Universität Heidelberg/Fachhochschule Heilbronn
Deutsches Krebsforschungszentrum Heidelberg

Studiengang Medizinische Informatik
Diplomarbeit

Genetische Algorithmen zur Molekülstrukturoptimierung

Bastien Chevreux

1. März 1997

Referentin: Prof. Diana Schmidt
Koreferent: Privatdozent Dr. Sándor Suhai

Vorwort

"You always find teh one typo in print that missed in galley proof."

Wie in so vielen Fällen, so ist auch dieses Vorwort erst ganz zum Schluss der vorliegenden Arbeit geschrieben worden. Eigentlich müsste es also Nachwort heissen und an das Ende der Diplomarbeit gesetzt werden – aber dann liest es ja niemand.

Allgemeines

Wer jetzt schon den Rotstift gezückt hat und im vorangegangenen Absatz neben diversen Kommafehlern auch das Fehlen des Buchstabens "ß" rot anstreichen wollte, den muss ich leider enttäuschen: diese Arbeit ist nach den Regeln der neuen Rechtschreibung verfasst worden. Neben einigem Unsinn sind viele intelligente Neuerungen darin aufgenommen worden, wobei ich die Vereinheitlichung von Kommaregeln, Groß-/Kleinschreibung und Doppel-S- bzw. ß-Schreibung eindeutig zur letztgenannten Kategorie zähle. Wie dem auch sei, große Unterschiede im Schriftbild ergeben sich nicht, nur schreibt (und liest) es sich nach einer kurzen Eingewöhnungsphase viel flüssiger.

Im Laufe dieser Arbeit hat es einige Hoch-, aber auch Tiefpunkte gegeben, und so mancher "Aha-Effekt" sorgte für Spannung, Freude und Niedergeschlagenheit. Nicht selten produzierte eine Mischung aus Beharrlichkeit und Zufall interessante Ergebnisse, auch wenn diese nicht immer wie erwartet – oder erhofft – ausfielen. Ich hoffe, diese facettenreichen Aspekte der Forschung in den folgenden Kapiteln einigermaßen interessant dargestellt zu haben, auch wenn es sich zum Teil nicht vermeiden ließ, riesige Tabellen zu deren Darstellung zu erzeugen.

Ein wenig reflektieren die Zitate am Anfang eines jeden Kapitels diese

ständigen Berg- und Talfahrten. Sie entstammen der Feder von David Gerrold, welcher sie Solomon Short – einer seiner Hauptpersonen – in seinen Romanen *A Day for Damnation, A Rage for Revenge, A Matter for Men* und *A Season for Slaughter*[1] in den Mund legte.

Typographisches

Werden Fachbegriffe im Text neu eingeführt, so sind sie *kursiv* gedruckt. Im anschließenden Fließtext werden sie als bekannt vorausgesetzt und nicht mehr hervorgehoben. Weniger wichtige Fachbegriffe werden in Fußnoten kurz erklärt. Wird ein Fachbegriff vor dessen Einführung benutzt, so wird er in der Regel von einem Verweis auf den Abschnitt mit dessen Einführung begleitet. Die Ausnahme von der Regel stellt Kapitel 1 dar, welches sich als zusammenfassende Einführung in die Problematik der Aufgabenstellung versteht und ansonsten praktisch nur noch aus Verweisen bestünde.

Danke

Großen Dank gebührt allen, die mir in irgendeiner Weise bei der Verfassung dieser Arbeit geholfen haben.
Dies sind

- Frau Prof. Schmidt und Herr Dr. Suhai, die für die intensive Betreuung dieser Arbeit zuständig waren.
- Dipl. Inform. Frank Herrmann, der mir als ständiger Ansprechpartner den Einstieg in die Materie sehr erleichterte und mir seine schnellen Energiefunktionen zur Verfügung stellte.
- Bernhard Przywara, für dessen Hilfe bei der Entdeckung von Fehlern in den GNU-C++Libs des Parsytec-Rechners und für die Bereitstellung von Rechenzeit auf diesem Rechner, als es bereits kurz vor knapp war.
- Dipl.-Inform. Med. Brigitte Euler, für Hilfestellungen beim SAS Programm und das Korrekturlesen dieser Arbeit schon in sehr frühen Stadien.
- Cand.-Inform. Med. Martina Busch, für das Korrekturlesen und ohne deren PPro200 der wichtigste Teil der Berechnungen von Kapitel 6 gar nicht zustande gekommen wäre.

[1] alle erschienen bei Bantam Books

– Dipl.-Inform. Med. Anke Häber, für das Auffinden von Fehlern in den sehr späten Stadien dieser Arbeit.

– Dipl.-Inform. Med Martin Haag, der für einen weiteren Teil der Berechnungen in Kapitel 6 seinen P133 zur Verfügung stellte.

Vergessen will ich auch nicht all diejenigen Leute, die mir im IRC auf #amigager und #unix geholfen haben, über größere und kleinere Probleme hinwegzukommen, sei es nun bei der C++–Programmierung oder LaTeX betreffend. Es sind ganz einfach zu viele.

Last but – certainly – not least sollen an dieser Stelle meine Eltern genannt werden: Ohne sie hätte ich kaum studieren können und – seien wir ehrlich – ohne sie hätte es diese Diplomarbeit niemals gegeben.

Bastien Chevreux. Heidelberg, den 28. Februar 1997

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung und Motivation

"In the beginning there was nothing. Then it all exploded."

Proteine

Unter den biologisch relevanten Makromolekülen nehmen Proteine eine wichtige Stelle ein. Sie sind der Hauptbestandteil des Zytoplasmas in biologischen Zellen und erfüllen im ganzen Körper vielfältige Funktionen, angefangen von Strukturbildung und Energieerzeugung über Biokatalyse bis hin zu Antikörpern des Immunsystems. Proteine bestehen aus zum Teil verzweigten Kettenmolekülen, die aus 20 natürlich vorkommenden Aminosäuren aufgebaut sind. Verschiedene Proteine unterscheiden sich sowohl in der Anzahl als auch in der Reihenfolge (Sequenz) der Aminosäuren. Die Sequenz oder Primärstruktur eines Proteins lässt sich heute relativ einfach bestimmen. Auch die lokale Faltung, die Sekundärstruktur, ist zu einem großen Teil vorhersagbar.

Für die Bestimmung der biologischen Funktion ist jedoch die räumliche, die Tertiärstruktur, von ausschlaggebender Bedeutung. Bedingt durch flexible Bindungen können die Kettenmoleküle im Prinzip sehr viele verschiedene räumliche Faltungen annehmen. Proteine mit derselben Primärstruktur falten sich aber immer in dieselbe Tertiärstruktur. Obwohl die Tertiärstruktur durch die Primärstruktur eindeutig festgelegt ist, lässt sie sich nicht direkt aus ihr ableiten. Sie lässt sich nur mit großem messtechnischen Aufwand bestimmen. Mit den Verfahren der Röntgenkristallographie und der Kernmagnetresonanz sind bisher einige tausend Strukturen aufgeklärt worden. Im Gegensatz dazu sind aber bereits mehrere hunderttausend Primärstrukturen bekannt, und mit wachsender Geschwindigkeit kommen neue Sequen-

zen hinzu. Deshalb ist man an rechnerunterstützten Verfahren interessiert, welche die Tertiärstruktur aus der Sequenz vorhersagen können.

Stand der Forschung

Ein Ansatz der Strukturoptimierung geht davon aus, dass die natürliche Faltung im Zustand der niedrigsten Energie vorliegt. Die Struktur des Moleküls wird optimiert, indem die Faltungsenergie mit einem geeigneten Verfahren minimiert wird. Das Optimierungsverfahren manipuliert dabei Parameter, welche die Struktur beschreiben. Eine übliche Vereinfachung benutzt dazu die Dihedralwinkel der flexiblen Bindungen und verwendet eine empirische Energiefunktion als Zielfunktion. Trotz dieser Vereinfachungen ist das Problem sehr komplex, da der Suchraum sehr viele lokale Minima enthält. Lokale Optimierungsverfahren wie Hillclimber oder Gradientenverfahren lokalisieren das nächstgelegene lokale Minimum. Andere Methoden wie die Build-up Methode oder Molekulardynamik untersuchen nur einen begrenzten Ausschnitt des Suchraums. Deshalb sind globale Optimierungsmethoden gefragt.

Das Hauptproblem bei globalen Optimierungsverfahren ist die Verlässlichkeit, mit der das globale Optimum gefunden wird. Selbst wenn bei mehrmaliger Wiederholung einer Optimierung dasselbe Optimum lokalisiert wird, ist das noch keine Garantie, dass es sich um das globale Optimum handelt. Deshalb werden hier für die Entwicklung der Methoden Probleme mit bekanntem globalen Minimum behandelt. So ist das Ergebnis einfach zu evaluieren. Neben der Monte Carlo Methode und dem Simulated Annealing sind besonders Genetische Algorithmen eine vielversprechende Methode.

Genetische Algorithmen

Genetische Algorithmen sind eine von der Natur motivierte heuristische Methode. In Analogie zur Darwinistischen Evolution wird die Entwicklung einer Population von Lösungen simuliert. Die Fitness der Individuen ergibt sich aus der Güte der Zielfunktion und entscheidet über deren Reproduktionschancen für die nächste Generation. Neue Lösungen werden durch Rekombination und Mutation erzeugt. Durch den Selektionsdruck setzen sich gute Lösungen in der Population mehr und mehr durch, bis die Population zu einer Lösung konvergiert ist.

Genetischen Algorithmen umfassen eine Klasse von Optimierungsalgorithmen deren einzelne Ausprägungen sich durch Konfigurationsparameter unterscheiden. Neben den verschiedenen grundlegenden Arten der Genetischen Algorithmen sind dies z.B. die Populationsgröße, der Selektionsdruck, der Rekombinationsoperator oder das Selektionsverfahren. Gesucht sind Genetische Algorithmen, die für eine bestimmte Problemkomplexität reproduzierbar das globale Optimum lokalisieren, da die Leistungsfähigkeit und die Reproduzierbarkeit der Ergebnisse von der Konfiguration des Algorithmus abhängt.

Ziele dieser Arbeit

Das Ziel dieser Diplomarbeit ist die Entwicklung von Konfigurationen Genetischer Algorithmen, die für ein Problem aus der Molekülstrukturoptimierung optimal sind. Die optimale Konfiguration benötigt möglichst wenig Evaluierungen der Energiefunktion, um das globale Optimum reproduzierbar zu lokalisieren. Das heisst, dass bei einer mehrmaligen Wiederholung der Optimierung mit derselben Konfiguration das globale Optimum jedesmal lokalisiert wird.

Dazu werden für jede Zielfunktion systematisch Optimierungen mit verschiedenen Konfigurationen durchgeführt. Dabei wird für jeden Konfigurationsparameter eine Menge von möglichen Werten festgelegt. Die Menge aller Kombinationen der für jeden Konfigurationsparameter möglichen Werte legt die Menge der zu untersuchenden Konfigurationen fest. Mit jeder Konfiguration werden mehrere Optimierungen durchgeführt. Nach Abschluss der Optimierungen wird überprüft, wie gut jede Konfiguration das globale Optimum lokalisiert hat und welche Konfiguration dies reproduzierbar mit der geringsten Anzahl von Energieevaluierungen erreicht.

Dieser Vorgang wird für kleine Peptide durchgeführt. Bei der Evaluierung der Ergebnisse soll festgestellt werden, ob man Regeln für die optimale Wahl der Konfigurationsparameter ableiten kann.

Zusätzlich gibt es die Möglichkeit, statt einer Population mehrere Subpopulationen zu verwenden. Da nur innerhalb der Subpopulationen gekreuzt wird, können sich diese in unterschiedliche Richtungen entwickeln. Da die Subpopulationen ihre besten Lösungen untereinander austauschen können sich gute Lösungen in anderen Subpopulationen durchsetzen, so dass schließlich der gesamte GA zu einer Lösung konvergiert. Es soll eine Methode ent-

wickelt werden, bei der mit Hilfe von Subpopulationen die Leistungsfähigkeit von Genetischen Algorithmen erhöht wird.

Aufbau dieser Arbeit

Nach der Erörterung der Problemstellung der Diplomarbeit in diesem Kapitel und einer kurzen Einführung in die Proteinchemie im folgenden zweiten Kapitel, werden Genetische Algorithmen als geeignetes Instrument zur Bearbeitung von Problemstellung innerhalb mathematisch nicht lösbarer Optimierungsaufgaben – wie zum Beispiel Molekülfaltung – in Kapitel 3 vorgestellt. Anschließend werden, in Kapitel 4, die für die Untersuchung relevanten Objekte aufgezeigt und die Methodik der Untersuchungen dargestellt. Kapitel 5 widmet sich ganz der Beschreibung und Bewertung von bekannten und neu eingeführten genetischen Operatoren, welche die Effizienz von Genetischen Algorithmen – mit einfachen Populationen – zur Molekülstrukturoptimierung beeinflussen. Diese Ergebnisse werden dann in Kapitel 6 für die Einführung des neuen Multipopulationsansatzes "pyramidale Kulturen" benutzt, um dessen Überlegenheit bei der Reproduzierbarkeit gesuchter Ergebnisse im Molekülkonformationsraum zu zeigen. Im abschließenden siebten Kapitel werden wichtige Ergebnisse noch einmal zusammengefasst, um dann Ausblicke auf weitere mögliche Forschungsrichtungen zu geben.

Kapitel 2

Einführung in die Proteinchemie

"Understanding the laws of nature does not mean that we are immune to their operations."

Dieses Kapitel stellt eine schematische und sehr kurze Einführung in die grundlegenden Mechanismen der Proteinchemie dar. Die Thematik wird unter dem Gesichtspunkt der in Kapitel 4 vorgestellten Energiefunktion besprochen und ist für deren Verständnis von Wichtigkeit. Angesichts des Umfangs dieses Gebiets kann allerdings nur ein Bruchteil des Wissens dargeboten und soweit erklärt werden, wie es für das Verständnis der vorliegenden Arbeit relevant ist.

2.1 Atome

Das aus dem griechischen stammende Wort Atom (a-tomos – "un-teilbar") bezeichnet das kleinste, mit chemischen Methoden herstellbare Teil des Elements, dessen typischen Eigenschaften es noch besitzt. Nach dem Modell von Rutherford und Bohr (1913) besteht das Atom aus einem Kern mit positiv geladenen Protonen und neutralen - d.h. ungeladenen - Neutronen, sowie einer Hülle aus negativ geladenen Elektronen. Der Durchmesser des Atomkerns liegt in der Größenordnung von 10^{-15}m und beherbergt 99,9% der Atommasse, das gesamte Atom hat einen Durchmesser von ungefähr $10^{-10}m$. In der Elektronenhülle des Atoms befinden sich beim elektrisch neutralen Atom genauso viele Elektronen wie sich Protonen im Kern befinden, so daß sich die Ladungen gegenseitig kompensieren. Die chemischen

Eigenschaften eines Atoms ergeben sich aus der Struktur und Anordnung der den Kern umgebenden Elektronen.

2.2 Chemische Bindungen

In Molekülen werden Atome durch chemische Bindungskräfte zusammengehalten. Diese chemischen Bindungskräfte sind stets auf Wechselwirkungen von Bindungs- bzw. *Valenzelektronen* zurückzuführen. Jede Bildung oder Veränderung eines Atomverbandes ist ein chemischer Vorgang, und ist stets von Energieänderungen durch Elektronendeplazierung innerhalb oder außerhalb der Hülle begleitet.

Atombindung / kovalente Bindung

Eine *kovalente Bindung* zwischen zwei Atomen kommt dadurch zustande, dass sich Valenzelektronen (negative Ladung) bevorzugt zwischen zwei Atomkernen (positive Ladung) aufhalten und von beiden gleichermaßen angezogen werden. Eine Veränderung des Abstands der beiden Atome wäre mit einem Energieaufwand verbunden, da bei einer Verringerung des Abstands die Abstoßungskräfte zwischen den positiv geladenen Kernen steigen würden und bei einer Vergrößerung die Anziehung zwischen den Valenzelektronen und beiden Atomrümpfen überwunden werden müßte. So bleiben beide Atome in einem weitgehend konstanten Abstand.

Die gemeinsamen Bindungselektronen befinden sich im zeitlichen Mittel nur dann genau in der Mitte zwischen beiden Atomen, wenn die positiven Ladungen der Kerne gleich groß sind. Dies trifft für alle Bindungen zu, die aus gleichen Atomen aufgebaut sind (z.B. C–C Bindungen). Anders ist dies bei ungleichen Bindungen, wie z.B. beim Wassermolekül (H_2O). Das Sauerstoffatom zieht aufgrund der größeren Protonenzahl – und somit der größeren positiven Ladung – im Kern die gemeinsamen Elektronen der beiden H-O-Bindungen stark an. Die Valenzelektronen halten sich im zeitlichen Mittel näher beim Sauerstoff als beim Wasserstoff auf. Der Sauerstoff hat somit einen Überschuß an negativer Ladung, währen die H-Atome infolge Elektronenmangels positiv geladen sind. Obwohl das Wassermolekül als ganzes elektrisch neutral ist, ist die elektrische Ladung innerhalb des Verbandes ungleichmäßig verteilt, und das Molekül verhält sich als *Dipol*.

Ionenbindung

Wenn bei zwei Bindungspartnern die Fähigkeit des einen Atoms, Elektronen anzuziehen, groß ist und die des anderen Atoms dagegen gering, so werden die Bindungselektronen ganz zum elektronegativen Partner hingezogen. Es entstehen positiv und negativ geladene Ionen, Kationen und Anionen. Diese ziehen einander infolge der gegensätzlichen Ladung an, es entsteht eine Ionenbindung. Eine *Ionenbindung* ist schwächer als eine kovalente Bindung und kann dementsprechend einfacher gelöst werden.

Wasserstoffbrückenbindung

Zwischen Dipolmolekülen herrschen zwischenmolekulare Kräfte. Besonders ausgeprägt sind diese, wenn ein Wasserstoffatom an ein stark elektronegatives Atom – z.B. Sauerstoff oder Stickstoff – gebunden ist. Das positiv polarisierte Wasserstoffatom kann dann mit einem negativ polarisierten Atom in Wechselwirkung treten. Wenn aufgrund der Größe und der räumlichen Struktur der Moleküle ein geeigneter Bindungsabstand möglich ist, entstehen relativ stabile Verbindungen, die als *Wasserstoffbrücken* bezeichnet werden.

Die Bindungsenergie einer Wasserstoffbrücke beträgt weniger als $\frac{1}{10}$ derjenigen einer einfachen Atombindung. Die Wasserstoffbrücke kann entsprechend leichter gelöst werden.

Hydrophobe Wechselwirkung

Die hydrophoben Wechselwirkungen (*Van-der-Waals*-Kräfte) beruhen auf der Tatsache, daß unpolare (hydrophobe) Aminosäureseitenketten eine enge Nachbarschaft zueinander bevorzugen und sich dabei in einer wässrigen Umgebung vor allem im Molekülinneren anordnen. Dadurch drängen sie die Wassermoleküle der das Proteinmolekül umgebenden wässrigen Lösung aus dem Inneren des (Protein-)Moleküls heraus.

Ionenbindungen, Wasserstoffbrücken und hydrophobe Wechselwirkungen sind schwache Bindungen. Sie sind mitbestimmend für die chemischen Umsetzungen einer Zelle und werden ständig mit hoher Geschwindigkeit gelöst und wieder neu geknüpft. Sie sind aber auch für die Stabilisierung von großen Makromolekülen verantwortlich.

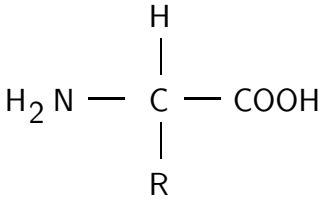

Abbildung 2.1: Grundstruktur von Aminosäuren

2.3 Proteine (Eiweiße)

Proteine sind die vielseitigste Stoffklasse im menschlichen Körper und spielen aus diesem Grund eine Schlüsselrolle für die Biochemie des Menschen. Sie stellen ca. 17% der Körpermasse zur Verfügung und übernehmen Aufgaben als Aufbaustoffe, Strukturelemente, Betriebsstoffe, Katalysatoren (Enzyme), Hormone und Antikörper.

Aminosäuren

Die einfachsten Bausteine der Eiweiße sind *Aminosäuren* (AS). Dies sind Karbonsäuren[1], bei denen ein Wasserstoffatom durch eine Aminogruppe – NH_2 ersetzt ist. Die 20 im menschlichen Körper vorkommenden wichtigesten Aminosäuren haben alle – mit Ausnahme des Prolins – den in Abbildung 2.1 skizzierten Aufbau.

Das bedeutet, dass alle Aminosäuren sich nur im Aufbau ihres Restes **R** unterscheiden. Der Rest kann sehr kurz – z.B. nur ein Wasserstoffatom bei Glycin – sein, er kann – wie bei Methionin – aus einem Kettenmolekül bestehen, es können einfache Kohlenstoffringe darin auftauchen (z.B. Phenylalanin und Tyrosin) oder sogar doppelte Ringe, wie bei Tryptophan.

Die im Eiweißstoffwechsel wichtigen Aminosäuren sind alle α–AS, haben Links-Konfiguration und sind – bis auf Glycin – optisch aktiv. Trotz ihres Namens haben Aminosäuren sowohl saure als auch basische Eigenschaften. Im menschlichen Körper sind 25 AS bekannt, davon 10 essentiell. Nur 20 verschiedene Aminosäuren werden für den Proteinaufbau benutzt.

[1] organische Verbindungen mit Karboxilgruppe(n): R-COOH

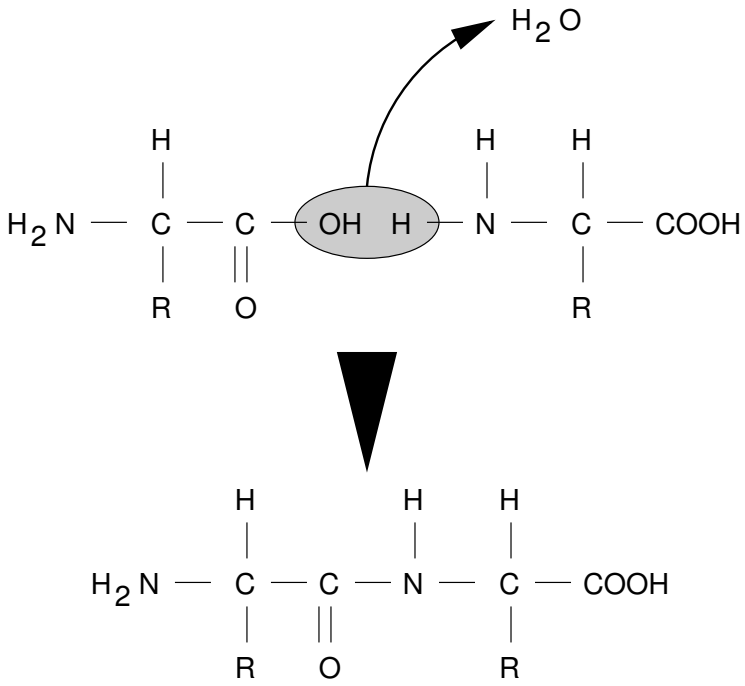

Abbildung 2.2: Peptidbindung

Peptide

Die wichtigste Eigenschaft von Aminosäuren ist ihre Fähigkeit zur Formation von Kettenmolekülen. Unter Abspaltung von Wasser bildet die COOH-Gruppe einer AS mit der NH_2-Gruppe einer anderen Aminosäure eine Peptidbindung (Abbildung 2.2).

Zwei Aminosäuren bilden ein Dipeptid, drei ein Tripeptid usw. Zwischen 2 und 10 AS bilden Oligopeptide, Polypeptide haben zwischen 10 und 100 Aminosäuren.

Proteine

Makromoleküle aus über 100 durch Peptidbindungen verbundenen Aminosäuren werden Proteine genannt. Nach ihrer Form werden globuläre bzw. sphärische[2] und langgestreckte fibrilläre[3] Proteine unterschieden, die allerdings alle bei Hitze und extremen pH-Werten denaturieren, d.h. ihre natürli-

[2]meist wasserlösliche Proteine

[3]meist nicht wasserlösliche Strukturproteine mit hoher Beständigkeit gegen Säuren, Alkalien und protelytischen Enzymen

che Form verlieren. Bei 100 Aminosäuren langen Ketten mit jeweils 20 Möglichkeiten ergeben sich schon ca. $1,27 \cdot 10^{130}$ verschiedene Kombinationsmöglichkeiten, jedoch sind "nur" $5 \cdot 10^6$ verschiedene Proteine bekannt.

Unter den biologisch relevanten Makromolekülen nehmen Proteine eine wichtige Stelle ein. Sie sind der Hauptbestandteil des Zytoplasmas in biologischen Zellen und erfüllen im ganzen Körper vielfältige Funktionen, angefangen von Zellkörperstrukturbildung und Energieerzeugung über Biokatalyse bis hin zu Antikörpern des Immunsystems.

2.4 Proteinmolekülstrukturen

Es gibt vier verschiedene Klassen bei der Strukturierung eines Proteins, wobei die Schwierigkeiten bei der Bestimmung des Proteinaufbaus von Klasse zu Klasse zunimmt. Die Kenntnis über die verschiedenen Strukturenklassen eines Moleküls erlaubt Vorhersagen über sein Verhalten auf molekularer Ebene.

2.4.1 Strukturklassen

Primärstruktur

Die Reihenfolge (*Sequenz*) von Aminosäuren ist für jedes Protein spezifisch. Diese - *Primärstruktur* genannte - Sequenz ist genetisch durch die Abfolge von Basen im Erbmaterial eines Lebewesens festgelegt. Ändert sich die Sequenz, so ändert sich das Protein und letztendlich auch seine Funktion.

Sekundärstruktur

Die *Sekundärstruktur* beschreibt die räumliche Anordnung der aneinandergereihten Aminosäuren, z.B. in Form einer α–Helix[4] oder einer Faltblattstruktur[5]. Bestimmend für die Faltung der Sekundärstruktur ist – ausgehend von der Sequenz – die Ausbildung intramolekularer Wasserstoffbrückenbindungen.

[4] schraubenförmige Struktur - ähnlich einer Wendeltreppe - mit Linksdrehung
[5] planar im Raum gefaltete Struktur

Tertiärstruktur

Die Bildung zusätzlicher Bindungsarten – z.B. Disulfid[6]- und Ionenbindungen – zwischen Seitenketten eines Proteins führt zu einer weiteren räumlichen Faltung des Moleküls, welches abhängig von der Lage, Form, Größe und Ladung der verknüpften Aminosäuren sowie den Molekülen seiner – zumeist wässrigen – Umgebung ist. Für die Bestimmung der biologischen Funktion ist jedoch diese *Tertiärstruktur* von ausschlaggebender Bedeutung.

Proteine mit derselben Primärstruktur falten sich immer in dieselbe Tertiärstruktur. Obwohl die Tertiärstruktur durch die Primärstruktur eindeutig festgelegt ist, lässt sie sich nicht direkt aus ihr ableiten. Sie lässt sich nur mit großem messtechnischem Aufwand bestimmen. Da jedoch Sekundär- und Tertiärstruktur oft nicht deutlich voneinander zu trennen sind, wird häufig von einer allgemeinen Kettenkonformation gesprochen.

Quartärstruktur

Durch die Verknüpfung mehrerer Polypeptid- bzw. Proteinketten entsteht ein funktionstüchtiger Proteinkomplex. Die räumliche Anordnung dieses Gesamtmoleküls wird *Quartärstruktur* genannt.

2.4.2 Strukturbestimmung

Die Analyse der Sequenz (Primärstruktur) eines Proteins ist heutzutage problemlos möglich und es sind bisher einige hunderttausend Sequenzen bekannt. Auch die Sekundärstruktur ist, soweit sie von der Tertiärstruktur zu trennen ist, in den meisten Fällen bestimmbar. Die Tertiär- und Quartärstrukturen allerdings müssen noch immer mit hohem gerätetechnischen Aufwand bestimmt werden, wenn es überhaupt möglich ist. Auch sind die Verfahren der Röntgenkristallographie und der Kernmagnetresonanz sind nicht auf alle Peptide anwendbar.

Eine Alternative dazu wird durch Rechnersimulation von Faltungen geboten. Ein Ansatz der Strukturoptimierung geht davon aus, dass die natürliche Faltung im Zustand der niedrigsten Energie vorliegt. Die Struktur des Moleküls wird optimiert, indem die Faltungsenergie mit einem geeigneten Verfahren minimiert wird. Das Optimierungsverfahren manipuliert dabei Parameter, welche die Struktur beschreiben. Es existieren jedoch keine Al-

[6] Atombindung aus zwei Schwefelatomen.

gorithmen, welche den in der Natur vorkommenden Ablauf des Faltungsvorganges simulieren können. Was fehlt, ist das entsprechende Wissen, wie und warum sich Proteine in genau ihre Form falten.

Kapitel 3

Genetische Algorithmen

"The universe has its own cure for stupidity. Unfortunately, it does not always apply it."

Genetische Algorithmen (GA) sind eine Klasse von nichtlinearen, adaptiven, heuristischen und zum Teil parallelen Methoden für Such- und Optimierungsprobleme. Sie basieren auf dem natürlichen Vorbild der Evolution und werden typischerweise als Black-Box Methoden angewandt. Über viele Generationen evolvieren Populationen in der Natur nach den Prinzipien der natürlichen Selektion und dem „Überleben des Stärkeren", welche zuerst von Charles Darwin in seinem Buch *The Origin of Species* postuliert wurden. Indem diese Prinzipien der Natur abgeschaut und imitiert werden, können Genetische Algorithmen künstliche Populationen generieren und einer Evolution unterziehen, deren einziges Ziel es ist, ihnen gestellte Probleme möglichst optimal zu lösen.

Genetische Algorithmen brauchen keine Gradienteninformationen oder andere Formen von problemspezifischem Wissen über die von ihnen zu lösende Aufgabe[1]. Aus diesem Grund werden sie vor allem in solchen Aufgabengebieten eingesetzt, welche entweder noch nicht gut verstanden werden oder deren komplette Modellierung aus mathematischen oder rechnerischen Gründen unmöglich ist [24].

3.1 Geschichte der Genetischen Algorithmen

Die Fundamente der GA wurden in den 60'er Jahren in den USA von John Holland gelegt und in seinem 1975 erschienenen Buch veröffentlicht [31]. Die

[1] siehe dazu jedoch NFL-Theorem (Abschnitt 3.2).

erste Anwendung dieser Theorie für Parameteroptimierung wurde von De Jong, einem Schüler Hollands, im selben Jahr publiziert. Jedoch erst ca. 10 Jahre später setzte sich langsam diese "neue" Art der Problemlösung durch und mit David E. Goldbergs Buch *Genetic Algorithms in Search, Optimization, and Machine Learning* [19] wurde dieses Thema einer breiten Öffentlichkeit zugänglich gemacht.

3.2 Funktionsweise der Genetischen Algorithmen

Der bis zu diesem Zeitpunkt größte Teil der Forschung hatte zum Ziel, empirische Regeln für eine optimale Performance von GA zu finden. Praktisch jedes Problem hat spezifische Parameter und (genetische) Algorithmen, mit dem es optimal gelöst werden kann. Nur allgemeine und äußerst grobe Werte können als Richtlinien angesehen werden. Es gibt bis jetzt keine generell akzeptierte Theorie, welche die diesen Algorithmen eigenen Eigenschaften erklärt. Trotzdem wurden viele Hypothesen in den Raum gestellt, welche zum Teil den Erfolg erklären können[2].

Markovketten

Theoretische Analysen kleiner, jedoch immer noch sehr komplexer, Genetischer Algorithmen können in einem gewissen Rahmen mit Markovketten erstellt werden. Es stellt sich jedoch die Frage nach Machbarkeit und Aussagekraft solcher Modelle, da trotz fundierter theoretischer Basis die Modelle entweder unendliche Populationsgrößen annehmen oder die kombinatorischen Möglichkeiten bei zunehmender Populations- oder Genomgröße ins Unendliche anwachsen [38].

No Free Lunch

In diesem Zusammenhang ist das von Wolpert und Macready postulierte "No Free Lunch"–Theorem (NFL) interessant[3].

Die grobe Vereinfachung diese Theorems besagt, dass für jedes beliebige Paar von Such- bzw. Optimierungsalgorithmen der eine Algorithmus für "genauso viele" Problemstellungen besser als der andere sein wird, wie es umgekehrt ebenso wahr ist. Dies gilt für alle Algorithmen.

[2] siehe dazu Abschnitte 3.6 (Schema-Theorem), 3.7 (Building-Block-Hypothese) und 3.8 (Kritik an ...).

[3] Manchmal auch TINSTAFL abgekürzt: There Is No Such Thing As Free Lunch.

Eine Konsequenz dieses Theorems ist, dass Genetische Algorithmen ohne die Einführung von problemspezifischem Wissen als Lösungshilfe nicht besser und nicht schlechter sind als z.B. reine Zufallssuche [27].

3.3 Die Natur als Vorbild

Das Spiel des Lebens

In einer natürlichen Umgebung treten einzelne Individuen einer Spezies in einen Konkurrenzkampf um Ressourcen wie Futter, Wasser und Schutz. Diesen fechten sie sowohl mit Vertretern anderer Spezies als auch untereinander aus. Zusätzlich müssen sie für Nachwuchs sorgen, das heißt, sie müssen sich nach einem geeigneten Partner umschauen und diesen dann erfolgreich umwerben. Diejenigen Vertreter einer Spezies, die auf den Gebieten des Überlebens und der Partnerwerbung am Erfolgreichsten agieren, werden mit hoher Wahrscheinlichkeit eine größere Anzahl an Nachkommen haben als ihre weniger glücklichen Artgenossen, welche z.B. einem Räuber zum Opfer gefallen sind oder für Paarungspartner zu unattraktiv sind.

Auf diese Weise werden Gene von erfolgreichen Individuen im Durchschnitt öfter vererbt als die der weniger erfolgreichen. Mit jeder Generation werden diese Gene also weiter verbreitet. In einigen Fällen können verschiedene Genkombinationen bei einer Paarung entstehen, die zusammen wiederum ihren Trägern eine bessere Adaption an dessen Umwelt ermöglichen.

Selfish Genes

Dass im Endeffekt nur Gene den alles entscheidende Faktor im Spiel des Lebens darstellen, ist zwischen und innerhalb von verschiedenen wissenschaftlichen Fachgebieten eine umstrittene Theorie. Letztlich sind es jedoch diese Ansammlungen aus den Grundbausteinen des Lebens, welche vererbt werden.

Die Erzeugung – und in den meisten Fällen auch die Aufzucht – von Nachkommen ist, vom Standpunkt eines Lebewesens betrachtet, für ein Individuum oder ein Elternpaar eine zeit- und kräftezehrende Aufgabe, welche im täglichen Überlebenskampf einen wesentlichen Bestandteil ausmacht. Im Laufe der Jahrmillionen hat die Evolution deshalb verschiedene Strategien hervorgebracht, um das Überleben einer Spezies – und somit auch deren Gene – zu sichern. Diese reichen von vererbten Instinkten zur Suche eines

Eiablage- oder Nistplatzes über einfache Beschützerinstinkte bis hin zu verschiedenen Ausprägungen des "Kindchenschemas".

Ansammlung von Wissen

Der in der Biologie geführte Nachweis von vererbten unbedingten Reflexen, Automatismen und Instinkten [41] spricht ebenso dafür, dass nicht nur Form und Funktion eines Lebewesens, sondern auch Wissen in jeglicher Form durch die Aneinanderreihung von Basenpaaren in der DNA von Generation zu Generation weitergegeben wird.

Andererseits steht aus eigener Anschauung sofort ein vermeintlicher Gegenbeweis parat: Das von den Eltern gelernte Wissen steht ihren Nachkommen nicht von Geburt an zur Verfügung. Es muss – mehr oder weniger mühsam – erlernt werden.

Doch das steht nicht im Widerspruch zur Weitergabe von "Wissen" durch Vererbung. Das durch Gene vererbte Wissen, und sei es nur in Form von Instinkten, stellt zusammen mit der allgemeinen Beschaffenheit eines Individuums eine Grundlage dar. Diese Grundlage wird zu 100% von den Genen bestimmt. Darauf aufbauend setzt erlerntes Wissen die Adaption an die Umgebung fort.

Vererbtes und erlerntes Wissen schließen sich demnach nicht gegenseitig aus, sondern stellen die zwei Hauptkomponenten zur erfolgreichen Verfolgung eines Zieles dar: der Weitergabe von Genen.

3.4 Genetische Algorithmen in der Computersimulation

Genetische Algorithmen basieren auf Prinzipien der Molekulargenetik und funktionieren in direkter Analogie zur biologischen Evolution. Es existiert ein Lebensraum, der Computer, in dem Bedingungen für das Überleben und die Vererbung gegeben sind. Die Bedingung für das Überleben stellen die *Objective Score* bzw. *Fitnessfunktionen*. Sie repräsentieren das Ergebnis der Projektion eines Satzes von Variablen auf die gewünschte *Zielfunktion*.

In diesem Lebensraum arbeiten Genetische Algorithmen mit *Populationen* von *Individuen*, wobei jedes Individuum einen Satz von Variablen zur vollständigen Lösung des vorgegebenen Problems beinhaltet. Jedem einzelnen Individuum wird aufgrund seiner "Leistung" beim Lösen dieses Pro-

blems eine *Fitness* zugeordnet. Zum Beispiel könnte es sich bei der Fitness um den Energiewert eines Moleküls handeln. In der Analogie zur Natur bedeutet dies, wie gut ein Wesen um Ressourcen kämpfen kann, wie viele es für sich vereinnahmen kann. Den gut angepaßten Individuen wird dann Gelegenheit gegeben, Nachkommen mit anderen zu zeugen, um sich damit fortzupflanzen. Bei dieser Fortpflanzung werden die natürlichen Mechanismen des *Crossovers* und der *Mutation* angewandt, so daß die Nachkommen Gene von jedem Elternteil bekommen und zusätzlich noch ein Unsicherheitsfaktor eingebaut ist. Es ist klar, daß weniger fitte Individuen ihre Gene nicht so oft weitervererben können sollten, um somit über die Zeit unproduktive Genkombinationen regelrecht aussterben zu lassen.

3.4.1 Einfacher Pseudoalgorithmus für Genetische Algorithmen

Als erste Übersicht für einen simulierten Evolutionslauf mit Genetischen Algorithmen kann folgendes Ablaufschema dienen:

1. Definition und Initialisierung der für den Lauf benötigten Parameter
2. Initialisierung einer Ausgangspopulation mit zufällig entstandenen Individuen
3. Ausrechnen des *Objective Score*[4] aller Individuen innerhalb der Population und Skalierung desselben anhand des eingestellten Skalierungsschemas.
4. Selektion der Eltern für die Nachfolgegeneration anhand der im vorigen Schritt berechneten Fortpflanzungswahrscheinlichkeiten.
5. Für jedes in der Population zu ersetzende Individuum wird mit den in Schritt 4 ausgewählten Eltern ein Kind erzeugt.
6. Mit den neuen Individuen (Kindern) die Population ganz oder teilweise ersetzen und diese Population als Ausgangspopulation definieren.
7. Solange das Abbruchkriterium für den Genetischen Algorithmus nicht erfüllt wird, zurück zu Schritt 3.

[4] siehe *Kodierung genetischer Information für Simulationen* in Abschnitt 3.4.2 auf Seite 19

8. Ausgabe der Endergebnisse.

Je nach Implementation der Genetischen Algorithmen können einzelne Schritte wegfallen oder etwas früher oder später ausgeführt werden. Andere, optionale Schritte können gegebenenfalls hinzukommen, dennoch kann der oben angegebene Algorithmus als gutes Verständnisfundament für die grundlegende Arbeitsweise der Genetischen Algorithmen dienen.

3.4.2 Genom

Die Grundlage der Vererbung bildet das *Chromosom*, welches aus zwei identischen *Chromatiden* aufgebaut ist. In der Zelle sind Chromosomen die Träger der Erbinformation, auf denen die Gene linear angeordnet sind. Sichtbar sind sie allerdings nur während der Zellteilung, wo sie als hakenförmige Gebilde mit Einschnürungen erscheinen. Im Normalzustand – der Teilungsruhe – liegen die Chromosomen als lange, dünne und vielfach gewundene Fäden vor, welche ein Netzwerk innerhalb des Zellkerns bilden. Ihr hauptsächlicher Bestandteil ist Desoxyribonukleinsäure – die DNA[5]. Dieses Makromolekül besteht aus fadenförmigen DNA-Molekülen, welche die Struktur einer Doppelspirale – eine Doppelhelix – aufweisen. Jede Spezies hat eine spezifische Anzahl an Chromosomen als Erbgutinformation. Dieser Chromosomensatz kann bei höheren Lebewesen in mehrfacher Ausführung vorhanden sein (multiploide Chromosomensätze). Als *Genom* wird in der Regel der haploide Chromosomensatz und die in ihm lokalisierten Gene bezeichnet, im weiteren Sinne bei Genetischen Algorithmen jedoch auch die Gesamtheit der Gene eines Individuums.

Genetische Information in der Zelle

Die genetische Information ist in der DNA-Doppelhelix durch Gene repräsentiert. Gene sind DNA-Abschnitte, welche ein Protein (z.B. ein Enzym) kodieren. Die Bauelemente der DNA bilden die *Nukleotide*, welche aus drei Elementen bestehen:

- der Pentose (ein Zuckermolekül mit 5 Kohlenstoffatomen) *Desoxyribose*
- einem Phosphorsäurerest

[5] aus dem englischen: Desoxyribonuclein*acid*

– und einer *Purin-* (*Adenin (A)* oder *Guanin (G)*) oder *Pyrimidinbase* (*Thymin (T)* oder *Cytosin (C)*)

Jeder DNA-Einzelstrang besteht so aus linearen und unverzweigten *Nukleinsäuremolekülen* (Desoxyribose mit anhängendem Phosphorsäurerest). An der Pentose hängt eine der vier möglichen Basen (Adenin, Thymin, Guanin, Cytosin), welche über Wasserstoffbrücken mit der jeweils komplementären Base im anderen Strang verbunden ist. Die vollständige Information zur *Proteinbiosynthese* ist ausschließlich in der Abfolge der Basen in der DNA enthalten, jedoch wird diese nicht als direkte Matrize benutzt. Sie stellt die Vorlage für die Herstellung einer Arbeitskopie dar, der *Messenger-RNA* oder auch *mRNA*. Die mRNA besteht im Gegensatz zur RNA aus einem Einzelstrang, zusätzlich ist die Desoxyribose durch *Ribose* und Thymin durch *Uracil (U)* ersetzt. Sie enthält jedoch genau die gleichen Informationen in komplementärer Form wie die DNA, weshalb dieser Kopiervorgang auch *Transkription* genannt wird.

Die mRNA wird anschließend aus dem Zellkern hinaus an den eigentlichen Ort der Biosynthese transportiert, dem Zytoplasma. Im Zytoplasma befinden sich die aus ribosomaler RNA (*rRNA*) und Proteinen aufgebauten Ribosome und die transfer-RNA (*tRNA*). Die Umsetzung der Information in ein Protein – die *Translation* – wird durch Ribosome sichergestellt. Diese fahren, vereinfacht ausgedrückt, die mRNA entlang und suchen zu jedem *Kodon* (Basentriplet) auf der mRNA eine tRNA mit passendem, d.h. komplementärem, *Anti-Kodon*. Jede tRNA transportiert zusätzlich eine zu ihrem Anti-Kodon spezifische Aminosäure. Diese wird vom Ribosom von der tRNA abgehängt (dabei wird der Rest der tRNA entlassen) und an die so mit der Zeit wachsende Aminosäurekette angehängt.

Kodierung genetischer Information für Simulationen

In den einfachen Ausführungen der Genetischen Algorithmen besteht das Genom aus einem haploiden Chromosomensatz. Dieses Genom G enthält alle Gene eines Individuums und wird durch einen Speicherblock simuliert, in dem die Variablen eines Problems in einer festgelegten, jedoch beliebig definierbaren Reihenfolge abgelegt sind.

$$V = (v_0, \ldots, v_i \,|\, i \in \mathbb{N}, v_i \in \mathbb{C}) \qquad \text{Variablensatz eines Problems}$$

Der Variablensatz v_i besteht üblicherweise aus genau den i Variablen, welche zur Lösung bzw. Berechnung der *Objective Score* oder auch *Ziel*funktion $f_s(V) = f_s(v_0, \ldots, v_i)$ nötig sind.

Wie oben schon angedeutet, können die Variablen aus jeder beliebigen Wertemenge kommen, bis hin zu den komplexen Zahlen[6]. Bei vielen Anwendungen der Genetischen Algorithmen – auch im naturwissenschaftlichen Bereich – ist die Menge der reellen Zahlen aber die am häufigsten anzutreffende, weshalb im weiteren Verlauf dieser Arbeit in der Regel $\mathbb{R}$ als Grundlage dienen soll.

Jede Variable v_i dieses simulierten Genoms wird durch eine festzulegende Anzahl an Bits repräsentiert. Die Anzahl der Bits j_i pro Variable kann gleich sein, muss es aber nicht. Ein Bit stellt demnach das Äquivalent zur Base innerhalb einer biologischen DNA, und eine Variable kann als Äquivalent eines Gensatzes gesehen werden. Im Gegensatz zur Base mit den vier Zuständen A, T, G und C – bzw. A, U, G und C in der RNA – kann ein Bit nur die beiden Zustände 0 und 1 annehmen. So kodiert ein Bitstrang mit n Bits genau 2^n Zustände, während ein RNA Strang mit n Basen genau $4^n = 2^{2n}$ Zustände kodiert.

Genotyp und Phänotyp

Es wird in der Biologie streng unterschieden zwischen der Anordnung der Basen auf einem DNA bzw. RNA Strang – dem Genotyp – und dessen Merkmalsausprägung beim Lebewesen – dem Phänotyp. An einer Stelle im Genom G könnte als Genotyp g_i für ein Merkmal z.B. die Folge –ATTGCTAGTCAT– stehen. Der Phänotyp v_i dieser Sequenz könnten dann womöglich braune Augen sein, oder grüne. Verschiedene Ausbildungsformen des gleichen Gens, die zu unterschiedlichen Phänotypen führen, werden *allele Gene* oder auch einfach *Allele* genannt.

Wie schon aus der Namensgebung ersichtlich, existiert in der Simulation kein Unterschied zwischen den Variablen der $Zielfunktion$ und den Phänotypen eines Genoms: sie sind identisch.

Jede dieser Variablen v_i mit jeweils j_i Bits kann demnach die Werte

[6] Jedoch lassen sich naturgemäß reelle oder komplexe Zahlen nicht genau im Rechner darstellen. Dies ist ein generelles Problem der Informatik, und deshalb soll an dieser Stelle nur ein Verweis auf die Fachliteratur stehen.

$\{0, \ldots, 2^{j_i} - 1\}$ annehmen. Für viele Probleme aus wissenschaftlichen Gebieten ist dies nicht adäquat. Dort werden reelle Werte, also Fließkommazahlen benötigt. Eine Lösung ist, die Anzahl der Bits pro Variable auf das Format einer benötigten Fließkommazahl zu erhöhen und die Variable von vornherein als ein solches Format zu behandeln. Neben dem Nachteil, dass dafür mindestens 32 Bits (üblicherweise jedoch 64 Bits) pro Variable gebraucht würden, bringt dieses Zahlenformat weitere Schwierigkeiten bzw. sub-optimale Ergebnisse[7].

In vielen Fällen ist der Wertebereich $\mathbb{W}^{\in\mathbb{R}}$ einer Variablen v_i zusammenhängend und kann diskretisiert werden. Man kodiert die Variable v_i binär auf dem Genom als g_i und rechnet bei Bedarf die Binärzahlen g_i in Fließkommazahlen um, welche dann die Variable v_i repräsentieren. Bei j_i Bits für eine Variable v_i und einem Wertebereich

$$\mathbb{W}^{\in\mathbb{R}} : \{U_{min} - U_{max}\}$$

ergeben sich folgende Umrechnungsformeln:

$$v_i = U_{min} + g_i \frac{U_{max} - U_{min}}{2^{j_i}}$$

und

$$g_i = \left\lfloor (v_i - U_{min}) \frac{2^{j_i}}{U_{max} - U_{min}} \right\rfloor$$

Die Gesamtlänge l des Genoms G in Bits beträgt bei n Variablen in jedem Fall

$$l = \sum_{i=1}^{n} j_i \qquad \text{bei unterschiedlicher Anzahl } j_i \text{ an Bits pro Variable } v_i$$

bzw.

$$l = n \cdot j \qquad \text{wenn } j_i \text{ konstant ist.}$$

Die *Transkription* wird in den meisten Genetischen Algorithmen vernachlässigt. Der Nukleus einer Zelle stellt primär eine Schutzfunktion für die DNA dar, welche in einem Rechner unnötig ist. Im Gegensatz zur Zelle kann ein Computer deshalb direkt auf den Genstrang zur Informationsgewinnung zugreifen.

[7] siehe Schema-Theorem und Building-Block-Hypothese in den Abschnitten 3.6 und 3.7.

3.4.3 Individuum und Objective Score

Das Genom G und der darauf basierende Variablensatz V werden zusammen mit der *Objective Score* Funktion $f_s(V)$ zu einem Individuum I zusammengefasst. $f_s(V)$ ist jeweils eine vom Benutzer des Genetischen Algorithmus zu entwickelnde Funktion, welche den Variablensatz V in geeigneter Weise in einen Zielwert (score) c umrechnet. Dabei muss c zwingend aus $\mathbb{N}, \mathbb{Z}$ oder $\mathbb{R}$ stammen, da dieser Wert als Grundlage für Vergleiche mit anderen Individuen dient und Vergleiche für komplexe Zahlen nicht definiert sind.

3.4.4 Population

Mehrere Individuen, welche zur gleichen Zeit einen simulierten Lebensraum bevölkern, werden zu einer Population P zusammengefasst. Die Population stellt eine logische Einheit zur besseren Handhabung und zum Vergleich mehrerer Individuen dar.

Die Anzahl der in der Population vorhandenen Individuen wird als Populationsgröße s bezeichnet und ist frei wählbar. In vielen Anwendungen aus dem Bereich der GA wird mit zeitinvarianten Populationsgrößen gearbeitet. Dies ist jedoch keine conditio sine qua non, sondern spiegelt lediglich den Mehraufwand an Arbeit wider, den zeitvariante Implementationen unweigerlich bedeuten.

Die Grundform eines Genetischen Algorithmus arbeitet mit genau einer Population von Individuen. Diese wird durch die den Genetischen Algorithmus bestimmenden Operatoren über Generationen hinweg als zusammenfassende Einheit für alle Individuen behandelt.

Es ist aber auch möglich, mehrere Populationen zu simulieren. Dies wird vor allem im Zusammenhang mit verschiedenen Migrations- und Kommunikationsmodellen benutzt, um den Einfluss von räumlicher Separation auf genetische Entwicklung zu untersuchen.

3.4.5 Skalierung

Wie in Abschnitt 3.4.3 gezeigt, definiert jedes Genom G über den daraus zu erstellenden Variablensatz V unter Zuhilfenahme der Objective Score Funktion $f_s(V)$ die grundlegenden Eigenschaften eines Individuums I durch die Reduzierung bzw. Verrechnung aller Eigenschaften auf einen Wert c, der Objective Score oder auch Zielwert genannt wird.

Jedes Individuum I ist zu jeder Zeit in eine Population P der Größe s eingebunden. Wie noch in Abschnitt 3.4.9 näher erläutert wird, braucht der Genetische Algorithmus jedoch im Endeffekt Fortpflanzungswahrscheinlichkeiten. Diese können ohne weiteres sofort aus den Zielwerten berechnet werden, allerdings stellen sich dabei verschiedene Hürden in den Weg:

- ein Teil oder auch alle Zielwerte können negativ sein.
- die Zielwerte können – vor allem bei exponentiellen Zielfunktionen – so ungünstig verteilt sein, dass ein oder wenige Individuen überragend gut sind und allen anderen Individuen keine Fortpflanzungschance lassen.

Um den daraus resultierenden negativen Effekten – wie z.B. verfrühte Konvergenz[8] und Takeover[9] – entgegenzuwirken, können die Zielwerte aller Individuen einer Population durch verschiedene Funktionen skaliert werden. Je nach Skalierungsfunktion können somit einzelne Individuen oder Gruppen von Individuen proportional zur Population in ihrer 'Fortpflanzungswahrscheinlichkeit' verbessert oder verschlechtert werden.

3.4.6 Generation und Generationsformen

Der Lauf eines Genetischen Algorithmus wird in Generationen gemessen. In jeder Generation wird aus der jeweilig aktuellen Population die nachfolgende Generation von Individuen berechnet. Dabei können die neuen Individuen die alte Population ganz ersetzen (*Simple GA* Ansatz von Goldberg) oder auch nur teilweise (*Steady State* Ansatz von de Jong). Beide Formen haben Vorbilder aus der Natur:

- Die vollständige Ersetzung aller Individuen kann bei einigen niederen Lebewesen beobachtet werden. Diese schlüpfen im Frühling, paaren sich und legen ihre Eier im Sommer und Herbst ab und sterben im Winter. Die Eier selbst überwintern und bilden die Nachfolgegeneration.
- Die teilweise Ersetzung mit überlappenden Populationen dagegen ist bei vielen höheren Lebewesen der Fall, wo die Lebensspanne der Eltern noch in die der Kinder hineinragt wie z.B. bei allen Säugetieren und bei den meisten Fischarten.

[8] siehe Abschnitt 3.4.11
[9] siehe Abschnitt 4.1.2

Die Generationen werden üblicherweise dann hochgezählt, wenn die neuen Individuen die Vorgängerpopulation ganz oder teilweise abgelöst haben und die Population zu einer neuen Elternselektion bereit steht.

3.4.7 Kreuzungsoperator

Der Crossover ist ein Genrekombinationsoperator, welcher die Entstehung von Individuen mit neuen Merkmalen ermöglicht.

In der Biologie ist Crossover bei allen höheren Lebewesen und bei den meisten niederen die Regel. Ein Crossover geht immer einher mit der sexuellen Reproduktion von Lebewesen und ermöglicht eine Durchbrechung der Genkopplung bei der Vererbung von Merkmalen bei höheren Lebewesen. Auf molekularer Ebene homologe[10] Chromosomen legen sich dabei aneinander und an bestimmten Stellen erfolgt ein Bruch der Chromatiden und eine kreuzweise Wiedervereinigung dieser Teilstücke. Durch eine solche Überkreuzung werden die auf abgetrennten Teilstücken liegenden Gene aus ihrer bisherigen Kopplungsgruppe gelöst und gegeneinander ausgetauscht. Dadurch werden die genetischen Informationen der Eltern gemischt, um den entstehenden Kindern Merkmalen von beiden Eltern zu vererben.

Dieses Verfahren ermöglicht die Durchbrechung der Genkopplung auch bei haploiden Chromosomensätzen und somit eine Erweiterung des 'Suchspektrums' für solche Lebewesen (z.B. Bakterien). Die Suche nach einer besseren, d.h. überlebens- oder reproduktionsfähigeren Lösung – als Lebewesen – wird gegenüber einfacher Mutation beschleunigt, da bei zunehmender Anzahl an Genen die Wahrscheinlichkeit, dass ein Kind einem der beiden Elternteile vollständig gleicht, abnimmt.

Genetische Algorithmen implementieren den Crossover-Operator in vielen verschiedenen Variationen und Spielarten, die sich alle mehr oder weniger auf das Beispiel aus der Biologie zurückführen lassen. Im einfachsten Fall werden zwei Genome an einer jeweils homologen Stelle gebrochen und die entstehenden Teilstücke ausgetauscht (1–Point Crossover). Weitere Möglichkeiten bestehen zum Beispiel aus mehreren solcher Bruchstellen (N–Point Crossover), vordefinierten Bruchstellen oder Austausch einzelner Basen. Die Möglichkeiten sind quasi unbegrenzt, und dieser Operator wird typischerweise sehr oft angewandt.

[10] gleiche

Der Effekt des Crossover-Operators lässt so im Laufe der Zeit typischerweise nach, je mehr die Population zu einer Lösung hinkonvergiert. Dadurch steigt die Wahrscheinlichkeit, dass zwei gleiche oder fast gleiche gekreuzte Genome mindestens ein Kindgenom erzeugen, welches mit einem der beiden Elternteile identisch ist.

Aus einer mathematischen Betrachtungsweise heraus dient dieser Operator im Rahmen der Exploitation von Informationen zweierlei Zwecken [24]. Der erste ist ein ständig fortlaufender Korrelationstest zwischen einer Anzahl an Allelen im Genom eines Individuums und dessen Fitness. Wenn während einer Genrekombination eine Gruppe von Genen vererbt wird, dann testet der Operator implizit die Hypothese, dass die Allele dieser Gene (oder auch eine Teilgruppe davon) mit einem guten Ziel- und damit auch Fitnesswert korrelieren[11]. Der zweite Zweck der Genrekombination ist es, gefundene Gruppen von Genen mit "guten" Werten, d.h. Gruppen, die wahrscheinlich die Fitness eines jeden Genoms erhöhen würden, miteinander zu rekombinieren. Hier wird von der Annahme ausgegangen, dass viele kleine dieser Building-Blocks zusammen eine weitere Verbesserung der Fitness eines Genoms bewirken[12].

3.4.8 Mutation

Als Mutation versteht man die Veränderung des Genoms an einer oder mehreren Stellen, welche durch innere oder äußere Einflüsse verursacht wird und eine bleibende Veränderung zurücklässt. Innerer Faktor für eine Mutation kann z.B. ein Kopierfehler bei der Vervielfältigung der DNA darstellen, äußere Faktoren können toxische Stoffe oder Strahlung sein, welche direkt das Erbgut verändern.

Die Simulation von Mutationen beschränkt sich im Allgemeinen auf die Mutation von Fortpflanzungszellen. Es gibt generell zwei Arten von Mutationen: die Basen- oder auch Punktmutation und die Phänotypmutation. Bei der Basenmutation wird das Kindgenom nach seiner Generierung – z.B. durch Crossover – einer Mutationsphase unterzogen, in der jede Base des Genoms eine bestimmte Mutationswahrscheinlichkeit hat. Dies ist die "natürliche" Mutation. Die Phänotypmutation arbeitet etwas anders und wird vorzugsweise bei Evolutionsstrategien verwendet: ein Gen wird als Phänotyp

[11] siehe auch Schema-Theorem in Abschnitt 3.6.
[12] siehe dazu die Building-Block-Hypothese in Abschnitt 3.7.

dekodiert und dann über eine Wahrscheinlichkeitsfunktion – typischerweise eine Gauss'sche Glockenkurve – mutiert. Dabei sind kleine Änderungen sehr, größere Änderungen wegen des zu beiden Seiten abnehmenden Charakters einer Glockenkurve weniger wahrscheinlich. Nach der Mutation wird der Phänotyp als Gen kodiert in das Genom zurückgeschrieben.

Mutation gibt jedem Punkt im Suchraum[13] eine Wahrscheinlichkeit die größer Null ist, evaluiert zu werden. Wie Crossover hat auch die Mutation sowohl destruktive als auch konstruktive Effekte auf eine Population. Die Mutationswahrscheinlichkeit sollte allerdings nicht zu hoch gesetzt werden, da sonst der Genetische Algorithmus in eine Zufallssuche abgleitet [31]. Goldberg und Segrest (1987) zeigten, dass höhere Mutationsraten in einem GA zu längeren Konvergenzzeiten führen und die Takeover-Zeit bei steigender Mutationsrate von einer zur Populationsgröße linearen Funktion zu einer exponentiellen wächst [39].

Das Problem der Mutationsrate lässt sich somit auf die Frage reduzieren, wie das Gleichgewicht zwischen der Zerstörung nützlicher Information und der Nutzung des Mutationsoperators als Suchoperator hergestellt werden soll.

3.4.9 Selektionsschemata

Die Selektion von Individuen zur Erzeugung von Nachkommen stellt einen Teil des darwinistischen Aspekts von Genetischen Algorithmen dar. Ein Selektionsschema definiert die Art und Weise, mit der die Eltern der Nachfolgegeneration bestimmt werden. Ausgehend vom errechneten Zielwert c jedes einzelnen Individuums und dessen Relation zur Gesamtpopulation durch die Skalierung ergibt sich für jedes Individuum eine bestimmte Fortpflanzungswahrscheinlichkeit. Diese bestimmt im Allgemeinen die ungefähre Anzahl der produzierten Nachkommen und somit die Weitergabe des jeweiligen Genoms.

Ist die Selektion von Eltern einzig und allein von deren Fitness abhängig und werden nur die besten Individuen für die Fortpflanzung in Betracht gezogen, so wird dies als *harte* Selektion bezeichnet. Im Gegensatz dazu steht die *weiche* Selektion, welche auch weniger fitten Individuen eine Reproduktionschance gibt [27].

[13] siehe Abschnitt 3.4.10 Suchraum: Exploration und Exploitation.

Mit der Selektion verbunden ist der Begriff *selection noise*. Dieser beschreibt den nichtdeterministischen Anteil des Selektionsoperators, welcher den Verlust von ansonsten fitten Individuen für den Reproduktionsprozess nach sich zieht. Ein gutes Maß für dieses Rauschen ist die Varianz der Anzahl von kreierten Instanzen einer Lösung i nach der Anwendung des Selektionsoperators; wobei i irgendein Punkt im Suchraum darstellt, der vor der Selektion in der Population vorhanden ist.

3.4.10 Suchraum: Exploration und Exploitation

Der *Suchraum* eines Problems wird durch dessen Variablensatz V bestimmt. Die darin enthaltenen n Variablen spannen den n-dimensionalen Suchraum auf. Wird jedem Punkt des Suchraums durch die Zielfunktion ein Ergebniswert zugeordnet, so kann der Suchraum als hyperdimensionale Ergebnislandschaft mit $n+1$ Dimensionen angesehen werden [27]. Eine Ergebnislandschaft mit mehreren Optima (Minima oder Maxima) wird als *multimodal* bezeichnet.

Als *Exploitation*[14] wird der Prozess bezeichnet, bei dem Information von früher besuchten Punkten im Suchraum für die Bestimmung der nächsten zu untersuchenden Punkte benutzt wird. Als Beispiel können einfache Hillclimber dienen, welche über Gradienteninformation oder über die Auswertung einiger benachbarter Punkte die Richtung ihres nächsten Schrittes bestimmen. Ein solches Verfahren eignet sich also gut, um ein lokales Optimum zu bestimmen.

Im Gegensatz dazu steht die *Exploration*. Hier werden Sprünge ins "kalte Wasser" unbekannter Regionen gewagt, zu denen keine Information über die Güte zur Verfügung steht. Aufgabenstellungen, deren Lösung viele lokale (Sub-) Optima enthält, können oftmals nur unter Zuhilfenahme von Exploration gelöst werden [27].

Entscheidend für den Erfolg eines Genetischen Algorithmus mit multimodaler Problemstellung wird also die richtige Mischung aus Exploration und Exploitation sein. Dazu dienen die beiden Operatoren Crossover – mit explorativem und exploitativem Charakter – und Mutation, mit rein explorativem Anteil.

[14] Verwertung, Ausnutzung

3.4.11 Konvergenz

Konvergenz ist der Prozess der Angleichung von Allelen in den Genen der Chromosomen einer Population. Haben z.B. 90% der Population in demselben Gen dasselbe Allel, so ist dieses Gen zu 90% konvergiert. Der Begriff ist auch auf ein komplettes Chromosom übertragbar.

Einfache Genetische Algorithmen lassen ihre Populationen relativ schnell zu einer Lösung hinkonvergieren, ganz im Gegensatz zu natürlichen Evolutionsprozessen, welche aufgrund von großen Populationen, räumlicher Trennung und Nischenphänomenen eine Vielzahl an Arten hervorbringen und auch halten können.

Einen starken Kritikpunkt an der üblicherweise schnellen Konvergenz von GAen stellt das Argument dar, dass die Evaluierung großer Mengen sich überlappender Schemata mehr Tests und eine langsamere, kontrollierte Konvergenz benötigt. Obwohl größere Populationen die Evaluationsrate steigern, müssen noch Methoden für eine genaue Kontrolle der Konvergenz erarbeitet werden. Deshalb sind Verfahren wie Niching und kontrollierte Konvergenz für Genetische Algorithmen in hochgradig multimodalen Suchräumen Gegenstand vertiefter Untersuchungen [39].

3.5 Evolutionsstrategien vs. Genetische Algorithmen

Genetische Algorithmen stellen nicht den einzigen Ansatz für die Simulation der Evolution in digitaler Form dar. In den sechziger und siebziger Jahren entwickelte Ingo Rechenberg zusammen mit Hans-Paul Schwefel an der TU Berlin seine Evolutionsstrategie (ES) und stellte sie 1973 in [45] vor. Evolutionsstrategien wurden von ihnen entwickelt, um technische Optimierungsaufgaben zu lösen und die ersten Ansätze ähnelten noch sehr damals bekannten iterativen, aber konventionellen Lösungsansätzen [27]. Der technische Aspekt der Evolutionsstrategien spiegelt sich deutlich in einem Gauss'schen Mutationsoperator wider, der auf phänotypischer Basis funktioniert [45].

Der grundsätzliche Ansatz – nämlich die Natur als Vorbild für eine Optimierungsstrategie zu nehmen – ist sowohl bei GA als auch bei ES gleich, jedoch stützen sich amerikanische Forscher um Holland und Goldberg mehr auf informations- und kodierungstechnische Aspekte der Evolution, während Rechenberg die Natur mehr als Leitfaden ansieht. Als direkte Folge da-

von wird die Funktionsweise von Genetischen Algorithmen durch Hollands Schema-Theorem und Goldbergs Building-Block-Hypothese erklärt, wobei der Mutation die Rolle eines "background" Operators geringer Wichtigkeit mit nur disruptiven Effekten zugewiesen wird [31, 19]. Im Gegensatz dazu arbeiteten die ursprünglichen Evolutionsstrategien ausschließlich mit dem Mutationsoperator und der Selektion von Individuen[15], wobei die Funktionsweise mit einem Korridormodell der Evolution erklärt wurde.

Beide Modellierungsansätze wurden in den letzten 20 Jahren weiterentwickelt, unabhängig voneinander und oftmals unter Nichtbeachtung der Ergebnisse der jeweils anderen Schule. Dies hat sich jedoch in letzter Zeit geändert, und man bemerkt in neueren Aufsätzen ein langsames Verschmelzen beider Modelle.

3.6 Das Schema-Theorem

Ein Schema ist ein Muster von Allelen eines Genoms, welches als Alphabet die möglichen Allele der jeweiligen Gene mit einem zusätzlichen Joker-Zeichen hat. Dieses Joker-Zeichen – meist ein # – steht für die Bedeutung "don't care". Ein Schema enthält ein bestimmtes Genom, wenn dieses dem vorgegebenen Muster entspricht.

Das Schema eines binären Genoms wird somit mit dem trinären Alphabet {0, 1, #} aufgebaut und könnte z.B. #1##00# sein. In diesem Schema ist z.B. das Genom 0101001 enthalten, aber auch 1101001. Ein Schema für ein menschliches Genom beinhaltet die Zeichen {A, T, G, C, #}, und das Schema A#G enthält die vier möglichen Genome {AAG, ATG, AGG, ACG}.

In einem Genom mit der Bitlänge l ist ein Schema mit einer Hyperebene in einem l-dimensionalen, bit-transformierten d.h. diskretisierten Suchraum gleichzusetzen.

Drei Maßzahlen sind für die theoretische Beschreibung eines Schemas von Bedeutung:

- die *Ordnung* eines Schemas ist die Anzahl der Zeichen, welche keine "don't cares" (#) sind.
- die *definierende Länge* eines Schemas entspricht der maximalen Distanz zwischen zwei Zeichen im Schema, welche keine "don't cares"

[15] sogenannte "naive" Evolution, siehe Abschnitt 3.8

sind.

– die *Fitness* eines Schemas wird durch den Mittelwert aller darin enthaltenen Genome beschrieben [19].

In den oben genannten Beispielen hat #1##00# die Ordnung 3 und die definierende Länge 4, A#G hat die Ordnung 2 und die definierende Länge 2. Die Fitness dieser Schemata lässt sich allerdings ohne dazugehörige Zielfunktion nicht bestimmen.

Das von Holland erstmals in seinem Buch [31] vorgestellte Schema-Theorem stellt einen Erklärungsansatz für das Verhalten von GAen dar. Die Kurzform besagt, dass Genetische Algorithmen überdurchschnittlichen Schemata exponentiell steigende Chancen zur Reproduktion geben. Da jedes Genom eine Vielzahl an Schemata enthält, ist die Schemaevaluationsrate sehr hoch. Dieses Phänomen nannte Holland "impliziten Parallelismus" und beschreibt die Steigerung der Bewertungsgeschwindigkeit durch kumulative Lerneffekte bei der sukzessiven Bewertung von Hyperebenen über mehrere Generationen hinweg [60]. Die Steigerungsrate wurde von ihm bei einer Population mit N Individuen mit N^2 angegeben [27]. Das bedeutet, dass die Schemaevaluationsrate einer Population mit N Individuen pro Generation N^3 beträgt.

3.7 Die Building-Block-Hypothese

Ein Building-Block (BB) ist im Bereich des Evolutionary Computing – und dort vor allem bei den Genetischen Algorithmen und bei der Genetischen Programmierung – eine eng gegliederte Gruppe von Genen, deren Allele, wenn sie in ein beliebiges Chromosom eingesetzt werden, die Fitness dieses Individuums mit großer Wahrscheinlichkeit verbessern würden.

Die zuerst von Goldberg in [19] vorgestellte Building-Block-Hypothese erklärt den Erfolg von Genetischen Algorithmen dadurch, dass diese in frühen Laufzeitstadien so viele Building-Blocks wie möglich finden und diese später, durch Crossover kombiniert, eine möglichst optimale Fitness ergeben [27]. Somit stellen sie überproportional fitte (Teil-)Schemata kleiner Ordnung und kurzer definierender Länge dar. Building-Blocks kurzer Länge werden als eng gekoppelt bezeichnet.

Als eine direkte Folge dieser Hypothese steht die Feststellung dass, je größer die Anfangspopulation eines GA, desto wahrscheinlicher das Vorhandensein einiger guter Fundamente für den raschen Aufbau von Building-

Blocks in Nachfolgegenerationen ist. Doch setzen in vielen Fällen Zeit- und Speicherplatzmangel enge Grenzen für die maximal mögliche Populationsgröße, weshalb ein wichtiger Forschungsschwerpunkt die Möglichkeit zum Aufbau guter Building-Blocks auch bei kleinen Populationen darstellt.

Eng verbunden mit der Building-Block-Hypothese ist der Begriff *Deception*[16]. Dieser beschreibt den Effekt, dass eine Rekombination guter Building-Blocks zu einer Verschlechterung der Fitness von Individuen führt anstatt zu einer Verbesserung. Dieses Verhalten von GAen bei manchen Kodierungen und Problemen wurde von Goldberg als einer der Gründe für deren Versagen bei manchen Aufgabenstellungen angesehen.

Eine erklärende Theorie, wann und wie welche im Lauf eines Genetischen Algorithmus auftauchenden und überlebenden Building-Blocks zusammengefügt eine größere Chance für bessere Fitness repräsentieren, ist zur Zeit noch nicht vorgestellt worden [20].

3.8 Kritik an dem Schema-Theorem und der Building-Block-Hypothese

Holland hatte in [31] einige vereinfachende Annahmen getroffen, darunter auch

1. unendliche Populationsgröße
2. kaum Interaktion (Epistasis[17]) zwischen einzelnen Genen

Die erste Annahme kann aus offensichtlichen Gründen niemals erfüllt werden, weshalb es bei GAen immer zu stochastischen "Fehlern" kommt. Als Ergebnis davon steht das aus der Natur bekannte Phänomen der genetischen Drift, welches in "kleinen" Populationen zur Weitergabe und Verbreitung bestimmter, nicht unbedingt guter Genkombinationen führt, weil sie rein zufällig etwas häufiger vorhanden sind als andere.

Die zweite Annahme ist sowohl in der Biologie als auch bei praktischen Anwendungsproblemen nicht einzuhalten. Eine geringe Epistasis von Genen

[16] Täuschung, Irreführung

[17] In der Biologie wird Epistasis als das Überdecken der phänotypischen Manifestation eines Gens durch ein anderes bezeichnet. Im Bereich des Evolutionary Computing wird Epistasis in der Regel als starke Interaktion zwischen verschiedenen Genen – die nicht unbedingt im Phänotyp des Individuums zum Ausdruck kommen muss – bezeichnet. Sie entsteht dadurch, dass Änderungen eines Gens die Wirkung eines anderen beeinflussen [27].

mag mit speziell entwickelten Testfunktionen für Genetische Algorithmen zu realisieren sein, wird jedoch ansonsten bei praktischen Problemen nie zu garantieren sein [9].

Aufbauend auf Hollands Schema-Theorem erzwingt die Building-Block-Hypothese von Goldberg noch eine dritte Annahme:

3. Interagierende Gene müssen eng zusammen im Genom liegen, um möglichst kleine Building-Blocks zu bilden.

Diese Annahme ist ebenfalls kaum einzuhalten, da eine nichtzufällige Ordnung von Genen in festgelegte Blöcke von vornherein ein tiefergehendes Verständnis des Problems und die "Erahnung" der richtigen Lösung voraussetzt. Beides sind Voraussetzungen die – wenn sie denn erfüllt werden könnten – in vielen Fällen die Programmierung problemspezifischer Such- und Optimierungsalgorithmen erlauben würden.

Das Schema-Theorem an sich wird von Lee Altenberg in [1] als Spezialfall von "Price's Theorem" über Kovarianz und Selektion aufgezeigt, welches seiner Meinung nach jedoch nicht ausreichend ist, um die Verhaltens- und Funktionsweise von GAen vollständig zu erklären. Altenberg schlägt deshalb vor, Statistiken der Übertragungsfunktion bei verschiedenen Operatoren und Kodierungen als Erklärungshilfe heranzuziehen.

Naive Evolution

Eine Beobachtung ganz anderer Art bringt sowohl Schema-Theorem als auch Building-Block-Hypothese in Erklärungsnot: naive Evolution. Diese nur aus Selektion und Mutation bestehende Evolution war die ursprüngliche Arbeitsweise von Evolutionsstrategien und wurde zwischenzeitlich in mehreren Untersuchungen auch für Genetische Algorithmen als äußerst leistungsfähig eingestuft. So nennen Beasley, Bull und Martin in [10] Ergebnisse von Schaffer et al, die 1989 einen "weit weniger wichtigen Einfluss von Crossover" feststellten als bislang vermutet wurde. In einer weiteren Untersuchung 1991 erkannten Schaffer et al. den Crossoveroperator als "nur" geschwindigkeitssteigernd an.

3.9 Genetische Algorithmen im Vergleich mit anderen Suchverfahren

3.9.1 Vollständige Enumeration

Die vollständige Enumeration des Suchraumes durch geordnetes Evaluieren aller Punkte im Suchraum – z.B. sequentiell oder mit Intervallen – stellt die einfachste und klassischste, rein explorative *brute force* Lösung für Such- und Optimierungsprobleme dar. Durch die Evaluierung aller Punkte besteht die Garantie, dass alle globalen Optima gefunden werden, jedoch in einer zur Größe des Suchraums N proportionalen Zeit, wobei N sich zur Variablenanzahl i exponentiell verhält. Kann in einigen seltenen Fällen mit Sicherheit festgestellt werden, dass ein evaluierter Punkt ein globales Optimum darstellt, so wird der Algorithmus im Durchschnitt ein globales Optimum in einer zu $\frac{N}{2}$ proportionalen Zeit finden.

Vor allem die exponentielle Beziehung der Variablenanzahl zur Suchraumgröße macht dieses Verfahren für größere i prohibitiv.

3.9.2 Zufallssuche

Diese auch Monte-Carlo-Simulation genannte Methode gehört ebenfalls zu den rein explorativen brute force Verfahren. Die zu evaluierenden Punkte im Suchraum werden hierbei per Zufall bestimmt. Würde dem Algorithmus ein Gedächtnis eingebaut, um den Besuch schon evaluierter Punkte zu vermeiden, so wären die Zeiten für ein vollständiges Evaluieren des Suchraums mit denen der Enumeration vergleichbar, wobei jedoch zusätzlich ein zu N proportionaler Verwaltungsoverhead für das Gedächtnis hinzukäme.

Da ein Gedächtnis aber normalerweise schon wegen der Größe der Suchräume nicht machbar ist, wird durch die im Laufe der Zeit erfolgende mehrfache Evaluierung von Punkten im Suchraum die Effizienz bzw. die Suchgeschwindigkeit der Zufallssuche erheblich reduziert. Zudem existiert keine Garantie, dass ohne Gedächtnis alle Punkte des Raumes evaluiert werden, und somit auch keine Garantie, dass das globale Optimum oder die globalen Optima gefunden werden. Dieses Verfahren arbeitet also immer schlechter als eine vollständige Enumeration.

Diese Ineffizienz führt dazu, dass dieses Verfahren kaum je für sich alleine implementiert wird, sondern meist in Kombination mit anderen Strategien Verwendung findet.

Solange die Anzahl an guten Lösungen eines Problems in Bezug auf die Größe des Suchraumes klein ist und diese Lösungen zusätzlich weit verteilt sind, werden Enumeration und Zufallssuche mit Sicherheit nicht die besten Methoden darstellen.

3.9.3 Gradientensuche

Eine Anzahl verschiedener Methoden wurde entwickelt, um für einfache, unimodale Suchräume die Information über die Steigung eines Punktes auszuwerten. Durch die Berechnung von Funktionsableitungen kann der Fokus der Suche in die Richtung der stärksten Steigung im aktuellen Punkt bewegt werden, weshalb diese Gruppe von Verfahren auch *Hillclimber* oder *-crawler* genannt wird. Sie stellen in der Grundform ein rein exploitatives Verfahren dar, da keinerlei Zufallselemente in der Suche enthalten sind und der nächste Zug nur von der zur Verfügung stehenden Information abhängt.

Wenn die Ableitung aus mathematischen oder anderen Gründen nicht berechnet werden kann, so bleibt als Annäherung an diese Methode jeweils nur die Möglichkeit, durch die Evaluierung einiger ausgesuchter Punkte in der Nachbarschaft den jeweils besten als Suchrichtung zu benutzen. Auch in diesem Fall handelt es sich immer noch um reine Exploitation von zur Verfügung stehender Information.

Der Nachteil dieser Verfahren liegt gleichzeitig in ihrer Effizienz, nahegelegene Optima mehr oder weniger direkt zu erreichen. Bei multimodalen Suchräumen zeichnen sich diese Methoden durch ihr Unvermögen aus, einmal gefundene und unter Umständen nur lokale Optima zu verlassen und weiterzusuchen. Sie bleiben darin stecken.

Aus diesem Grund werden Gradientensucher sehr oft mit abgeschwächten Versionen explorativer brute force Methoden kombiniert. Zum Beispiel kann ein konvergierter Hillclimber per Zufall ein neues Suchgebiet zugewiesen bekommen, um dort nach einem anderen Optimum zu suchen. Jedoch werden bei diesen iterativen Läufen keinerlei Daten weitergereicht. Ein kumulativer Lerneffekt kann so nicht eintreten.

Die Einfachheit dieses Systems macht es jedoch zu einem beliebten Verfahren, um Suchräume stichprobenartig anzutesten.

3.9.4 Simulated Annealing

Dieses Verfahren ist ebenfalls an ein natürliches Phänomen angelehnt: der Abkühlung eines erhitzten Stoffes. Von einem zufällig gewählten Punkt im Suchraum aus wird ein mehr oder weniger zufällig gewählter nächster Punkt evaluiert. Der Suchfokus wird unter 2 Bedingungen auf diesen neuen Punkt gerichtet:

- immer wenn der neue Punkt besser ist als der alte oder
- mit der Wahrscheinlichkeit $p(t)$, wenn der neue Punkt schlechter ist als der alte.

Die Funktion $p(t)$ ist hierbei eine von der Zeit t abhängige Funktion, welche für t=0 nahe 1 beginnt und für wachsende t gegen 0 strebt. Diese monoton fallende Funktion simuliert somit den Abkühlungsprozess, bei dem im heissen Zustand Atome große Sprünge oder Drehungen vollziehen und bei der Abkühlung diese Sprünge und Drehungen nur noch erfolgen, wenn dadurch die Energie eines Systems verringert wird.

Simulated Annealing stellt in gewissem Sinne einen modifizierten Hillclimber dar, dem eine Zufallsakzeptanz für schlechtere Punkte eingebaut ist, um aus lokalen Optima wieder herauszufinden. Die Methoden zur Überführung des anfangs rein explorativen Verfahrens in ein exploitatives und das richtige Mischungsverhältnis ist immer noch Gegenstand zahlreicher Untersuchungen [9].

Kapitel 4

Rahmen und Umfeld der Untersuchungen

"How much intelligence does one need to sneak upon lettuce?"

Anhand der Probleme Genetischer Algorithmen und der gestellten Aufgabe wird im ersten Teil die allgemeine Lösung zur Verknüpfung der beiden Themengebiete – Molekülstrukturoptimierung und Genetische Algorithmen – aufgezeigt. Anschließend werden die Methodik und Vorgehensweise der Untersuchungen vorgestellt und die Darstellungsformen im weiteren Verlauf dieser Diplomarbeit erläutert.

4.1 Probleme von Genetischen Algorithmen

Die Leistungsfähigkeit von Genetischen Algorithmen und deren Erfolg beim Einsatz für verschiedenste Aufgabenstellungen darf nicht dazu verleiten, GAen ohne Bedenken für jedes mögliche Problem einsetzen zu wollen. In der Tat gibt es Aufgaben, bei denen dieses Verfahren sehr schlecht funktioniert.

4.1.1 Problemverursacher

Goldberg et al. konnten in [20] vier wichtige Problemkreise identifizieren, welche für Genetische Algorithmen – wie allerdings für andere Such- und Optimierungsstrategien auch – schwierig zu lösen sind:

1. pikartige oder isolierte Optima,

2. Deception,

3. Noisiness,

4. und massive Multimodalität

in der Ergebnisfunktion wirken einzeln oder kombiniert massiv erschwerend.

Mit Blick auf das Schema-Theorem und die Building-Block-Hypothese impliziert dies nach einer Untersuchung von Harik in [24] folgende kodierungs- oder ablauftechnische Probleme, von denen mindestens eines erfüllt sein muss:

1. es gibt für diesen Genetischen Algorithmus in der vorliegenden Informations- bzw. Variablenkodierung keine Building-Blocks.

2. die Fortpflanzung von Building-Blocks, die eine optimale Lösung formen, muss erschwert sein.

3. die Rekombination gefundener kleiner Building-Blocks zu einer optimalen Lösung ist nicht trivial oder nicht existent.

Besonders das erste Problem kann bei unbedachter bzw. falscher Informationskodierung entstehen und sollte bei schwachen Leistungen des GA als erstes überprüft werden[1].

Eine Fortpflanzung von Building-Blocks wird in erster Linie durch Blöcke hoher Ordnung und großer definierender Länge erschwert. Es ist nachvollziehbar, dass je niedriger die Ordnung und die definierende Länge von Blökken ist, desto einfacher ein GA diese finden und rekombinieren können wird.

Als Schwierigkeitsgrad eines Problems kann deshalb die Ordnung des größten Building-Blocks mit "guten" Eigenschaften gelten, der nicht durch Rekombination kleinerer Blöcke erzeugt werden kann. Je größer diese Ordnung ist, desto schwieriger ist es für den GA, diesen zu finden, wenn ein solcher Block nicht per Zufall oder dank einer "ausreichend großen" Anfangspopulation vorhanden ist. Goldberg & Bridges zeigten 1990, dass GAen bei Problemen mit Building-Blocks großer definierender Länge sehr häufig zu lokalen Optima hinkonvergieren anstatt zu einem globalen Optimum [24]. Eine Umordnung der Genrepräsentation kann Besserung in der Leistung schaffen und wenn dies nicht möglich ist, so kann die Wahl eines anderen Crossover-Operators helfen.

[1] siehe dazu auch Abschnitt 4.4: Kodierung der Moleküle.

Das dritte der oben genannten Probleme stammt aus dem Bereich der Deception und ist bisher nicht gelöst worden. Die Tatsache, dass Rekombination kleiner, potentiell guter Building-Blocks zusammen eine schlechtere Fitness ergeben, zeigt deutlich die theoretischen Grenzen von Schema-Theorem und Building-Block-Hypothese.

4.1.2 Richtige Parameterwahl

Abhängig von der Informationsrepräsentation im Genom – und den sich daraus ergebenden, oben beschriebenen Problemen – müssen die "richtigen" Parameter für einen Genetischen Algorithmus gewählt werden. Als direktes Ergebnis dieser Aussage gilt die Feststellung, dass es nicht *die* optimalen Parameterkombination gibt, sondern dass diese für jede spezielle Problemstellung, deren Kodierung im Genom und den benutzten Genetischen Algorithmen ermittelt werden muss. Stellvertretend für die gesamte Problematik soll dies hier kurz an zwei Beispielen aufgezeigt werden.

Crossover

Selbst wenn ein GA gute Building-Blocks findet, so kann ein ungünstiger Crossover-Operator eine Rekombination der Blöcke erschweren und in ungünstigen Fällen sogar ganz verhindern. Der am häufigsten genutzte 1-Point Crossover z.B. favorisiert kleine, eng gekoppelte Blöcke. Je länger ein Building-Block, desto wahrscheinlicher wird ein Kreuzungspunkt innerhalb dieses Blockes liegen. Eine Rekombination mit einem anderen, guten Teil eines Blockes ist in solchen Fällen sehr unwahrscheinlich.

Selektion

Die Wahl des Selektionsschemas ist eng verknüpft mit der zugrunde liegenden Problemstellung, deren Realisierung in der Zielfunktion und dem ausgewählten Skalierungsschema. Ungünstige Kombinationen dieser Faktoren resultieren entweder in einem zu niedrigen oder einem zu hohen Selektionsdruck innerhalb der Population. Im ersten Fall dauert es dadurch länger, bis ein Genetischer Algorithmus gegen eine Lösung konvergiert. Die Anzahl der Zielfunktionsevaluationen wird dadurch unnötig in die Höhe getrieben. Im zweiten Fall wird in der Regel zu Anfang des Laufes die Population durch einige wenige, relativ gute Individuen innerhalb kürzester Zeit "übernommen". Dieser Effekt wird auch als *Takeover* bezeichnet und hat eine zu

schnelle Konvergenz der genetischen Vielfalt innerhalb einer Population zur Folge. Dadurch kommt es in den allermeisten Fällen zu einem vorzeitigen Abbruch des Algorithmus, da nicht mehr genug Genkombinationen für die Suche zur Verfügung stehen[2].

Blickle und Thiele [12] haben 1995 zu diesem Thema einen Vergleich der gängigsten Selektionsschemata publiziert, in welchem sie theoretische und mathematische Modelle zu allen Schemata herleiten und durch Versuche ihre Ergebnisse untermauern. Bäck und Hofmeister [6] zeigten in einer Untersuchung über erweiterte Selektionsschemata in evolutionären Algorithmen, dass "sanfte" Selektionsschemata wie *uniform* oder *linear ranking*[3] bei einer multimodalen Funktion bessere Ergebnisse liefern als andere Selektionsschemata. Jedoch warnen sie davor, dass bei zu sanften Schemata unter Umständen eine Population die gefundenen Teillösungen nicht halten kann und diese somit für weitere Exploitation und als Startpunkt für umgebungsbezogene Exploration verloren gehen.

4.2 NP-Vollständigkeit

Die Energieminimierung von Molekülen ist deshalb eine nicht-triviale Aufgabenstellung, weil die Anzahl lokal stabiler Strukturen – lokale Energieminima in der Ergebnishyperebene – exponentiell mit der Anzahl der im Molekül vorhandenen Atome zunimmt [16]. J.T. Ngo und J. Marks haben 1992 in [43] für ein erweitertes Faltungsmodell gezeigt, dass eine Energieminimierung von Proteinmakromolekülen ein NP-vollständiges Problem ist.

4.3 Untersuchte Moleküle

Die Variationsbreite der möglichen Untersuchungsobjekte reicht von kleinen und kleinsten Peptidmolekülen mit nur wenigen Atomen bis hin zu Proteinen mit mehreren hundert Atomen. Die zur Atomzahl exponentiell benötigte Rechenzeit für die Bestimmung der Konformationsenergie engt jedoch für eine Breitenuntersuchung die Zahl der Kandidaten auf kleinere Repräsentanten ein.

[2] Oft genug kann es sogar passieren, dass innerhalb weniger Generationen die komplette Population von einem einzigen Individuum dominiert wird.

[3] siehe auch Abschnitt 5.8

4.3.1 Peptide

Für diese Arbeit wurden zwei kleine Peptide ausgesucht, Tetraalanin und Metenkephalin. Tetraalanin ist ein kleines Tripeptid (drei Alanine mit den "blocking groups" Acethyl und Methylamid – 42 Atome und 15 drehbare Bindungen) ohne Seitenketten und wurde sowohl von F. Herrmann [28] als auch während der Entwicklung der für dieser Arbeit benutzten GA-Bibliothek eingehend untersucht. In diesen Untersuchungen wurden keine besseren Energiestrukturen gefunden als das bisher bekannte Minimum, doch konnte dieses sehr oft reproduziert werden. Die sehr kleine Struktur und das Fehlen von Seitenketten machen Tetraalanin zu einem relativ einfach zu minimierenden Molekül.

Aus diesem Grund konzentriert sich diese Arbeit auf Metenkephalin, was von seiner Komplexität her – zum einen wegen seiner größeren Struktur, als auch wegen seiner Seitenketten – schwieriger zu lösen ist. Metenkephalin ist ein Pentapeptid (Tyrosin, Glycin, Glycin, Phenylalanin, Methionin – 77 Atome und 24 drehbare Bindungen) mit Seitenketten bei Tyrosin, Phenylalanin und Methionin. Im Zuge der noch andauernden Untersuchung von F. Herrmann konnte die minimale gefundene Struktur von einem Energieniveau von bisher bekannten 4,277345 $\frac{kcal}{mol}$ auf 1,752383 $\frac{kcal}{mol}$ gesenkt werden.

4.3.2 Energiefunktion

Genetische Algorithmen stützen sich von ihrer Funktionsweise her auf eine Vielzahl von Evaluationen. Dies macht eine schnelle Funktion zur Berechnung von Konformationsenergie zur Pflicht. Eine genaue Simulation atomarer und quantenmechanischer Kräfte fällt aus diesem Grunde aus.

Als Alternative wurde die Energiefunktion *Amber* gewählt. Amber berechnet die Konformationsenergie eines Moleküls unter Zuhilfenahme eines in mehreren Untersuchungen empirisch ermittelten Kraftfeldes [57, 58, 59]. Diese Funktion hat sich als genau genug erwiesen, um damit die Energie auch größerer Moleküle mit geringer Fehlertoleranz zu bestimmen. Die Energiefunktionen wurden durch ein Programmpaket von F. Herrmann erzeugt, das aus Referenzstrukturen der jeweils zu untersuchenden Moleküle C-Quellcode zur Berechnung der Amber-Energie erzeugt. Aufgrund der effizienten Implementierung können auf schnellen Rechnern pro Sekunde circa 900 Konformationen von Tetraalanin und circa 300 Konformationen von Metenkephalin evaluiert werden (PPro200).

Die C-Energiefunktion ist jeweils spezifisch für ein bestimmtes Molekül und besitzt als Argument den vom GA erzeugten Variablensatz zur Darstellung der Konformation[4].

4.4 Kodierung der Moleküle

Wie schon erwähnt, ist die Wahl einer geeigneten Kodierung des Variablensatzes für die jeweilige Problemstellung ein wichtiger Punkt für den erfolgreichen Einsatz von Genetischen Algorithmen. Im Falle der Darstellung von Molekülen lassen sich zwei verschiedene Ansätze ermitteln, deren Vor- und Nachteile im folgenden besprochen werden.

4.4.1 Kodierung durch absolute Koordinaten

Die Positionierung von Atomen in einem Molekül unter Zuhilfenahme eines kartesischen – oder transformierten – Koordinatensystems stellt einen anschaulichen und intuitiven Ansatz dar. Dieser bringt jedoch bei einer direkten Umsetzung für Genetische Algorithmen einige gravierende Probleme. Die Darstellung in absoluten Koordinaten führt dazu, dass die Anwendung des Crossover-Operators in vielen Fällen völlig ungültige Molekülkonfirmationen ergibt, weil Atome entweder zu weit oder zu nah für eine Bindung wären. Bei einer Kodierung der Position ohne Gray-Code würde dieser Einwand in großem Maße auch für Mutation gelten. Die für eine Aussortierung der ungültigen Konfirmationen notwendigen Filter sind zu zeitintensiv, um eine vernünftige Leistung der Algorithmen noch zu gewährleisten.

Weiterhin kann die Äquivalenz von Molekülen in dieser Darstellungsform nicht ohne zeitaufwendige und komplexe 3D-Transformationen und Translationen mit Ankerpunkten festgestellt werden. Als Beispiel könnte die zweidimensionale Darstellung eines O_2-Moleküls (Sauerstoff) sowohl mit dem Punktepaar $\left(\binom{1}{1}, \binom{2}{2}\right)$ als auch mit dem Paar $\left(\binom{4}{5}, \binom{5.1}{3.9}\right)$ angegeben werden: die Koordinaten des zweiten Paares sind eine um 90° gedrehte, zusätzlich verschobene und etwas gestreckte Version des ersten Koordinatenpaares.

4.4.2 Relative Kodierung

Bei der relativen Kodierung werden für jedes Atom die Winkel seiner Bindungen, die Bindungslängen und der Drehwinkel der Bindungen kodiert.

[4] siehe auch Abschnitt 4.4.2: Relative Kodierung (von Molekülen).

Unter der Annahme, dass sowohl die Bindungswinkel als auch die Bindungslängen fest sind, werden als Vereinfachung zu jeder Bindung eines Atoms nur die Drehwinkel kodiert. So entstehen nur in sehr wenigen Fällen illegale Konfirmationen durch zu nah positionierte Atome (clashing), welche jedoch in den Energiefunktionen durch sehr hohe Energien direkt aussortiert werden können. Die Äquivalenz zweier Moleküle kann in dieser Darstellung durch einfache standardisierte Rücktransformation in ein kartesisches Koordinatensystem und anschließender Root-Mean-Square-Vergleichsberechnung (RMS) von Atomkoordinaten zweier Moleküle bewerkstelligt werden.

Ein weiteres Problem besteht in der benötigten Genauigkeit der Drehwinkel. Untersuchungen von F. Herrmann [29] ergaben, dass eine Auflösung von 10 Bit pro 360°-Drehwinkel als unterste Grenze anzusehen ist. Bei 24 Drehwinkeln für Metenkephalin ergibt dies eine in dieser Arbeit benutzten Genomlänge von 240 Bit, was weit über der typischen Bitanzahl von 50 bis 100 Bit in theoretischen und praktischen Untersuchungen von Testfunktionen bei GAen liegt.

4.5 Programme für Genetische Algorithmen

4.5.1 Öffentliche Pakete

In den letzten Jahren ist eine Vielzahl von Programmbibliotheken erschienen, welche als Grundlagenbibliotheken für Genetische Algorithmen dienen. Die Vielfalt der benutzten Programmiersprachen reicht von Pascal über C und C++ zu Lisp, Fortran und Smalltalk. Eine erste Übersicht darüber liefert [27]. Für eine erste Voruntersuchung wurde das Programmpaket GAlib in den Versionen 2.3.2 bzw. 2.4.x vom Massachusetts Institute of Technology (Matthew Wall) als Grundlage gewählt.

Im Zuge dieser ersten Untersuchungen wurden jedoch zwei Fehler festgestellt, welche die Arbeit beeinträchtigten[5]. Bei der Portierung und anschließenden Benutzung des Pakets auf einem Parsytec Hochleistungsrechner stellten sich zusätzliche, auf anderen Plattformen nicht reproduzierbare Abstürze ein[6]. Deshalb wurde für diese Arbeit eine eigene Bibliothek für Ge-

[5]Es gab falsche Werte bei Minimierungsfunktionen und einen Fehler in einer Sortierroutine, welcher große Populationen mit exponentiellen Laufzeiten belegte. Letzteres soll in der Version 2.4.2 laut Autor beseitigt worden sein.

[6]Diese stellten sich als nicht durch das Paket selbst verursacht heraus, siehe nächster Abschnitt.

netische Algorithmen entwickelt und die Berechnungen damit durchgeführt.

4.5.2 DiplGA Bibliothek

Die Entwicklung der Bibliothek erfolgte unter Linux mit dem gcc 2.7.1 und 2.7.2 Compiler. Das DiplGA genannte Paket ist in C++ geschrieben und objektorientiert aufgebaut. Der Compiler hat allerdings noch einige Probleme mit Templates, weshalb auf diese Funktionalität leider verzichtet werden musste.

Die GA-Bibliothek erlaubt die Einbindung der zu optimierenden Zielfunktion in Form einer C/C++–Funktion (oder sonstigen Sprache, welche Module in einem vom GNU-Linker unterstütztem Format schreiben).

Bei der Portierung des DiplGA Paketes auf den Parsytec Rechner stellten sich symptomatisch dieselben unerklärlichen Abstürze ein wie bei dem GAlib Paket. Nach eingehender Untersuchung konnte die Ursache gefunden und beide Pakete hinsichtlich dieser Abstürze schuldlos gesprochen werden: Eine offensichtlich fehlerhafte Implementation der dortigen GNU C++–Ausgaberoutine *cout* verursachte reproduzierbare Abstürze. Bei alternativer Benutzung der C-Funktion *printf* traten diese in der DiplGA Bibliothek nicht mehr auf. Eine testweise Portierung und stichprobenartige Überprüfung einiger Rechenergebnisse auf alle anderen verfügbaren Plattformen zeigte außer notwendigen Anpassungen für den Pfad der *include*-Dateien keinerlei Probleme.

In die Bibliothek wurden alle aus der Literatur bekannten und für diese Untersuchung wichtigen Funktionalitäten eingebaut. Zusätzlich wurden speziell für die Problemstellung der Molekülstrukturoptimierung einige neue, experimentelle Funktionen – wie unterschiedliche Elternanzahl, Double prevention, adaptive Mutation und pyramidale GAen – entwickelt, implementiert, getestet und untersucht. Eine detaillierte Liste aller untersuchten Funktionalitäten, ihre genaue Funktionsweise und die erbrachten Ergebnisse finden sich in den Kapiteln 5 und 6.

4.6 Benutzte Plattformen

Zur Verfügung standen diverse Einzelrechner an der Universität Heidelberg (IBM RS/6000 unter AIX 3.2.5 und SUN sparcstation 20 unter Solaris 5.5), der Fachhochschule Heilbronn (SUN sparcstation 20 unter Solaris 5.5.1),

dem Deutschen Krebsforschungszentrum (SGI Silicon Graphics unter IRIX 5.3) sowie private Arbeitsrechner (Pentium und PentiumPro PCs unter Linux 1.2.x und 2.0.x). Durch einen Kooperationsvertrag des DKFZs mit dem Interdisziplinären Zentrum für Wissenschaftliches Rechnen (IWR) der Universität Heidelberg war zudem der Zugriff auf einen Parsytec Hochleistungsparallelrechner mit 192 Prozessoren möglich.

Nach einer ersten Laufzeitevaluierung wurde beschlossen, die Entwicklung unter Linux auf dem privaten PC und die Rechnungen für die Untersuchungen auf der Parsytec vorzunehmen. Die Benutzung weiterer Einzelrechner der Universität Heidelberg und der Fachhochschule Heilbronn für Berechnungen hätte nur einen geringen Geschwindigkeitsvorteil bei gleichzeitiger Beeinträchtigung des Benutzerbetriebs bedeutet.

4.7 Vorgehensweise

In ersten Voruntersuchungen und diversen Testläufen während der Erstellung der Programmbibliothek konnten einige Erfahrungswerte bezüglich des Laufzeitverhaltens gesammelt werden. Gleichzeitig wurden benötigte Parameter spezifiziert und implementiert. Es wurde aufgrund der großen Anzahl an Parametrierungsmöglichkeiten beschlossen, die Untersuchung in drei Teile zu gliedern: Basisuntersuchungen, ergänzende Untersuchungen und Erweiterungen zu Genetischen Algorithmen (siehe Kapitel 6).

4.7.1 Untersuchte Parameter

Eine Aufzählung der zur Verfügung stehenden Parameter gibt untenstehende Liste, eine genaue Aufschlüsselung der Arbeits- und Funktionsweise eines jeden Parameters ist im jeweiligen Untersuchungsabschnitt in Kapitel 5 nachzulesen. In eckigen Klammern werden oft gebrauchte Abkürzungen der Parameter angegeben, in runden Klammern stehen die Abkürzungen der jeweiligen Optionen.

- Genetischer Algorithmus: modifizierter Simple Genetic Algorithm nach Goldberg, modifizierter Steady State Genetic Algorithm nach de Jong
- Populationsgröße [popsize]: von 20 bis mehreren tausend Individuen pro Population[7]

[7] je nach Speicherausbau des verfügbaren Systems

- Ersetzungsanteil an Individuen pro Generation [pRepl]: von 0 bis 1 einschließlich
- Mutationswahrscheinlichkeit [pMut]: von 0 bis 1 einschließlich, wobei in der Literatur übereinstimmend Werte $\geq$ 0,1 als ungeeignet angegeben werden.
- Skalierungsfunktion [scale]: keine Skalierung und Skalierung nach Rang mit Rangmultiplikationsfaktor von 1 bis beliebig hoch.
- Minimierungsfensterfunktion [minwindow]: Dynamic Reversed Fitness (DRF), Positive Scores 1 Div (PS1DIV), Ceil Sub Score (XCEIL)
- Selektionsschema für Eltern [select]: Roulettewheel Multi-Spin (RMS), Roulettewheel Single-Spin (RSS), Tournament, Uniform.
- Anzahl der Eltern pro Kind [nparents]: einstellbar von 2 bis beliebig hoch.
- Adaptive Mutation [adapmut]: aus und an, wobei die Adaptionsschranke von > 0 bis beliebig hoch einstellbar ist.
- Verhinderung von individuellen Dubletten [dprev]: aus und an,
- Übernahmeprävention[8] [fallback]: aus und an, mit beliebig zwischen 0 und 1 einstellbarer Schranke.
- Terminierungsfunktion des Genetischen Algorithmus [donetype]: nach n Generationen (ngen) oder wenn nach n Generationen kein besseres Individuum gefunden wurde (nobetter).
- Crossover Mechanismus [xover]: N-Point mit 1 bis beliebig vielen Bruchstellen, Randomwalk mit für jedes Elternteil beliebig zwischen 0 und 1 einstellbaren Bruchwahrscheinlichkeiten, Uniform mit für jedes Elternteil beliebig zwischen 0 und 1 einstellbaren Übernahmewahrscheinlichkeiten (wobei die Summe aller Wahrscheinlichkeiten gleich 1 sein muss)

[8] Verhinderung verfrühter Konvergenz durch Superindividuen

Parametername	Untersuchte Optionen
Algorithm	Simple, Steady State
popsize	100
pRepl	0,1; 0,5; 0,9
pMut	0,001; 0,01
scale	none, rank 2, rank 5, rank 10
minwindow	drf, xceil, ps1div
select	rms, rss, tournament, uniform
nparents	2; 3
adapmut	off
dprev	none
donetype	nobetter 50
xover	npoint mit 1, 2, 3 und 5 Bruchstellen randomwalk mit 0,05 und 0,1 Wahrscheinlichkeit uniform; uniform mit 0,5 Wahrsch. für den ersten Elternteil

Tabelle 4.1: Parametrierung der ersten Untersuchung.

4.7.2 Basisuntersuchung 1

Eine einfache Überschlagsrechnung, welche alle möglichen Kombinationen sinnvoller Parametereinstellungen obiger Liste ausrechnet, ergibt eine Zahl von ca. 66,5 Millionen möglichen Kombinationen nur für die hier dargestellten Parameter. Um die Rechenzeit innerhalb erträglicher Grenzen zu halten, wurden für eine erste Basisuntersuchung die in Tabelle 4.1 angegebenen Parameter zusammengestellt.

Unter Auslassung sich ausschließender Parameterkombinationen ergeben sich daraus 5040 verschiedene Kombinationen. Ziel dieser ersten umfassenden Untersuchung war es,

1. Unterschiede und Gemeinsamkeiten vom Simple Genetic Algorithm und Steady State Algorithm herauszuarbeiten.

2. den Einfluss verschiedener Parameter zu erkennen und zu qualifizieren.

3. Entscheidungshilfen für eine Vorauswahl der zu untersuchenden Parameter der zweiten Basisuntersuchung zu geben.

Aufgrund genetischer Drift in kleinen Populationen ist jede dieser Kombinationen zur statistischen Sicherheit 8 mal mit unterschiedlichen Zufalls-

Parametername	Untersuchte Optionen
Algorithm	Steady State
popsize	500, 1000, 2000, 5000
pRepl	0,5; 0,9
pMut	0,001; 0,01
scale	rank 2
minwindow	ps1div
select	rss, tournament, uniform
nparents	2; 3
adapmut	on, off
dprev	none, reinitialize
adapmut	0, 1
donetype	nobetter 50
xover	npoint mit 3 und 5 Bruchstellen randomwalk mit 0,05 Wahrscheinlichkeit

Tabelle 4.2: Parametrierung der zweiten Untersuchung.

zahlenreihen gerechnet worden. Die Ergebnisse wurden anschließend gemittelt. Hinweise zur Darstellungsform in dieser Arbeit finden sich in Abschnitt 4.8.

4.7.3 Basisuntersuchung 2

Die zweite Untersuchung wurde konzipiert, um Ergebnisse der ersten zu bestätigen bzw. zu relativieren. Als hauptsächliche Einflussgröße sollte hier die neu hinzugekommene Variation der Populationsgröße sowie das Zusammenspiel mit den aus der ersten Basisuntersuchung bekannten Optionen untersucht werden.

Die Auswahl der zu untersuchenden Parameter musste kurz nach Beendigung der ersten Basisuntersuchung und aufgrund erster vorläufiger Ergebnisse geschehen. Die begrenzt zur Verfügung stehende Rechenzeit wurde für eine eingehende Untersuchung des Steady State Algorithmus benutzt.

Auch hier wurden zu jeder Parameterkombination 8 Läufe mit verschiedenen Zufallszahlenreihen gerechnet und deren gemittelten Ergebnisse als Ergebnis der Parameterkombination angesehen.

4.7.4 Ergänzende Untersuchungen

Außerhalb der Hauptachsen untersuchter Parameterkombinationen wurden mehrere kleine Zusatzuntersuchungen gemacht. Einem der Mathematik entlehnten Verfahren der partiellen Ableitung gleich sollte zum einen der Einfluss adaptiver Mutation und Double prevention auf einige ausgewählte Parameterkombinationen für den Simple GA und den Steady State GA untersucht werden. Zum anderen sollten die aus der ersten Basisuntersuchung gewonnenen Erkenntnisse über Gemeinsamkeiten und Unterschiede zwischen Simple GA und Steady State GA für größere Populationen stichprobenartig verglichen werden.

4.8 Methoden zur Darstellung der Ergebnisse

Die Darstellung hyperdimensionaler Ergebnisebenen auf dem Papier gestaltet sich naturgemäß schwierig. In dieser Arbeit werden zwei verschiedene Formen verwendet: Diagramme und Tabellen. Beide können natürlich nur zweidimensional dargestellt werden und bedürfen einiger Interpretationserklärung, doch stellen sie wertvolle visuelle und numerische Resultate bereit.

4.8.1 Diagramme

Die X-Achse aller Diagramme hat als Einheit die Parameterkombination. Jeder Punkt stellt somit eine bestimmte Parameterkombination dar. Diese Kombinationen sind üblicherweise gruppiert, um einen bestimmten Zusammenhang visuell darstellen zu können. Die jeweilige Gruppierung ist der Bildunterschrift zu entnehmen. Dort werden die Gruppierungen rekursiv nach Name der jeweiligen Kombinationsgruppierung und Anzahl der enthaltenen Elemente angegeben. Beispiel (Abbildungen 4.1 und 4.2): *Populationsgröße - 500:1000:2000:5000 (144), , Ersetzungsgröße - 0,5:0,9 (72), Mutationsrate - 0,001:0,01 (36)* benennt die Aufteilung der X-Achse nach aufsteigender Populationsgröße. Jede der 4 verschiedenen Größen (500, 1000, 2000, 5000) besteht aus 144 Kombinationen, deren gemeinsamer Nenner die jeweilige Populationsgröße ist. Als erste rekursive Untergruppe ist die Ersetzungsgröße angegeben. Dies bedeutet, dass jede der 4 Gruppen der Populationsgröße noch einmal in zwei Gruppen à 72 Elementen unterteilt ist, wovon erstere als gemeinsamen Nenner die Ersetzungsgröße von 0,5 und die zweite von 0,9 hat. Als zweite rekursive Untergruppe ist die Mutationsrate

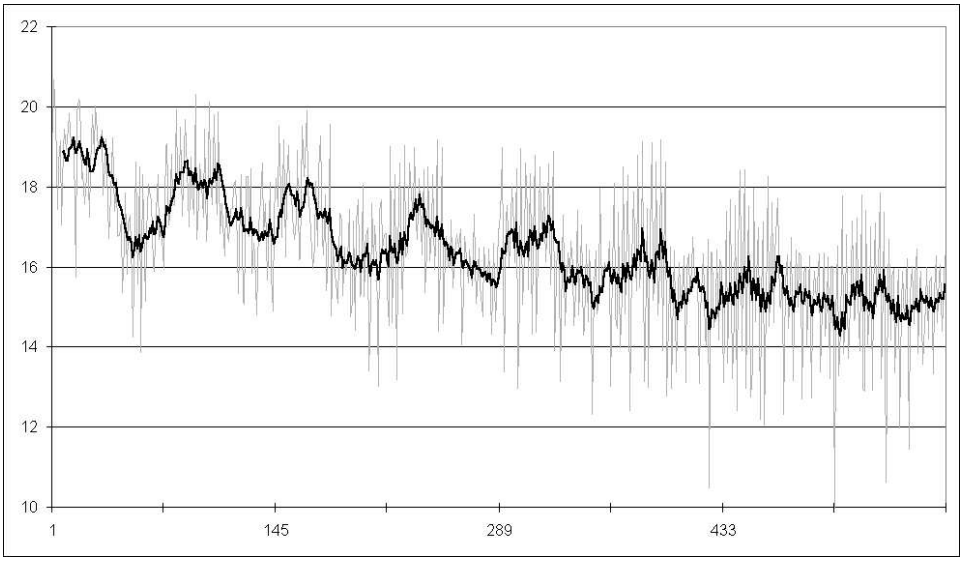

Abbildung 4.1: Energiewerte: Populationsgröße - 500:1000:2000:5000 (144); Ersetzungsgröße 0,5:0,9 (72); Mutationsrate - 0,001:0,01 (36)

angegeben. Dies bedeutet, dass jede der nach Ersetzungsgröße gruppierten Mengen noch einmal in zwei Gruppen à 36 Elementen unterteilt ist, wovon erstere als gemeinsamen Nenner die Mutationsrate von 0,001 und die zweite von 0,01 hat.

Die Y-Achse stellt die gemittelten Ergebnisse dar. Dies sind meistens das durchschnittlich erreichte Energieniveau (Einheit ist Kilokalorien pro Mol: $\frac{kcal}{mol}$) einer Parameterkombination oder die Anzahl der dafür benötigten Evaluationen. Diese Werte sind hellgrau aufgetragen. Um Trends deutlicher hervorzuheben wurde eine Tiefpassfilterung durch Mittelung (üblicherweise zwischen 8 und 16 Werten) diesen Werten in schwarz unterlegt.

4.8.2 Tabellen

Tabellen sind üblicherweise voll beschriftet und in wichtigen Auszügen im Text erläutert. Aufgrund chronischen Platzmangels wurden die Tabellenköpfe zumeist abgekürzt, die Abkürzungen werden jedoch in jeder Tabellenunterschrift aufgelistet und erklärt. Alle Energiewerte (Mittelwert, Minimum, Maximum) sind in $\frac{kcal}{mol}$ angegeben.

Eine Besonderheit der Tabellen stellt der |Z|-Wert dar. Dieser ist entweder einem Wilcoxon-Rangsummentest (2 Zeilen) oder einem Kruskal-Wallis-Test (3 oder mehr Zeilen) entnommen. Beide Verfahren testen gegebene

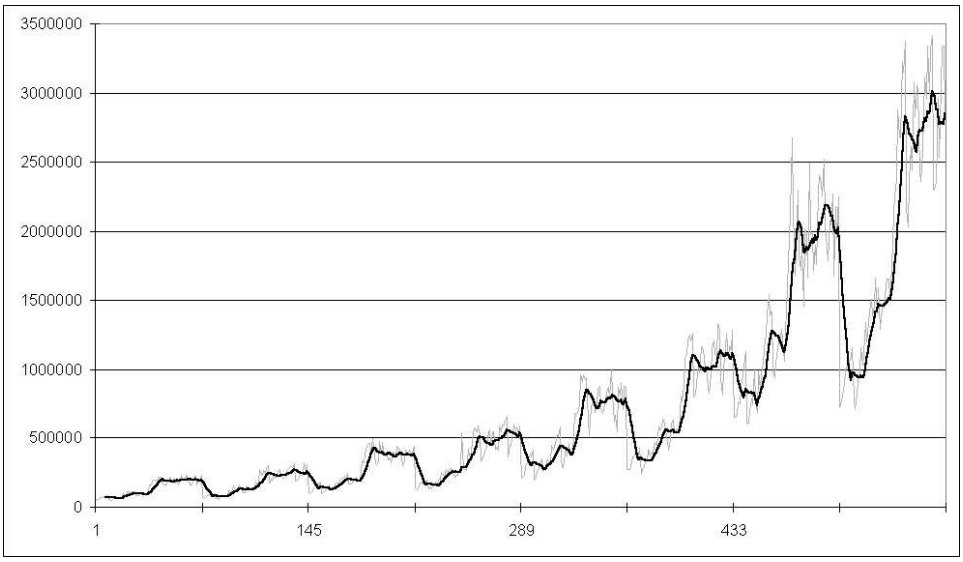

Abbildung 4.2: Benötigte Anzahl der Evaluationen: Aufteilung wie in Abb. 4.1.

unbekannte Distributionen auf Ungleichheit. Testwert (Zielvariable) ist jeweils immer der durchschnittlich erreichte Mittelwert des Objective Scores einer Parameterkombination. Der |Z|-Wert beschreibt hierbei die Fehlerwahrscheinlichkeit der getesteten Hypothese. Ist diese unterhalb eines gegebenen Schwellenwertes – in dieser Arbeit üblicherweise 0,01; also 1% – so können die Distributionen als verschieden angesehen werden. Ist die Fehlerwahrscheinlichkeit größer als der Schwellenwert, so ist die Ungleichheitshypothese falsch und muss verworfen werden.

Kapitel 5

Konfiguration Genetischer Algorithmen

"Never trust a tall dwarf. He's lying about something."

Ein Hauptziel dieser Arbeit bestand in einer breiten Untersuchung möglicher Einflussfaktoren von Genetischen Algorithmen unter Einbeziehung der speziellen Problemstellung der Molekülstrukturoptimierung bei kleinen Polypeptiden.

Dieses Kapitel stellt die untersuchten Parameter und Konfigurationen von Genetischen Algorithmen vor und untersucht die Ergebnisse jedes Parameters in Hinsicht auf seine Tauglichkeit in Bezug zur Problemstellung. Es wird aufbauend auf dem Vorwissen aus den Kapiteln 2 bis 4 die grundsätzliche Arbeitsweise eines jeden Parameters und dessen Optionen beschrieben und anschließend die Ergebnisse diskutiert.

Die dargestellten Ergebnisse beziehen sich vorwiegend auf die Versuchsreihen mit dem Steady State Algorithmus. Wie in Abschnitt 5.1 gezeigt werden wird, sind die Ergebnisse dieses Algorithmus weitgehend vergleichbar mit denen des Simple Genetic Algorithm. Bei einer Nennung von Ergebnissen aus den Läufen mit dem Simple GA wird jeweils explizit darauf hingewiesen.

5.1 Generationsformen

Die Methoden zur Selektion der für die nächste Generation zu entfernenden Individuen sind als zwei Generationsformen implementiert und untersucht worden: dem Simple GA Ansatz nach Goldberg und dem Steady State GA

Ansatz nach de Jong.

5.1.1 Simple GA

Beim Simple Genetic Algorithm nach Goldberg wird in jeder Generation die komplette Population durch die Kindgeneration ersetzt. Dieser ursprünglich nicht überlappende Algorithmus ist für die Zwecke dieser Arbeit um einen überlappenden Teil erweitert worden: die Ersetzungsgröße. Werden pro Generation n neue Nachkommen gezeugt, so werden aus der Ursprungspopulation die n Individuen mit der schlechtesten Fitness entfernt, um den "Neugeborenen" Platz zu schaffen.

Ist n gleich der Größe der Population, entspricht dies dem Goldbergschen Simple GA, welcher in der Fitnessverlaufsfunktion des jeweils besten Genoms einer Generation nicht monoton ist. Bei den Evolutionsstrategien von Rechenberg entspricht dies der (μ, λ) Ersetzungsstrategie.

Elitismus

Die Elitismusoption wurde von Goldberg eingeführt, um eine gefundene 'beste' Lösung nicht durch Zufall in der nächsten Generation zu verlieren, was durch den nicht-überlappenden Charakter der kompletten Populationsersetzung durchaus oft passieren kann. Die Elitismusoption prüft, ob das beste Individuum der aktuellen Generation einen schlechteren Fitnesswert als das beste Individuum der Elterngeneration hat. Ist dies der Fall, wird das beste Individuum der Vorgängergeneration in die aktuelle "hinübergerettet".

Die Elitismusoption ist durch den überlappenden Zusatz in der für diese Untersuchung benutzten Bibliothek unnötig geworden. Wird ein elitäres Verhalten gewünscht, so wird als Ersetzungsgröße pro Generation genau die Größe der Population minus einem Individuum gewählt.

5.1.2 Steady State GA

Dieser Algorithmus arbeitet auf der Basis von überlappenden Populationen beim Generationenwechsel. Ausgehend von der Ersetzungsgröße, d.h. dem Anteil der Population, der pro Generation entfernt wird, um Individuen der Kindgeneration Platz zu machen, werden für jede Generation entsprechend viele Nachkommen erzeugt. Anstatt danach jedoch die schlechtesten Individuen aus der Elterngeneration ohne Prüfung zu entfernen (wie beim Simple

Genetic Algorithm), werden alle Individuen der Elterngeneration und alle Individuen der neu erzeugten Kindgeneration miteinander verglichen und nur die besten aus dieser Gesamtmenge aller Individuen werden in die nächste Generation hinüber genommen.

Diese Ersetzungsstrategie ist mit der $(\mu + \lambda)$ Evolutionsstrategie von Rechenberg vergleichbar. Der Verlauf der Fitnessfunktion des jeweils besten Genoms ist über die Zeit monoton. Insgesamt führt dies zu einer wesentlich aggressiveren Suche als ein Simple Genetic Algorithm, da sehr gute Schemata immer mindestens eine Generation überleben.

5.1.3 Diskussion der Ergebnisse

Abb. 5.1 und Abb. 5.2 zeigen eine grafische Gegenüberstellung des Simple Genetic Algorithm und des Steady State Algorithm. Der erste Eindruck vermittelt eine weitgehende Gleichheit der Ergebnisse bei fast allen Parametereinstellungen. Als sehr deutlicher Unterschied jedoch sind beim Simple GA regelmäßig wiederkehrende Piks zu erkennen, welche beim Steady State nicht oder nur sehr undeutlich ausgeprägt sind. Die Regelmäßigkeit ist auf die Parameterkonstellation “scaling none” und “selection rss” bzw. “selection rms” zurückzuführen. Hier treffen genau die Faktoren zusammen, welche proportionale Selektionsschemata ungünstig erscheinen lassen: die ungehemmte Selektion eines Superindividuums führt in vielen Fällen zu einer verfrühten Konvergenz, wobei durch diesen Takeover zu viele potentiell gute Building-Blocks anderer Individuen vernichtet werden und der Genetische Algorithmus in einem Suboptimum stecken bleibt. Weitere Einzelheiten zu diesem Thema werden in den Abschnitten über Selektion (5.8) und Skalierung (5.9) besprochen.

Interessant dabei ist, dass der Steady State Algorithmus je nach Mutationsrate diese Übernahmen abschwächt oder sogar fast komplett verhindern kann (Abb. 5.1 und Abb. 5.2). Die charakteristischen Piks sind bei genauer Betrachtung für die Mutationsrate von 0,001 noch als solche zu erkennen, für eine Mutationsrate von 0,01 hingegen kaum noch. Diese Beobachtung läßt sich auch theoretisch durch das Zusammenspiel von Steady State GA und hoher Mutation bei Superindividuen erklären: taucht ein solches Individuum auf, so wird es fast komplett die gesamten Nachkommen für die nächste Generation produzieren. Bei einer durchschnittlichen Mutation von 2,4 Bit pro Genom ist jedoch zu erwarten, dass ein Teil der Nachkommen

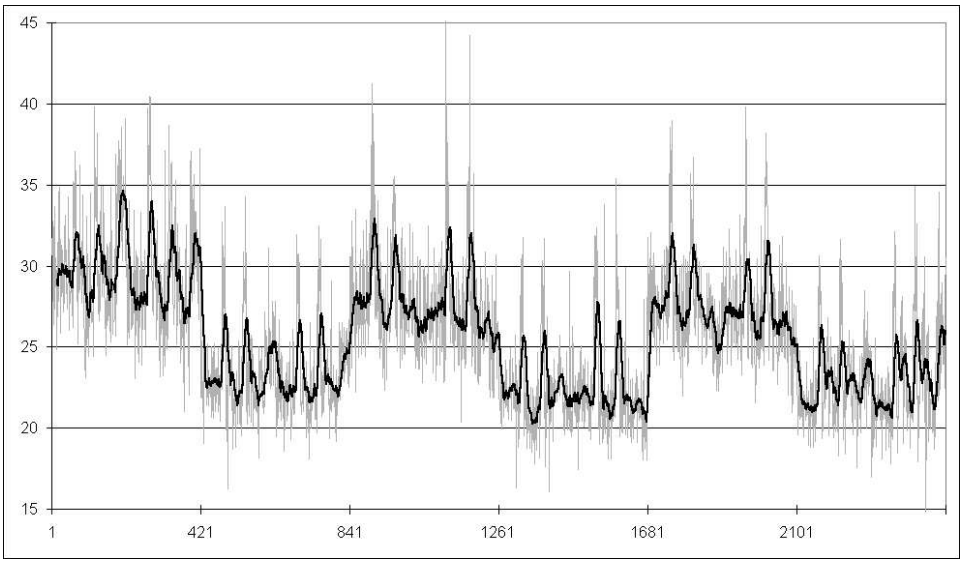

Abbildung 5.1: Simple GA Energiewerte: Ersetzungsgröße- 0,1:0,5:0,9 (840); Mutationsrate - 0,001:0,01 (420). Ergebnisse aus Basisuntersuchung 1.

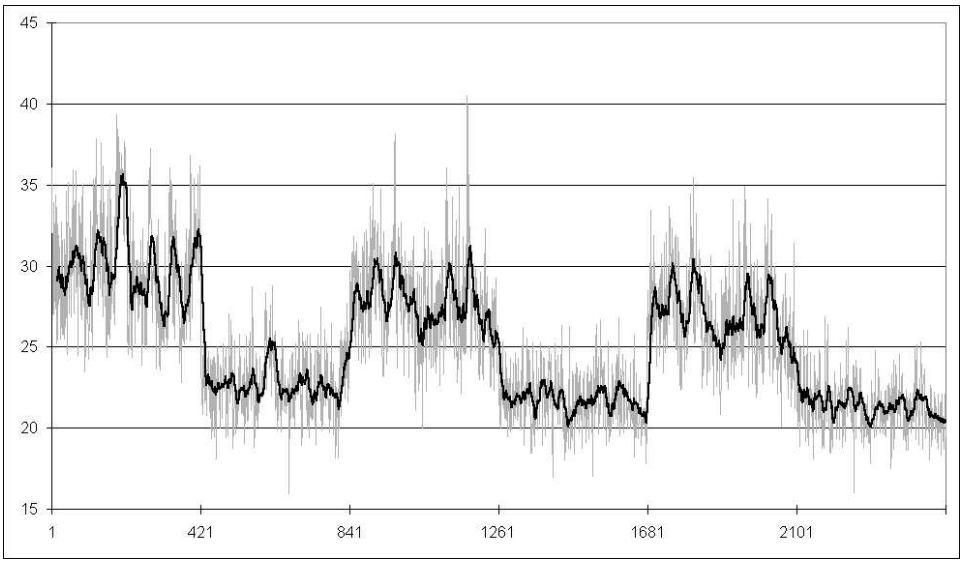

Abbildung 5.2: Steady State GA Energiewerte: Ersetzungsgröße - 0,1:0,5:0,9 (840); Mutationsrate - 0,001:0,01 (420). Ergebnisse aus Basisuntersuchung 1.

Algorithmus	N	∅ En.	σ	Min	Max	nEvals	|Z|
Simple	2280	25,02	3,68	14,79	39,07	31601	0,0082
Steady State	2280	24,83	3,90	15,99	39,30	25609	

Tabelle 5.1: Vergleich von Steady State Algorithm und Simple GA unter Auslassung der Ergebnisse bei proportionaler Selektion ohne Skalierung.

N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der beiden Distributionen

schlechter sein wird als die schon in der Population vorhandenen Individuen. Letztere werden bei einem Simple GA bedingungslos ersetzt, bei einem Steady State GA jedoch nur, wenn sie schlechter sind als die Nachkommen. Deshalb werden beim Steady State GA immer einige Individuen mit einem wesentlich anderen Genom als das des Superindividuums noch in die nächste Generation hinübergerettet. Dies verzögert die Konvergenz und verhindert eine vollständige Übernahme einer Population innerhalb nur einer Generation. Damit bleibt der Genpool über längere Zeit variabel, womit auch die Suche in andere Richtungen als die des Superindividuums nicht aufgegeben wird.

Um noch bessere Vergleichsmöglichkeiten zu erhalten, können die oben genannten schlechten Kombinationen ausgeblendet und beide Algorithmen dann verglichen werden. Tabelle 5.1 zeigt die Ergebnisse eines Wilcoxon 2-Stichprobentests für diese gesäuberten Parameterkombinationen beider Algorithmen.

Tab. 5.1 ist zu entnehmen, dass der Hypothesentest auf Ungleichheit der Ergebnisdistributionen bei einer Fehlerwahrscheinlichkeit von unter 1% als richtig angenommen wird. Dies bedeutet, dass – aufgrund des besseren Mittelwerts – der Steady State Algorithmus auch unter Auslassung von für den Simple GA ungünstiger Parameterkombinationen besser ist als der Simple Genetic Algorithm. Bemerkenswert hierbei ist, dass die Zahl der notwendigen Evaluationen beim Steady State Algorithmus um fast 19% niedriger als beim Simple Genetic Algorithm liegt. Einen visuellen Vergleich der benötigten Evaluationen liefern die Abbildungen 5.3 und 5.4.

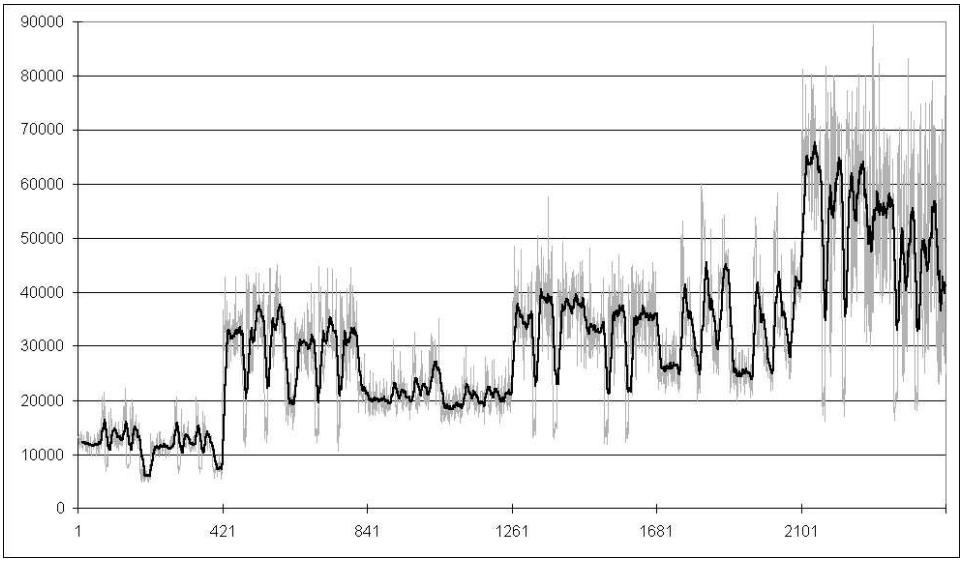

Abbildung 5.3: Anzahl Evaluationen Simple GA: Ersetzungsgröße - 0,1:0,5:0,9 (840); Mutationsrate - 0,001:0,01 (420)

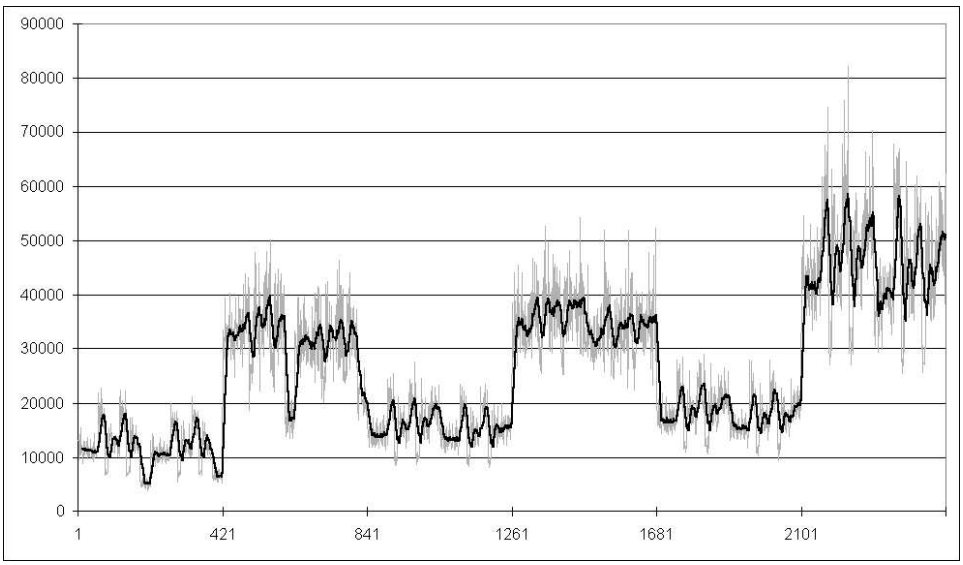

Abbildung 5.4: Anzahl Evaluationen Steady State GA: Ersetzungsgröße - 0,1:0,5:0,9 (840); Mutationsrate - 0,001:0,01 (420)

pRepl	pMut	GA	N	∅ En.	σ	Min	Max	nEvals	|Z|
0,1	0,001	SI	380	29,22	3,06	21,06	39,07	11749	0,1073
		SS	380	29,61	3,27	22,87	39,30	10965	
	0,01	SI	380	22,85	1,91	16,28	31,11	31020	0,4261
		SS	380	22,73	1,94	15,99	28,80	30489	
0,5	0,001	SI	380	27,09	2,44	20,36	35,68	21283	0,2362
		SS	380	27,29	2,60	19,99	34,87	15073	
	0,01	SI	380	21,70	1,92	16,10	33,81	36220	0,6484
		SS	380	21,62	1,70	16,97	28,50	34788	
0,9	0,001	SI	380	26,80	2,15	20,98	33,20	33762	0,0174
		SS	380	26,44	2,54	20,04	34,06	17588	
	0,01	SI	380	22,45	2,46	14,79	34,55	55570	0,0001
		SS	380	21,29	1,58	16,00	26,91	44753	

Tabelle 5.2: Vergleich von Steady State GA und Simple GA unter Berücksichtigung der Ersetzungsgröße und der Mutationsrate.

pRepl: Populationsersetzungsgröße; *pMut:* Mutationswahrscheinlichkeit;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
GA: Genetischer Algorithmus: SI=SImple GA, SS=Steady State GA;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der beiden Distributionen;

Um eine weitere Aufschlüsselung eventueller Gemeinsamkeiten und Unterschiede zu erhalten, können verschiedene Parameter gruppiert dargestellt werden. Dies geschieht in den Tabellen 5.2 und 5.3, welche die Ergebnisse nach Ersetzungsgröße und Mutation bzw. nach Selektionsschemata gruppiert zeigen. Die Ergebnisse sind insofern erstaunlich, als sie eine Ungleichheit beider Algorithmen über weite Teile der Parameterkombinationen nicht bestätigen können, da diese Hypothesen wegen zu großen Fehlerwahrscheinlichkeiten meist abgelehnt werden müssen.

In Tabelle 5.2 wird gezeigt, dass nur bei einer Ersetzungsgröße von 0,9 und einer Mutationsrate von 0,01 eine Ungleichheit beider Distributionen angenommen werden kann. Eine Ungleichheit der Distributionen für eine Ersetzungsgröße von 0,9 und einer Mutation von 0,001 wird hingegen schon knapp abgelehnt. Alle Hypothesen auf Ungleichheit anderer Kombinationen

Selekt	GA	N	∅ En.	σ	Min	Max	nEvals	|Z|
rms	SGA	600	24,92	3,48	14,79	36,68	33063	0,4414
	SS	600	24,88	3,83	16,00	37,62	26584	
rss	SGA	600	24,70	3,57	16,28	37,17	32805	0,9934
	SS	600	24,76	3,73	18,47	35,34	26155	
tournament	SGA	720	24,95	3,61	16,30	34,83	30503	0,3070
	SS	720	24,79	3,63	15,99	36,07	24429	
uniform	SGA	360	25,85	4,12	18,07	39,07	29352	0,0001
	SS	360	24,91	4,79	17,82	39,30	25435	

Tabelle 5.3: Vergleich von Steady State GA und Simple GA unter Berücksichtigung der Selektionsschemas.

N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
GA: Genetischer Algorithmus: SI=SImple GA, SS=Steady State GA;
Selekt: Selektionsschema: rms=Roulette Multi Spin, rss=Roulette Single Spin;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der beiden Distributionen;

von Ersetzungsgröße und Mutation werden deutlich abgelehnt. Tabelle 5.3 zeigt, dass auch die Crossover Parameter unter beiden Algorithmen weitgehend gleich arbeiten. Nur der Uniform Crossover wird beim Hypothesentest auf Ungleichheit der Distributionen nicht abgelehnt. Somit arbeitet dieser Operator mit dem Steady State Algorithm besser zusammen als mit dem Simple Genetic Algorithm.

Auch in diesen beiden Tabellen ist die Anzahl der durchschnittlich notwendigen Evaluationen beim Simple Genetic Algorithm immer - zumeist sogar deutlich - höher als beim Steady State Algorithm. Dies und die weiter oben genannten Erkenntnisse lassen sich folgendermaßen zusammenfassen: Ein Steady State Ersetzungsalgorithmus wird zumindest gleich gute Ergebnisse erzielen wie ein Simple Genetic Algorithmus, in den meisten Fällen sogar bessere. Auch wird ein Steady State GA für diese Ergebnisse weniger Evaluationen brauchen als ein Simple GA und kann somit als effektiver angesehen werden.

5.2 Ersetzungsgröße

Die Möglichkeit, durch die Ersetzungsgröße pro Generation mehrere Individuen in die Nachfolgegeneration zu retten, stellt eine Art “Super-Elitismus” dar. Ein “unbeabsichtigtes Verlieren” guter Genome von einer Generation zur anderen wird damit effektiv verhindert. Dies erlaubt den Einsatz von sehr weichen Selektionsschemata zur Maximierung des Exploitationseffektes im Genetischen Algorithmus. Die Funktionsweise dieses Parameters ist schon im vorangegangenen Abschnitt über Generationsformen eingehend erläutert worden.

Diskussion der Ergebnisse

Die Ersetzungsgröße ist in dieser Arbeit der einzige Parameter, der nicht gesondert abgehandelt wird. Dies liegt mit in der Tatsache begründet, dass sich aufgrund der engen Interaktion dieses Parameters mit anderen Parametern – z.B. Generationsform und Mutation – keine getrennte Darstellung der Ergebnisse erzielen lässt. Deshalb wird in den folgenden Abschnitten immer wieder auf die Ersetzungsgröße eingegangen werden.

5.3 Populationsgrößen

Die in den Basisuntersuchungen verwendeten Populationsgrößen variieren über ein Spektrum von 100 bis 5000 Individuen pro Population. Die relativ kleine Populationsgröße von 100 Individuen ist für die erste Basisuntersuchung gewählt worden, um eine möglichst große Bandbreite an verschiedenen Parametern innerhalb eines vernünftigen Zeitrahmens austesten zu können. In der zweiten Basisuntersuchung sollte dann der Einfluss der Populationsgröße auf einige ausgewählte Parameterkombinationen der ersten Versuchsreihe untersucht werden, um mögliche Interaktionen bei steigender Populationsgröße zu erkennen.

5.3.1 Diskussion der Ergebnisse

Die Ergebnisse der ersten Untersuchung brachten hinsichtlich der verwendeten Größe von Populationen die – von vornherein absehbare – Erkenntnis, dass 100 Individuen bezüglich der relativen Länge des Genoms (240 Bit) bei allen getesteten Parameterkombinationen eine zu kleine Population darstellen, um den Suchraum in korrekter Weise zu testen.

In Tabelle 5.4 sind die Ergebnisse für einen ersten Überblick zusammengestellt. Die Zeile mit der Populationsgröße 100 stammt aus der ersten Basisuntersuchung und ist wegen der sehr unterschiedlichen Parameterkombinationen beider Untersuchungen mit den folgenden Zeilen nur bedingt vergleichbar. Zwei Merkmale stechen besonders hervor: einerseits die kontinuierliche Verbesserung des durchschnittlich gefundenen Mittelwertes, andererseits der zur Populationsgröße proportionale Anstieg der benötigten Anzahl an Evaluationen. In Abb. 5.5 ist die schrittweise Verbesserung der gemittelten Ergebnisse bei steigender Populationsgröße dargestellt, Abb. 5.6 zeigt die hierzu steigende Anzahl der Evaluationen.

In diesen Zahlen sind auch einige sehr ungünstige Parameterkombinationen enthalten. In Tabelle 5.5 wurden deshalb nur gute Kombinationen betrachtet, um den Progressionsverlauf guter bis sehr guter Parameterkombinationen darzustellen. Diese Parameterkombinationen arbeiten mit den Optionen

- Populationsersetzungsgröße 0,5 und 0,9.
- Mutationsrate 0,001 und 0,01.

Popgr.	N	∅ En.	σ	Min	Max	nEvals
100	2520	25,08	4,12	15,99	40,55	13551
500	144	17,75	1,38	13,85	20,68	157784
1000	144	16,68	1,40	13,03	19,90	321816
2000	144	15,94	1,66	10,49	19,18	660567
5000	144	15,23	1,62	9,97	18,43	1753215

Tabelle 5.4: Gegenüberstellung der Ergebnisse für den Steady State Algorithmus bei verschiedenen Populationsgrößen.

Popgr.: Populationsgröße;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;

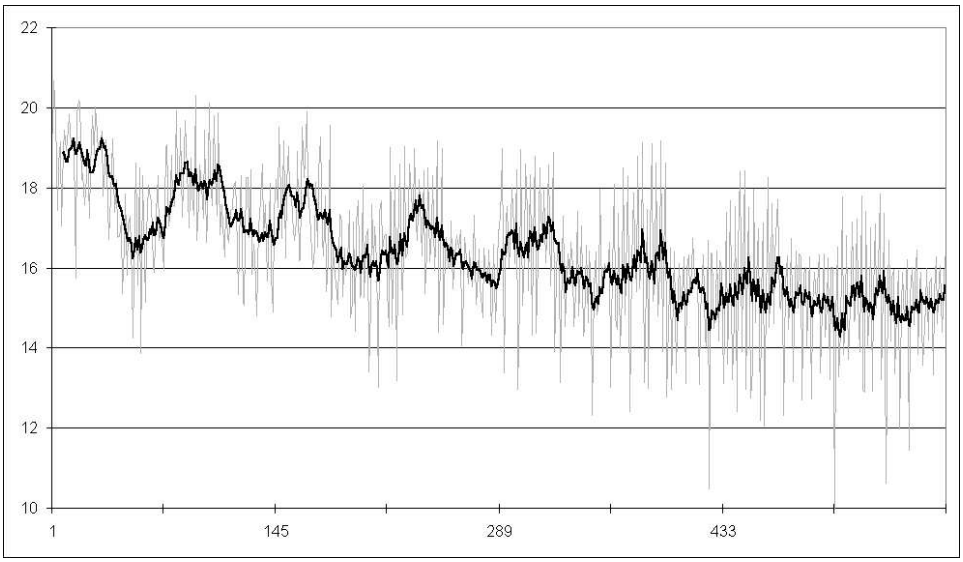

Abbildung 5.5: Energiewerte für den Steady State Algorithm bei verschiedenen Populationsgrößen: Populationsgröße - 500:1000:2000:5000 (144); Ersetzungsgröße - 0,5:0,9 (72); Mutationsrate - 0,001:0,01 (36)

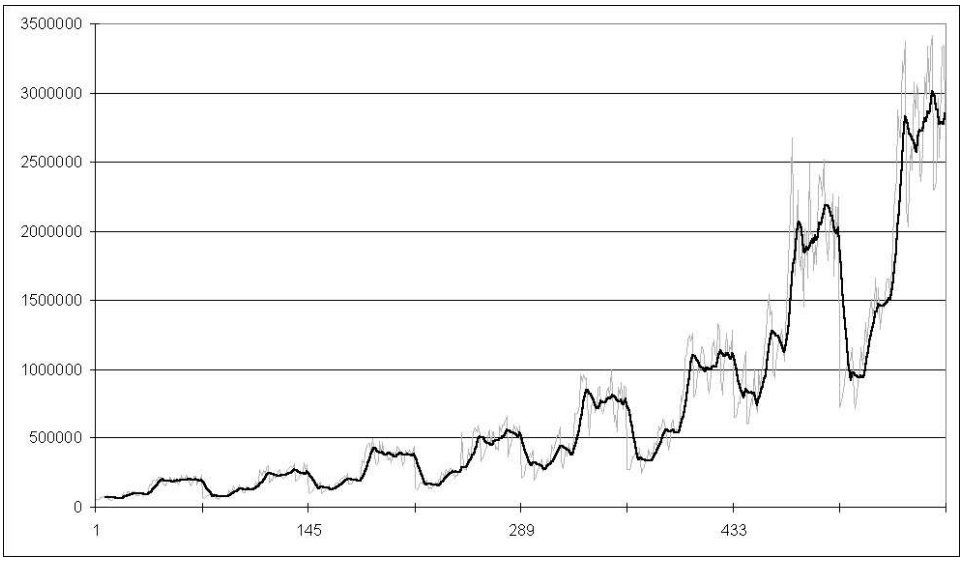

Abbildung 5.6: Benötigte Anzahl der Evaluationen: Aufteilung wie in Abb. 5.5.

- Crossover N-Point mit 3 und 5 Kreuzungspunkten.
- Selektionsschema Uniform, Tournament und RSS (RSS mit Ranking Faktor 2).
- 2 und 3 Eltern.
- Ohne Double prevention.

Die dargestellten Werte für den durchschnittlichen Mittelwert, das Minimum und das Maximum lassen sich durch einfache Formeln interpolieren. Diese haben einen zur Populationsgröße degressiven Charakter und können so für die Vorhersage der Ergebnisse bei verschiedenen Populationsgrößen benutzt werden. Die Formeln setzen sich zusammen aus einem skalierten, degressiven und von der Populationsgröße abhängigen Anteil und einem Offset, welcher durch das kleinste bisher bekannte Energieniveau für Metenkephalin gebildet wird. Diese Formeln bedürfen allerdings noch experimenteller Überprüfung für größere Populationen ($\geq$ 10000 Individuen).

Annäherung an die Mittelwert-Reihe: $50 \cdot \left(\frac{1}{\ln(popsize \cdot 3)}\right) + 1,75$

Nach dieser Formel liegt bei den genutzten Parameterkombinationen und einer Populationsgröße von 100000 der durchschnittlich erreichte Mittelwert des Energieniveaus bei 10,8.

Popgr.	N	∅ En.	σ	Min	Max	nEvals
100	96	23,80	3,26	18,47	32,10	13561
500	96	17,39	1,36	13,85	20,68	157025
1000	96	16,10	1,17	13,03	18,70	316083
2000	96	15,18	1,33	10,49	17,90	650391
5000	96	14,44	1,32	9,97	16,47	1735222

Tabelle 5.5: Gegenüberstellung ausgewählter Parameterkombinationen für den Steady State Algorithmus bei verschiedenen Populationsgrößen. Parametrisierung ist im Text angegeben.

Popgr.: Populationsgröße;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;

Annäherung an die Min-Reihe: $30 \cdot \left(\frac{1}{\ln(popsize \cdot 0{,}6)}\right) + 1,75$

Diese Kurve ist etwas steiler als die Mittelwert-Kurve: sie fällt bei zunehmender Populationsgröße also stärker ab. Trotzdem liegt der Mittelwert der erreichten Minimalwerte bei einer Populationsgröße von 100000 nur bei 8,03.

Annäherung an die Max-Reihe: $50 \cdot \left(\frac{1}{\ln(popsize \cdot 0{,}6)}\right) + 1,75$

Diese Kurve ist nur geringfügig steiler als die Min-Kurve, liegt jedoch etwas höher. Der Mittelwert der erreichten Maximalwerte liegt für eine Populationsgröße von 100000 bei 12,21.

Es zeigt sich, dass auch für sehr große Populationen die im Durchschnitt ermittelten Werte nicht in die Nähe des bisher bekannten Optimums (bei 1,7524) kommen werden. Die Gründe hierfür werden in Abschnitt 5.7 (Crossover) näher untersucht und beschrieben, eine mögliche Abhilfe wird mit dem in Kapitel 6 vorgestellten pyramidalen Genetischen Algorithmus aufgezeigt.

5.4 Abbruchkriterien

Die Entscheidung, wann ein Genetischer Algorithmus anzuhalten sei, ist keine triviale Angelegenheit [8]. Abgesehen von dem Fall, in dem alle Genome einer Population gleich sind, gibt es keine einfache Möglichkeit zur Bestim-

mung des optimale Zeitpunktes zur Beendigung eines GA Laufes. Es kommt dazu, dass entweder zu früh abgebrochen wird – die gefundene Lösung ist dann unter Umständen nicht optimal und der Genpool der Population hätte nach dem Schema-Theorem und der Building-Block-Hypothese noch ein Potential zur Verbesserung besessen – oder zu spät. Dann hat der Algorithmus unnötig viele Funktionsevaluationen durchgeführt und so Zeit verloren.

5.4.1 Bitkonvergenz

Die Bitkonvergenz aller Genome einer Population als Kriterium zu benutzen ist in doppelter Hinsicht schwierig. Einerseits muss entschieden werden, ob alle oder nur ein Teile (und wenn ja, welcher Teile) der Gene zum Vergleich herangezogen werden. Andererseits sind die zeitlichen Kosten zur Ermittlung der Bitkonvergenz eine zur Genomlänge quadratische und zur Populationsgröße exponentielle Funktion und somit für große Populationen extrem zeitaufwendig[1].

5.4.2 Schwellenwerte

Ein augenscheinlich einfaches Haltekriterium stellen Schwellenwerte dar. Wird ein solcher Schwellenwert über- oder unterschritten, so terminiert der GA-Lauf. Als Schwellenwert in Frage kommen z.B. die Standardabweichung des Mittelwerts, die Varianz oder ganz einfach der Objective Score des besten Individuums einer Population. Der Vorteil dieser Methode liegt sicherlich in der Berechnungsgeschwindigkeit des Kriteriums. Der gravierende Nachteil jedoch besteht in der Tatsache, dass die Schwelle sehr schnell zu hoch oder zu niedrig geschätzt werden kann. Unter Umständen wird diese also nie erreicht und der Algorithmus läuft unterbrechungslos weiter. Ein weiterer Nachteil ist bei der Benutzung des Simple GA bei kompletter Populationsersetzung zu beachten: die großen Werteschwankungen zwischen den einzelnen Generationen und der nicht-monotone Verlauf der Ergebniskurve verhindern in den meisten Fällen auch bei korrekt gewählter Schwelle ein Erreichen derselben.

5.4.3 Haltefenster

Im allgemeinen wird der Begriff Konvergenz etwas weniger streng gehandhabt, als es die Theorien zur Bitkonvergenz beschreiben. Im Normalfall wird

[1] siehe Abschnitt 3.4.11 (Konvergenz).

er so interpretiert, dass ein GA keine besseren Ergebnisse mehr produziert. Symptomatisch für diesen Pragmatismus ist folgendes Zitat aus [27]:

> "What people really mean is: I'm not willing to wait the GA to find a new, better solution, because I've already waited longer than I wanted to and it hasn't improved in ages."

Daraus lässt sich eine einfach zu implementierende und schnelle Möglichkeit für ein Haltekriterium realisieren. Für die Funktion eines sogenannten Haltefensters wird eine Anzahl h an Generationen definiert, innerhalb derer sich die Fitness des jeweils besten Individuums verbessern muss, damit der Algorithmus weiterläuft. Sind also h Generationen verstrichen, ohne dass der GA ein besseres Individuum als das bisher beste finden konnte, wird abgebrochen.

Die Ermittlung von h richtet sich allerdings nach vielen Faktoren, unter anderem der Zielfunktion, der Populationsgröße, dem Selektionsschema usw. Deshalb sollte dieser Wert experimentell ermittelt und eher zu groß als zu klein bemessen werden.

5.4.4 Diskussion der Ergebnisse

Alle Versuchsreihen wurden mit dem Kriterium des Haltefensters gerechnet. Das Verfahren ist effektiv, wenn die Anzahl der Generationen richtig abgeschätzt wird. Als Anzahl der notwendigen Generationen für das Haltefenster wurde nach einigen Voruntersuchungen der Wert 50 angenommen.

Dies kann allerdings auch als Beispiel für eine in Teilen falsche Abschätzung herangezogen werden (Abb. 5.7). Hier wurden alle Parameterkombinationen mit einem Haltefenster von 50 berechnet. Zu bemerken ist in der Gruppe mit einem Populationsersetzungsanteil von 0,1 die hohe Variabilität der Ergebnisse. Es zeigte sich, dass für eine so geringe Ersetzungsrate das Haltefenster bedeutend vergrößert werden muss. Bei einer Neuberechnung der Gruppe mit dem Ersetzungsanteil von 0,1 und einem Haltefenster von 200 ergeben sich Werte wie in Abb. 5.8 dargestellt.

Im allgemeinen erwies sich der Wert von 50 Generationen jedoch als ausreichend effektiv. Bei Beendigung der Läufe war die Standardabweichung des Objective Scores immer sehr niedrig, meistens kleiner als 0,001. Dies bedeutet, dass die Populationen bei Abbruch des Genetischen Algorithmus eine

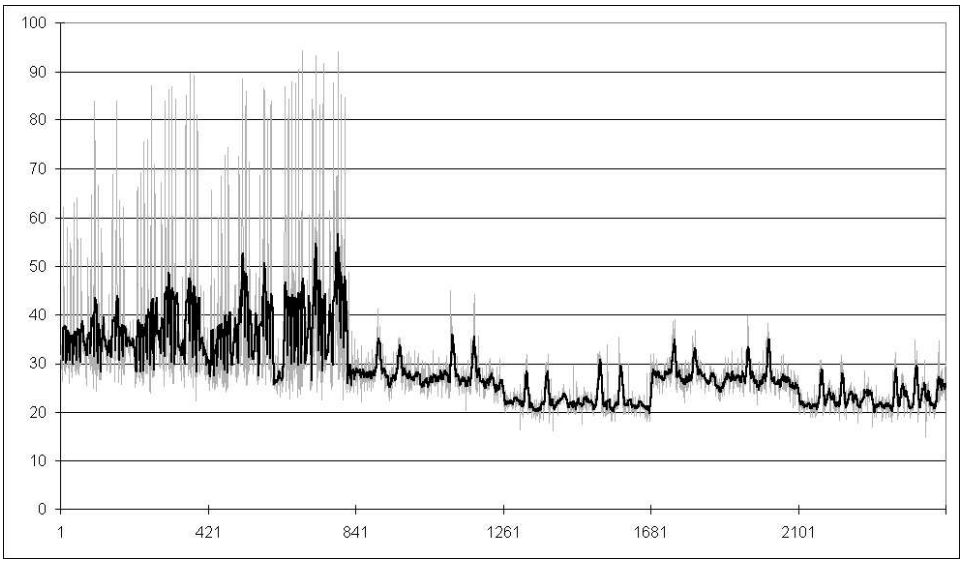

Abbildung 5.7: Simple GA Energiewerte: Ersetzungsgröße - 0,1:0,5:0,9 (840); Mutationsrate - 0,001:0,01 (420). Alle Werte mit einem Termination Window von 50 Generationen berechnet.

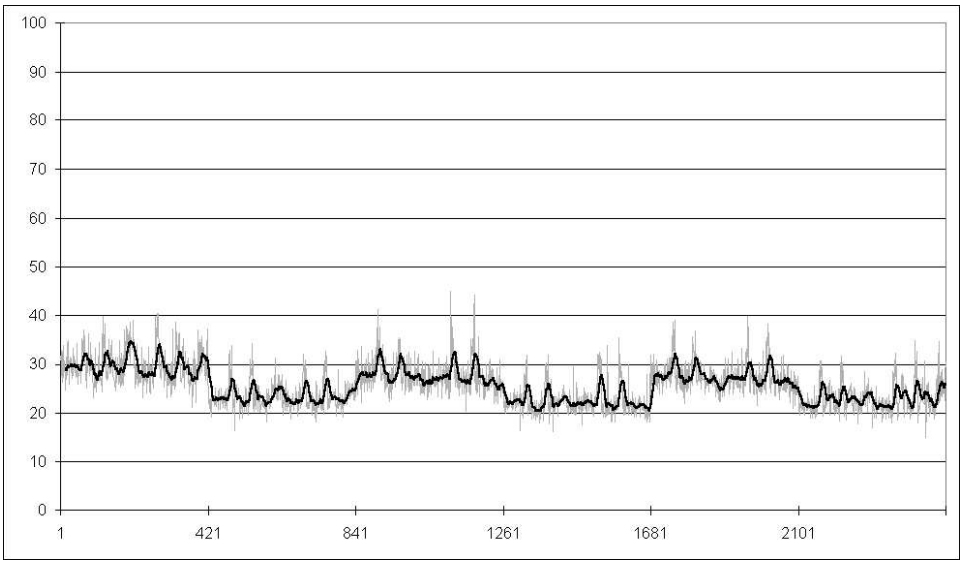

Abbildung 5.8: Simple GA Energiewerte: Ersetzungsgröße - 0,1:0,5:0,9 (840); Mutationsrate - 0,001:0,01 (420). Die Werte für die Ersetzungsgröße von 0,1 sind mit einem Termination Window von 200 Generationen berechnet, die übrigen mit einem Fenster von 50 Generationen.

hohe Konvergenzrate erreicht und gleichzeitig niemals mehr als 50 Generationen umsonst berechnet hatten.

5.5 Anzahl der Eltern

5.5.1 Motivation

Eine in der Literatur bisher wenig untersuchte Option ist die Anzahl der Eltern pro Kind. Eine solche Option mag sich paradox anhören, da alle aus der Natur bekannten Organismen mit nur maximal 2 Elternteilen sich reproduzieren. Doch geben Genetische Algorithmen nicht vor, die Verfahren der Natur genau kopieren zu wollen. R.A. Fisher kommentiert dieses Thema folgendermaßen [27]:

> "No practical biologist interested in (e.g.) sexual reproduction would be led to work out the detailed consequences experienced by organisms having three or more sexes; yet what else should [s/]he do if [s/]he wishes to understand why the sexes are, in fact, always two?"
>
> (three sexes would make for even weirder grammar, [s/] said ...)

Es ist jedoch ohne weiteres möglich, nicht-natürliche, aber verwandte Verfahren in einem GA zu implementieren. Hierzu gehört auch die Option, ein Kind von 3 oder mehr Eltern erzeugen zu lassen.

Dies bedingt gewisse Anpassungen der Algorithmen für Selektion und Crossover. Diese sind als natürlich erscheinende Erweiterungen so implementiert, dass sie in den jeweiligen Kontext und im Vergleich zu den Algorithmen für 2 Eltern nahtlos hineinpassen. Die genaue Beschreibung der Änderungen sind in den jeweiligen Passagen über Selektion (5.8) und Crossover (5.7) zu finden.

5.5.2 Diskussion der Ergebnisse

Wie Tabelle 5.6 zu entnehmen, wird nur für kleine Populationsgrößen und niedrige Mutationsraten die Hypothese angenommen, dass ein Unterschied im Ergebnis besteht. Dies führt zu der Annahme, dass eine Erhöhung der Anzahl der Eltern als zusätzlicher Diversifikations- bzw. Disruptionsfaktor innerhalb eines begrenzten Rahmens – nämlich bei kleinen Populationsgrößen – sinnvoll ist. Die Elternanzahl stellt nur einen möglichen solchen Faktor

Populationsgröße	Mutation 0,001	Mutation 0,01	Zusammen
100	0,0001	0,0179	0,0005
500	0,2577	0,3952	0,8028
1000	0,6934	0,4956	0,8510
2000	0,7143	0,1542	0,2590
5000	0,1677	0,3646	0,1335

Tabelle 5.6: Fehlerwahrscheinlichkeiten des Wilcoxon Rangsummentest bei Vergleich der Wertedistrubutionen für 2 oder 3 Eltern bei verschiedenen Mutationsraten. Werte aus der ersten Basisuntersuchung für den Steady State Algorithmus.

dar, welcher aber auch durch höhere Mutationsraten erreicht werden kann. Allerdings sollte bei der Arbeit mit kleinen Populationen in hochgradig multimodalen Suchräumen immer auch diese Option untersucht werden.

5.6 Mutation

Die Mutation und ihre Wahrscheinlichkeit ist als sogenannte Basen– bzw. Bitmutation implementiert worden. Dies ist auch der Ansatz, welcher die in der Zelle ablaufenden Mechanismen am ehesten abbildet.

Diverse Untersuchungen über eine “optimale” Mutationsrate kommen in Ihrer Gesamtheit zu keinem konkreten Ergebnis. Praktisch jede Aufgabe und jeder verwendete Genetische Algorithmus verhält sich auf das Optimum dieses Werts anders. Als Referenzlinie kann jedoch die Aussage von de Jong (1975) aufgegriffen werden, dass alle Wahrscheinlichkeiten $\geq 0{,}1$ ein Abdriften der GAen in den Random Search bewirken [39].

Zum Beispiel geben Schaffer, Caruna, Eshelman und Das in [46] als optimale Mutationsrate die empirisch ermittelte Formel $p_m \approx \frac{1.75}{n\sqrt{l}}$ an, mit n als Populationsgröße. Theoretische Arbeiten von Bäck [2] schlagen eine Mutationsrate von $p_m = \frac{1}{l}$ vor, wobei l die Genomlänge in Bits ist. In einer späteren Untersuchung von Bäck [4] erweitert der Autor seine Aussage, indem er vorschlägt, die Mutationsrate auf einen anderen Wert zu setzen, um das Ausbrechen aus lokalen Optima zu ermöglichen. Letztgenannte Arbeit basiert auf Experimenten mit individueller Selbstadaption der Mutationsraten und es werden leider keine genaueren Angaben über die Mutationsraten gemacht.

Popgr	pMut	N	∅ En.	σ	Min	Max	nEvals	|Z|
100	0,001	1260	28,19	3,33	19,99	40,44	15112	0,0001
	0,01	1260	21,96	1,88	15,99	28,80	37348	
500	0,001	72	18,56	1,10	15,74	20,68	94024	0,0001
	0,01	72	16,95	1,14	13,85	19,23	221543	
1000	0,001	72	17,32	1,43	13,21	19,90	192896	0,0001
	0,01	72	16,03	1,04	13,03	18,03	450737	
2000	0,001	72	16,48	1,80	12,43	19,18	399522	0,0002
	0,01	72	15,40	1,31	10,49	17,97	921108	
5000	0,001	72	15,33	1,95	9,97	18,43	1108388	0,3615
	0,01	72	15,12	1,19	11,47	16,75	2398310	

Tabelle 5.7: Vergleich der Mutationeinflüsse für verschiedene Populationsgrößen mit dem Steady State Algorithmus.

Popgr: Populationsgröße; *pMut:* Mutationswahrscheinlichkeit;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
|Z|: Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der beiden Distributionen

Es wurden deshalb als Basis Wahrscheinlichkeiten von 0,01 und 0,001 angenommen. Diese liegen einmal weit über und einmal weit unter den Vorschlägen von Bäck, sind aber immer noch bedeutend höher als die Mutationsraten aus der Natur. Dies wird durch die im Vergleich zur natürlichen Evolution kleinen Populationsgrößen bedingt. Setzt man zum Beispiel Mutationsraten von eins zu einer Million ein, so ist der Effekt nachweislich vernachlässigbar klein und man könnte sich die Mutation ganz sparen.

5.6.1 Diskussion der Ergebnisse

Die Interpretation der Ergebnisse für den Einflussfaktor Mutation ergibt etwas überraschende Erkenntnisse. Tabelle 5.7 gibt einen ersten Überblick.

Wie an der Spalte mit den Fehlerwahrscheinlichkeiten der Hypothesentests zu erkennen ist, wird nur für eine Populationsgröße von 5000 die Annahme auf Ungleichheit der Distributionen für die Mutationsraten von 0,001 und 0,01 verworfen. In allen anderen Populationsgrößen hat die höhere Mu-

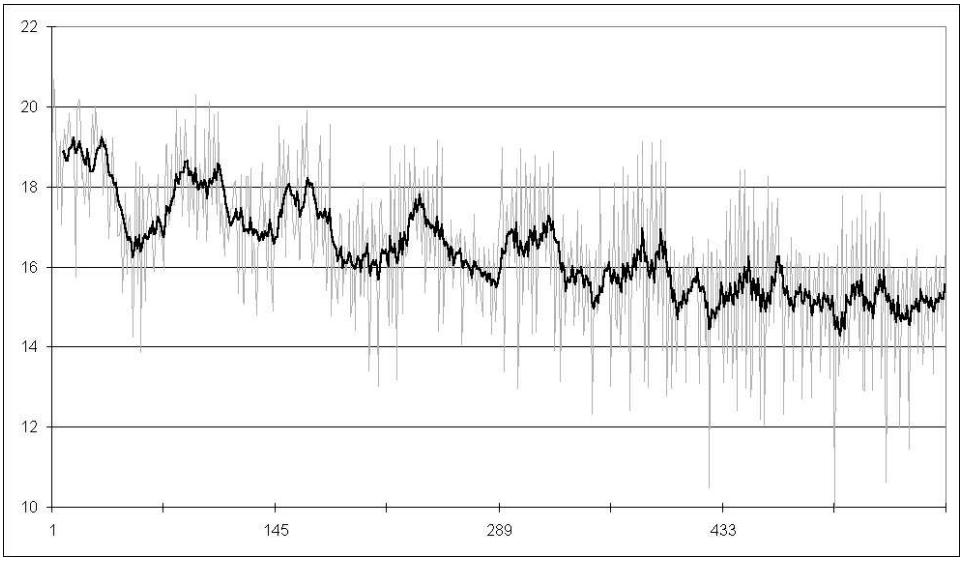

Abbildung 5.9: Energiewerte: Populationsgröße - 500:1000:2000:5000 (144); Ersetzungsgröße 0,5:0,9 (72); Mutationsrate - 0,001:0,01 (36). Ergebnisse aus Basisintersuchung 2.

tation einen belegbaren (positiven) Einfluss auf den Objective Score. Dieses Ergebnis soll jedoch anhand zweier weiterer Darstellungen relativiert werden.

Abb. 5.9 zeigt die Mittelwerte der verschiedenen Parameterkombinationen für steigende Populationsgrößen und verschiedene Ersetzungsgrößen, in Tabelle 5.8 werden die wichtigsten Werte noch einmal zusammengefasst.

Deutlich zu erkennen ist, dass zusätzlich zur Populationsgröße auch die Ersetzungsrate einen leichten Einfluss auf die Effektivität der Mutationsrate hat. Die Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit ist bei einer Ersetzungsgröße von 0,5 stets niedriger als bei 0,9. Bei einer Populationsgröße von 2000 wird die Hypothese auf Ungleichheit bei der Ersetzungsgröße von 0,9 sogar verworfen, während sie für 0,5 noch angenommen wird. Zusammen mit der Abb. 5.9 zeigt sich hiermit, dass Mutation mit der Zunahme der Populations- und Ersetzungsgröße eine immer weniger wichtige Rolle spielt. Es ist zu erwarten, dass bei noch größeren Populationen der Einfluss der Mutation nur noch marginal wird.

Die Folgerungen aus diesem Verhalten sind eine Bestätigung sowohl der Annahme von Holland und Goldberg, dass Mutation ein “background” Operator ist, als auch der Behauptung von Rechenberg und anderen Forschern,

Popgr	pRepl	pMut	N	∅ En.	σ	Min	Max	nEvals	|Z|
500	0,5	0.001	36	18.89	1.00	15.74	20.68	82320	0,0001
		0.01	36	16.96	1.20	13.85	19.23	196701	
	0,9	0.001	36	18.22	1.12	15.97	20.32	105728	0,0001
		0.01	36	16.94	1.08	14.81	18.61	246385	
1000	0,5	0,001	36	17,66	1,21	15,44	19,90	166074	0,0001
		0,01	36	16,15	1,22	13,03	18,03	394738	
	0,9	0,001	36	16,97	1,55	13,21	19,18	219717	0,0015
		0,01	36	15,92	0,83	14,06	17,34	506737	
2000	0,5	0,001	36	16,83	1,58	12,99	18,98	353114	0,0006
		0,01	36	15,65	1,18	12,33	17,97	783187	
	0,9	0,001	36	16,13	1,96	12,43	19,18	445931	0,0819
		0,01	36	15,16	1,41	10,49	16,75	1059029	
5000	0,5	0,001	36	15,53	1,97	12,04	18,43	990354	0,2975
		0,01	36	15,17	1,21	12,32	16,75	2021133	
	0,9	0,001	36	15,14	1,94	9,97	17,84	1226423	0,8570
		0,01	36	15,07	1,19	11,47	16,44	2775488	

Tabelle 5.8: Vergleich der Mutationeinflüsse für verschiedene Populations- und Ersetzungsgrößen mit dem Steady State Algorithmus.

Popgr:	Populationsgröße	*pMut:*	Mutationswahrscheinlichkeit
N:	Anzahl Beobachtungen	*∅ En.:*	Mittelwert
Min:	Minimaler Mittelwert	*Max:*	Maximaler Mittelwert
pRepl:	Ersetzungsgröße der Population		
σ*:*	Standardabweichung des Mittelwerts		
nEvals:	Durchschnittliche Anzahl benötigter Evaluationen		
|Z|:	Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der beiden Distributionen		

demnach die Mutation ein Hauptfaktor der Evolution ist. Wie sich zeigt, gewinnt Mutation vor allem in kleinen Populationen großen Einfluss. Der zusätzliche explorative Anteil gibt somit kleinen Gruppen die Gelegenheit, der genetischen Drift entgegenzuwirken und sich nicht in kleinen, lokalen Optima zu verfangen.

Dies kann zum Beispiel für kleine Gruppen von Auswanderern wichtig sein, welche auf absehbare Zeit kaum oder keinen Kontakt zu Individuen derselben Spezies haben werden. Bei großen Populationen wird hingegen die Mutation zu einem nebensächlichen Faktor. Hier greifen andere Mechanismen der Evolution stärker – zum Beispiel das Schema-Theorem oder die Building-Block-Hypothese – ohne jedoch den Einfluss der Mutation ganz zu verdrängen.

5.7 Crossover

Der Crossover-Operator ist neben der Selektion als einer der entscheidenden Faktoren für den Erfolg der Genetischen Algorithmen anzusehen. Die erarbeiteten Erweiterungen in der Implementation von GAen für diese Diplomarbeit haben einige Änderungen nötig gemacht, deren Hauptmerkmal die Erhöhung der möglichen Elternanzahl pro Kindgenom ist.

Allgemeine Funktionsweise

Als Erweiterung des “natürlichen” Crossovers mit zwei Eltern arbeitet der Crossover mit beliebig vielen Eltern, die in einer zyklischen Liste abgelegt werden. Während bei zwei Eltern jeweils immer nur bei Kreuzungspunkten zwischen den beiden elterlichen Genomen gekreuzt werden kann, wird hier bei einer Kreuzung das jeweils nächste Genom aus der zyklischen Liste als elterliches Genom angefordert. Auf diese Weise wird immer ein Kindgenom von zwei oder mehr Eltern erschaffen. Ist die Anzahl der Bruchstellen kleiner als die zur Verfügung stehenden Anzahl der Eltern, so werden einige Eltern ihre Gene nicht weitervererben, ist die Anzahl der Bruchstellen größer, so vererben einige Eltern mehrere voneinander räumlich getrennte Gensequenzen.

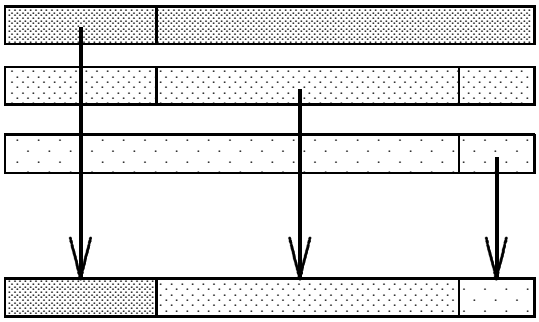

Abbildung 5.10: N-Point Crossover bei 3 Eltern und 2 Bruchstellen

Multiple Kreuzungspunkte

Die Frage nach der Anzahl und der Verteilung der Kreuzungspunkte bei der Genrekombination wird in der GA-Forschung immer wieder neu aufgeworfen. Beasley, Bull und Martin nennen in [10] als Beispiel Untersuchungen von Eshelman et al. (1989) und von Spears & de Jong (1991). Während die erste Untersuchung keine signifikanten Unterschiede zwischen N-Point (N=1, 2 und mehr) und Uniform Crossover konstatiert, demonstriert letztere anhand theoretischer Analysen die Überlegenheit von 1-Punkt gegenüber 2-Punkt und Uniform Crossover. Allerdings wird dort von Spears und De Jong die Einschränkung gemacht, das dies nicht für kleine Populationen gelte. In diesem Fall spekulieren sie, dass "der disruptive Effekt von Uniform und N-Punkt (N>>2) Crossover bei kleineren Populationen die beschränkte Informationskapazität durch mehr Homogenität unterstützt" [60].

5.7.1 N-Point

Der N-Point Crossover-Operator ist *der* Standardoperator in den Genetischen Algorithmen. Das bedeutet jedoch nicht, dass er in seiner Grundform auch der effizienteste ist.

Dieser Operator arbeitet mit einer einstellbaren Anzahl n von Bruchstellen zwischen 1 und (maximal) der Länge l des Kindgenoms G_K. Es werden dann per Zufall n voneinander verschiedene Sollbruchstellen innerhalb der Länge des Genoms ermittelt. An jeder Bruchstelle wird auf das Genom des jeweils nächsten Elternteils übergesprungen, um dessen Gene an das Kindgenom weiterzugeben.

Als einfachste Form dieses Operators ist der 1-Point Crossover in der

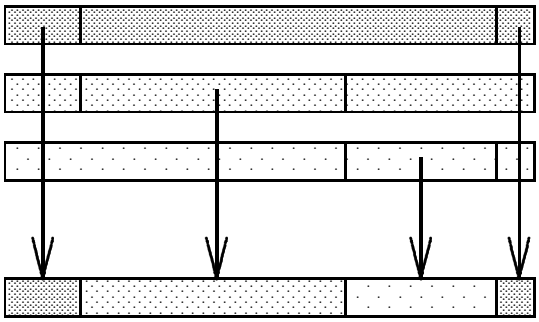

Abbildung 5.11: Randomwalk Crossover mit 3 Eltern. Bei einer angenommenen Genomlänge von 240 Bit und einer Crossoverwahrscheinlichkeit von 1% kämen rein rechnerisch im Durchschnitt 2,4 Kreuzungspunkte pro Genom. In diesem Beispiel sind zufällig 3 Kreuzungspunkte.

Literatur schon eingehend untersucht worden. Ein Ergebnis dieser Untersuchungen ist die Bestätigung der von Holland und Goldberg aufgestellten Theorie, dass der 1-Point Crossover Problemstellungen mit eng gekoppelten Building-Blocks bevorzugt. In solchen Fällen wurde durch Goldberg et al. 1991 empirisch nachgewiesen, dass Genetische Algorithmen Testprobleme mit diesem Kreuzungsoperator in einer Zeit proportional zu $l \cdot \log(N)$ gelöst haben, mit l der Länge des Genoms [24][2].

5.7.2 Randomwalk

Der Randomwalk ist in seiner Funktionsweise dem N-Point Crossover nicht unähnlich. Jedoch wird hier anstatt einer festen Anzahl an Bruchstellen eine relative Wahrscheinlichkeit für einen Bruch angegeben. Für jedes Genom G_{E_i} eines Elternteiles existiert die Wahrscheinlichkeit p_{E_i}, die für jedes Bit im Genom die Bruchwahrscheinlichkeit angibt.

Beginnend mit dem ersten Bit des ersten Elternteiles wird für jedes Bit des gerade aktiven Elternteils E_i per Los entschieden, ob das Bit in das Kindgenom G_K übernommen wird oder ob ein Bruch passiert und das Bit aus dem Genom des jeweils nächsten Elternteils stammen soll.

Randomwalk kann so konfiguriert werden, dass es sich fast wie ein N-Point verhält. Ein Nachteil bei Randomwalk ist jedoch der zusätzliche, nicht-

[2]Anmerkung: Wird die Annahme getroffen, die DNA sei – wie bei einigen Bakterien – ringförmig aufgebaut, so ist ersichtlich, dass der 1-Point Crossover eine Sonderform des 2-Point Crossovers ist, bei welcher der zweite Punkt immer am Ende bzw. Anfang des Genoms liegt.

deterministische Anteil in der Anzahl der Crossover Punkte. Soll die Anzahl klein gehalten werden und ist die genaue Zahl wichtig, so empfiehlt sich eher N-Point. Bei vielen Kreuzungspunkten allerdings werden ein oder zwei Punkte mehr oder weniger nicht so viel schaden, wofür der Randomwalk geeignet ist.

5.7.3 Uniform

Auch beim Uniform Crossover wird mit relativen Wahrscheinlichkeiten p_{E_i} für jedes Genom G_{E_i} gearbeitet und nicht mit einer festen Anzahl an Bruchstellen. Es muss gelten, dass die Summe aller Wahrscheinlichkeiten den Wert 1 annimmt: $\sum_{i=1}^{n} p_{E_i} = 1$. Anstatt jedoch die Bruchwahrscheinlichkeiten zu definieren, wird damit die Übertrittswahrscheinlichkeit jedes einzelnen Gens des Elternteil E_i in das Kindgenom G_K angegeben. Bei zwei Eltern z.B. und einer Übertrittswahrscheinlichkeit von jeweils 0,5 pro elterliches Genom, werden im Kindgenom ca. 50% der Gene vom ersten Elternteil und ca. 50% der Gene vom zweiten Elternteil stammen, allerdings nicht am Stück, sondern bunt gemischt.

Wie sofort ersichtlich, ist dieses Verfahren in der Funktionsweise potentiell viel destruktiver gegenüber Building-Blocks im Genom als der N-Point und der Randomwalk Crossover. Die Wahrscheinlichkeit, dass ein Teilstück eines Genoms an einem Stück vererbt wird, sinkt bei wachsender Länge des Stückes viel schneller als bei N-Point oder Randomwalk.

5.7.4 Diskussion der Ergebnisse

Wie in Abschnitt 3.4.7 bereits festgestellt, wird der Crossover-Operator sowohl durch disruptive als auch durch konstruktive Effekte charakterisiert. Diese der Exploration – respektive Exploitation – dienenden Mechanismen sind sowohl in der ersten als auch in der zweiten Basisuntersuchung eingehend analysiert worden.

Verhalten von Crossover-Operatoren bei kleinen Populationen

Wie in den Abschnitten 5.3 und 5.6 bereits aufgezeigt wurde, sind kleine Populationsgrößen in Hinsicht auf Exploration stärker auf disruptive Aspekte von GA Operatoren angewiesen als große Populationen. In der ersten Basisuntersuchung wurden aus diesem Grund auch die beiden völlig entgegengesetzten Ansätze des minimalen und maximalen Crossovers untersucht. Als

pMut	XOvTyp	N	∅ En.	σ	Min	Max	nEvals	|Z|
0,001	N-Point	504	28,73	3,04	21,07	40,55	15398	0,0001
	RndWlk	504	27,83	3,64	19,99	39,85	15162	
	Uniform	252	27,86	3,10	22,16	38,17	14442	
0,01	N-Point	504	21,95	1,92	16,01	28,80	37945	0,0151
	RndWlk	504	21,86	1,81	16,97	28,76	38170	
	Uniform	252	22,23	1,89	15,99	28,50	34511	

Tabelle 5.9: Vergleich von Crossover-Operatoren bei verschiedenen Mutationsraten. Populationsgröße 100. Werte stammen aus dem Steady State Algorithmus, sind jedoch für den Simple GA repräsentativ.

pMut: Mutationswahrscheinlichkeit;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
XOvTyp: Crossover-Typ (RndWlk=Randomwalk);
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der Distributionen;

Vertreter des minimalen Ansatzes kann dabei der N-Point Crossover mit 1 bis 5 Kreuzungspunkten angesehen werden. Für den maximalen Ansatz wurde der Uniform Crossover mit bis zu 240 (Genomlänge in Bit) Kreuzungspunkten gewählt. Für eine mittlere Anzahl an Kreuzungspunkten wurde der Randomwalk mit ca. 10 und 20 Punkten eingesetzt.

Tabelle 5.9 gibt einen ersten Überblick über die Ergebnisse mit einer Populationsgröße von 100 Individuen. Wie der Spalte mit den Fehlerwahrscheinlichkeiten zu entnehmen ist, sind bei dieser Betrachtungsweise die Ergebnisdistributionen aller Operatoren in ihrer Leistungsfähigkeit nicht sehr weit voneinander entfernt. Bei der schwachen Mutationsrate von 0,001 unterscheiden sich zwar die Verteilungen so deutlich, dass Randomwalk und Uniform eindeutig besser sind als N-Point, doch bei einer höheren Mutationsrate muss die Hypothese auf unterschiedliche Distributionen – wenn auch nur knapp – abgelehnt werden.

Diese Werte lassen jedoch die unterschiedliche Anzahl von Kreuzungspunkten – also den disruptiven Effekt – in der Untersuchung vollkommen unberücksichtigt. In Tabelle 5.10 erfolgt deshalb eine weitere Differenzierung der Crossover-Operatoren.

pMut	XOvTyp	# XOvP	N	∅ En.	σ	Min	Max	nEvals
0,001	N-Point	1	126	30,45	2,69	24,89	37,62	17159
		2	126	29,09	2,82	23,88	40,55	15581
		3	126	28,19	2,88	21,97	39,30	14836
		5	126	27,18	2,83	21,07	37,71	14013
	RndWlk	≈ 2-48	252	29,56	3,08	22,38	38,46	16836
		≈ 10	126	26,01	3,28	19,99	36,06	13525
		≈ 20	126	26,16	3,39	20,83	39,85	13451
	Uniform	≈ 120	189	28,14	3,04	22,16	38,17	14598
		≈ 180	63	27,00	3,11	22,68	37,30	13974
0,01	N-Point	1	126	22,75	2,12	16,01	28,80	41444
		2	126	21,98	1,87	18,75	26,82	38773
		3	126	21,74	1,73	18,07	27,36	36584
		5	126	21,34	1,67	17,81	28,42	34981
	RndWlk	≈ 2-48	252	22,29	1,89	17,86	28,78	41635
		≈ 10	126	21,41	1,66	16,97	26,21	34920
		≈ 20	126	21,40	1,60	17,56	27,63	34289
	Uniform	≈ 120	189	22,26	1,84	15,99	28,50	35233
		≈ 180	63	22,11	2,05	18,18	27,47	32346

Tabelle 5.10: Vergleich von Crossover-Operatoren unter Berücksichtigung der Zahl von Kreuzungspunkten bei verschiedenen Mutationsraten. Steady State Algorithmus mit kleinen Populationen (100 Individuen), ist für den Simple GA repräsentativ.

pMut: Mutationswahrscheinlichkeit; *# XOvP:* Anzahl Kreuzungspunkte;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
XOvTyp: Crossover-Typ (RndWlk=Randomwalk);
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der Distributionen;

Hier zeigt sich ein deutlicher Einfluss der Anzahl der Kreuzungspunkte. Der N-Point Crossover mit nur 1 Punkt ist allgemein bei Molekülstrukturoptimierung mit kleinen Populationen immer schlechter als mit mehreren Kreuzungspunkten. Interessanterweise sind die Distributionen für N-Point Crossover bei 1, 2, 3 und 5 Kreuzungspunkten alle verschieden, bei steigender Anzahl an Kreuzungspunkten sinkt der durchschnittlich gefundene Energiewert. Dies ist bei einer Mutationsrate von 0,001 sehr deutlich ausgeprägt, bei einer Rate von 0,01 ist es immerhin noch quantifizierbar. Dies erhärtet die Aussage, dass bei kleinen Populationsgrößen stärkere Disruption erwünscht ist.

Ein weiteres Indiz für die Richtigkeit dieser Aussage geben die Werte für Randomwalk mit einer durchschnittlichen Anzahl von 10 bzw. 20 Kreuzungspunkten. Im Ergebnis sind die durchschnittlich gefundenen Energiewerte denen des N-Point Crossovers entweder überlegen (bei einer Mutationsrate von 0,001) oder zumindest gleichwertig (bei einer Mutationsrate von 0,01).

Wie bei den Uniform-Operatoren dargestellt, bewirkt dann aber eine zu starke Disruption wieder einen Abfall der Effizienz im Verhalten der Genetischen Algorithmen. Die Erklärung hierfür liegt in der Tatsache, dass Building-Blocks bei diesem Operator kaum jemals komplett vererbt werden. Dadurch wird die Rekombinationshäufigkeit dieser Blöcke drastisch herabgesetzt und somit die Leistung des GA gemindert. Dass trotzdem noch "akzeptable" Ergebnisse erzielt werden, spricht für die These, dass die Building-Block-Hypothese allein noch nicht den Erfolg von GAen erklären kann.

Als bemerkenswerte Tatsache am Rande sei noch auf die Anzahl der benötigten Evaluationen hingewiesen. Beim N-Point Crossover sinkt die Zahl der Evaluationen beständig mit der Zunahme der Kreuzungspunkte bzw. der Verminderung des durchschnittlich gefundenen Energiewertes. Wird dies unter Berücksichtigung der Werte für Random Crossover extrapoliert, so kann die Hypothese aufgestellt werden, dass es eine feste Anzahl an Kreuzungspunkten geben muss, bei welcher die durchschnittlichen Energiewerte nicht mehr weiter verbessert werden, aber die Zahl der Evaluationen bei mehr Kreuzungspunkten trotzdem auf demselben Niveau konstant bleibt. Bei der Mutationsrate von 0,01 muss diese Zahl an Kreuzungspunkten zwischen 4 und 5 liegen, da die Anzahl der Evaluationen von N-Point 3 nach N-Point 5 sinkt, aber dann für Randomwalk 10 und Randomwalk 20 konstant geblieben ist. Auch der Uniform Crossover mit durchschnittlich 120 Kreuzungspunkten

XOvTyp	# XOvP	N	∅ En.	σ	Min	Max	nEvals	|Z|
N-Point	3	192	15,27	1,90	9,97	20,14	713506	0,0001
	5	192	16,28	1,28	12,90	20,68	715933	
RndWlk	≈10	192	17,65	1,20	15,38	20,32	740508	

Tabelle 5.11: Vergleich von Crossover-Operatoren mit unterschiedlicher Anzahl an Kreuzungspunkten für Populationsgrößen 500-5000 mit dem Steady State Algorithmus.

*XOvTyp:*Crossover-Typ (RndWlk=Randomwalk);
*# XOvP:*Anzahl Kreuzungspunkte;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
|Z|: Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der Distributionen;

braucht ungefähr genauso viele Evaluationen, wenn auch mit einem schlechteren Ergebnis. Bei Uniform Crossover mit ca. 180 Punkten sinkt dann die Zahl der Evaluationen noch einmal bei vergleichbaren Werten in den gefundenen Energieminima.

Verhalten von Crossover-Operatoren bei steigenden und großen Populationen

Als Fortführung der oben gewonnenen Erkenntnisse wurde das Verhalten von Crossover-Operatoren bei steigenden und großen Populationszahlen untersucht. Als Ansatzpunkt diente dabei die Vermutung, dass bei steigender Populationsgröße geringere Disruptionseffekte benötigt würden.

Aus diesem Grund wurden die Crossover-Operatoren N-Point 3 und N-Point 5 als Vertreter geringer Disruption und Randomwalk 10 als Vertreter mittlerer Disruption untersucht. Tabelle 5.11 zeigt eine erste Zusammenfassung der ermittelten Ergebnisse über alle Populationsgrößen hinweg. Es zeigt sich, dass hier genau eine Umkehrung des bei kleinen Populationen beobachteten Trends vorliegt. Je mehr Kreuzungspunkte vorhanden sind, desto schlechter ist das durchschnittlich gefundene Energieniveau.

Eine genauere Unterteilung in verschiedene Populationsgrößen (Tabelle 5.12) klärt die Zusammenhänge. Deutlich zu erkennen ist, dass bei steigen-

Popgr	XOvTyp	# XOvP	N	∅ En.	σ	Min	Max	nEvals	$\lvert Z_{35}\rvert$	$\lvert Z_{All}\rvert$
500	N-Point	3	48	17,16	1,40	13,85	20,14	158318	0,1499	0,0001
		5	48	17,61	1,28	14,25	20,68	155732		
	RndWlk	≈ 10	48	18,49	1,13	16,06	20,32	159301		
1000	N-Point	3	48	15,88	1,29	13,03	18,27	313979	0,0793	0,0001
		5	48	16,32	1,00	14,33	18,70	318188		
	RndWlk	≈ 10	48	17,83	1,09	16,08	19,90	333282		
2000	N-Point	3	48	14,52	1,44	10,49	17,46	645839	0,0001	0,0001
		5	48	15,84	0,79	14,06	17,90	654687		
	RndWlk	≈ 10	48	17,46	1,13	15,38	19,18	680420		
5000	N-Point	3	48	13,53	1,15	9,97	15,19	1753890	0,0001	0,0001
		5	48	15,34	0,71	12,90	16,47	1735126		
	RndWlk	≈ 10	48	16,81	0,78	15,75	18,43	1789033		

Tabelle 5.12: Vergleich von Crossover-Operatoren mit unterschiedlicher Anzahl an Kreuzungspunkten für Populationsgrößen 500-5000 mit dem Steady State Algorithmus.

Popgr: Populationsgröße;
XOvTyp: Kreuzungsart; *# XOvP:* Anzahl Kreuzungspunkte;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
XOvTyp: Crossover-Typ (RndWlk=Randomwalk);
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z_{35}|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der N-Point 3 und N-Point 5 Distributionen;
$|Z_{All}|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der 2 N-Point und der Randomwalk Distributionen;

den Populationsgrößen die größere Disruption durch mehr Kreuzungspunkte einen eindeutig negativen Effekt hat. Die Randomwalk Distribution mit durchschnittlich 10 Kreuzungspunkten erzielt hier immer schlechtere Ergebnisse als die beiden N-Point Distributionen. Die Hypothese, dass die N-Point 3 und N-Point 5 Distributionen für Populationsgrößen von 500 und 1000 ungleich sind, kann aufgrund zu großer Fehlerwahrscheinlichkeit nicht bestätigt werden. Jedoch wird der bei den Populationsgrößen 500 und 1000 sichtbare Trend, dass N-Point 3 besser ist als N-Point 5, bei den Populationsgrößen 2000 und 5000 durch entsprechend geringe Fehlerwahrscheinlichkeiten bei der Überprüfung der Ungleichheitshypothese der Distributionen vollständig bestätigt.

Einer weiteren Aufteilung unter Einbeziehung der Mutationsrate bestätigt sowohl Ergebnisse aus der Beurteilung der Mutationsrate allein (siehe Abschnitt 5.6), als auch Ergebnisse aus der Analyse der Crossover-Operatoren bei einer Populationsgröße von 100 Individuen (Tabelle 5.10). Die in Tabelle 5.13 aufgelisteten Werte weisen die kombinierten Merkmale beider Einflussfaktoren auf: bis zu einer Populationsgröße von 1000 einschließlich liegt die Randomwalk Distribution mit ca. 10 Kreuzungspunkten immer nachweisbar schlechter als die beiden N-Point Distributionen, in beiden Mutationsmodi. Dem entgegen können die Distributionen von N-Point 3 und N-Point 5 sowohl bei einer Mutationsrate von 0,001 als auch bei einer Rate von 0,01 nicht als unterschiedlich angesehen werden. Ab einer Populationsgröße von 2000 Individuen sind alle Distributionen untereinander verschieden. Der in Abschnitt 5.6 schon nachgewiesene schwindende Einfluss der Mutationsrate bei steigenden Populationsgrößen und die relativ geringe Anzahl an Werten – je 24 aus jeweils 8 gemittelten Tests – machen genauere Angaben über die Interaktion von Mutationsrate und Anzahl der Kreuzungspunkte allerdings nicht mehr möglich.

5.8 Selektionsschemata

Den Schemata für die Selektion von Eltern zur Erzeugung einer Nachfolgegeneration ist die Möglichkeit eröffnet worden, entsprechend den Vorgaben der Elternanzahl zu operieren. Die Wahrscheinlichkeit, dass ein bestimmtes Individuum als Elternteil selektiert wird, ist durch den skalierten Zielwert - dem Fitnesswert - oder dem Rang eines Individuums in einer Population

pMut	Popgr	XOvTyp	# XOvP	N	∅ En.	σ	Min	Max	nEvals	$\lvert Z_{35}\rvert$	$\lvert Z_{35R}\rvert$
0,001	500	N-Point	3	24	17,83	1,06	15,74	20,14	97222	0,0851	0,0001
			5	24	18,41	1,01	17,03	20,68	92885		
		RndWlk	10	24	19,43	0,52	18,37	20,32	91965		
	1000	N-Point	3	24	16,46	1,30	13,21	18,27	187147	0,4037	0,0001
			5	24	16,77	0,97	14,54	18,70	187244		
		RndWlk	10	24	18,72	0,73	16,88	19,90	204295		
	2000	N-Point	3	24	14,91	1,50	12,43	17,46	395466	0,0059	0,0001
			5	24	16,09	0,83	14,21	17,90	389146		
		RndWlk	10	24	18,45	0,53	17,38	19,18	413954		
	5000	N-Point	3	24	13,31	1,27	9,97	15,17	1068098	0,0001	0,0001
			5	24	15,21	0,91	12,90	16,47	1068586		
		RndWlk	10	24	17,48	0,51	16,52	18,43	1188482		
0,01	500	N-Point	3	24	16,49	1,39	13,85	19,23	219414	0,2611	0,0041
			5	24	16,82	1,00	14,25	18,30	218578		
		RndWlk	10	24	17,54	0,67	16,03	18,51	226636		
	1000	N-Point	3	24	15,30	0,99	13,03	17,18	440811	0,0814	0,0001
			5	24	15,86	0,82	14,33	17,43	449131		
		RndWlk	10	24	16,94	0,52	16,08	18,04	462269		
	2000	N-Point	3	24	14,13	1,28	10,49	15,93	896212	0,0001	0,0001
			5	24	15,60	0,68	14,06	16,64	920227		
		RndWlk	10	24	16,48	0,54	15,38	17,97	946885		
	5000	N-Point	3	24	13,75	1,00	11,47	15,19	2403681	0,0001	0,0001
			5	24	15,47	0,39	14,88	16,35	2401668		
		RndWlk	10	24	16,14	0,24	15,75	16,75	2389583		

Tabelle 5.13: Vergleich von Crossover-Operatoren mit unterschiedlicher Anzahl an Kreuzungspunkten für Populationsgrößen 500-5000 mit dem Steady State Algorithmus unter Berücksichtigung verschiedener Mutationsraten.

pMut: Mutationrate; *Popgr:* Populationsgröße; *XOvTyp:* Kreuzungsart;
 # XOvP: Anzahl Kreuzungspunkte; *N:* Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
XOvTyp: Crossover-Typ (RndWlk=Randomwalk);
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z_{35}|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der N-Point 3 und N-Point 5 Distributionen;
$|Z_{35R}|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der 2 N-Point und der Randomwalk Distributionen;

gegeben. Aus der Gesamtheit aller individuellen Wahrscheinlichkeiten ermitteln die spezifischen Selektionsschemata die Eltern der nächsten Generation. Die vier wichtigsten in der Literatur beschriebenen Verfahren wurden auf ihre Eignung für die gesetzte Aufgabe untersucht und implementiert.

5.8.1 Roulette Multi-Spin

Das RMS Verfahren ist an das aus Glücksspielen bekannte Rouletterad angelehnt. Jedes Individuum bekommt als Slot einen dem relativen Prozentsatz seines Fitnesswertes entsprechend breiten Anteil an einem virtuellen Rouletterad. Für jedes zu selektierende Elternteil wird das Rad gedreht. Dasjenige Individuum wird als Elternteil selektiert, in dessen Anteil das Rad zum Stehen kommt.

Dieses Selektionsschema nach Goldberg ist äußerst sensibel gegenüber Superindividuen in einer ansonsten weniger fitten Population. Es kann bei Problemstellungen mit exponentiellen Fitnessfunktionen durchaus passieren, dass in einer Generation ein Individuum fast 100% der Reproduktionswahrscheinlichkeit auf sich konzentriert und so die Nachfolgegeneration mit wenig mutierten Kopien seiner selbst auffüllt. Bei dieser extremsten Form des beschleunigten Takeover ist die Population innerhalb kürzester Zeit natürlich fast komplett konvergiert und die Ergebnisse des Laufes wenn nicht nutzlos, so doch von minderem Wert.

Eine weitere unerwünschte Eigenschaft besteht in der nicht vorhandenen Translationsinvarianz dieses speziellen Selektionsschemas [40]. Das bedeutet, eine Addition oder Subtraktion eines genügend großen Wertes in der Zielwertfunktion reicht aus, um die Fortpflanzungswahrscheinlichkeit eines jeden Individuums innerhalb einer Population gegenüber einer Bewertung ohne diese Translation dramatisch zu ändern. Daraus resultiert ein ungewollter Nicht-Determinismus in der Bewertung der relativen Fitness eines Individuums gegenüber der Gesamtpopulation.

Ein sehr interessanter Kommentar zu diesem Schema stammt aus dem ausführlichen Schemataverglеich von Blickle & Thiele (1995) [12]:

> "All the undesired properties led us to the conclusion that proportional selection is a very unsuited selection scheme. Informally one can say that the only advantage of proportional selection is that it is so difficult to *prove* the disadvantages."

5.8.2 Roulette Single-Spin

Dieses Verfahren nach J.E. Baker – welches er 'stochastic universal sampling' nannte – arbeitet ähnlich wie das oben beschriebene Roulette Multi-Spin Verfahren. Auch hier werden Anteile an einem Rouletterad entsprechend den Fitnesswerten vergeben. Dann wird das Rad einmal gedreht und das erste Individuum als Elternteil selektiert. Anstatt jedoch für jede weitere Selektion das Rad zu drehen, wird entsprechend der Elternanzahl – und ausgehend von der Position des Rades nach der ersten Drehung – das Rad in gleichmäßig kleine Abschnitte unterteilt. Jede Abschnittsgrenze definiert je eine Selektion für die Reproduktion.

Das RSS-Verfahren garantiert im Gegensatz zu der RMS-Methode, dass Eltern mit einem relativ hohen Anteil an der Gesamtfitness der Population nicht mehr – aber auch nicht weniger – selektiert werden, als es ihrem Prozentsatz entspricht. Das RSS-Verfahren berücksichtigt also weniger fitte Individuen besser als Roulette Multi-Spin. Ansonsten gelten jedoch alle zu RMS gemachten Aussagen und Einschränkungen auch für diese Selektionsschema.

5.8.3 Tournament

Das Tournamentverfahren ist ein rangbasiertes Selektionsverfahren und wählt zufällig k Individuen aus der Gesamtpopulation. Diese werden anhand ihres Fitnesswertes verglichen. Das Individuum mit der besseren Fitness wird selektiert, die anderen $k-1$ gehen leer aus. Alle anderen k Individuen werden anschließend aus dem Turnier entfernt. Dies wird solange wiederholt, bis entweder die festgelegte Anzahl an selektierten Eltern erreicht ist oder die Turniermenge leer ist, d.h. dieses Selektionsschema veranstaltet n Turniere um n Individuen zu selektieren. Sollten keine Individuen mehr im Turnier vorhanden sein und noch Eltern selektiert werden müssen, werden alle Individuen wieder hineingenommen.

Dieser Selektor reagiert also mehr auf die relative Fitness von Individuen untereinander als auf die absolute Fitness. Für steigende k ergibt sich ein steigender Selektionsdruck, da die Wahrscheinlichkeit, dass ein gutes Individuum im jeweiligen Turnier vorhanden ist, ebenfalls steigt. Deshalb ist die meistbenutzte Form des Turniers das Duell mit $k = 2$ Individuen. Wie in [60] dargestellt wird, implementiert dieser Selektor somit eine verrauschte Form des Ranking-Selektionsschemas.

5.8.4 Uniform

Dieser Selektor stellt eine Ausnahme dar: als einziger berücksichtigt dieser die Fitness der Individuen nicht. Alle Individuen einer Population werden der Reihe nach selektiert. Ist der Selektor am Ende einer Population angelangt und müssen weitere Eltern ausgewählt werden, wird wieder von vorne angefangen. Der Selektor stellt kein wirklich rangbasiertes Verfahren dar, er steht gewissermaßen als Gruppe für sich.

In der Implementierung der Bibliothek für Genetische Algorithmen arbeitet dieser Algorithmus mit nach Fitness sortierten Populationen. Deshalb werden Individuen mit besseren Fitnesswerten etwas begünstigt wenn die Größe der Population kein ganzer Teiler der Anzahl der zu selektierenden Eltern ist. Bei sehr kleinen Ersetzungsgrößen führt dies allerdings zu einem zusätzlichen, sehr harten Selektionsdruck: bei einer Ersetzungsgröße von 0,1 und 2 Eltern werden zum Beispiel immer nur die besten 20% der Individuen als Eltern selektiert. Dies führt zu einem konstant harten, zusätzlichen Selektionsdruck und im Normalfall zu einer schlechten Effektivität einer solchen Kombination.

5.8.5 Diskussion der Ergebnisse

Die Werte im Vergleich der Selektionsschemata bekräftigen die schon in Abschnitt 5.1 gewonnenen Erkenntnisse, dass die proportionalen Selektionsschemata RMS und RSS im Schnitt deutlich schlechter abschneiden als die beiden rangbasierten Schemata. Tabelle 5.14 fasst die Ergebnisse zusammen.

Es zeigt sich sehr deutlich die nachteilige Verfahrensweise von proportionalen Selektoren, demgegenüber das Tournament Verfahren bei allen Ersetzungsgrößen besser arbeitet. Etwas überraschend mag auf den ersten Blick das schlechte Abschneiden des Uniform Selektors bei einer Ersetzungsgröße von 10% erscheinen. Es erklärt sich jedoch durch die in der Arbeit benutzte Implementierungsweise dieses Operators: bei sehr kleinen Ersetzungsgrößen – wie 0,1 – tritt der in Abschnitt 5.8.4 schon beschriebene Effekt des zusätzlichen Selektionsdrucks auf. Die führt zu einer schnelleren, allerdings verfrühten Konvergenz der Allele innerhalb der Population und somit zu einem im Durchschnitt schlechteren Ergebnis, wie der Tabelle zu entnehmen ist.

Insgesamt verfestigt dieser Untersuchungsabschnitt die Erkenntnis, dass sanfte elterliche Selektionsverfahren einen Vorteil für Genetische Algorithmen darstellen, wenn sie gut konfiguriert sind und die Individuenselektion

pRepl	Selekt	N	∅ En.	σ	Min	Max	nEvals	|Z|
0,1	rms	240	26.59	4.42	18.21	39.79	21898	0,0001
	rss	240	26.49	4.67	16.28	40.49	21889	
	trnmnt	240	25.63	3.66	18.62	34.83	21606	
	unifrm	120	28.87	4.63	21.77	39.07	13632	
0,5	rms	240	25.76	4.37	16.10	44.20	27136	0,0001
	rss	240	25.48	4.80	17.86	45.12	27370	
	trnmnt	240	24.84	3.46	16.30	33.52	26932	
	unifrm	120	23.59	3.11	18.07	30.69	30380	
0,9	rms	240	25.72	4.04	14.79	38.25	42419	0,0012
	rss	240	25.67	3.97	18.30	39.81	41400	
	trnmnt	240	25.09	3.60	16.96	33.20	42970	
	unifrm	120	24.37	2.29	18.04	34.55	44042	

Tabelle 5.14: Selektionsschemata nach Populationsersetzungsgrößen gruppiert. Werte aus der ersten Basisuntersuchung für Simple GA.

N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
pRepl: Populationsersetzungsgröße;
Selekt: Selektionsschema: rms=Roulette Multi Spin, rss=Roulette Single Spin;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
|*Z*|*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der beiden Distributionen;

für die Nachfolgeneration durch das Ersetzungsverfahren gewährleistet wird.

5.9 Skalierungsschemata

Skalierungsschemata werden dann bedeutsam, wenn als Selektionsschema eine wie auch immer geartete proportionale Selektion gewählt wird. Aufgrund einiger Publikationen von J.E. Baker (1985), D. Whitley (1989) und Frank Herrmann [30] wurden für diese Arbeit nur zwei Skalierungsschemata ausgewählt. Die Untersuchungen von Baker, Whitley und Herrmann zeigten eine Überlegenheit von Ranking gegenüber anderen Skalierungsmethoden und Selektionsschemata [9].

5.9.1 None

Dieses Skalierungsschema übernimmt direkt die Objective Score Werte ohne weitere rechnerische Zwischenschritte.

5.9.2 Ranking

Das Ranking weist jedem Individuum nur in Abhängigkeit von seinem Platz in der Objective Score Rangliste der Population eine Reproduktionschance zu; der effektive Wert bleibt dabei unbeachtet. Ein Rankingfaktor muss angegeben werden, welcher die Steigung der durch die Reproduktionschancen definierten Geraden angibt. Konkret gibt der Rankingfaktor den Multiplikationsfaktor der Reproduktionschance zwischen dem schlechtesten und dem besten Individuum an. Ein Rankingfaktor von 2 bedeutet also, dass bei einer proportionalen Selektion das beste Individuum doppelt so große Chance auf Selektion besitzt als das schlechteste Individuum der Population.

5.9.3 Diskussion der Ergebnisse

Die gewonnenen Resultate stehen in enger Übereinstimmung mit theoretischen Überlegungen aus der Selektion und den Ergebnissen von Baker, Whitley und Herrmann. Tabelle 5.15 gibt einen ersten Überblick über die Skalierungsschemata None und Rank bei den proportionalen Selektoren RMS und RSS.

Wird keine Skalierung gewählt (None), so werden bei RMS und RSS Selektion fast unweigerlich im Laufe einer GA-Rechnung überproportional gute Superindividuen entstehen, die anschließend einen Großteil der nächsten

Skal	Rankf	N	∅ En.	σ	Min	Max	nEvals	$\lvert Z_R \rvert$	$\lvert Z_{All} \rvert$
rank	2	240	23,73	3,55	16,97	34,87	29141	0,0001	0,0001
	5	240	24,25	3,39	16,01	35,30	26828		
	10	240	24,92	3,63	18,18	35,34	25738		
none		720	26,21	4,51	18,71	40,55	27424		

Tabelle 5.15: Vergleich von verschiedenen Skalierungsschemata mit den proportionalen Selektoren RMS und RSS für den Steady State Algorithmus mit 100 Individuen. Die Werte sind für alle Ersetzungsgrößen und Mutationsraten repräsentativ.

Skal: Skalierungschema;
Rankf: Rankingfaktor bei Skalierungsschema rank;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z_R|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der Ranking Distributionen;
$|Z_{All}|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der alle Skalierungsdistributionen;

Generation zeugen. Diese verfrühte Konvergenz zeigt sich in dem deutlich schlechteren Mittelwert für die Skalierungsart None gegenüber dem Ranking Verfahren. Das Ranking selbst erzielt diese relativ guten Werte durch die Tatsache, dass Superindividuen mit ihren Reproduktionschancen nicht komplett die nächste Generation bestimmen können. Im Grunde genommen handelt es sich nämlich bei der proportionalen Selektion mit Ranking Skalierung um eine leichte Abwandlung der Uniform Selektion mit fest verteilten Reproduktionschancen und einem zusätzlichen stochastischem Anteil. Für diese Tatsache spricht auch die Beobachtung, dass die durchschnittlichen Energiewerte für kleine Rankingfaktoren besser sind als für höhere Rankingfaktoren.

Somit stellt die Benutzung von Skalierungen bei proportionalen Selektoren eine Verbesserung dar, wenn dadurch Superindividuen verhindert werden. Allerdings sollte von komplexen Skalierungsschemata abgesehen werden, da diese immer spezifisch auf die zugrundeliegende Zielfunktion und deren erwartete bzw. tatsächliche Ergebniswerteverteilung zugeschnitten werden müssen. Ein solches Verfahren ist aus der Natur nicht bekannt und kann es dort auch nicht geben.

5.10 Die Minimierungsfensterfunktion

Die Fitness eines Individuums bestimmt die Fortpflanzungswahrscheinlichkeit dieses Individuums: je fitter ein Lebewesen in der Natur ist, desto wahrscheinlicher ist, dass es Nachkommen haben wird.

In einer Rechnersimulation mit Genetischen Algorithmen kann nach zwei verschiedenen Optima in einem Suchraum gesucht werden: nach einem Minimum oder einem Maximum, je nach der Definition der Problemstellung. Bei rangabhängigen Selektionsschemata stellt dies kein Probleme dar, da hier einfach die Sortierreihenfolge umgedreht werden kann. Bei proportionalen Selektoren kann die Fitness eines Individuums bei einer Maximumsuche direkt in eine entsprechend große Reproduktionswahrscheinlichkeit umgerechnet werden. Bei einer Minimumsuche mit proportionalen Selektoren müssen jedoch den niedrigen Fitnesswerten hohe Reproduktionswahrscheinlichkeiten zugewiesen werden. Dies kommt praktisch einer Invertierung der Fitness gleich.

Es gibt für diesen Sachverhalt keine entsprechenden Vorbilder aus der

Natur, da dort grundsätzlich alle Optimierungsprozesse der Evolution auf eine Maximierung der Nachkommenzahl hinauslaufen. Auch kann in ungünstigen Fällen durch die Fensterfunktion an sich ein Bias in den Ergebnissen erzeugt werden. Es ist deshalb sehr schwierig, eine gute Minimierungsfensterfunktion zu finden, welche für alle Problemstellungen gleichermaßen gut funktioniert.

5.10.1 Dynamic Reversed Fitness

(drf) Bei dieser Methode tauschen die Individuen einer Population ihre Fitnesswerte aus: das beste Individuum bekommt den Fitnesswert des schlechtesten und umgekehrt. Das zweitbeste ... etc.

$$\forall_i \quad i \leq \tfrac{s}{2} : \qquad \text{swap}(\text{fitness}_i, \text{fitness}_{s-i})$$

Dieser Algorithmus entkoppelt jedoch die Ziel- bzw. Fitnesswerte vollständig von dem darunterliegenden Genom. Die grundlegende Kopplung von Gen zu Phänotyp und somit zu Fitness wird zerstört.

5.10.2 Positive Scores 1 Div

(ps1div) Hier werden die Fitnesswerte der Population auf einen Wertebereich von 0 bis 1 gebracht. Dazu werden die Kehrwerte aller Fitnesswerte der Population mit dem größten vorhandenen Fitnesswert multipliziert. Als Grundvoraussetzung müssen jedoch alle Fitnesswerte > 0 sein.

$$\forall_i \quad i \leq s : \qquad \text{fitness}_i = \frac{\text{highest\_fitness}}{\text{fitness}_i}$$

5.10.3 Ceil Sub Score

(xceil) Der Fitnesswert jedes Individuums wird von einer oberen Schranke abgezogen und dieser Wert als neuer Fitnesswert genommen. Als besonderes Problem taucht hier der große Wertebereich von Energieniveaus bei Molekülen auf, der sich innerhalb einer Population ohne weiteres über mehrere Zehnerpotenzen erstrecken kann. Nimmt man eine feste obere Schranke, so kann diese zu groß oder zu klein sein. Ist sie zu groß, so wird der Selektionsdruck in der Population sehr klein. Ist die Schranke zu klein, so erhalten

MinWin	N	∅ En.	σ	Min	Max	nEvals	|Z|
drf	360	26,49	4,93	18,43	40,55	29691	
ps1div	380	24,98	3,86	18,79	34,77	28339	0,0001
xceil	380	24,70	3,66	18,71	37,62	20685	

Tabelle 5.16: Vergleich verschiedener Minimierungsfenster für proportionale Selektionsschemata. Werte gelten für den Steady State Algorithm mit 100 Individuen, sind jedoch für den Simple GA repräsentativ.

MinWin: Minimierungsfenster;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
|Z|: Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der Distributionen;

viele Individuen einen Wert kleiner 0. Dann fallen sie entweder aus der Population heraus[3], oder die Fitnesswerte müssen nach der Fensterfunktion noch einmal normiert werden. Diese zusätzliche Verschiebung und Normierung bringt allerdings einen weiteren Bias, bei welcher aus relativ guten Individuen plötzlich Superindividuen entstehen können.

Als Kompromiss kann als obere Schranke ein Wert genommen werden, bei dem immer mindestens $x\%$ der aktuellen Population noch einen Fitnesswert größer 0 bekommen. Allerdings ist auch dieser Prozentwert mit Bedacht und in Abhängigkeit der erwarteten Ergebnisse zu wählen, auch hier können zu kleine oder zu große Werte negative Effekte nach sich ziehen. Bei den in dieser Arbeit verwendeten Läufen wurde mit 90% gearbeitet.

5.10.4 Diskussion der Ergebnisse

Eine genaue Bewertung der Ergebnisse ist aufgrund ihrer hochgradig nichtlinearen und kontextabhängigen Arbeitsweise mit gewissen Problemen behaftet. Tabelle 5.16 zeigt die Ergebnisse, die aus der ersten Basisuntersuchung gewonnen wurden.

Wie den |Z|-Werten zu entnehmen ist, unterscheiden sich alle Fensterfunktionen weit genug in ihren Ergebnissen, um als verschiedene Distributionen erkannt zu werden. Auch eine Überprüfung durch einen Wilcoxon-Test

[3] was vielleicht mit einer Totgeburt zu vergleichen wäre

für ps1div mit xceil und drf mit xceil ergibt eine Fehlerwahrscheinlichkeiten unter 1%. Beachtenswert ist die Tatsache, dass das xceil-Verfahren mit fast einem drittel weniger Evaluationen auskommt und dabei noch das beste durchschnittliche Energieniveau erzielt.

Trotz dieser Ergebnisse wird von dem xceil-Verfahren abgeraten, denn es bedingt durch den zu wählenden Schwellenwert einen zusätzlichen Unsicherheitsfaktor in der Parameterwahl für einen Genetischen Algorithmus. Das ps1div-Verfahren braucht zwar mehr Evaluationen und ist etwas schlechter im Ergebnis, jedoch arbeitet es bei allen Ergebnisverteilungen ohne großen Bias.

Das drf-Verfahren ist in dieser Hinsicht sehr schlecht. Besonders bei Anwendung eines Simple Genetic Algorithm (ohne Tabelle) zeigt es noch schlechtere Werte als bei dem in der Tabelle dargestellten Steady State Algorithm.

Als Ergebnis dieses Untersuchungsabschnitts soll jedoch festgehalten werden, dass bei Verwendung von rangbasierten Selektionsverfahren das Problem der zu wählenden Minimierungsfensterfunktion verschwindet und so eine weiterer nicht-deterministischer Faktor vermieden wird.

5.11 Erweiterung des Suchfokus

Das Verhalten Genetischer Algorithmen tendiert dazu, im Verlauf der simulierten Evolution eine Konzentration des Genpools hin auf wenige, aber leistungsstarke Individuen zu forcieren. Die relativ kleinen Populationen, welche für Simulationsläufe benutzt werden, eliminieren durch den Konvergenzeffekt einzelne Genkombinationen aus der gesamten Population. Diese können durch den Crossover-Operator nicht mehr zurückgeholt werden, nur noch zufällige Mutationen an genau den betroffenen Stellen bieten eine geringe Chance dafür an.

Somit verlangsamt sich das Tempo der Suche beträchtlich, da durch eine Selektion immer mehr Individuen einer Population sich im Genotyp immer mehr ähneln oder sogar gleich sind, so dass deren erzeugte Nachkommen ebenfalls zum Teil identische Suchräume abdecken.

5.11.1 Adaptive Mutation

Als eine mögliche Gegenmaßnahme kann die Adaption der Mutationsrate dienen. Es wird eine Schwelle definiert, oberhalb welcher die Streuung im Genpool einer Population noch als ausreichend angesehen wird. Fällt dieser Wert unterhalb die Schwelle, so wird aus dem Verhältnis von Schwelle zu tatsächlichem Wert ein Multiplikationsfaktor für die Mutationsrate der jeweils nächsten Generation errechnet.

Als Vergleichswert für die Schwelle lassen sich recht viele Kriterien heranziehen. Die beiden naheliegendsten Möglichkeiten zum Beispiel sind

– die genetische Vielfalt (Konvergenz) der Population.

– die Standardabweichung aller Ziel- oder Fitnesswerte

Wie in 3.4.11 schon erläutert, wird besonders für große Populationen die Berechnung der genetischen Vielfalt zu einem zeitkritischen Faktor, da jedes Gen jedes einzelnen Individuums mit den jeweils korrespondierenden Genen aller anderen Individuen verglichen und bewertet werden muss. Dies führt zu einer exponentiellen Zunahme von Vergleichen.

Deshalb ist für diese Arbeit der Standardabweichungsoperator benutzt worden. Fällt der Wert der Standardabweichung aller Objective Score Werte in der Population unterhalb eines festgelegten Schwellenwertes, wird die Mutationsrate für die Nachfolgegeneration anhand folgender Formel festgelegt:

$$mutation_{effektiv} = mutation_{gegeben} \cdot \frac{Schwellenwert}{Standardabweichung der Zielwerte}$$

Dieses Verhalten zielt darauf ab, am Anfang eines Laufes den Crossover-Operator zu unterstützen und eine maximale Exploitation bei geringer Exploration zu ermöglichen. Bei fortgeschrittener Konvergenz und abnehmender Bedeutung des Crossover wird die Exploration stärker gewichtet, um ein Optimum in der jeweiligen Umgebung zu finden oder möglicherweise aus einem Suboptimum herauszufinden.

Im Gegensatz zu Cobb und Greffenstette's Hypermutation (1993) mit einem Standardmodus von $p_m = 0,001$ und einem Hypermutationsmodus von $p_m = 0,5$ setzt die adaptive Mutation die durch Exploitation gewonnenen Informationen für eine eingegrenzte Exploration ein. Der Hypermutationsmodus hingegen kann eher mit einem multiplen sequentiellen Lauf mit teilweiser Reinitialisierung verglichen werden, der viel größere Sprünge bei der Exploration des Suchraumes absolviert.

5.11.2 Double prevention

Als eine Ursache für die Monotonisierung des Genpools gilt die Erzeugung von identischen Klonen. Diese entstehen, wenn elterliche Individuen sich in ihren Genomen stark ähneln und bei einer Crossover-Operation durch Zufall die elterlichen Genome so geschnitten werden, dass das Kindgenom identisch ist mit einem der beiden Eltern. Ein solches Kind wird auch Dublette genannt.

Vor allem bei der Arbeit mit überlappenden Populationen stellen sich Dubletten als störend heraus: sie bringen keine neuen Genkombinationen ins Spiel und erhöhen nur die Wahrscheinlichkeit für diese eine Genkombination, Nachkommen zu zeugen. Als Abhilfe ist für diese Untersuchungen die *Double prevention* Option eingeführt worden.

Dieser in der Natur nicht existente Mechanismus vergleicht in jeder Generation die neu erzeugten Individuen mit den Individuen der Elterngeneration *bevor* die Kinder für die Folgegeneration eingegliedert werden. Wird eine Dublette gefunden, so wird diese entweder mit einer hohen Mutationswahrscheinlichkeit mutiert oder das komplette Genom der Dublette wird mit Zufallswerten reinitialisiert. Im Idealfall wird dadurch gewährleistet, dass sich alle Individuen einer Population in mindesten einer Base (einem Bit) unterscheiden.

5.11.3 Diskussion der Ergebnisse

Verhalten von adaptiver Mutation

Die Abbildungen 5.12 und 5.13 geben eine gute Zusammenfassung der in Basisuntersuchung 2 gewonnenen Ergebnisse. Die ersten 576 Parameterkombinationen auf der linken Seite sind schon aus Abschnitt 5.6 (Mutation) bekannt und sind ohne adaptiver Mutation berechnet. Die 288 Kombinationen der rechten Seite stellen eine Untermenge der linken Parameterkombinationen dar – die Mutationsrate hat nur 0,001 anstatt 0,001 und 0,01 – beinhalten als eingeschaltete Option die adaptive Mutation. Deutlich zu erkennen ist die markante Verschlechterung der durchschnittlichen Ergebnisse bei gleichzeitiger Abnahme der Anzahl an Evaluationen, wenn adaptive Mutation eingeschaltet ist.

Eine genauere Untersuchung der Läufe liefert den Grund für dieses Verhalten: die eingesetzte Funktion zur Anpassung der Mutation erzeugt zu

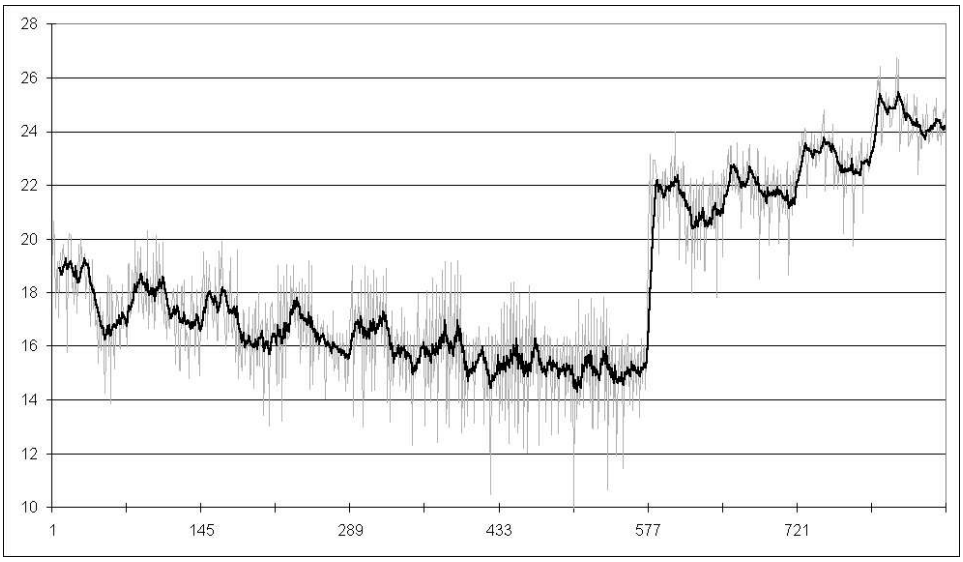

Abbildung 5.12: Verhalten von adaptiver Mutation, Energiewerte;
Adaptive Mutation - Aus:An (576:288);
Aufteilung im Teil ohne adaptive Mutation: Populationsgröße - 500:1000:2000:5000 (144); Ersetzungsgröße 0,5:0,9 (72); Grundmutationsrate - 0,001:0,01 (36);
Aufteilung im Teil mit adaptiver Mutation (Grundmutationsrate durchgehend 0,001): Populationsgröße - 500:1000:2000:5000 (72); Ersetzungsgröße 0,5:0,9 (36);

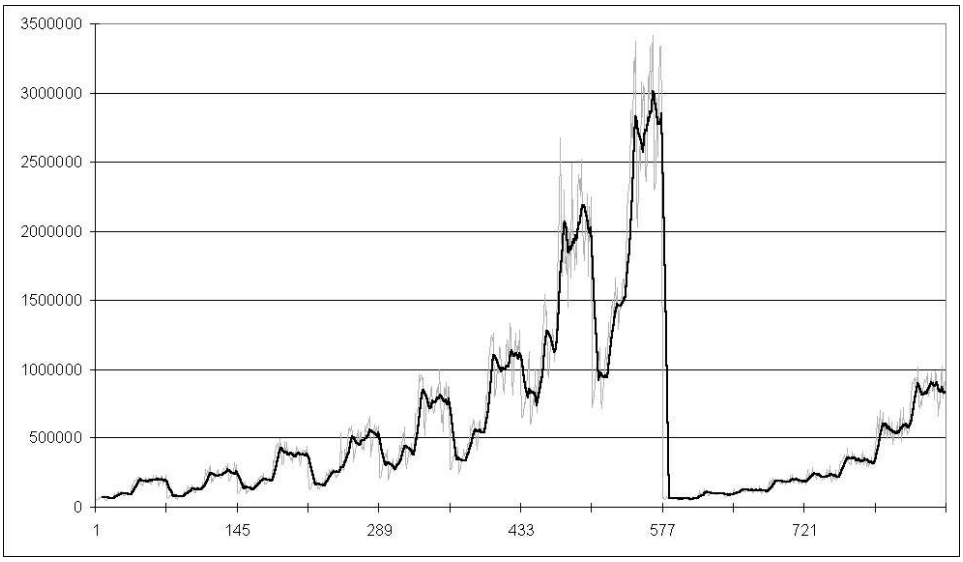

Abbildung 5.13: Verhalten von adaptiver Mutation, benötigte Anzahl an Evaluationen. Aufteilung wie in Abb. 5.12.

DPrev	N	∅ En.	σ	Min	Max	nEvals	|Z|
n	72	17,36	1,64	13,07	23,45	351030	0,0001
j	72	16,49	1,39	13,73	22,69	501607	

Tabelle 5.17: Vergleich der Ergebnisse mit und ohne Double prevention bei einer Populationsgröße von 1000 und dem Simple Genetic Algorithm. Parametrierung der Zusatzuntersuchung wie die zweite Basisuntersuchung, mit Ausnahme des Ersetzungsalgorithmus.

N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
DPrev: Double-Prevention Option (ja/nein);
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der Distributionen;

starke Erhöhungen der Mutationsrate. Diese lassen die Suche, wie schon in Abschnitt 5.6 beschrieben, in die Zufallssuche abgleiten und die Effizienz des GA abnehmen. Diese Erkenntnis und die in Abschnitt 5.6 gewonnenen Ergebnisse über den Einfluss von Mutation vor allem bei kleineren Populationen, geben Grund zur Annahme, dass bei einer sanfteren Anpassung der Mutationsrate – unter Umständen auch mit Deckelung des Maximalwertes – eine adaptive Mutation Vorteile bringen wird.

Verhalten von Double prevention beim Simple GA

Die Ersetzungsstrategie beim Simple Genetic Algorithm schließt eine Nichtbeachtung des durch Double prevention erzeugte Individuen aus: das reinitialisierte Individuum wird auf jeden Fall in die Nachfolgepopulation aufgenommen. Bei entsprechend weicher Elternselektion bekommt das Genom dann in den nächsten Generationen auch eine reelle Chance, selektiert zu werden.

Wie in Tabelle 5.17 dargestellt, ist hier ein Einfluss des Parameters deutlich nachweisbar. Die besseren Ergebnisse bei eingeschaltetem Double prevention zeigen, dass in Verbindung mit weicher Elternselektion die Einführung neuer Allele in eine schon stark konvergierte Population den Suchfokus kurzfristig wieder genug erweitert, um aus kleineren lokalen Optima wieder herauszufinden. Allerdings wird dieser Vorteil durch einen An-

Popgr	DPrev	N	∅ En.	σ	Min	Max	nEvals	|Z|
100	n	100	22,83	2,77	16,01	32,35	28675	0,7032
	j	100	22,75	2,68	18,07	31,15	33221	
500	n	72	17,80	1,38	14,25	20,68	143234	0,7629
	j	72	17,71	1,39	13,85	20,32	172333	
1000	n	72	16,72	1,41	13,21	19,53	296411	0,6998
	j	72	16,64	1,40	13,03	19,90	347222	
2000	n	72	15,92	1,63	12,43	18,98	613202	0,9125
	j	72	15,97	1,71	10,49	19,18	707428	
5000	n	72	15,22	1,68	9,97	18,43	1605319	0,9395
	j	72	15,23	1,56	10,64	18,26	1901380	

Tabelle 5.18: Vergleich der Ergebnisse mit und ohne Double prevention bei verschiedenen Populationsgrößen. Werte für Populationsgröße 100 aus einer Zusatzuntersuchung zur ersten Basisuntersuchung, Werte für alle andere Größen aus der zweiten Basisuntersuchung (Steady State GA).

Popgr: Populationsgröße;
N: Anzahl Beobachtungen; *∅ En.:* Mittelwert der Energie;
Min: Minimaler Mittelwert; *Max:* Maximaler Mittelwert;
DPrev: Double-Prevention Option (ja/nein);
σ*:* Standardabweichung des Mittelwerts;
nEvals: Durchschnittliche Anzahl benötigter Evaluationen;
$|Z|$*:* Fehlerwahrscheinlichkeit bei Annahme der Hypothese auf Ungleichheit der Distributionen;

stieg von 43% in der Zahl der benötigten Evaluationen erkauft.

Verhalten von Double prevention beim Steady State Algorithm

Wie Tabelle 5.18 zu entnehmen ist, hat die Double prevention Option außer für eine Populationsgröße von 100 Individuen keinerlei nachweisbaren Einfluss auf die Güte der Ergebnisse. Eine weitere Gruppierung nach Mutationsrate oder Ersetzungsgröße zeigt ebensowenig nachweisbare Eigenschaften. Auch hier liegt in allen Populationsgrößen die benötigte Anzahl der Evaluationen bei eingeschalteter Double prevention deutlich höher – zwischen 15% und 20% – als ohne diese Option. Somit bliebe als Ergebnis, dass Double prevention mit dem Steady State GA keine Verbesserung bringt.

Die Erklärung für dieses Verhalten findet sich in dem Zusammenwirken von Double prevention und Ersetzungsstrategie des Steady State Algorith-

mus: eine Kopie des als doppelt vorhanden identifizierten Individuums wird vollständig neu initialisiert. Bei schon stark konvergierten Populationen bedeutet dies, dass diese Kopie mit großer Sicherheit sehr viel schlechter ist als alle anderen Individuen. Somit wird im Ersetzungsschritt dieses "neue" Individuum niemals für die Nachfolgegeneration in Betracht gezogen, die neuen Allele werden nicht in die Population übernommen.

Um also Double prevention für den Steady State Algorithmus effektiv werden zu lassen, muss die Suche nach Dubletten *nach* dem Einfügen der Kindgenome in die Population erfolgen. In diesem Fall sollten ähnliche Ergebnisse wie bei dem Simple GA zu erwarten sein.

Als Fazit bleibt die Aussage, dass die Verhinderung von Dubletten eine nützliche Eigenschaft darstellt – dies allerdings nur, wenn die neu initialisierten Genome auch in die Nachfolgegeneration übernommen werden.

Kapitel 6

Erweiterungen zu Genetischen Algorithmen

"Probability is a constant: Even if Murphy's Law doesn't work all the time, it is always a good excuse."

Die grundsätzliche Arbeitsweise von Genetischen Algorithmen und ihre Leistungen bei der Molekülstrukturoptimierung wurden in den vorangegangenen Kapiteln beschrieben und die Wechselwirkungen von Parametern dargestellt.

Dieses Kapitel widmet sich einigen Ausbaumöglichkeiten, welche die Leistungsfähigkeit der Algorithmen erhöhen und deren Vielseitigkeit anschaulich demonstrieren. Es wird gezeigt, dass GAen sowohl allein durch Rekombination vorhandener Methoden als auch durch Unterstützung mittels evolutionsexterner – aber problemspezifischer – Optimierungsalgorithmen verbessert werden können.

6.1 Einfache Multipopulationsansätze

Grundsätzlich gibt es drei Ansätze zur Benutzung von mehreren Populationen bei Genetischen Algorithmen: parallele, sequentielle und Kombinationen davon. Alle Ansätze arbeiten nach dem Prinzip, dass die in Computersimulationen verwendeten Populationsgrößen in keinster Weise mit dem massiven Parallelismus der Natur und der Interkonnektivität der darin enthaltenen Einflussgrößen konkurrieren können.

6.1.1 Multiple parallele Läufe

Populationen von Individuen können quasi gleichzeitig einen Evolutionsprozess durchlaufen. Dabei wird jede Population als räumlich abgeschlossene Einheit betrachtet, bei welcher jedoch von Zeit zu Zeit einige Individuen migrieren und sich in anderen Populationen integrieren. Dieser Aufbau entspricht dem aus der Natur bekannten Beispiel von Herden, welche als geschlossene Gemeinschaft in geringer Zahl herdenfremde Tiere aufnehmen und eigene Mitglieder verstoßen.

Als bedeutende Faktoren in diesen Ansätzen sind sowohl die Anzahl der Populationen und deren interne Parameter – wie Populationsgröße, Mutationsrate, Crossover-Algorithmus etc. – als auch die räumliche Anordnung dieser Populationen für den darunterliegenden Individuen-Migrationsalgorithmus anzusehen. Die Gewichtung dieser Faktoren ist Gegenstand mehrerer Untersuchungen und S. W. Mahfoud zitiert in [39] einige Ergebnisse von Tanese (1989), demnach sich die Wahrung des Gleichgewichts als äußerst prekäre Angelegenheit erweist. Zu wenig oder keine Migration resultiert z.B. in hoher Redundanz im Suchprozess, da die Populationsläufe in diesen Fällen stark multiplen sequentiellen Läufen mit vollständiger Reinitialisierung ähneln. Zu hohe Migrationsraten führen andererseits vielfach zu verfrühtem Takeover vieler Populationen durch wenige Superindividuen und vernichten so die Vorteile paralleler Evolution.

Die Anzahl der verschiedenen Ansätze ist fast genauso groß wie die Anzahl der darüber veröffentlichten Publikationen und umfasst so unterschiedliche Populationsverteilungenen wie Maschenanordnung mit Pollenisierung durch simulierten Wind, zirkuläre Anordnung mit Ringaustausch von Individuen, Voll- und Teilvernetzung aller Populationen und Simulation komplexer Sozialgefüge inklusive Lebensplanung durch Individuen (Goldbergs Kommune).

6.1.2 Multiple sequentielle Läufe

Die zeitlich sequentielle Anordnung von Populationen kann einer Züchtung oder auch Missionierung gleichgesetzt werden. Hierbei wird ein Genetischer Algorithmus darauf ausgelegt, dass die besten gefundenen Individuen nach einem vollständigen Lauf die Möglichkeit bekommen, in einem neuen Lauf direkt von Anfang an dabei zu sein und so von vornherein dessen Evolution in der Anfangsphase zu beschleunigen, beziehungsweise in eine bestimmte

Richtung zu lenken.

Die vermehrten Evaluationen sequentieller Läufe verringern einerseits den selection noise, andererseits wird dadurch dem Problem der verfrühten Konvergenz aus dem Weg gegangen. Anstatt mit allen Mitteln die Konvergenz kontrollieren zu wollen, wird hier ein unter Umständen suboptimales Genom quasi als Sprungbrett für weitere Exploitation und Exploration benutzt. Die Wiedereinführung verlorener Allele durch die teilweise Reinitialisierung erlaubt es somit, die Suche von einer breiteren Basis an Suchpunkten durchzuführen.

6.2 Pyramidale Kulturen

Sowohl die zeitliche als auch die räumliche Anordnung von Populationen produzieren Ergebnisse, welche denen eines einfachen Genetischen Algorithmus zumeist überlegen sind[39]. Als Untersuchungsgebiet für diese Arbeit wurde – neben der im vorherigen Kapitel besprochenen einfachen Population – ein Ansatz entwickelt, der sowohl sequentielle als auch parallele Komponenten aufweist. Dieses *Pyramidenkultur* getaufte Konzept soll dabei die Vorteile aus beiden Welten vereinen.

6.2.1 Vorbilder

Das Vorbild für diesen Algorithmus stellen Auswanderungsströme in der Natur dar, bei denen einzelne Lebewesen oder ganze Bevölkerungsgruppen aus ihrer angestammten Heimat ausziehen, um sich einen neuen Lebensraum zu suchen. Unter Umständen treffen sie dort entweder auf Einheimische oder auf "Kolonisten" aus anderen Gegenden. Aus der Menschheitsgeschichte können die keltischen Wanderungen oder die Besiedelung Amerikas und Australiens von Europäern herangezogen werden.

6.2.2 Funktioneller Ablauf

Der Lauf ist in mehrere Ebenen unterteilt, welche von unten nach oben eine abnehmende Zahl von Populationen aufweisen. Jede Population ist von allen anderen Populationen derselben Ebene komplett abgetrennt und autark. Es existieren also – in dieser Basiskonfiguration – keine Migrationsbewegungen innerhalb der Populationen einer Ebene. Bei Wanderungen in eine höhere Ebene jedoch treffen Individuen aus verschiedenen Populationen zusammen.

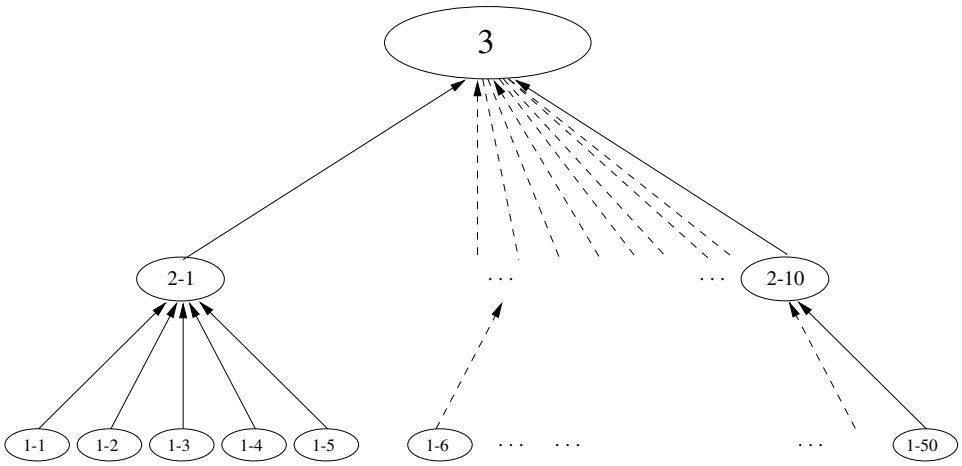

Abbildung 6.1: Beispiel eines pyramidalen GA mit 3 Ebenen. Je 5 Populationen der ersten Ebene geben ihre besten Individuen an eine der zehn Populationen der zweiten Ebene. Diese wiederum geben ihre besten Individuen an die Population der dritten Ebene.

Formal: Es existiert eine Pyramide PP mit n_e Ebenen $\{E_1, E_2, \ldots, E_{n_e}\}$. Jede Ebene E_i enthält n_p initialisierte, d.h. mit zufällig generierten Individuen gefüllte, Populationen. Es gilt fernerhin $n_p(E_i) \geq n_p(E_{i+1})$.

Zu Beginn wird durch $i = 1$ die Ebene E_1 als aktuelle Ebene deklariert. Für die aktuelle Ebene E_i werden alle $n_p(E_i)$ Populationen einem Evolutionsprozess durch den Lauf eines GAen unterzogen. Sind alle Läufe beendet, emigriert aus jeder Population $P_j \in E_i$ eine festgelegte Anzahl n_m an besten Individuen in die Populationen der darüberliegenden Ebene E_{i+1}. Jedes emigrierende Individuum $I(P(E_i))$ emigriert jedoch in genau eine Population $P(E_{i+1})$ und ersetzt dort ein beliebiges anderes Individuum, welches nicht vorher immigriert ist. Dann wird durch $i = i + 1$ die obere Ebene zur aktuellen Ebene und der Algorithmus arbeitet diese ab. Der Algorithmus endet in der obersten Ebene: das Ergebnis des gesamten Laufes wird durch das beste Individuum der letzten Ebene repräsentiert.

6.2.3 Theoretische Überlegungen zur Funktionsweise

Wie eingangs bemerkt, soll der pyramidale Genetische Algorithmus (PGA) die Vorteile der multiplen sequentiellen und parallelen Läufe in sich vereinen. Dies wird durch die massive Parallelität der Populationen innerhalb

einer Ebene und der sequentiellen Abarbeitung der Ebenen ermöglicht. Sind diese Populationen autark, wie in der für diese Arbeit benutzten Versionen, so kann davon ausgegangen werden, dass bei kleinen Populationen viele relativ gute, aber unterschiedliche Molekülkonfigurationen generiert wurden. In einer Ebene wird so der Nebeneffekt der genetischen Drift kleiner Populationen instrumentalisiert und ausgenutzt. Durch die Weitergabe und Zusammenführung mehrerer unterschiedlicher Individuen in eine Population der nächsthöheren Ebene wird so ein genetischer Grundstock geschaffen, der dem eines multiplen sequentiellen Laufes oder dem einer zufälligen Initialisierung im Hinblick auf die genetische Vielfalt überlegen ist. Statt eine gutes Individuum oder viele zufällige als "Saat" für die nächste Generation einzusetzen, werden viele verschiedene – aber schon bewährte – benutzt.

Hinter diesem Ansatz steht die Überlegung, dass es in einem so hochgradig multimodalen Suchraum wie dem der Molekülstrukturen viele Optima mit großem Potential an Deception geben muss, d.h wenn Genome mit diesen Allelen auftreten, so werden sie die Suche aufgrund der abgeschätzten Schemafitnesswerte in eine falsche Richtung lenken. Bei einem einfachen sequentiellen Lauf besteht das Risiko, ein solches Optimum zu finden und trotz mehrmaliger, teilweiser Reinitialisierung der Population nicht mehr aus diesem Optimum ausbrechen zu können. Werden mehrere verschiedene Individuen dafür benutzt, so steigt die Wahrscheinlichkeit, in diesen Lösungen mindestens ein Individuum mit Allelen zu haben, dessen Potential an Deception weniger groß ist. Diese treiben dann die Entwicklung der gesamten Population in eine andere Richtung und durch Ausnutzung des Schema-Theorems und der Building-Block-Hypothese werden bessere Ergebnisse erzielt.

6.2.4 Kombination mit lokalem Hillclimbing

Nach der Beendingung eines Laufes kann das Ergebnis in vielen Fällen noch etwas verbessert werden, indem ein lokaler Hillclimber benutzt wird. Dies liegt daran, dass – bei der hohen Anzahl von 10 Bits pro Drehwinkel – die Least Significant Bits (LSBs) eines Drehwinkels wegen der relativ kleinen Populationen und aufgrund genetischer Drift auf Zufallswerte konvergieren. So kann zum Beispiel ein Winkel in einem Lauf auf $178,764°$ konvergieren, im nächsten auf $180,946°$. Die Hauptdrehung um ungefähr $180°$ ist damit im Prinzip gefunden, es müssen nur noch die Feinheiten ausgearbeitet werden.

Um dies zu demonstrieren, wurde deshalb zusätzlich ein einfacher Hillclimber implementiert, der alle Ergebnisse überprüft und gegebenenfalls verbessert. Dieser Hillclimber arbeitet mit einfacher Schrittweitenadaption, d.h. es werden erst große Änderungen der Drehwinkel evaluiert, um dann bei Misserfolg die Schrittweite sukzessiv gegen eine untere Grenze – in diesem Fall $\frac{360}{2^{10}} = 0,3515625$ Grad – laufen zu lassen.

6.2.5 Getestete Konfigurationen

Der Lauf eines pyramidalen GA musste sowohl die Überlegenheit von pyramidalen Genetischen Algorithmen gegenüber einfachen GAen zeigen, als auch in vertretbarer Zeit zu berechnen sein.

Aus diesem Grund wurde, nach einigen Tests und einer Abschätzung des Laufzeitverhaltens, eine 4-stufige Pyramide gewählt. Jeweils 5 Populationen (à 20 Individuen) der ersten Ebene reichen nach Beendigung ihrer Läufe die beiden besten Individuen an eine Population der zweiten Ebene. Je 5 dieser Populationen mit jeweils 40 Individuen reichen ihre beiden besten Lösungen an eine Population der dritten Ebene. In der dritten Ebene arbeiten dann nur noch 10 Populationen mit 80 Individuen, welche ihre besten Individuen am Ende ihrer Läufe an die Population der 4 Ebene mit 160 Individuen weiterreichen. Die beste Lösung dieser Population stellt das Ergebnis der Pyramide dar.

Da für den Untersuchungszeitraum auf der Parsytec keine Rechenzeit zur Verfügung stand, wurden die Läufe auf einem[1] Pentium 100 durchgeführt. Zusätzlich wurden auf einem PentiumPro 200 einige weitere Läufe mit dreimal größeren Populationen berechnet.

Die Parameter der einzelnen Populationen basierten auf einer Vorauswertung der Ergebnisse aus Kapitel 5, welche sich im Nachhinein als erstaunlich zutreffend erwiesen. Tabelle 6.1 gibt eine Übersicht über die in jeder Population benutzten Parameter.

Die Parameterwahl für Algorithmus, Mutationsrate, Anzahl der Eltern, Selektionsschema und adaptive Mutation ist somit – wie in Kapitel 5 gezeigt – optimal für kleine Populationen. Die Parameter für Ersetzungsgröße und Crossover sind nicht ganz optimal, doch immer noch als gut zu bezeichnen. Die Wahl der Double prevention hat keine negativen Einflüsse auf das Ergebnis, allerhöchstens in der Zahl der benötigten Evaluationen.

[1] später zwei

Parameter	Option
Algorithmus	Steady State
Populationsgröße	20, 40, 80 oder 160 (je nach Ebene) bzw. 60, 120, 240 oder 480 (je nach Ebene)
Ersetzungsgröße	0,5
Mutationsrate	0,01
Crossover	N-Point mit 5 Kreuzungspunkten
Anzahl Eltern	3
Selektionsschema	Uniform (somit keine Skalierung und kein Minimierungsfenster notwendig)
Haltekriterium	Haltefenster mit 40, 400, 800 oder 1600 Generationen (je nach Ebene)
Double prevention	Reinitialisierung
Adaptive Mutation	Aus

Tabelle 6.1: Getestete Konfigurationsparameter für pyramidale Genetische Algorithmen.

6.2.6 Ergebnisse

Insgesamt konnten 20 Läufe mit Populationsgrößen von (20, 40, 80 und 160)[2] Individuen und 12 Läufe mit (60, 120, 240, 480)[3] Individuen berechnet werden. Die Ergebnisse werden in den Tabellen 6.2 und 6.3 zusammengefasst.

Gefundene Konformationen

Wie auf den ersten Blick zu erkennen ist, sind die mit pyramidalen GAen gefundenen Molekülkonformationen weit besser als alle in Kapitel 5 analysierten Parameterkonfigurationen mit einfachen Populationen. Zur Erinnerung: Populationen mit 5000 Individuen fanden durchschnittlich bestenfalls Konformationen bei 14,44 $\frac{kcal}{mol}$, eine Extrapolation für Populationen von 100000 Individuen ergab 8,03 $\frac{kcal}{mol}$. Im Gegensatz dazu finden im Durchschnitt die PGA-20 Konfigurationen Energiewerte bei 2,56 $\frac{kcal}{mol}$, die PGA-60 Konfigurationen sogar bei 1,95 $\frac{kcal}{mol}$.

Als Entscheidungskriterium, ob eine Molekülkonformation mit dem bekannten Optimum identisch ist, kann der Root-Mean-Square (RMS) Wert[4] herangezogen werden. Nach F. Herrmann [30] ist die Äquivalenz zweier Mo-

[2] im folgenden "pyramidaler GA mit Populationsbasis 20" (PGA-20) genannt
[3] im folgenden PGA-60 genannt
[4] siehe Abschnitt 4.4.2 (Relative Kodierung von Molekülen.

	Original			Hillclimber		Verbesserung in %	
Nr.	Energie	RMS	nEvals	Energie	RMS	Energie	RMS
1	1,82780	0,025	3382566	1,78338	0,015	2,43	40,00
2	3,89903	1,227	3353493	3,23398	1,227	17,06	0,00
3	3,60343	0,122	3379076	1,82686	0,022	49,30	81,97
4	1,81704	0,019	3317898	1,77929	0,012	2,08	36,84
5	1,95299	0,050	3288769	1,80372	0,019	7,64	62,00
6	2,34347	0,048	3450250	1,81090	0,020	22,73	58,33
7	1,83495	0,023	3300681	1,81322	0,018	1,18	21,74
8	4,32353	0,789	3313527	3,91717	0,784	9,40	0,63
9	1,87173	0,049	3287234	1,78976	0,013	4,38	73,47
10	3,77698	1,219	3351977	3,26235	1,245	13,63	-2,13
11	2,98980	0,783	3344132	1,86995	0,773	37,46	1,28
12	2,72303	0,793	3391681	2,00519	0,777	26,36	2,02
13	1,90802	0,029	3273314	1,78063	0,019	6,68	34,48
14	1,88120	0,030	3372550	1,82718	0,024	2,87	20,00
15	1,92326	0,774	3295525	1,84730	0,773	3,95	0,13
16	3,66501	1,193	3308564	3,19242	1,231	12,89	-3,19
17	1,84408	0,018	3324922	1,81396	0,016	1,63	11,11
18	1,81106	0,019	3435155	1,79559	0,017	0,85	10,53
19	1,77715	0,011	3253685	1,77294	0,011	0,24	0,00
20	3,40092	1,214	3369528	3,17091	1,233	6,76	-1,57
∅	2,55872	0,422	3339726	2,20484	0,41	13,830	2,37

Tabelle 6.2: Pyramidaler GA mit Populationsbasisgröße 20.

Original: Vom PGA ermittelte Werte
Hillclimber: PGA Ergebnisse, die durch den einfachen Hillclimber nachbearbeitet wurden
Verb. ...: Durch den Hillclimber erzielte Verbesserungen in % gegenüber den PGA Ergebnissen
Nr.: Laufnummer
Energie: Energiewert der gefundenen Konformation;
RMS: Root-Mean-Square Werte dieser Konformation zum bekannten Optimum bei 1,75382
nEvals: Anzahl der benötigten Evalutionen
∅: Durchschnittliche Werte

	Original			Hillclimber		Verbesserung in %	
Nr.	Energie	RMS	nEvals	Energie	RMS	Energie	RMS
1	1,78915	0,015	9035551	1,77583	0,015	0,74	0,00
2	1,85655	0,023	8889266	1,81454	0,017	2,26	26,09
3	1,78184	0,016	8818212	1,76107	0,016	1,17	0,00
4	1,84045	0,029	9153995	1,79251	0,029	2,60	0,00
5	1,77529	0,014	8916169	1,76486	0,012	0,59	14,29
6	1,78584	0,014	8880046	1,77836	0,015	0,42	-7,14
7	1,84759	0,033	8931873	1,79370	0,025	2,92	24,24
8	1,81033	0,025	8864185	1,76520	0,011	2,49	56,00
9	1,78998	0,017	8956939	1,77100	0,019	1,06	-11,76
10	1,83847	0,020	8884249	1,80620	0,012	1,76	40,00
11	3,51972	1,222	9016416	3,21913	1,232	8,54	-0,82
12	1,79801	0,015	9047794	1,79287	0,016	0,29	-6,67
∅	1,95277	0,120	8949558	1,90294	0,118	2,55	1,67

Tabelle 6.3: Pyramidaler GA mit Populationsbasisgröße 60.

Original: Vom PGA ermittelte Werte
Hillclimber: PGA Ergebnisse, die durch den einfachen Hillclimber nachbearbeitet wurden
Verb. ...: Durch den Hillclimber erzielte Verbesserungen in % gegenüber den PGA Ergebnissen
Nr.: Laufnummer
Energie: Energiewert der gefundenen Konformation;
RMS: Root-Mean-Square Werte dieser Konformation zum bekannten Optimum bei 1,75382
nEvals: Anzahl der benötigten Evalutionen
∅: Durchschnittliche Werte

leküle bei einem RMS-Wert $\leq 0{,}2$ gegeben. Die PGA-20 Konfiguration (Tab. 6.2) erfüllt dieses Kriterium in 12 von 20 Läufen. Noch besser ist die PGA-60 Konfiguration: In 11 von 12 Läufen wurde das Optimum gefunden, wobei Lauf Nr. 11 zwar nicht das globale Minimum finden konnte, jedoch ein besseres Ergebnis produzierte (3.51972 $\frac{kcal}{mol}$) als das vor Beginn der Untersuchung bekannte Minimum bei 4,277345 $\frac{kcal}{mol}$. Dies bedeutet eine Erfolgsquote von ca. 60% für die PGA-20 Konfiguration und von ca. 92% für die PGA-60 Konfiguration.

Benötigte Evaluationen

Ein weiteres interessantes Merkmal stellt die benötigte Anzahl an Evaluationen dar. Diese ist für beide PGA Konfigurationen relativ konstant und beträgt im Durchschnitt ca. 3,3 Millionen für die PGA-20 Konfiguration und ca. 9 Millionen für die PGA-60 Konfiguration. Unter Berücksichtigung der in Abschnitt 5.3 (Populationsgrößen) gewonnenen Erkenntnisse läßt sich für diese PGA Konfigurationen eine äquivalente Populationsgröße für einfache Populationen bestimmen: Die benötigte Anzahl von 3,3 Millionen Evaluationen für die PGA-20 Konfiguration entspricht einer Populationsgröße von ca. 9500 Individuen, die PGA-60 Konfiguration mit 9 Millionen Evaluationen entspricht einer Populationsgröße von ca. 25500 Individuen. Allerdings sind die von den PGA Konfigurationen gefundenen Ergebnisse weit besser, als Läufe mit äquivalenten einfachen Populationsgrößen erwarten lassen (siehe Formeln in Abschnitt 5.3).

Hillclimber

Die ebenfalls in Tab. 6.2 und 6.3 angegebenen Werte bei Anwendung des lokalen Hillclimbers auf die Ergebnisse der PGA Konformationen zeigen deutlich das vorhandene Verbesserungspotential, aber auch die Grenzen dieser Methode. Bei allen Läufen konnte der Hillclimber die von den pyramidalen Genetischen Algorithmen gefundenen Ergebnisse noch verbessern, von wenigen Prozentpunkten bei gut gefundenen Optima bis zu fast 50% bei schlechter lokalisierten Optima. Allerdings bleiben die RMS-Werte größtenteils unverändert. Dies zeigt, dass der Hillclimber im gleichen Konformationsgebiet wie das vom PGA gefundene Optimum bleibt. Es werden also keine grundsätzlich anderen Optima gefunden, sondern die vorhandenen Drehwinkel so weit wie möglich verbessert. Die Tatsache jedoch, dass die

per RMS-Wert als äquivalent ermittelten und per Hillclimber optimierten Molekülkonformationen noch einen Energiebereich zwischen $1,76\frac{kcal}{mol}$ und $1,82\frac{kcal}{mol}$ abdecken, beweist die hohe Multimodalität des Suchraums auch im "Zielbereich" des Optimums.

Kapitel 7

Fazit

"Reliable information lets you say 'I don't know' with real confidence."

In den vorangehenden sechs Kapiteln wurde das Untersuchungsgebiet "Genetische Algorithmen zur Molekülstrukturoptimierung" erörtert und in Bezug auf die gestellte Aufgabe vertieft. Altbekannte und neu entwickelte Operatoren für Genetische Algorithmen wurden in extensiven Testläufen untersucht und die praktisch ermittelten Ergebnisse unter Zuhilfenahme theoretischer Überlegungen bewertet. Zuletzt wurde ein genetischer Multipopulationsalgorithmus entwickelt, dessen Leistungsfähigkeit die von bekannten Einfachpopulationsalgorithmen übertrifft und Optima für ein gegebenes Molekül reproduzierbar ermittelt.

Neben vielen kleinen Teilergebnissen stehen vier wichtige Erkenntnisse als Fazit dieser umfassenden theoretischen und empirischen Untersuchungen:

1. Wie auch schon von L. Altenberg in [1] dargestellt, können die zwei wichtigsten Fundamente der GA-Forschung – Hollands *Schema-Theorem* und die *Building-Block-Hypothese* von Goldberg – als Erklärung für die Funktionsweise von Genetischen Algorithmen so nicht alleine stehen bleiben, sondern sie können nur als Erklärungsansätze angesehen werden. Die für diese beiden Thesen getroffene Annahme unendlicher Populationsgrößen erweist sich dabei als größtes Problem: der Fehler bei der Abschätzung der Schematfitness ist umso größer, je kleiner die verwendete Populationsgröße ist. Die in Kapitel 5 dargestellten Untersuchungen für kleine und große Populationen beweisen eindeutig eine Änderung der Effizienz der beiden Hauptoperatoren Crossover

und Mutation bei unterschiedlichen Populationsgrößen. Diese Änderungen im Laufzeitverhalten bei kleinen Populationen werden weder durch das Schema-Theorem, noch durch die Building-Block-Hypothese hinreichend erklärt.

2. Die Benutzung von Populationsersetzungsstrategien mit überlappenden Anteilen ist eine gute Möglichkeit, den Selektionsdruck bei der Elternselektion zu minimieren bzw. ganz wegfallen zu lassen. Dadurch können die weithin bekannten Problemquellen der proportionalen Elternselektion durch konsequenten Einsatz rangbasierter Elternselektion effektiv vermieden werden. Als Folge dieser Strategie entfallen zusätzlich die Probleme durch Superindividuen – weil sie nicht mehr entstehen können – und die Probleme von Minimierungsfenstern, weil diese nicht mehr gebraucht werden.

3. Die Konvergenzgeschwindigkeit von Genetischen Algorithmen kann, zusammen mit den schon in 2. erwähnten sanften Schemata zur Elternselektion, durch zusätzliche Operatoren zur Erweiterung des Suchfokus – wie z.B. die Verhinderung von Dubletten – verlangsamt werden. Dadurch wird der Suchraum gründlicher durchsucht und die Ergebnisse der Genetischen Algorithmen werden verbessert.

4. Der Einsatz von Multipopulationsansätzen, wie den in dieser Arbeit entwickelten pyramidalen Kulturen, erhöht die Effizienz von Genetischen Algorithmen spürbar. Dadurch können Problemkomplexe bearbeitet werden, die ansonsten nicht durch Genetische Algorithmen mit Einfachpopulationen – auch nicht sehr große – gelöst werden können.

In dieser Arbeit wurde gezeigt, dass Energieoptima für kleine Peptide mit Seitenketten reproduzierbar durch Genetische Algorithmen lokalisiert werden können. Die dafür notwendige Erweiterung zu den Genetischen Algorithmen wurde entwickelt und die Funktionsweise erklärt.

Die Forschung an und mit Genetischen Algorithmen zur Molekülstrukturoptimierung ist jedoch bei weitem noch nicht abgeschlossen, sie hat vielmehr erst das Anfangsstadium hinter sich gelassen. Die Erfolge bei kleinen Peptiden geben Anlass zur Hoffnung, dass sich GAen auch bei mittleren und großen Peptiden bzw. Proteinen zur Strukturbestimmung bewähren werden. Im Zuge dieser Arbeit haben sich einige Themenkomplexe herauskristalli-

siert, deren eingehendere Untersuchung sicherlich zum Verständnis des Problems und zu dessen Lösung beitragen kann:

- Der entwickelte Ansatz der pyramidalen Kulturen entält noch sehr viel Entwicklungspotential. Ihre grundlegende Funktionsweise und Effizienz ist in dieser Arbeit demonstriert worden. Es sollte allerdings noch untersucht werden, wie diese Pyramiden aufgebaut werden sollten, um möglichst effizient zu arbeiten. Prinzipell gibt es hier drei verschiedene Möglichkeiten: hohe Pyramiden (viele Ebenen mit wenig Populationen pro Ebene), breite Pyramiden (wenig Ebenen mit vielen Populationen pro Ebene) und Kombinationen davon.

- Eine umfassende Kombination von GAen mit anderen Optimierungsverfahren ist in dieser Arbeit nicht untersucht worden. Jedoch deuten erste Untersuchungen von F. Herrmann (bisher noch unveröffentlicht) darauf hin, dass globale bzw. lokale Optimierungsverfahren in Kombination mit *Lamarckismus*[1] eine Effizienzsteigerung von Genetischen Algorithmen zu leisten vermögen. Es sollte deshalb untersucht werden, ob pyramidale Kulturen mit Lamarckismus kombiniert werden können, um eine zusätzliche Effizienzsteigerung zu erzielen.

- Die Strukturoptimierung großer Proteinmoleküle wird in Zukunft sehr große Genome mit mehreren hundert bis tausend Bits erfordern. Bisher sind allerdings die Ordnung und die definierende Länge der Genome zur Molekülstrukturoptimierung nicht bekannt. Es sollte deshalb untersucht werden, in welcher Größenordnung diese Kennwerte liegen. In diesem Zusammenhang interessant ist dann die Fragestellung, ob ein durch den Genetische Algorithmus automatisch[2] durchgeführtes *Reordering*[3] der Gene möglich ist und eine Leistungsverbesserung von Genetischen Algorithmen mit dieser Problemstellung bringen kann.

Die in dieser Arbeit gewonnenen Erkenntnisse werden sicherlich dabei helfen, die anstehenden Untersuchungen schneller und genauer durchführen

[1] Von Jean-Baptiste de Monet (Chevalier de Lamarck, 1744-1829) aufgestellte Evolutionstheorie, demnach von Lebewesen erlernte oder antrainierte Eigenschaften vererbt werden können.

[2] manuelle oder halbautomatische Verfahren erzwingen, wie in Abschnit 3.8 schon erwähnt, ein tiefergehendes Verständnis der Problemstellung oder zumindest die "Erahnung" der richtigen Lösung.

[3] eine Umkodierung bzw. Umsetzung einzelner Teilstränge im Genom.

zu können. Eine Erkenntnis ist allerdings jetzt schon sicher: mit Genetischen Algorithmen wird für die Aufklärung von Proteinstrukturen in Zukunft ein bedeutsames Hilfsmittel zur Verfügung stehen.

Literaturverzeichnis

[1] Lee Altenberg: *The Schema Theorem and Price's Theorem*; Institute of Statistics and Decision Sciences; Duke University, Durham, NC, USA.

[2] Thomas Bäck: *The Interaction of Mutation Rate, Selection and Self–Adaptation within a Genetic Algorithm*; University of Dortmund, Department of Computer Science, D-44221 Dortmund, Germany

[3] Thomas Bäck: *Optimization by Means of Genetic Algorithms*; University of Dortmund, Department of Computer Science, D-44221 Dortmund, Germany

[4] Thomas Bäck: *Optimal Mutation Rates in Genetic Search*; University of Dortmund, Department of Computer Science, D-44221 Dortmund, Germany

[5] Thomas Bäck: *Self–Adaptation in Genetic Algorithms*; University of Dortmund, Department of Computer Science, D-44221 Dortmund, Germany

[6] Thomas Bäck, Frank Hoffmeister: *Extended Selection Mechanisms in Genetic Algorithms*; University of Dortmund, Department of Computer Science, D-44221 Dortmund, Germany

[7] Thomas Bäck, Frank Hoffmeister, Hans-Paul Schwefel: *A Survey of Evolution Strategies*; University of Dortmund, Department of Computer Science, D-44221 Dortmund, Germany

[8] David Beasley, David R. Bull, Ralph R. Martin: *A Sequential Niche Technique for Multimodal Function Optimization*; Technical Report No. 93001; February 1993; University of Wales, College of Cardiff, CF2 4YN, UK

[9] David Beasley, David R. Bull, Ralph R. Martin: *An Overview of Genetic Algorithms: Part 1, Fundamentals*; University of Wales, College of Cardiff; University Computing, 1993, 15(2) 58-69; ©Inter-University Committee on Computing

[10] David Beasley, David R. Bull, Ralph R. Martin: *An Overview of Genetic Algorithms: Part 2, Research Topics*; University of Wales, College of Cardiff; University Computing, 1993, 15(4) 170-181; ©Inter-University Committee on Computing

[11] Alberto Bertoni, Martino Dorigo: *Implicit Parallelism in Genetic Algorithms*; TR 93-001-Revised, April 1993; Appeared in Artificial Intelligence (61) 2, 307-314; Universitá Statale di Milano, Italy

[12] Tobias Blickle, Lothar Thiele: *A Comparison of Selection Schemes used in Genietic Algorithms*; TIK-Report Nr. 11 Version 2, December 1995; Computer Engineering and Communication Networks Lab (TIK) at the Swiss Federal Institute of Technology, ETH Zürich

[13] I.N. Bronstein†, K.A. Semandjajew: *Taschenbuch der Mathematik*; 25., durchgesehene Auflage herausgegeben von G.Grosche, V. Ziegler†und D. Ziegler;
Verlag Nauka, Moskau
B.G. Teubner Verlagsgesellschaft, Stuttgart · Leipzig
Verlag Harri Deutsch, Thun und Frankfurt/Main

[14] Lance Chambers (editor): *Practical Handbook of Genetic Algorithms. Volume 1: Applications*; 1995; CRC Press, Boca Raton, New York, London, Tokyo

[15] Lance Chambers (editor): *Practical Handbook of Genetic Algorithms. Volume 2: New Frontiers*; 1995; CRC Press, Boca Raton, New York, London, Tokyo

[16] D.M. Deaven, N. Tit, J.R. Morris, K.M. Ho: *Structural optimization of Lennard-Jones clusters by a genetic algorithm*; Submitted December 18, 1995; Chem. Phys. Lett. 256, 195 (1996)

[17] D.M. Deaven, K.M. Ho: *Molecular geometry optimization with a genetic algorithm*; Physics Department, Ames Laboratory USDOE, Ames; Appeared in: Physical Review Letters 75, 288 (1995)

[18] Duden "Informatik": *Ein Sachlexikon fuer Studium und Praxis*; hrsg. vom Lektorat des BI-Wiss.-Verl. unter Leitung von Hermann Engesser. Bearb. von Volker Claus und Andreas Schwill. - 2., vollst. ueberarb. und erw. Aufl. - Mannheim, Leipzig, Wien, Zuerich: Dudenverl., 1993. ISBN 3-411-05232-5.

[19] David E. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning*;Addison Wesley (1989); 0-201-15767-5

[20] David E. Goldberg, Kalyanmoy Deb, Hillol Kargupta & Georges Harik: *Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Gentic Algorithms* (IlliGAL Report No. 93004);February 1993; Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

[21] David E. Goldberg, Jeffrey Horn, Kalyanmoy Deb: *What makes a Problem hard for a Classifier System?* (IlliGAL Report No. 92007);May 1992; Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

[22] David E. Goldberg, Kalyanmoy Deb: *A comparative Analysis of selection schemes used in genetic algorithms*; Appeared in: Foundations of Genetic Algorithms, G.J.E. Rawlins (editor), Morgan Kaufmann 1991

[23] A.P. Gultyaev: *The computer simulation of RNA folding involving pseudoknot formation*; January 1991; Appeared in *Yearbook of Medical Informatics 92*; Schattauer Verlahsgesellschaft mbH - IMIA's Publications Office

[24] G. Harik: *Finding multiple solutions in problem bounded difficulty* (IlliGAL Report No. 94002); March 1994; Department of Computer Science, University of Illinois at Urbana-Champaign.

[25] Inman Harvey: *Evolutionary Robotics and SAGA: the case for Hill Crawling and Tournament Selection*; Cognotive Science Research Paper 222, 1992; The University of Sussex, School of Cognitive and Computing Sciences, Brighton, England U.K.

Appeared in: C. Langton, editor, Artificial Life 3 Proceedings, Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol XVI.

[26] Jochen Heistermann: *Genetische Algorithmen*; Teubner-Texte zur Informatik; B.G. Teubner Verlagsgesellschaft Stuttgart · Leipzig (1994); ISBN 3-8154-2057-1

[27] Jörg Heitkötter, David Beasley (eds.): *The Hitch-Hiker´s Guide to Evolutionary Computation: A list of Frequently Asked Question (FAQ)*; 1996; USENET: comp.ai.genetic; Available via anonymous FTP from rtfm.mit.edu:/pub/usenet/news.answers/ai-faq/genetic

[28] Frank Herrmann: *Gentic Algorithm used in Protein Structure Prediction by Energy Minimization*; Department of Molecular Biophysics; German Cancer Research Center

[29] Frank Herrmann and Sándor Suhai: *Genetic Algorithms in Protein Structure Prediction*;Department of Molecular Biophysics; German Cancer Research Center; appeared in *Computational Methods In Genome Research*, Edited by S. Suhai, Plenum Press, New York 1994

[30] Frank Herrmann and Sándor Suhai: *Energy Minimization of Peptide Analogues Using Genetic Algorithms*;Department of Molecular Biophysics; German Cancer Research Center; appeared in *Journal of Computational Chemistry, Vol. 16, No. 11*, ©1995 by John Wiley & Sons, Inc.

[31] John H. Holland: *Adaptation in natural and artificial systems*; 1975; Ann Arbor: The University of Michigan Press

[32] Jeffrey Horn: *Thesis: Genetic Algorithms, Problem Difficulty, and the Modality of Fitness Landscapes* (IlliGAL Report No. 95004);July 1995; Department of Computer Science, University of Illinois at Urbana-Champaign.

[33] Herbert Immich: *Medizinische Statistik*;1974; F.K. Schattauer Verlag · Stuttgart – New York

[34] Ari Juels, Martin Wattenberg: *Stochastic Hillclimbing as a Baseline Method for Evaluating Genetic Algorithms*; September 28, 1994; University of California at Berkley, CA, USA

[35] Sami Khuri, Thomas Bäck, Jörg Heitkötter: *An Evolutionary Approach to Combinatorial Optimization Problems*; ©1993 ACM Press; Appeared in the proceedings of CSC'94; Phoenix Arizona, March 8-10, 1994

[36] Sami Khuri, Thomas Bäck, Jörg Heitkötter: *The Zero/One Multiple Knapsack Problem and Genetic Algorithms*; Appeared in the ACM Symposium of Applied Computation (SAC'94) proceedings. ©1993 ACM Press

[37] James R. Levenick: *Inserting Introns Improves Genetic Algorithm Success Rate: Taking Cue from Biology* (GARAGe Report No. 95-01-06); 1995; Computer Science Department; Willamette University; Salem, Oregon

[38] Samir W. Mahfoud: *Simple Analytical Models of Genetic Algorithms for Multimodal Function Optimization* (IlliGAL Report No. 93001);February 1993; Department of Computer Science, University of Illinois at Urbana-Champaign.

[39] Samir W. Mahfoud: *PhD Thesis: Niching Methods for Genetic Algorithms* (IlliGAL Report No. 95001);May 1995; Department of General Engineering, University of Illinois at Urbana-Champaign.

[40] Michael de la Maza: *The Boltzmann Selection Procedure*; Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA; Appeared in [15].

[41] J.B. Metzler: *Linder Biologie*; 19. neubearbeitete Auflage 1983; J.B. Metzlersche Verlagsbuchhandlung Stuttgart; ISBN 3-476-20261-5

[42] J.R. Morris, D.M. Deaven, K.M. Ho: *Genetic Algorithm Energy Minimization for Point Charges on a Sphere*; Ames Laboratory - U.S. Department of Energy, Department of Physics and Astronomy, Iowa State University, Ames

[43] J.T. Ngo, J. Marks: *Computational complexity of a problem in molecular-structure prediction*; Appeared in: Protein Engineering, vol 5, no 4, pp. 313-321, 1992

[44] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn: *Genetic Programming for Improved Data Mining – Application to the Biochemistry*

of Protein Interactions (GARAGe Report No. 96-04-01); 1996; Michigan State University, East Lansing MI

[45] Ingo Rechenberg: *Evolutionsstrategie*; Friedrich Frommann Verlag Stuttgart-Bad Cannstatt (1973); ISBN 3-7728-0373-3

[46] J.D. Schaffer, R.A. Caruna, L.J. Eshelman, R. Das: *A study of control parameters affecting online performance of genetic algorithms for function optimisation*; 1989; Appeared in: Proceedings of the Third International Conference on Genetic Algorithms and Their Applications; edited by J.D. Schaffer, San Mateo, California. Morgan Kaufmann Publishers.

[47] Eberhard Schöneburg, Frank Heinzmann, Sven Feddersen: *Genetische Algorithmen und Evolutionsstrategien*; Addison Wesley (1995); ISBN 3-89319-493-2

[48] Nicol N. Schraudolph, Richard K. Belew: *Dynamic Parameter Encoding for Genetic Algorithms*; UCSD Technical Report CS 90-175, LANL Technical Report LAUR 90-2795; July 20, 1992; Computer Science & Engineering Department; University of California, San Diego

[49] Robert Sedgewick: *Algorithmen in C++*;1. Auflage 1992 / 1. korrigierter Nachdruck 1994; Addison Wesley; ISBN 3-89319-462-2

[50] Stefan Silbernagl, Agamemnon Despopoulos: *dtv-Atlas der Physiologie*; 4. Auflage 1991; Georg Thieme Verlag Stuttgart; ISBN 3-423-03182-4

[51] Uwe Schoening: *Theoretische Informatik kurz gefasst*; Mannheim, Leipzig, Wien, Zuerich: BI-Wiss.-Verl., 1992, ISBN 3-411-15641-4

[52] Dr. Steffen Schulze-Kremer: *Genetic Algorithms and Protein Folding*; Juni 1995; Universität Bielefeld

[53] Arne Steuer: *Genetische Algorithmen: Natürlich optimiert*; Appeared in: iX Magazin 1/97, Heise Verlag

[54] Daniel L. Swets, Bill Punch: *Genetic Algorithms for Object Localization in a Complex Scene* (GARAGe Report No. 95-01-06); 1995; Michigan State University, East Lansing MI

[55] Gerhard Thews, Ernst Mutschler, Peter Vaupel: *Anatomie, Physiologie, Patophysiologie des Menschen*; 4. durchgesehene Auflage 1991; Wissenschaftliche Verlagsgesellschaft mbH Stuttgart; ISBN 3-8047-1148-0

[56] Günter Rudolph: *Convergence Analysis of Canonical Genetic Algorithms*; Department of Computer Science; University of Dortmund, D-44221 Dortmund, Germany

[57] Paul K. Weiner, Peter A. Kollman: *AMBER: Assisted Model Building with Energy Refinement. A general Program for Modeling Molecules and Their Interactions*; Appeared in: *Journal of Computational Chemistry* 1981, Volume 2, Number 3

[58] Scott J. Weiner, Peter A. Kollman, David A. Case, U. Chandra Singh, Caterina Ghio, Giuliano Alagona, Salvatore Profeta Jr., Paul Weiner: *A new Force Field for Molecular Mechanical Simulation of Nucleic Acids and Proteins*; Apeared in: *Journal of the American Chemical Society* 1984, Volume 106

[59] Scott J. Weiner, Peter A. Kollman, Dzung T. Nguyen, David A. Case: *An All Atom Force Field for Simulations of Proteins and Nucleic Acids*; Appeared in: *Journal of Computational Chemistry* 1986, Volume 7, Number 2

[60] Darrel Whitley: *A Genetic Algorithm Tutorial*; November 10, 1993; Technical Report CS-93-103 (Revised); Department of Computer Science, Colorado State University, Fort Collins, US