Optimierung von Peptidsequenzen mittels multikriterieller Heuristiken

Berechnung der Peptidfaltung in gitterbasierten Modellen

Diplomarbeit

Tim Hohm
tim.hohm@uni-dortmund.de

Philipp Limbourg
philipp.limbourg@uni-dortmund.de

Universität Dortmund

Dezember 2003

Prof. Dr. Wolfgang Banzhaf
banzhaf@ls11.cs.uni-dortmund.de
Fachbereich Informatik
Lehrstuhl für Systemanalyse
Universität Dortmund
44221 Dortmund

Dr. Daniel Hoffmann
daniel.hoffmann@caesar.de
research center caesar
Ludwig-Erhard-Allee 2
53175 Bonn

„I love deadlines. I especially love the whooshing sound they make as they fly by.“
Douglas Adams, Author, Hitchhiker's Guide to the Galaxy

„Ich sage nur ein Wort: Vielen Dank.“
Andreas Brehme, dt. Fußballspieler

Wir danken...

Dr. Daniel Hoffmann und Prof. Wolfgang Banzhaf für die exzellente Betreuung.
Jörn, der uns Sonne in unser Leben brachte.
Der besten Currymaus der Welt.
Holger und Jens, die sie sich den Quatsch freiwillig durchgelesen haben.
Leo, der sich nie zu schade für einen guten Rat war (dict. leo. org).
Der netten Bibliotheksdame des MPI.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

> „Es wird langsam Zeit, daß wir Köpfe mit Nägeln machen.“
>
> Klaus Täuber, Fußballspieler

Die Bioinformatik stellt Werkzeuge zur Verfügung, die Biochemiker bei ihrer Arbeit unterstützen sollen. Zu diesen Werkzeugen gehören nicht nur Datenbanken zur Organisation großer Mengen biologischer Daten, sondern auch Methoden zur rechnergestützten Simulation, mit deren Hilfe teure Experimente effizienter durchgeführt werden können.

Es ist das Ziel dieser Arbeit, ein Simulationsprogramm für den Entwurf neuer Wirkstoffe auf Peptidbasis zu entwickeln. Die zentrale Problemstellung ist das Proteinfaltungsproblem, bei dem versucht wird, aus einer Aminosäurensequenz deren räumliche Anordnung zu ermitteln. Dieses Problem kann als Optimierungsproblem aufgefasst werden. Es wird versucht, zu einer gegebenen Sequenz die optimale Faltung zu finden.

Unsere Arbeit beginnt in Kapitel 2 mit einer Einführung in die biochemischen Grundlagen, in der wir zunächst Aufbau und Funktion der Proteine erklären und das Proteinfaltungsproblem vorstellen. Im Anschluss erläutern wir wichtige Verfahren, um das Faltungsproblem experimentell zu lösen und in der Praxis häufig verwendete Screeningmethoden, um Wirkstoffkandidaten auf ihre Rezeptorkompatibilität hin zu untersuchen.

Da das Proteinfaltungsproblem zu komplex ist, um es am Rechner zu simulieren, werden vereinfachende Modelle verwendet. Die Faltungsberechnung ist trotz der Vereinfachung ein NP-hartes Problem [22]. In Kapitel 3 stellen wir Modelle vor, die anstatt der quantenmechanisch korrekten Energieberechnung einfachere lokale Kräfte verwenden. Die in Kapitel 4 beschriebenen Modelle gehen noch einen Schritt weiter, indem nur noch die Aminosäuren als Entitäten bei der Energieberechnung berücksichtigt werden.

Aufgrund der NP-Härte des Problems ist es nicht möglich, den Raum aller denkbaren Faltungen in polynomieller Zeit vollständig zu explorieren. Um dennoch in akzeptabler Zeit Ergebnisse erzielen zu können, haben wir uns für den Einsatz randomisierter Suchheuristiken entschieden. Die von uns eingesetzten Verfahren stammen aus dem Bereich der evolutionären Algorithmen, die sich bereits bei vielen Gelegenheiten als sehr robust erwiesen haben. In Kapitel 5 führen wir zunächst in die Grundlagen der Optimierung ein. Wir motivieren den Einsatz evolutionärer Verfahren, um abschließend die Grundformen der von uns verwendeten Algorithmen zu beschreiben.

In den darauf folgenden Kapiteln erläutern wir unsere Umsetzung der Aufgabenstellung. Nachdem wir in Kapitel 6 zunächst das Design und das einbettende Framework beschreiben, gehen wir in den Kapiteln 7, 8 und 9 auf die genaue Funktionsweise der einzelnen Einheiten unseres Designs ein. Das Problem lässt sich in zwei Stufen teilen. Die erste Stufe befasst sich mit der Peptidfaltung (Kapitel 7), die übergeordnete Stufe widmet sich der Suche nach geeigneten Sequenzen bzw. dem Screening mehrerer Kandidaten (Kapitel 9). Um die Qualität der einzelnen Sequenzen zu bewerten, entwickeln wir verschiedene Kriterien, die in Kapitel 8 beschrieben sind.

In Kapitel 10 beschreiben wir von uns durchgeführten Versuche und die dabei erzielten Ergebnisse. Während der Versuche machten wir einige interessante Beobachtungen, die in Kapitel 11 aufgegriffen werden. Anhand dieser Beobachtungen entwickelten wir Hypothesen, wir ebenfalls beschrieben und teilweise verifizierten.

Im Anschluss fassen wir in Kapitel 12 unsere Ergebnisse zusammen. Damit endet der Hauptteil unserer Arbeit. Abschließend führten wir ein erstes praktisches Experiment durch. Es ist in Kapitel 13 beschrieben und soll die Funktionsweise des Programms verdeutlichen. In diesem Experiment versuchten wir, die Rezeptorgegend eines Proteins zu kopieren und in einem deutlich kleineren Peptid zu stabilisieren. Den Abschluß unserer Arbeit bildet das Kapitel 14, in dem wir einen Ausblick auf mögliche Erweiterungen des Programms geben und möglich praktische Anwendungen skizzieren.

Kapitel 2

Biochemische Grundlagen

„Die Grundlage ist das Fundament der Basis.“

Le Corbusier (1887 - 1965), französisch-schweizerischer Architekt

Bevor wir beginnen, möchten wir darauf hinweisen, dass wir im Folgenden die Begriffe Peptid und Protein Synonym verwenden. Diese beiden Molekülarten unterscheiden sich lediglich in ihrer Länge, gehören chemisch aber zur gleichen Klasse. Wir befassen uns jedoch nur mit der kürzeren Variante, den Peptiden. Die meisten von uns zitierten Untersuchungen wurden jedoch an Proteinen durchgeführt.

Dieses Kapitel dient der Einführung in die biochemischen Grundlagen, die für das weitere Verständnis dieser Arbeit notwendig sind. Zu Beginn gehen wir auf den Aufbau und die Funktion der Proteine ein. Im Anschluss stellen wir das Proteinfaltungsproblem vor. Das Kapitel schließen wir mit der Erläuterung in der Praxis genutzter Verfahren zur Faltungsermittlung und dem Sequenzscreening ab.

2.1 Proteine

Eine der wichtigsten Klassen molekularer Bausteine des Lebens sind die Proteine. In diesem Abschnitt wollen wir zunächst den Aufbau der Proteine erklären. Im Anschluss werden wir auf ihre Funktionen im Körper und den körpereigenen Synthesemechanismus eingehen.

2.1.1 Aufbau der Proteine

Proteine bestehen aus Aminosäuren, die durch Peptidbindungen verknüpft sind. Aminosäuren sind organische Säuren, die mindestens eine Carboxyl-[1] und eine Aminogruppe enthalten. Alle in natürlichen Polypeptiden vorkommenden Aminosäuren haben die gleiche Grundstruktur, die in Abbildung 2.1 dargestellt ist. Sie bestehen aus einem zentralen Kohlenstoffatom (α-C-Atom), an das ein Wasserstoffatom, die Aminogruppe (NH_2), die Carboxylgruppe (COOH) und eine Restgruppe gebunden sind. Abhängig vom pH-Wert können die Carboxylgruppe und die Aminogruppe protoniert oder deprotoniert vorliegen. Falls beide geladen sind, spricht man von einem Zwitterion. Eine Aminosäure kann jedoch auch nur positiv-, negativ- oder ungeladen vorliegen, wie in Abbildung 2.2 dargestellt. Die Aminosäuren unterscheiden sich durch ihre Reste. Eine Liste der 20 in natürlichen Proteinen vorkommenden Aminosäuren ist in Tabelle 2.1 enthalten.

Bei der Bildung einer Peptidbindung bindet die Aminogruppe eines Moleküls unter Abspaltung von Wasser (H_2O) mit der Carboxylgruppe eines zweiten Moleküls (siehe Abbildung 2.3). Es kommt dabei zu einer Elektronenverschiebung zwischen den beteiligten Kohlenstoff-, Sauerstoff-

[1] Carboxylgruppen –COOH oder $–COO^-$ sind organische Säuregruppen.

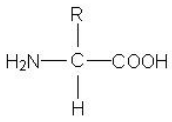


Abbildung 2.1: Grundstruktur der Aminosäuren

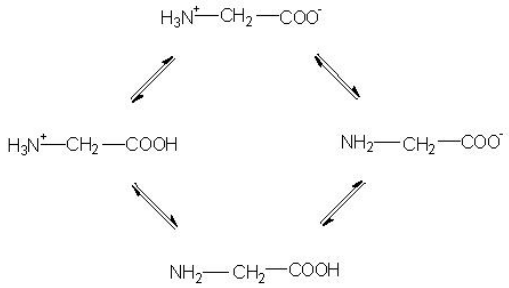


Abbildung 2.2: Ladungsänderungen der Aminosäuren

und Stickstoffatomen. In Folge dieser Elektronenverschiebung handelt es sich bei der Peptidbindung in etwa um eine Eineinhalbfachbindung. Dieser leichte Doppelbindungs-Charakter erhöht die Energiebarriere der Torsion um die Peptidbindung. Die Torsionen um die C-C_α- und C_α-N-Bindungen sind dagegen, in Abhängigkeit vom Rest, freier änderbar.

Die Gruppe der so aufgebauten Makromoleküle wird häufig begrifflich aufgeteilt in die Menge der Peptide, die alle Oligopeptide und Polypeptide bis zu einer Aminosäurenzahl von ca. 50 umfasst, und die Menge der Proteine, die aus allen Polypeptiden mit mehr als 50 Aminosäuren besteht. Diese Aminosäuresequenzen ordnen sich nach bestimmten Prinzipien im Raum an. Viele Proteine nehmen dabei eine feste „Faltung“ (räumliche Anordnung der Atome) ein, während Peptide oft zwischen einer Reihe von Faltungen wechseln oder ungefaltet vorkommen. Üblicherweise gibt es vier verschiedene Strukturarten, in denen die Proteine dargestellt werden. In Abbildung 2.4 sind die verschiedenen Arten schematisch dargestellt.

Primärstruktur Die lineare Sequenz der Aminosäuren.

Sekundärstruktur Die Faltung linearer Segmente der Sequenz. Dabei werden die Seitenketten nicht berücksichtigt. Es wird zwischen Helices (spiralförmigen Strukturen wie in Abbildung 2.6 dargestellt) und β-Faltblättern (planaren Strukturen ebenfalls in Abbildung 2.6 dargestellt) unterschieden. Sequenzteile, die auf diese Weise nicht zugeordnet werden können, bezeichnet man als Random Coil. Sie übernehmen z. B. die Funktion von Schleifen bzw. Kurven, die die einzelnen Helizes und Faltblätter bzw. die einzelnen β-Stränge untereinander verbinden. Die Struktur dieser Segmente wird durch die sogenannten Diederwinkel ψ und ϕ (siehe Abbildung 2.5) nahe der Peptidbindung bestimmt und durch Wasserstoffbrücken zwischen den einzelnen Aminosäuren stabilisiert. Regionen in der ψ- ϕ-Ebene entsprechen

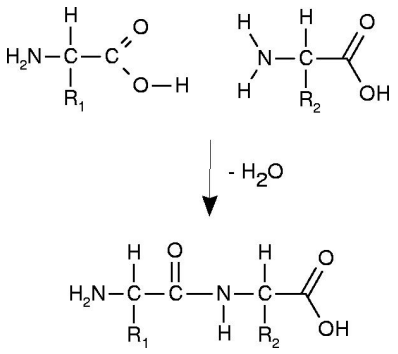


Abbildung 2.3: Ausbildung einer Peptidbindung

Tabelle 2.1: Liste der 20 in natürlich Proteinen vorkommenden Aminosäuren

Name	Abk.	linearisierte Strukturformel
Alanin	Ala	CH_3-CH(NH_2)-COOH
Arginin	Arg	HN=C(NH_2)-NH-3(CH_2)-CH(NH_2)-COOH
Asparagin	Asn	H_2N-CO-CH_2-CH(NH_2)-COOH
Asparaginsäure	Asp	HOOC-CH_2-CH(NH_2)-COOH
Cystein	Cys	HS-CH_2-CH(NH_2)-COOH
Glutamin	Gln	H_2N-CO-2(CH_2)-CH(NH_2)-COOH
Glutaminsäure	Glu	HOOC-2(CH_2)-CH(NH_2)-COOH
Glycin	Gly	NH_2-CH_2-COOH
Histidin	His	NH-CH=N-CH=C-CH_2-CH(NH_2)-COOH
Isoleucin	Ile	CH_3-CH_2-CH(CH_3)-CH(NH_2)-COOH
Leucin	Leu	2(CH_3)-CH-CH_2-CH(NH_2)-COOH
Lysin	Lys	H_2N-4(CH_2)-CH(NH_2)-COOH
Methionin	Met	CH_3-S-2(CH_2)-CH(NH_2)-COOH
Phenylalanin	Phe	C_6H_5-CH_2-CH(NH_2)-COOH
Prolin	Pro	NH-3(CH_2)-CH-COOH
Serin	Ser	HO-CH_2-CH(NH_2)-COOH
Threonin	Thr	CH_3-CH(OH)-CH(NH_2)-COOH
Tryptophan	Trp	C_6H_4-NH-CH=C-CH_2-CH(NH_2)-COOH
Tyrosin	Tyr	HO-C_6H_4-CH_2-CH(NH_2)-COOH
Valin	Val	2(CH_3)-CH-CH(NH_2)-COOH

verschiedenen Strukturen. Der Diederwinkel ω liegt relativ fest bei 180^o, da er durch die Peptidbindung stabilisiert wird. Die Gruppe der Helices wird unterteilt in:

- α-Helizes, die häufigste Helix-Variante. Bei ihnen geht die Carboxylgruppe der i-ten Aminosäure mit der Aminogruppe der $(i+4)$-ten Aminosäure eine Wasserstoffbrücke ein. Die α-Helices kommen rechts- und linksdrehend vor, wobei der rechtsdrehende Fall häufiger vorkommt. Zur Unterscheidung wird der linksdrehende Fall als Lα-Helix bezeichnet, der rechtsdrehende als Rα-Helix.
- 3_{10}-Helizes, bei denen die Carboxylgruppe der i-ten Aminosäure mit der Aminogruppe der $(i+3)$-ten Aminosäure eine Wasserstoffbrücke ausbildet. Auch bei dieser Variante existiert der selten vorkommende linksdrehende Fall, die L3_{10}-Helix.
- π-Helizes, bei denen eine Wasserstoffbrücke zwischen der Carboxylgruppe der i-ten Aminosäure und der Aminogruppe der $(i+5)$-ten Aminosäure ausgebildet wird.

Bei allen Helices liegen die Wasserstoffbrücken etwa parallel zur Achse der Helix.

Die Gruppe der Faltblätter lässt sich weiter unterteilen in:

- parallele Faltblätter, bei denen die einzelnen Stränge der Faltblätter gleiche Orientierung haben
- antiparallele Faltblätter, bei denen benachbarte Stränge entgegengesetzte Orientierung haben.

Bei paralleler Anordnung haben alle Wasserstoffbrücken gleiche Länge, während es bei antiparalleler Anordnung immer im Wechsel zu kurzen und zu längeren Brücken zwischen den einzelnen Strängen kommt. In seltenen Fällen kommt es auch zu Mischformen zwischen parallelen und antiparallelen Faltblättern.

In Erweiterung der Sekundärstruktur spricht man auch manchmal von Super-Sekundärstruktur. Bei dieser erweiterten Darstellung werden auch Strukturen beschrieben, die aus mehreren verschiedenen Faltblättern und Helices bestehen. Ein kurzer Überblick über gängige

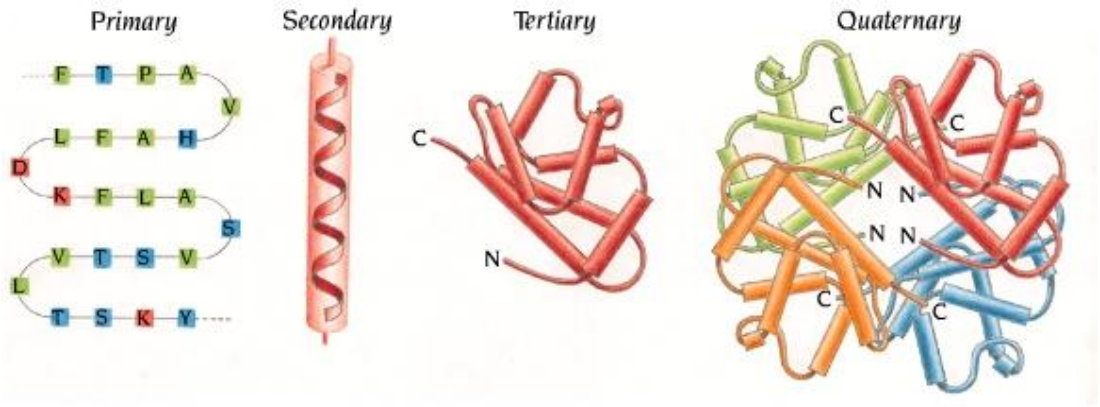


Abbildung 2.4: Strukturarten der Proteindarstellung (aus [17])

Super-Sekundärstrukturen wird in Abbildung 2.7 gegeben. Man unterscheidet dabei im Wesentlichen vier Gruppen:

- α-Proteine, die hauptsächlich aus Helices bestehen,
- β-Proteine, die hauptsächlich aus Faltblättern bestehen,
- α-β-Proteine, die häufig β-α-β-Motive enthalten,
- die Gruppe der Proteine, die in keine der drei anderen Gruppen passt.

Tertiärstruktur Sie ist die dreidimensionale Struktur eines Proteins, wobei auch die Seitenketten und ihre Wechselwirkungen berücksichtigt werden. Bei der Tertiärstruktur werden die in der Sekundärstruktur angegebenen Einheiten räumlich zueinander angeordnet. Hier zu beachtende Kräfte sind:

- Disulfidbindungen
- Wasserstoffbrücken
- van der Waals Kräfte
- hydrophobe Wechselwirkungen
- elektrostatische Wechselwirkungen

Quartärstruktur Manche Proteine lassen sich in einzelne Domänen einteilen. Diese Domänen können als kleinere Proteine angesehen werden, die miteinander durch nichtkovalente Bindungen verbunden sind. In der Quartärstruktur wird die räumliche Anordnung der einzelnen Domänen dargestellt. Da jedoch nicht alle Proteine aus verschiedenen Domänen bestehen, gibt es auch nicht zu allen eine Quartärstruktur.

2.1.2 Funktion und Synthese der Proteine im Körper

Einen ersten Eindruck der Bedeutung der Proteine für das Leben gewinnt man durch die Tatsache, dass ein wesentlicher Teil der in der DNS gespeicherten Erbinformationen Baupläne für Proteine enthält. Die im Körper vorkommenden Proteine und Peptide übernehmen viele verschiedene Funktionen. Anhand dieser Aufgaben lassen sie sich in verschiedene Gruppen einteilen:

Enzyme Diese Proteine dienen als Katalysator für viele biochemische Prozesse. Sie sind substratspezifisch, setzen also nur bestimmte Moleküle um (z. B. das Verdauungsenzym Lipase spaltet nur Fette). Durch die genaue Anpassung der räumlichen Struktur der Enzyme an die jeweiligen Substrate wird die Umsetzung anderer Substrate verhindert.

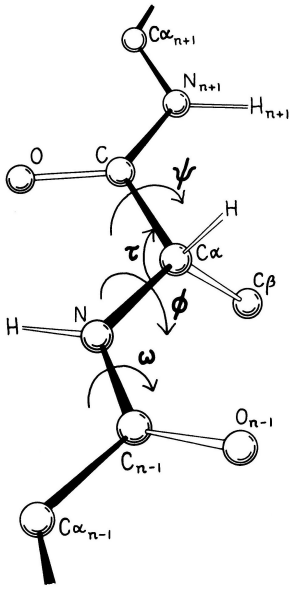


Abbildung 2.5: Winkel, die die Sekundärstruktur der Proteine bestimmen (aus [45])

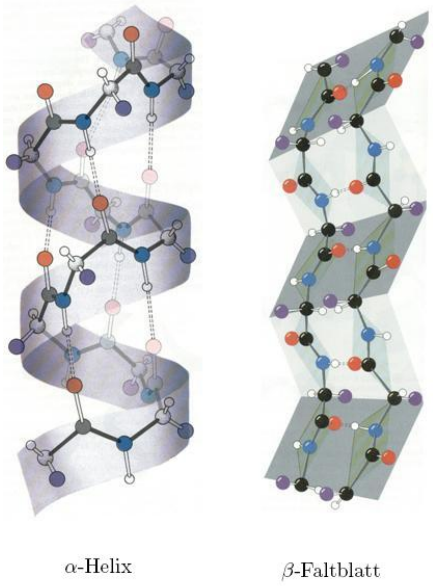


Abbildung 2.6: Strukturarten der Proteindarstellung: spiralförmige Helices und nahezu planare Faltblätter (aus [17])

Strukturproteine Diese Proteine werden an verschiedenen Stellen im Körper verwendet, um körpereigene Stützstrukturen zu verstärken. Das Protein Kollagen kommt z. B. in der Haut vor und ist wichtig für deren mechanische Eigenschaften.

Membranproteine Membranproteine sitzen in Membranen, die zelluläre Strukturen einschließen und sorgen für einen selektiven, kontrollierten Transport von Informationen oder Stoffen durch die Membranen hindurch. Beispiele sind Ionenpumpen, Ionenkanäle, Signaltransduktions-Proteine, etc.

Hormone, Wirkstoffe Manche Proteine funktionieren als Wirkstoffe, die bestimmte Abläufe im Körper aktivieren oder hemmen. Üblicherweise binden diese Proteine an Protein-Rezeptoren und lösen dort die entsprechende Änderung aus. Bekannte Beispiele sind Insulin, Wachstumshormone oder Neurotransmitter.

Transportproteine Sie ermöglichen den Transport von Stoffen wie z. B. Sauerstoff (Hämoglobin) oder Lipiden (β2-Lipoprotein).

Die an diesen Stellen benötigten Proteine werden in den Zellen anhand von in der DNS gespeicherten Bauplänen synthetisiert. Dieser Vorgang lässt sich in zwei Phasen (vgl. Abbildung 2.8) einteilen. Zuerst wird in der Transkription eine Kopie des Bauplans erstellt. Die DNS-Doppelhelix wird aufgetrennt und vom codogenen Strang der DNS wird eine komplementäre Kopie (zu jeder von der DNS abgelesenen Base wird die entsprechende komplementäre Base in die Kopie aufgenommen) erzeugt. Diese Kopie nennt man mRNS. In der zweiten Phase, der Translation, wird die mRNS in die Ribosomen (Zellorganellen in denen die eigentliche Proteinsynthese stattfindet) „eingefädelt“. Die mRNS wird in drei Basen umfassende Teilstücke, die Codons, eingeteilt. Zu jedem Codon gibt es komplementäre tRNS-Stücke, die jeweils eine bestimmte Aminosäure tragen. Die mRNS wird Codon für Codon abgelesen. Ein zu dem gelesenen Codon passendes tRNS-Stück gibt die von ihm getragene Aminosäure ab. So wird Aminosäure für Aminosäure das Protein aufgebaut.

Die Funktion der Proteine wird durch ihre Tertiärstruktur bestimmt. Die Bestimmung der Tertiärstruktur ist ein sehr zeitaufwändiger und schwieriger Prozess. Aus diesem Grund werden

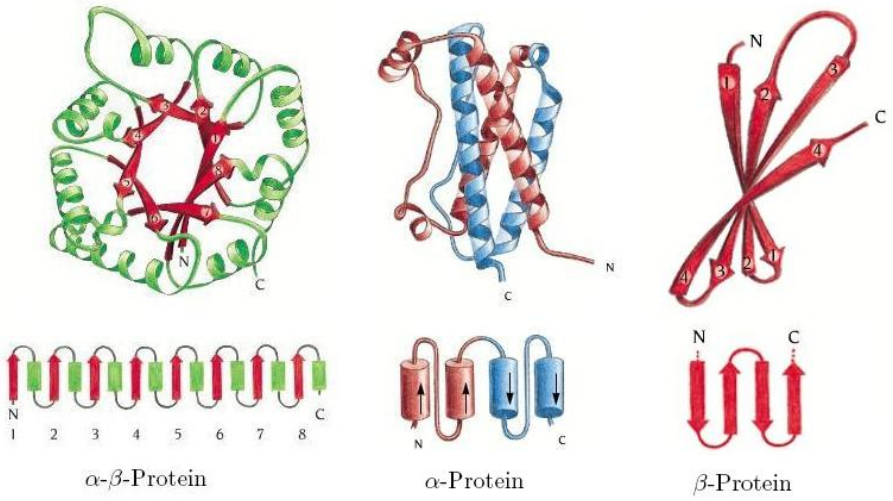


Abbildung 2.7: Überblick über gängige Super-Sekundärstrukturen (aus [17])

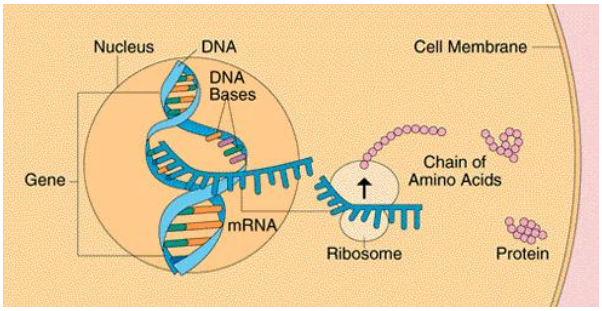


Abbildung 2.8: Phasen der Proteinbiosynthese (aus [79])

Strukturdaten zu diversen Proteinen in öffentlich zugänglichen Datenbanken, z. B. der Brookhaven Protein Data Bank (PDB), gesammelt.

2.2 Das Proteinfaltungsproblem

Wie Anfinsen 1973 in [5] in einem für die Proteinforschung bahnbrechenden Experiment zeigt, ist die native Tertiärstruktur eines Proteins fast ausschließlich abhängig von der Primärstruktur, also der Sequenz. Dies bedeutet, es existiert eine Abbildung folgender Form:

$$Sequenz \rightarrow Tertiärstruktur$$

Mit diesem Experiment war die Wissenschaft um ein Forschungsgebiet reicher: Die Erforschung dieser Abbildung, auch bekannt als das Proteinfaltungsproblem. Ein Protein hat keine im Sinne der klassischen Mechanik feste Struktur. So kann beispielsweise durch Kontakt mit einem Lösungsmittel das Protein seine Form verändern. Abhängig von seiner Struktur besitzt das System aus Protein und Umgebung ein energetisches Potential, wir haben es also mit einer Funktion der Form

$$Tertiärstruktur \rightarrow Energie$$

zu tun. Dies ist die so genannte Energiefunktion (genauer: Freie Enthalpie), die beim Proteinfaltungsproblem minimiert werden soll. Nach den Gesetzen der Thermodynamik kann sich ein

Protein ausgehend von einem beliebigen Zustand für $t \to \infty$ in jedem möglichen Zustand mit einer von der Energie E_0 abhängigen Wahrscheinlichkeit $p(E_0)$ befinden:

$$p(E_0) = \frac{e^{-\frac{E_0}{k_b T}}}{Z}$$

$$Z = \sum_{i=0}^{N} e^{-\frac{E_i}{k_b T}}$$

T sei hier die Temperatur in Kelvin, k_b die Boltzmannkonstante. Die Zustandssumme Z normiert die Wahrscheinlichkeit, indem sie über alle möglichen Konformationen (Zustände) aufsummiert. Das bedeutet, dass eine Faltung, deren Energie weit unter anderen Energieniveaus liegt, mit hoher Wahrscheinlichkeit auftritt. Die Faltung mit minimaler Energie (genauer: minimaler Freier Enthalpie) wird als native Faltung bezeichnet. Die meisten Proteine besitzen eine natürliche Faltung mit einem energetischen Minimum. Da wir nur $t \to \infty$ betrachten, sagt dies wenig über den Vorgang der Faltung aus. Es kann noch andere Zustände mit hoher Wahrscheinlichket geben, die geometrisch sehr weit entfernt von der nativen Faltung liegen. Dies sind so genannte metastabile Zustände, lokale Minima in der Energielandschaft, die das Protein nur schwer verlassen kann. Gerät ein Protein während seines Faltungsprozesses in ein solches Minimum, kann es dort sehr lange verweilen. Die Übergangswahrscheinlichkeit eines Zustandes in einen anderen benachbarten Zustand hängt ab von der Höhe der energetischen Schwelle zwischen den beiden Zuständen. In der Natur verläuft der Faltungsprozess (für chemische Maßstäbe) sehr langsam.

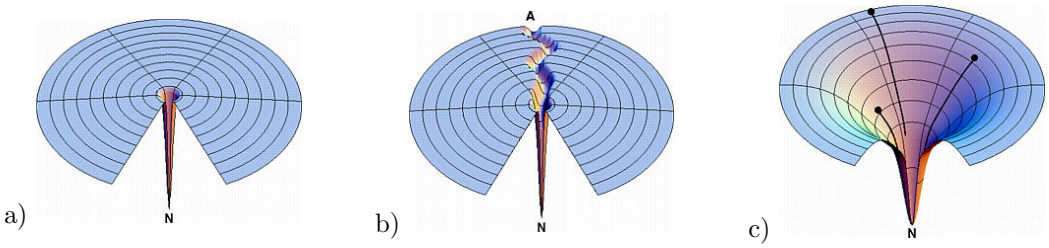


Abbildung 2.9: Energielandschaften (idealisiert), Faltungsdauer: a) sehr lange, b) normal, c) schnell (aus [33])

Proteine brauchen normalerweise zwischen 10^{-1} und 10^3 Sekunden, um ihren nativen Zustand zu erreichen. In den dicht gepackten Zellen wird die Faltung von Proteinen durch Wechselwirkung mit anderen Molekülen behindert. Um dieses Problem zu lösen gibt es sogenannte Chaperone, eine Klasse von "Faltungsenzymen" die die Faltung anderer Proteine beschleunigen. Wie kann das Protein nun überhaupt seinen nativen Zustand erreichen? In Abbildung 2.9 sind drei idealisierte Energielandschaften dargestellt, die die Energie eines Proteins bezüglich der Faltung darstellen. Landschaft a) ist eine sogenannte „Needle in the Haystack"Funktion. Proteine mit einer solchen Energielandschaft bräuchten in der Natur einen langen Zeitraum, um ihr energetisches Optimum zu erreichen. Landschaft b), oft als „Pathway-Typ"bezeichnet, ermöglicht eine schnellere Faltung. In letzter Zeit wurde ein neuer Ausdruck in den Sprachgebrauch der Proteinforscher eingeführt: Der so genannten Faltungstrichter (engl. Funnel), im Bild Landschaft c). Mit diesem Ausdruck soll die Form der Energiefunktion beschrieben werden, die anscheinend eine gewisse „Suchrichtung" aufweist, die es den Proteinen ermöglicht, ihr Optimum zu finden. Natürliche Proteine scheinen meist eine trichterförmige Energielandschaft zu haben; wahrscheinlich ist auch diese Eigenschaft ein Produkt biologischer Evolution.

Oft finden sich Proteine während des Faltungsprozesses in einem lokalen Minimum, welches der nativen Faltung in Sekundärstruktur und Tertiärstruktur bereits sehr ähnlich ist. Dieser so genannte „molten globule"-Zustand zeichnet sich meist dadurch aus, dass die Faltung noch nicht kompakt ist, da die Seitenketten noch nicht optimal gepackt sind (vgl. [48]).

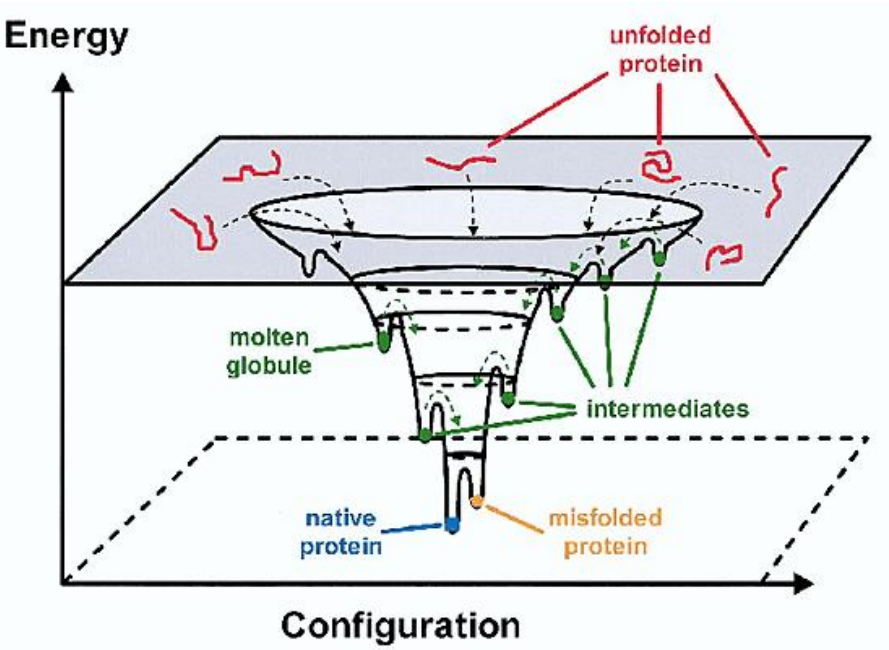


Abbildung 2.10: Faltungstrichter (aus [80])

Auf atomarer Ebene wird der Faltungsvorgang hauptsächlich durch Kräfte wie die elektrostatische Wechselwirkung und die van der Waals-Kraft beeinflusst (siehe Abschnitt 3). Auf der Abstraktionsebene der Residuen (Aminosäuren) hat sich gezeigt, dass diese verschiedene Eigenschaften haben, die sehr bedeutsam für den Faltungsprozess sind und nach denen sie klassifiziert werden können. Hier sind vor allem die Hydrophobizität, die Polarität und die Ladung maßgeblich. Der hydrophobe Effekt entsteht dadurch, dass nichtpolare Gruppen die Tendenz haben, ihre Interaktionsfläche mit Wasser zu minimieren. Dies ist ein entropischer Effekt. Wasser neigt dazu, entropisch ungünstige regelmäßige Strukturen um unpolare Residuen zu bilden. Daher führt eine Minimierung der Interaktionsfläche zwischen unpolaren Aminosäuren und Wasser zu einer Minimierung der freien Energie. Auf diese Weise bildet sich der typische hydrophobe Kern im Innern wasserlöslicher Proteine. Polarität bezeichnet eine ungleichmäßige Ladungsverteilung eines Residuums; z.B. kann eine Aminosäure ein elektrisches Dipolmoment besitzen. Dadurch neigt sie dazu, in Kontakt mit anderen Dipolen zu treten. Da Wasser ebenfalls ein Dipol ist, finden sich polare Residuen meist an der Oberfläche des Proteins in Kontakt mit dem Wasser ("hydrophile" Residuen).

Aminosäuren können Restladungen aufweisen, wenn sich positive und negative Ladungen innerhalb ihrer eigenen Struktur nicht ausgleichen. Gegensätzliche Ladungen ziehen sich an, so dass Residuen unterschiedlicher Ladungen oft beieinander zu finden sind.

Anhand der Stärke dieser Eigenschaften werden die Aminosäuren in verschiedene Gruppen klassifiziert, wie z. B. in [17] beschrieben.

2.3 Experimentelle Strukturbestimmung – eine Einführung

In diesem Abschnitt stellen wir zwei gängige experimentelle Methoden zur Strukturbestimmung von Proteinen vor. Wir beschränken uns auf eine sehr kurze Darstellung der Funktionsweisen von Röntgenkristallographie und kernmagnetischer Resonanzspektroskopie. Eine ausführlichere Erläuterung dieser Verfahren findet sich in Anhang A.

Tabelle 2.2: Hydrophobizität verschiedener Aminosäuren (nach [15])

Name			Molekulares Gewicht	Hydrophobizität
Alanin	Ala	A	89,09	0,616
Cystin	Cys	C	121,16	0,680
Asparaginsäure	Asp	D	133,10	0,028
Glutaminsäure	Glu	E	147,13	0,043
Phenylalanin	Phe	F	165,19	1,00
Glycin	Gly	G	75,07	0,501
Histidin	His	H	155,16	0,165
Isoleucin	Ile	I	131,18	0,943
Lysin	Lys	K	146,19	0,283
Leucin	Leu	L	131,18	0,943
Methionin	Met	M	149,21	0,738
Asparagin	Asn	N	132,12	0,236
Prolin	Pro	P	115,13	0,711
Glutamin	Gln	Q	146,15	0,251
Arginin	Arg	R	174,20	0,000
Serin	Ser	S	105,09	0,359
Threonin	Thr	T	119,12	0,450
Valin	Val	V	117,15	0,825
Tryptophan	Trp	W	204,23	0,878
Tyrosin	Tyr	Y	181,19	0,880

2.3.1 Röntgenkristallographie

Röntgenstrahlung ist kurzwellige elektromagnetische Strahlung von einer Wellenlänge, die ungefähr der Abmessung typischer biologischer molekularer Strukturen entspricht. Sie wird an regelmäßigen dreidimensionalen Strukturen wie Protein-Einkristallen gestreut wie sichtbares Licht an einem Beugungsgitter. In bestimmten räumlichen Richtungen interferieren die gestreuten Röntgenstrahlungen additiv, in anderen löschen sie sich aus. Aus dem dadurch entstehenden Beugungsbild oder Diffraktionsmuster (siehe Abbildung 2.11) versucht man, auf die beugende räumliche Struktur (also die 3D-Struktur der Proteine) zurückzuschließen. Bei der Anwendung der Röntgenkristallographie für die Untersuchung von Proteinen ist allerdings zu beachten, dass aus einer Lösung mit sehr hohem Proteinanteil Proteinkristalle gezüchtet werden müssen. Dieser Vorgang ist sehr zeitintensiv und funktioniert bei weitem nicht für alle Proteine. Eine wichtige Limitierung der Proteinkristallographie liegt darin, dass es nicht ohne weiteres möglich ist, dynamische Vorgänge wie den Prozess der Faltung zu untersuchen.

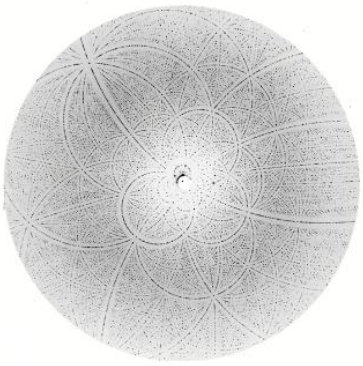


Abbildung 2.11: Diffraktionsmuster (aus [17])

Tabelle 2.3: Klassifikation der Aminosäuren (nach [17])

Name	Kürzel	Klassifikation
Ala	A	hydrophob
Cys	C	polar neutral
Asp	D	polar negativ
Glu	E	polar negativ
Phe	F	hydrophob
Gly	G	hydrophob
His	H	polar positiv
Ile	I	hydrophob
Lys	K	polar positiv
Leu	L	hydrophob
Met	M	hydrophob
Asn	N	polar neutral
Pro	P	hydrophob
Gln	Q	polar neutral
Arg	R	polar positiv
Ser	S	polar neutral
Thr	T	polar neutral
Val	V	hydrophob
Trp	W	hydrophob
Tyr	Y	hydrophob

2.3.2 Kernmagnetische Resonanzspektroskopie

Die NMR-Spektroskopie (Nuclear Magnetic Resonance) ist ein Verfahren, das basiert auf der Wechselwirkung von magnetischen Momenten von Atomkernen mit externen Magnetfeldern, elektromagnetischer Strahlung und vor allem der molekularen Umgebung der Kerne. Man legt dazu ein starkes externes Magnetfeld an, das die Atomkerne ausrichtet. Dann werden Radiowellen eingestrahlt, die diese Ausrichtung ändern. Das Ausmaß dieser Änderung läßt sich quantifizieren über die dabei absorbierte und wieder abgegebene Energie, die wiederum abhängt von der molekularen Umgebung und damit von der dreidimensionalen Struktur des Moleküls. Es gibt also einen – komplexen – Zusammenhang zwischen gemessener Absorbtion und Emission und räumlicher Struktur. Anhand dieses Zusammenhangs kann man die 3D-Struktur von Proteinen aus den Messresultaten ableiten.

2.4 Wirkstoffentwicklung

Nachdem wir verschiedene Methoden der experimentellen Strukturbestimmung vorgestellt haben, gehen wir jetzt über zu Verfahren der Wirkstoffentwicklung. Hier geht es darum, Affinität zwischen Proteinen und Rezeptoren festzustellen, also z. B. zu einem gegebenen Rezeptor ein möglichst stark bindendes Protein zu finden. Wir beschränken uns auf zwei wichtige Verfahren, die wir kurz beschreiben.

2.4.1 Phage Display

Phage Display, erstmals 1985 publiziert in [44] ist ein Verfahren, mit dem man Peptidbibliotheken enormer Größe auf ihre Bindungseigenschaften untersuchen kann. Es ist eine Anwendung des Evolutionsprinzips. Durch die Proteinbiosynthese ist es möglich, jedem Protein eine DNA-Sequenz zuzuordnen. Phage Display liegt der Gedanke zu Grunde, diese Sequenzen in selbstreplizierende Organismen zu übertragen und sie einem Selektionsprozess zu unterziehen. In der Tat ist dies das Prinzip von Phage Display. Phagen sind Viren, die Bakterien infizieren, meist das Bakterium

Escherichia coli, welches in der biochemischen Forschung und Industrie große Verwendung findet. Eine besondere Eigenschaft dieser Phagen ist, dass sie fremde DNA-Stücke in sich tragen können. Infizieren sie ein Bakterium, fügen sie ihre DNA (und somit auch die Fremd-DNA) in das Genom des Bakteriums ein. Dadurch repliziert sich diese injizierte DNA im Wirt mit der DNA des Phagen. Für den Selektionsprozess ist es auch wichtig, dass sich die fremden DNA-Stücke im Phänotyp des Phagen widerspiegeln. Hierzu werden die DNA-Stücke in Gene der Hüllenproteine des Phagen eingefügt. Dadurch wird das Peptid auf der Oberfläche des Phagen ausgeprägt (präsentiert). In Ab-

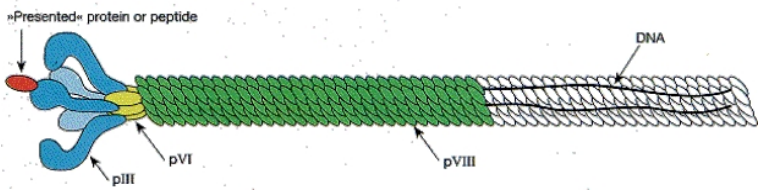


Abbildung 2.12: Phage mit „präsentiertem“ Peptid (aus [66])

bildung 2.12 ist ein M13-Phage mit verschiedenen Mantelproteinen dargestellt. Die gängigen Phage Display-Systeme verwenden die Mantelproteine pIII, pVI und pVIII zur Ausprägung der fremden Peptide. Typ3-, Typ6- und Typ8-Systeme enthalten jeweils ein verändertes Gen III/VI/VIII, so dass an jedem Mantelprotein das fremde Peptid ausgeprägt wird. Beim Typ8-System kann der Phage allerdings seine Struktur verlieren, wenn das Peptid zu groß ist. In Typ88-Systemen werden zwei VIII-Gene eingefügt, das eine verändert, das andere in seinem natürlichen Zustand. Hierdurch können auch große Proteine und Peptide an der Virenoberfläche dargestellt werden. Eine Phagenbibliothek ist ein Gemisch vieler Phagen, die jeweils unterschiedliche DNA-Stücke in sich tragen. Sie stellen im übertragenen Sinne eine Population im Evolutionsprozess dar. Gängige Bibliotheken enthalten zwischen 10^7 und 10^{10} verschiedene Sequenzen. Meist werden die Bibliotheken zufällig erzeugt, basierend auf der „Mimotopstrategie“ von Geysen et al. (vgl. [46]). Degenerierte Oligonucleotide werden hierbei als „Wildcards“ verwendet, um zufällige Sequenzen zu erzeugen. Es ist möglich, nur bestimmte Punkte im DNA-Strang zu randomisieren und somit Schemata zu erhalten. Diese Bibliotheken können dann z. B. auf verschiedene Bindungseigenschaften untersucht werden. Dazu werden sie beim so genannten Biopanning auf einen Rezeptor übertragen und die nichtbindenden Phagen abgewaschen. Danach können die erhalten gebliebenen Phagen eluiert (vom Rezeptor gelöst) und erneut repliziert werden. Dies geschieht in mehreren Iterationen mit zunehmendem Selektionsdruck (dargestellt durch die Intensität des Waschprozesses). Die am Ende der letzten Iteration übriggebliebenen Phagen können dann sequenziert und analysiert werden. Der Selektionsdruck spielt hierbei eine wichtige Rolle. Ist er zu groß, ist es möglich, dass selbst gute Individuen verloren gehen können oder z. B. Phagen, die das Lösungsmedium binden, in der Population überrepräsentiert sind. Bei der Elution unterscheidet man zwischen generellen Verfahren, die sämtliche gebundene Phagen lösen und speziellen Verfahren. Diese lösen nur die Phagen, die tatsächlich an den Rezeptor gebunden haben und verringern so die Messungenauigkeit. Hierbei wird z. B. ein bekannter Ligand verwendet, um die Phagen vom Rezeptor zu verdrängen. Diese Methode ist wünschenswerter, allerdings nicht immer durchführbar. Andere Methoden der Selektion erzeugen künstlich Mutationen, um einen größeren Teil der Fitnesslandschaft zu erkunden. Oft wird eine Greedy-Strategie gewählt, in der der bestbindende Phage repliziert und mutiert wird, oder es wird eine größere Auswahl von Phagen getroffen, um das Abgleiten in lokale Minima zu verhindern. Phage Display-Systeme werden an vielen Orten in der Biochemie erfolgreich eingesetzt. Als Rezeptoren wurden schon Antikörper, Hormonrezeptoren und sogar Kunststoffe verwendet. Die Anwendungen lassen sich in folgende Gebiete unterteilen:

Epitopmapping Hierbei wird versucht, von den Ergebnissen des Phage Displays Rückschlüsse auf die Beschaffenheit des aktiven Zentrums (Epitop) des Rezeptors zu ziehen. Liegen die aktiven Aminosäuren des Rezeptors auch in der Sequenz nah beisammen, ist dies relativ leicht möglich. Liegen sie jedoch nur in der 3D-Struktur beieinander, ist die Methode nutzlos.

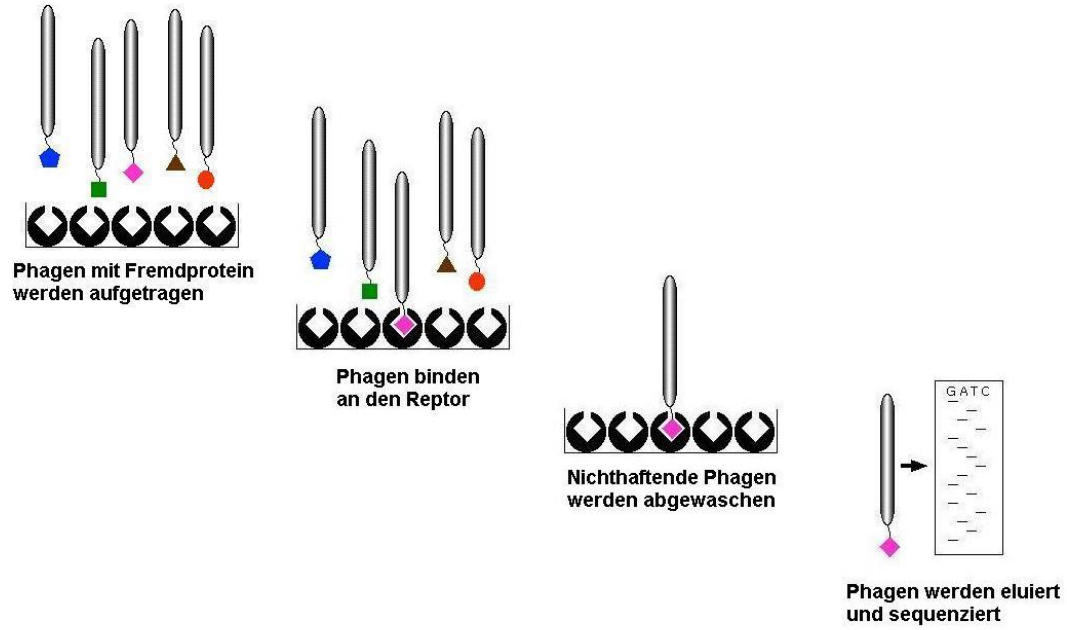


Abbildung 2.13: Phage Display

Auffindung natürlicher Liganden Die Ergebnisse des Phage Displays über einem natürlichen Rezeptor werden einem DNA-Expressionstest unterzogen. So wird festgestellt, ob ein gefundenes Peptid bereits im Organismus erzeugt werden kann und noch nicht entdeckt wurde.

Medikamentenentwicklung Viele beim Phage Display verwendete Rezeptoren sind Objekte der medizinischen Forschung und die aufgefundenen Peptide können als Rezeptoragonisten oder -antagonisten wirken. Hierbei wird das gleiche Verfahren mit einer vielfach höheren Geschwindigkeit durchgeführt, als bei gängigen chemischen Methoden. Leider sind Proteine oft nicht gut als Medikamente geeignet, da sie oft schnell von körpereigenen Enzymen zersetzt werden.

Diagnose und Impfstoffentwicklung Wird als Rezeptor ein Antikörper verwendet, so können die resultierenden Peptide im Weiteren als Indikatoren für diese Antikörper verwendet werden. In manchen Fällen können sie sogar als Impfstoffersatz verwendet werden. Dies bietet Möglichkeiten zur billigen und schnellen Entwicklung neuer Impfstoffe.

2.4.2 Peptidarrays

Protein- und Peptidarrays haben sich seit Ende der 80er Jahre zu einem aktiven Zweig der biochemischen Forschung entwickelt. Die Technik ist den DNA-Microarrays sehr ähnlich und wird heutzutage auch in der biochemischen Industrie stark genutzt. Ziel ist es, eine große Bibliothek von Proteinen parallel auf ihre Bindungseigenschaften zu einer Zielsubstanz zu untersuchen. Hierzu werden die Proteine maschinell auf eine Trägerplatte aufgetragen. Dabei wird eine Dichte von 20 bis 200 so genannter Spots pro Quadratzentimeter erreicht. Als Trägermaterial wird hauptsächlich Glas, Silizium oder Nitrozellulose verwendet. Die Platten werden mit einer Substanz überzogen, die die Proteinproben bindet. Das Problem hierbei ist, dass die Proteinproben einerseits immobilisiert werden müssen, andererseits aber nicht in ihren Reaktionsmöglichkeiten eingeschränkt werden dürfen. Die Substanzen sollten dabei die Messung möglichst nicht beeinträchtigen und nicht unspezifisch binden. Es werden kovalent und nichtkovalent bindende Stoffe verwendet. Dann

werden die fertigen Arrays mit der Zielsubstanz in Kontakt gebracht und ausgewertet. Auch hier werden wieder verschiedene Methoden genutzt. Beim so genannten Fluorescence Labelling werden die Proteine mit verschiedenen fluoreszierenden Stoffen markiert. Farbwechsel zeigen dann Reaktionen mit der Zielsubstanz an. Resonance Light Scattering verwendet die Reflektion von Laserstrahlen, um Rückschlüsse auf die Reaktionen an den einzelnen Spots zu ziehen. Die Messungen werden dann mit leistungsfähigen Detektoren durchgeführt und manuell oder mit Bildverarbeitungsprogrammen ausgewertet.

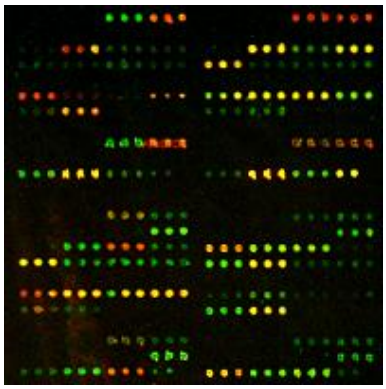


Abbildung 2.14: Peptidarray (aus [90])

Kapitel 3

All-Atomare Modelle

„Ein Physiker ist der Versuch der Atome, etwas über Atome herauszufinden."

George Wald, amerik. Biologe und Nobelpreisträger

Im folgenden Kapitel wird zuerst ein einleitender Blick auf die Geometrie eines Proteins aus mathematischer Sicht geworfen. Der Begriff Potentialfeld wird eingeführt und die Modellierung der einzelnen Teilkräfte wird erläutert. Auf der Grundlage dieser Felder wird die molekulare Dynamik als spezifische Optimierungsmethode eingeführt. CHARMM wird als praktische Anwendung kurz vorgestellt. Abschließend werfen wir einen Blick auf die Grenzen molekularer Mechanik.

3.1 Proteingeometrie

Nachfolgend werden einige geometrische Begriffe eingeführt, die ein Protein charakterisieren und in den nächsten Abschnitten verwendet werden sollen. Die räumliche Struktur eines Proteins mit n Atomen kann durch n dreidimensionale Koordinatenvektoren definiert werden:

$$\overrightarrow{x_i} = \begin{pmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \end{pmatrix}$$

Gehen zwei Atome (oder Pseudo-Atome, CH_4-Gruppen werden z. B. oft als solche behandelt) i und j eine chemische Bindung ein, lassen sich der Bindungvektor $\vec{r}_{ij}$ und die Bindungslänge r_{ij} folgendermaßen ausdrücken:

$$\vec{r}_{ij} = \vec{x}_j - \vec{x}_i$$

$$r_{ij} = \|\vec{r}_{ij}\|$$

Die Torsionswinkel θ_{ijk} und Diederwinkel (siehe Abbildung 3.1 und 3.2) ϕ_{ijkl} ergeben sich aus den Formeln

$$\cos\theta_{ijk} = \frac{\vec{r}_{ij} \cdot \vec{r}_{jk}}{r_{ij} r_{jk}}$$

$$\cos\phi_{ijkl} = \frac{(\vec{r}_{ij} \times \vec{r}_{jk}) \cdot (\vec{r}_{jk} \times \vec{r}_{kl})}{\|\vec{r}_{ij} \times \vec{r}_{jk}\| \|\vec{r}_{jk} \times \vec{r}_{kl}\|}$$

3.2 Potentialfelder - Grundlagen

Der Ansatz der Molekularen Mechanik (MM) beruht auf der Born-Oppenheimer-Näherung, nach der die Schrödingergleichung eines Moleküls in einen Teil für die Elektronbewegung und einen für die Atomkernbewegung separiert werden kann. Bei der MM wird die Komplexität reduziert, indem nur die Kernbewegung betrachtet wird. Die Moleküle werden als mechanische Einheiten

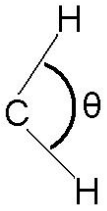


Abbildung 3.1: Bindungswinkel

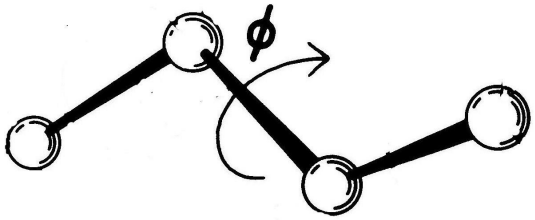


Abbildung 3.2: Diederwinkel (aus [45])

betrachtet, die man sich als fest oder mit Federn verbundene Kugeln vorstellen kann. In dieser Mechanik wirken verschiedene Kräfte, die durch verschiedene Kraftfelder simuliert werden. Aus ihnen ergibt sich die innere Energie des Moleküls. Die Felder werden der Komplexität nach in zwei Klassen eingeteilt. Klasse-I-Felder enthalten maximal quadratische Terme, während Klasse-II-Felder komplexer sind. Man unterteilt die Wechselwirkungen zwischen den einzelnen Atomen in kovalente und nicht-kovalente Kräfte. In diesem System lässt sich die Geometrie ("Konformation") eines Moleküls beschreiben durch die Bindungslängen der kovalenten Bindungen r_{ij}, durch den Bindungswinkel θ und den Diederwinkel ϕ. Ein Potentialfeld ergibt sich aus der Summe aller Kräfte, welche auf die einzelnen Atome wirken. Die Potentialfunktion bildet jede Konformation des Moleküls auf ein Potential ab.

3.2.1 Kovalente Kräfte

Die Bindungskräfte setzen sich aus der Bindungsdehnung, den Kräften der Bindungswinkel und den Kräften der Diederwinkel (Torsionsenergie) zusammen. Das Potential der Bindungsdehnung $E_{Bindungslänge}$ ist abhängig von der optimalen Bindungslänge r_0 und der tatsächlichen Bindungslänge r. Das tatsächliche Potential wird oft durch eine harmonische Näherung (Hookssches Gesetz) simuliert:

$$E_{Bindungslänge} = \frac{k_b}{2}(r - r_0)^2$$

k_b sei hier die Dehnungskonstante der Bindung. Das Morse-Potential wird ebenfalls häufig genutzt:

$$E_{Bindungslänge} = E_{min}(1 - e^{(-k(r-r_0))})^2$$

wobei E_{min} die minimale Energie ist und k eine Konstante. Auch für die Bindungswinkel θ wird

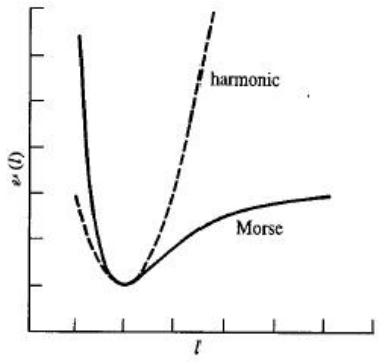


Abbildung 3.3: Vergleich zwischen Hookschem Gesetz und Morse-Potential (aus [19])

eine Näherung des harmonischen oder des Morse-Potentials verwendet. Die Torsionsenergie hat eine periodische Form, die im Zusammenhang mit der „Rotationsbarriere" steht. Diese entsteht durch Überlappung der Atome während der Rotation um die Bindungen (siehe Abbildung 3.4).

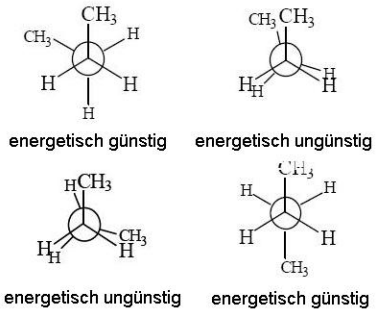


Abbildung 3.4: Verschiedene Torsionen, Beispiel

3.2.2 Nicht-kovalente Kräfte

Die modellierten nicht-kovalenten Kräfte sind die anziehende van der Waals-Wechselwirkung, die abstoßende Wechselwirkung zwischen überlappenden Elektronenhüllen (Paulisches Ausschließungsprinzip) und die elektrostatische Wechselwirkung. Die van der Waals-Wechselwirkung und die Abstoßung zwischen überlappenden Elektronenhüllen wird meist im Lennard-Jones-Potential zusammengefasst:

$$E_{vdW} = 4\epsilon \left(\frac{\sigma}{r_{ij}^{12}} - \frac{\sigma}{r_{ij}^{6}} \right),$$

wobei ϵ und σ die Lennard-Jones-Parameter sind. Sie variieren je nach interagierendem Atomtyp. Für Wasser ist z. B. $\epsilon = 0,3165nm$ und $\sigma = 0,6501\frac{kJ}{mol}$. r_{ij} sei auch hier wieder der Abstand zwischen den Atomen i und j. Die elektrostatische Energie entsteht durch die Wechselwirkung von elektrischen Ladungen. Diese werden meist als punktförmig aufgefasst. Durch das Coulomb-Potential errechnet sich dann die elektrostatische Energie aus den Ladungen q_i und q_j und der sog. Influenzkonstanten ϵ_0:

$$E_{elektrostatisch} = \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$$

3.3 Molekulare Dynamik

Wie berechnet sich nun die Konformation eines Atoms aus einem Potentialfeld? Es gibt hierzu verschiedene Ansätze. Zum einen kann die Potentialfunktion mit üblichen Black-Box-Strategien optimiert werden (siehe Kapitel 5), jedoch wird diesen Strategien meist eine langwierigere und genauere Methode vorgezogen. In der molekulare Dynamik wird die Bewegung der einzelnen Atome nach den Newtonschen Gesetzen simuliert. Die kontinuierliche Dynamik wird diskretisiert, man unterteilt sie in Zustände und Übergänge. Die Schrittweite zwischen zwei Zuständen liegt normalerweise im Femtosekundenbereich (10^{-15}s). Newtons 2. Gesetz

$$\frac{\vec{F}}{m} = \vec{a}$$

wird hier verwendet, um aus der Kraft die atomare Bewegung zu errechnen. Die Kraft, die auf ein Atom wirkt, errechnet sich als Ableitung der Potentialfunktion über der Position:

$$\vec{F} = -\nabla E$$

Anhand der folgenden Gleichungen ergeben sich Geschwindigkeit und Position des Atoms folgendermaßen:

$$\vec{v} = \int \vec{a}\, dt, \quad \vec{x} = \int \vec{v}\, dt$$

Für die Simulation kann z.B. die LeapFrog-Methode verwendet werden, die in [38] beschrieben wird.

3.4 Anwendungen

Eine Vielzahl von Simulationsprogrammen zur molekularen Dynamik wird in der Wissenschaft benutzt. Die meistverwendeten sind AMBER und CHARMM (Chemistry at HARvard Macromolecular Mechanic), GROMOS und NAMD. CHARMM ist der Marktführer und wurde 1983 von Brooks et al. (vgl. [16]) entwickelt. Es wird heute von Accelrys Inc. (www.accelrys.com) kommerziell vertrieben. CHARMM ist ein mächtiges Tool, mit dem man alle oben genannten Potentialfunktionen nutzen kann. CHARMM bietet viele Möglichkeiten der Energieminimierung, darunter verschiedene Gradientenmethoden. CHARMM bietet die Möglichkeit, ein beliebiges atomares Kraftfeld zu spezifizieren.

3.5 Grenzen

Es ist der Traum vieler Wissenschaftler, chemische Reaktionen nicht mehr im Reagenzglas herbeiführen zu müssen, sondern sie in silico simulieren zu können. Bis dahin ist es jedoch noch ein weiter Weg. Zum einen stellen die genannten Modelle nur ungenaue Näherungen an die Wirklichkeit und sogar an den Stand der modernen Physik dar. Wie überall muss man auch hier mit einen Trade-Off zwischen Berechenbarkeit und Genauigkeit leben. Ein weiteres Problem ist die Vielzahl der Konstanten, die in diesen Modellen verwendet wird. Viele Werte lassen sich zwar mit Experimenten ermitteln, aber da die Modelle nur Näherungen an die wirkliche (unbekannte) Energiefunktion sind, gibt es keine Garantie, dass die gewählten Konstanten gute Ergebnisse liefern werden. Am schwersten zu bestimmen sind nach [92] die Lennard-Jones-Konstanten und das Diederpotential. Das wichtigste Problem ist jedoch die immer noch enorme rechnerische Komplexität. Um die molekulare Dynamik akkurat simulieren zu können, benötigt man eine Auflösung von zwei Femtosekunden, d.h. eine Simulationsschrittlänge von 10^{-15} Sekunden. Ein Protein braucht, um seine natürliche Faltung zu erreichen jedoch zwischen 10^{-1} und 10^{3} Sekunden ([6]). Wie man unschwer erkennen kann, klafft immer noch eine große Lücke zwischen der zur Zeit zur Verfügung stehenden und der benötigten Rechenleistung. Einen interessanten Ansatz verfolgt hier das Folding@Home-Projekt (http://folding.stanford.edu). Auf der Webseite steht ein Client zum Download bereit, der die freie Rechenzeit des Anwender-PCs nutzt, um vom Server empfangene Tasks zu bearbeiten. Hierzu wird die eigens entwickelte „Distributed Dynamic“ [39] eingesetzt, eine parallele MD-Variante mit fast linearer Geschwindigkeitszunahme bezüglich der CPU-Zahl. Alle diese Ansätze haben jedoch eine Gemeinsamkeit. Sie eignen sich zwar dazu, die Konformation eines Proteins zu ermitteln, jedoch sind sie zu langsam, um Millionen von möglichen Proteinsequenzen zu untersuchen. Hierfür werden Modelle und Heuristiken benötigt, die schneller arbeiten.

Kapitel 4

Residuenbasierte Modelle

„Alle Dinge sind schwierig, bevor sie einfach werden."

Dr. Thomas Fuller (1654 - 1734)

Residuenbasierte Modelle verdanken ihren Namen ihrer vereinfachten Darstellung eines Proteins. Ein Protein wird nur durch wenige molekulare Untereinheiten dargestellt, die untereinander in einer Kettenstruktur verbunden sind. Es werden einfache Energiefunktionen eingesetzt und auch der Faltungsraum ist häufig in der Zahl seiner Freiheitsgrade reduziert. Diese Modelle fanden häufig Verwendung, da sie einfach zu verwenden sind und vor allem die Komplexität des Proteinfaltungsproblems auf einen Bruchteil reduzieren. Allerdings sind sie wie jede Abstraktion der Realität fehlerbehaftet. Die Modelle unterscheiden sich in der Entwicklung (anhand von allgemeinen Überlegungen oder empirischen Untersuchungen), in ihrer Komplexität und natürlich in ihrer Genauigkeit. Ein residuenbasiertes Modell besteht aus drei Komponenten, die im Folgenden vorgestellt werden:

- Die Proteinrepräsentation
- Der Faltungsraum
- Die Energiefunktion

4.1 Proteinrepräsentationen

Eine häufig verwendete Darstellung von Proteinen ist die Perlenkettendarstellung. Die anderen vorgestellten Darstellungen sind lediglich selten verwendete Erweiterungen.

4.1.1 Perlenkettendarstellung

Die Perlenkettendarstellung („beads on a string") ist eine übliche vereinfachte Repräsentation eines Proteins. Eine gesamte Aminosäure einschließlich zugehörigem Backboneteil wird als eine Einheit aufgefasst. Ein Protein vereinfacht sich hierdurch zu einer „Perlenkette" dieser so genannten „Beads" oder Kügelchen. Die Kügelchen können durch starre Kettenglieder verbunden sein, wie z. B. in den gitterbasierten Modellen. Sie können auch durch elastische „Federn" verbunden sein, wie in [63].

4.1.2 Seitenketten

Eine Erweiterung dieser „klassischen" Proteinrepräsentation stellen Strings mit Seitenketten dar. Klimov und Thirumalai (vgl. [67]) verwenden z. B. ein dreidimensionales Gittermodell, in welchem ein HP-String (H=hydrophober Rest, P=polarer Rest) mit Seitenketten gefaltet wird (siehe Abbildung 4.1). Ein Residuum wird durch zwei Kügelchen dargestellt, zum einen den verbundenen

Backbone, zum anderen die Seitenkette. Interaktionen werden nur zwischen den Seitenketten betrachtet. Dies soll die Rotationsfreiheit der Aminosäuren um den Backbone simulieren und so zu genaueren Ergebnissen führen. Wie immer ist auch dies wieder ein Trade-Off zwischen Genauigkeit und Einfachheit (und damit schneller Simulation). Ein Protein ohne Seitenketten der Länge 15 im kubischen Gitter hat 296.806 symmetrisch verschiedene Faltungsmöglichkeiten. Mit Seitenketten wächst diese Zahl auf 653.568.850 an. Trotz der gestiegenen Komplexität ist allerdings bei geringer Länge eine vollständige Enummeration aller Sequenzen möglich. In einem neueren Ansatz

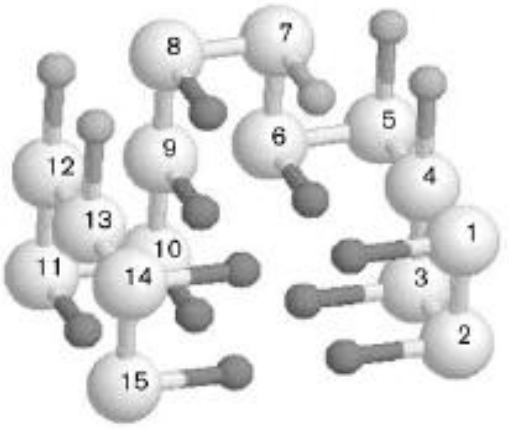


Abbildung 4.1: Protein mit Seitenketten in dreidimensionalem Gitter (aus [67])

verwenden Kussell und Shakhnovich (vgl. [36]) zur Modellierung der Seitenketten anstatt zusätzlicher Kügelchen Spinzustände, Zustände, in denen sich Residuen befinden können. Jedes Residuum kann sich entweder in seinem natürlichen Zustand oder in einem anderen Zustand befinden. Residuen interagieren hier nur, wenn sie sich beide in einem natürlichen Zustand befinden. Ist dies bei einem oder beiden Individuen nicht der Fall, so trägt der Kontakt nicht zur Gesamtenergie bei. Kussell und Shakhnovich berichten, mit diesem Modell Vorgänge ähnlich des Glasübergangs beobachtet zu haben, welche in anderen Modellen bisher noch nicht festgestellt wurden. Allerdings scheint die räumliche Interpretation unserer Ansicht nach nicht ganz schlüssig zu sein, denn durch Annahme eines nicht natürlichen Zustands kann ein Residuum z. B. ohne Interaktionen mitten im Kern des gefalteten Proteins liegen. In einer Modifikation (siehe [37]) ihres Modells kann eine Interaktion zweier Residuen auch dann zur Gesamtenergie beitragen, wenn nur ein Interaktionspartner im natürlichen Zustand ist. Der Energiebetrag ist dann allerdings nur ein Bruchteil des Betrags zweier Residuen im natürlichen Zustand.

4.2 Faltungsraum

Der Faltungsraum beschreibt den Raum, in dem sich das vereinfachte Protein bewegen kann. Es gibt hier diskrete Räume (die so genannten Gittermodelle) und kontinuierliche Räume. Diese unterscheiden sich in der Form des Proteins und den verwendeten Freiheitsgraden.

4.2.1 Das 2D-Modell

Die einfachste Form eines verwendeten Gitters wird in [32] beschrieben. Dill verwendet ein zweidimensionales quadratisches Gitter (siehe 4.2). Die maximale Anzahl von Nachbarn (verbunden und unverbunden) liegt bei vier. Bei diesem Modell ist die Komplexität noch so gut handhabbar, dass selbst lange Sequenzen durch vollständige Enumeration bestmöglichst gefaltet werden. Abhängig von der Sequenzlänge n ist die Komplexität beschränkt durch $O(3^n)$. Jedoch ist dieses Modell selbstverständlich recht weit von der Realität entfernt. Allerdings wurden auch in diesen Modellen

Eigenschaften (wie z. B. der Glasübergang) beobachtet, die auch bei realen Proteinen vorkommen. Die Hoffnung bei der Verwendung dieses Modells ist es, trotz geringster Komplexität einen Einblick in die Eigenschaften eines Proteins zu erhalten. Z. B. können sich hydrophobe Kerne auch in einer zweidimensionalen Landschaft manifestieren. Verwendung fand es unter anderem in [84, 54].

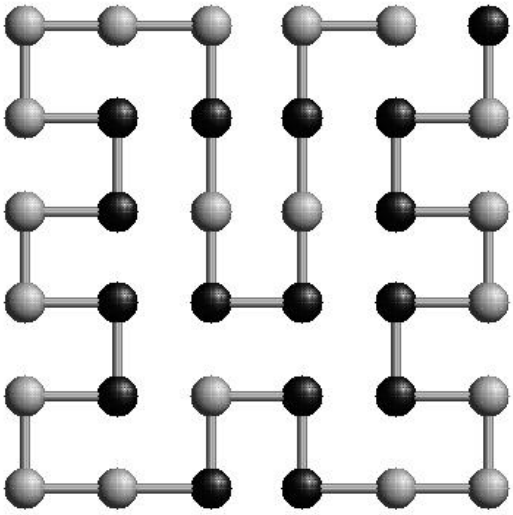


Abbildung 4.2: Zweidimensionales Gitter (aus [72])

4.2.2 Das kubische 3D-Modell

Ein einfaches Modell im dreidimensionalen Raum ist das kubische Gittermodell (siehe 4.3). Die maximale Zahl der Nachbarn eines Residuums liegt bei sechs. Möchte man hier eine vollständige Enumeration durchführen, so steigt die Komplexität auf $O(6^n)$ an. Dies wird schon für kleine Sequenzen mit heutigen Rechnern nicht mehr handhabbar. Das Modell ist realitätsnäher als die zweidimensionale Variante. Es wurde festgestellt, dass sich die Residuen tatsächlich in gitterähnlichen Strukturen anordnen. Durch eine optimale Faltung in diesem Modell kann man mehr über Form und Verhalten realer Proteins erfahren. Allerdings bleibt zu beiden bisher vorgestellten Gittern anzumerken, dass sie eher aus allgemeinen Überlegungen entstanden, als aus empirischen Untersuchungen. Das kubische Modell ist das meistbenutzte Gittermodell, wohl auch deshalb, weil es sich einfach implementieren und darstellen lässt (siehe z. B. [64]).

4.2.3 Das Jernigan-Modell

Das dreidimensionale Gitter von Raghunathan und Jernigan [82] beschreibt die Form eines Kubooktaeders (siehe 4.4). Jedes Residuum hat zwölf mögliche Nachbarn. Die Form des Gitters lässt sich folgendermaßen definieren:

p_0 ist Gitterpunkt, dann sind folgende Punkte auch Gitterpunkte:

$$p_{1-4} = p_0 + \begin{pmatrix} \pm 1 \\ \pm 1 \\ 0 \end{pmatrix}, \; p_{5-8} = p_0 + \begin{pmatrix} \pm 1 \\ 0 \\ \pm 1 \end{pmatrix}, \; p_{9-12} = p_0 + \begin{pmatrix} 0 \\ \pm 1 \\ \pm 1 \end{pmatrix}$$

Eine Einheit des Gitters entspricht 4.5 Å, so dass der Abstand zwischen zwei Residuen 6.5 Å beträgt. Raghunathan und Jernigan gingen empirisch vor, um die Form dieses Gitters zu bestimmen. Sie untersuchten die Anzahl der Nachbarn von Residuen in experimentell bestimmten

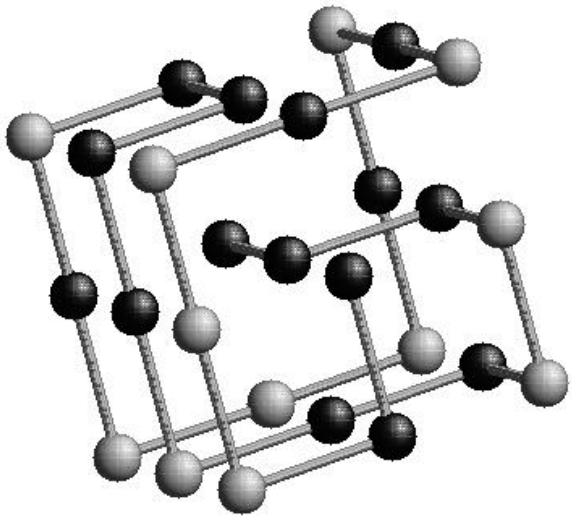


Abbildung 4.3: Dreidimensionales kubisches Gitter (aus [72])

Proteinstrukturen. Es stellte sich heraus, dass die Residuen sehr genau zwölf Nachbarn hatten. Wir verwenden das Jernigan-Gitter aus diesem Grund. Die Komplexität einer vollständigen Enumeration wird allerdings nur noch durch $O(11^n)$ beschränkt. Dadurch entziehen sich interessante Sequenzlängen komplett einer effizienten fehlerfreien Betrachtung.

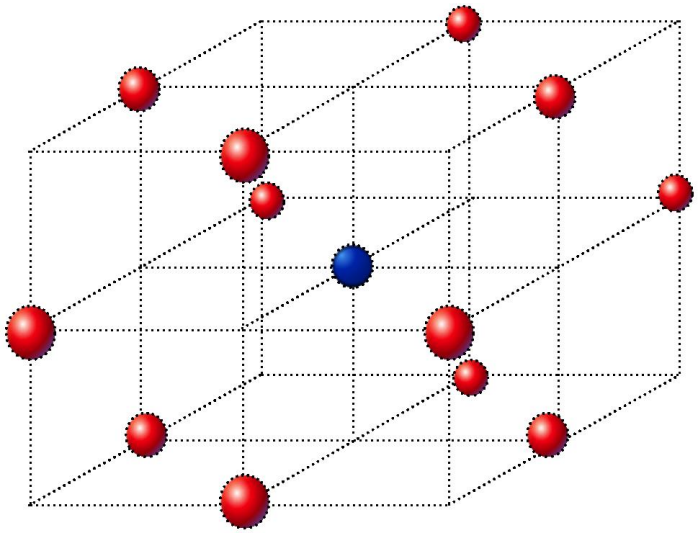


Abbildung 4.4: Dreidimensionales Gitter nach Jernigan

4.2.4 Kontinuierliche Modelle

Kontinuierliche Modelle sind die komplexesten aller möglichen residuenbasierten Modelle. Residuen haben die Möglichkeit, beliebige Positionen im Raum einzunehmen. Manche Anwendungen beschränken aber z. B. die Bindungslänge zwischen zwei Aminosäuren auf einen bestimmten Wert oder Bereich, so dass hier die Analogie zur Perlenkette, die im Raum zusammengelegt wird, sehr treffend ist. Verwendet z. B. in [87, 63] sind diese Modelle wahrscheinlich die realitätsnächsten auf Residuenbasis. Allerdings sind sie wegen ihrer hohen Zahl von Freiheitsgraden schwierig zu

optimieren.

4.3 Energiefunktionen

Die hier vorgestellten Potentialfunktionen versuchen, die tatsächlichen Kräfte zwischen den einzelnen Atomen des Proteins durch Kontaktenergien zwischen einzelnen Residuen auszudrücken. Einige Funktionen sind für gitterbasierte Modelle bestimmt: sie geben ein Potential ausschließlich für eine feste Distanz der Aminosäuren an. Andere Potentialfunktionen beziehen den Abstand der Residuen mit ein. Diese Modelle sind auch für die kontinuierliche Faltung verwendbar. Oft versucht man, die Energiefunktion zu minimieren; in diesem Sinne wird die Energie häufig behandelt wie die freie Enthalpie, also wie die thermodynamische Größe, die in der Natur unter physiologischen Bedingungen minimiert wird.

4.3.1 Das HP-Potential

Das HP-Modell von Chan und Dill [32] arbeitet mit lediglich zwei verschiedenen Residuentypen. Dies ist zum einen die Gruppe der hydrophoben Aminosäuren, und zum anderen die Gruppe der polaren. Das Energiepotential ist ebenfalls sehr simpel gehalten. Lediglich hydrophobe Kontakte werden bewertet, diese negativ. Hiermit soll dem hydrophoben Effekt Rechnung getragen werden, nach dem sich solche Residuen gerne zusammenlagern, um ihre Kontaktfläche mit Wasser zu minimieren (siehe Abschnitt 2.2).

Tabelle 4.1: Das HP-Potential (nach [32])

	H	P
H	-1	0
P	0	0

4.3.2 Das HPNX-Potential

Backofen stellt in [83] eine Erweiterung des HP-Modells vor. Er unterteilt die 20 Aminosäuren in vier Klassen. Es wird nicht nur der hydrophobe Effekt in Betracht gezogen, sondern auch die elektrischen Ladungen der polaren Residuen. Das Alphabet erweitert er, indem er die Klasse P in die Klassen P, N und X aufteilt. Diese stellen positive (P), negative (N) und neutrale (X) polare Residuen dar. Positive resp. negative stoßen einander ab, ziehen aber den jeweils anderen Ladungstyp an. Da dieser Effekt nicht so stark wie der hydrophobe ist, wird letzterer viermal stärker bewertet.

Tabelle 4.2: Das HPNX-Potential[83]

	H	P	N	X
H	-4	0	0	0
P	0	1	-1	0
N	0	-1	1	0
X	0	0	0	0

Backofen hält hier die Komplexität, die ein großes Alphabet mit sich bringt, noch im Rahmen. Allerdings argumentiert er, dass mit diesem Modell die Charakteristika der realen Aminosäuren besser getroffen werden, die ebenfalls in hydrophobe, polar-positive, polar-negative und polar-neutrale Gruppen eingeteilt werden können. Durch diese Verfeinerung des Potentials sinkt die Zahl der Konfigurationen mit gleichem Energieniveau, so dass die Suchlandschaft kontinuierlicher und damit realistischer wird.

4.3.3 Das empirische Modell von Miyazawa und Jernigan

Anders als z.B. Dill et al. [32] wählten Miyazawa und Jernigan eine empirische Herangehensweise, um die Kontaktpotentiale zu bestimmen (siehe Abschnitt 4.3). Sie teilten die Aminosäuren nicht in Klassen wie z. B. hydrophob und polar auf, sondern betrachteten die Wechselwirkungen aller 20 Aminosäuren (vgl. [77, 78]). Hierzu untersuchten sie 1.168 Proteinstrukturen der Brookhaven Protein Data Bank (PDB) und insgesamt 113.914 Residuenkontakte. Sie betrachteten zwei Residuen als miteinander in Kontakt stehend, wenn ihre Distanz unter 6.5 Å liegt. Auch hier lassen sich gewisse Tendenzen erkennen, die schon in den vorangegangenen Potentialfunktionen deutlich geworden sind. So tritt das niedrigste Potential bei einem Leucin-Leucin-Kontakt auf. Leucin gilt als eine der hydrophobsten Aminosäuren. Das Maximum wird bei Kontakten zwischen Lysin und Lysin erreicht. Diese Aminosäure ist als stark polar bekannt. Auch hier ist also wieder die Hydrophobizität die treibende Kraft. Allerdings ist es nach [78] nicht möglich, den Effekt aus den Kontaktpotentialen der Aminosäuren zu separieren. Eine Einteilung in Klassen wie H und P würde nach Miyazawa und Dill das Modell nicht nur vereinfachen, sondern auch verfälschen.

4.3.4 Gausssches HP-Potential

Das Gausssche HP-Modell (vgl. [11]) ist im Gegensatz zu den vorangegangenen Modellen nicht gitterbasiert. Die einzelnen Residuen werden als Kugeln aufgefasst, die durch Federn (kovalente und nonkovalente Bindungen) verbunden sind. Dadurch wird das Modell reellwertig und eine vollständige Enumeration des Suchraums unmöglich. Auch hier werden die Aminosäuren in die Klassen H und P unterteilt. Die Eingabeparameter der zu optimierenden Funktion sind dreidimensionale Vektoren (die Positionen der Residuen). Die Energiefunktion ergibt sich aus der Summe verschiedener Potentialfelder:

$$
\begin{aligned}
E &= E_{kov} + E_{nonkov} + E_{HP} + E_{el} \\
E_{kov} &= \sum_{i<n} a^{kov}(r_{i,i+1} - \overline{r}_{i,i+1})^2 \\
E_{nonkov} &= \sum_{i<j} a^{stat}_{i,j} r^2_{i,j} \\
E_{HP} &= \sum_{i} a^{H/P}_{i} r^2_{i} \\
E_{el} &= \sum_{i,j} a^{el}_{i,j}(r_{i,j} - \overline{r}_{i,j})^2
\end{aligned}
$$

$r_{i,j}$ sei die Distanz zwischen den Residuen i und j, r_i sei die Distanz zwischen Atom i und dem Mittelpunkt des Moleküls. E_{kov} stellt die Bindungsstreckung dar (in der Federanalogie die stärkste Verbindung), $\overline{r}_{i,i+1} = 3.8$Å ist hier die optimale Bindungslänge. E_{nonkov} stellt die nonkovalenten Kräfte zwischen den Residuen dar. E_{HP} soll die hydrophobe Kernbildung unterstützen. Die Funktion wird also minimiert, wenn hydrophobe Residuen im Kern und polare in den äußeren Regionen des Proteins zu finden sind. Dieser Ansatz geht über die Modellierung des herkömmlichen HP-Modells der wechselseitigen Anziehung zwischen H-Residuen hinaus. Letzteres kann z. B. durch verschiedene $a^{nonkov}_{i,j}$ dargestellt werden. E_{el} bietet die Möglichkeit, Wissen über die nichtkovalente Bindungen (z. B. Disulfidbrücken oder bekannte Sekundärstrukturen) zwischen Residuen einfließen zu lassen. [63] verwendeten eine Gittervariante dieses Modells mit einer Bindungslänge von 3.8Å.

4.3.5 Andere Potentiale

Woese et al. [43] entwickelten früh ein residuenbasiertes Energiepotential. Es arbeitet auf dem Alphabet der 20 Aminosäuren. Zur Energieberechnung wird die Polarität der einzelnen Aminosäuren

Tabelle 4.3: Kontaktpotentiale nach Miyazawa und Jernigan ([78])

	Cys	Met	Phe	Ile	Leu	Val	Trp	Tyr	Ala	Gly	
Cys	-5,44	-4,99	-5,80	-5,50	-5,83	-4,96	-4,95	-4,16	-3,57	-3,16	Cys
Met		-5,46	-6,56	-6,02	-6,41	-5,32	-5,55	-4,91	-3,94	-3,39	Met
Phe			-7,26	-6,84	-7,28	-6,29	-6,16	-5,66	-4,81	-4,13	Phe
Ile				-6,54	-7,04	-6,05	-5,78	-5,25	-4,58	-3,78	Ile
Leu					-7,37	-6,48	-6,14	-5,67	-4,91	-4,16	Leu
Val						-5,52	-5,18	-4,62	-4,04	-3,38	Val
Trp							-5,06	-4,66	-3,82	-3,42	Trp
Tyr								-4,17	-3,36	-3,01	Tyr
Ala									-2,72	-2,31	Ala
Gly										-2,24	Gly
Thr											Thr
Ser											Ser
Asn											Asn
Gln											Gln
Asp											Asp
Glu											Glu
His											His
Arg											Arg
Lys											Lys
Pro											Pro
	Thr	Ser	Asn	Gln	Asp	Glu	His	Arg	Lys	Pro	
Cys	-3,11	-2,86	-2,59	-2,85	-2,41	-2,27	-3,60	-2,57	-1,95	-3,07	Cys
Met	-3,51	-3,03	-2,95	-3,30	-2,57	-2,89	-3,98	-3,12	-2,48	-3,45	Met
Phe	-4,28	-4,02	-3,75	-4,10	-3,48	-3,56	-4,77	-3,98	-3,36	-4,25	Phe
Ile	-4,03	-3,52	-3,24	-3,67	-3,17	-3,27	-4,14	-3,63	-3,01	-3,76	Ile
Leu	-4,34	-3,92	-3,74	-4,04	-3,40	-3,59	-4,54	-4,03	-3,37	-4,20	Leu
Val	-3,46	-3,05	-2,83	-3,07	-2,48	-2,67	-3,58	-3,07	-2,49	-3,32	Val
Trp	-3,22	-2,99	-3,07	-3,11	-2,84	-2,99	-3,98	-3,41	-2,69	-3,73	Trp
Tyr	-3,01	-2,78	-2,76	-2,97	-2,76	-2,79	-3,52	-3,16	-2,60	-3,19	Tyr
Ala	-2,32	-2,01	-1,84	-1,89	-1,70	-1,51	-2,41	-1,83	-1,31	-2,03	Ala
Gly	-2,08	-1,82	-1,74	-1,66	-1,59	-1,22	-2,15	-1,72	-1,15	-1,87	Gly
Thr	-2,12	-1,96	-1,88	-1,90	-1,80	-1,74	-2,42	-1,90	-1,31	-1,90	Thr
Ser		-1,67	-1,58	-1,49	-1,63	-1,48	-2,11	-1,62	-1,05	-1,57	Ser
Asn			-1,68	-1,71	-1,68	-1,51	-2,08	-1,64	-1,21	-1,53	Asn
Gln				-1,54	-1,46	-1,42	-1,98	-1,80	-1,29	-1,73	Gln
Asp					-1,21	-1,02	-2,32	-2,29	-1,68	-1,33	Asp
Glu						-0,91	-2,15	-2,27	-1,80	-1,26	Glu
His							-3,05	-2,16	-1,35	-2.25	His
Arg								-1,55	-0,59	-1,70	Arg
Lys									-0,12	-0,97	Lys
Pro										-1,75	Pro

Tabelle 4.4: Beispielparameter für das Gausssche HP-Modell (nach [63])

Parameter	Relative Stärke
a^{kov}	6.0
a^{nonkov}	0.05
a^{H}	0.1
a^{P}	-0.1
a^{el}	0–5.0

genutzt. Jeder Aminosäure wird ein Polaritätswert w_i zugeordnet. Die Interaktionsenergie ergibt sich dann wie folgt:

$$e_{ij} = \begin{cases} \frac{1}{d_{ij}} - (\frac{1}{w_i} + \frac{1}{w_j}) - 1 & d_{ij} < 1 \\ -(\frac{1}{w_i} + \frac{1}{w_j})e^{1-d_{ij}} & d_{ij} \geq 1 \end{cases}$$

Miller [76] verwendete ebenfalls empirisch ermittelte Hydrophobizitätswerte, um die Kontaktpotentiale zu gewinnen.

$$e_{ij} = \begin{cases} \frac{1}{d_{ij}} + (m_i + m_j) - 1 & d_{ij} < 1 \\ (m_i + m_j)e^{1-d_{ij}} & d_{ij} \geq 1 \end{cases}$$

Die Modelle sind wie das Gausssche HP-Modell nicht gitterbasiert. Beide Varianten entsprechen in der Komplexität dem Modell von Jernigan und Myiazawa, haben aber den Nachteil, dass die Kontaktpotentiale nicht direkt empirisch ermittelt wurden, sondern aus den Hydrophobizitäts- und Polaritätswerten ermittelt wurden. Bei gleicher Komplexität sollten sie daher dem Modell von Jernigan und Miyazawa unterlegen sein. Deswegen werden wir diese Modelle im Weiteren nicht mehr betrachten.

Tabelle 4.5: Potentiale nach Woese und Miller [76, 43]

Name	Abk.	Symbol	Woese w_i	Miller m_j
Alanin	Ala	A	7.0	-0.20
Arginin	Arg	R	9.1	1.34
Asparagin	Asn	N	10.0	0.69
Asparaginsäure	Asp	D	13.0	0.72
Cystein	Cys	C	4.8	-0.67
Glutamin	Gln	Q	8.6	0.74
Glutaminsäure	Glu	E	12.5	1.09
Glycin	Gly	G	7.9	-0.06
Histidin	His	H	8.4	-0.04
Isoleucin	Ile	I	4.9	-0.74
Leucin	Leu	L	4.9	-0.75
Lysin	Lys	K	10.1	2.00
Methionin	Met	M	5.3	-0.71
Phenylalanin	Phe	F	5.0	-0.67
Prolin	Pro	P	6.6	-0.44
Serin	Ser	S	7.5	0.34
Threonin	Thr	T	6.6	0.26
Tryptophan	Trp	W	5.2	-0.45
Tyrosin	Tyr	Y	5.4	0.22
Valin	Val	V	5.6	-0.61

Tabelle 4.6: Energiefunktionen auf einen Blick

Modell	Alphabet	Alphabetgröße	Typ
HP	{H,P}	2	Gitter
HPNX	{H,P,N,X}	4	Gitter
Miyazawa, Jernigan	Alle Aminosäuren	20	Gitter
Gausssches HP	{H,P}	2	Kontinuierlich
Miller, Woese	Alle Aminosäuren	20	Kontinuierlich

Kapitel 5

Suchheuristiken, Optimierung

> „If the minimum wasn't acceptable it wouldn't be called the minimum."
>
> George Muncaster

In diesem Kapitel stellen wir einige randomisierte Suchverfahren vor. Bei dieser Klasse von Algorithmen handelt es sich um Verfahren, die häufig zu Optimierung verwendet werden. Ihre Entscheidungen über den jeweils nächsten Suchschritt hängen von (Pseudo-)Zufallsbits ab (randomisiert) und man kann keine Garantie über Güte oder Rechenzeit (heuristisch) abgeben. Zum einen werden wir die Gruppe der Monte Carlo Strategien, eine Gruppe häufig verwendeter Suchheuristiken, vorstellen. Zum anderen befassen wir uns mit der Gruppe der evolutionären Algorithmen (EA) und aus dieser Gruppe speziell mit genetischen Algorithmen (GA), genetischer Programmierung (GP) und Evolutionstrategien (ES), die wir für das uns vorliegende Optimierungsproblem verwenden wollen. Einen vollständigeren Überblick über die Klasse der evolutionären Algorithmen findet man in [18, 31, 38].

Zunächst wollen wir jedoch kurz darauf eingehen, was wir unter einem Optimierungsproblem verstehen und dabei einige notwendige Begrifflichkeiten einführen.

5.1 Optimierungsprobleme

Jeder Mensch trifft täglich eine Reihe von Entscheidungen, bei denen er aus mehreren Möglichkeiten auswählen muss. In den meisten Fällen schließen sich dabei die einzelnen Möglichkeiten gegenseitig aus und haben unterschiedliche Folgen. So ergibt sich z. B. häufig das Problem, den besten Weg von einem Punkt A, z. B. der Heimatstadt, zu einem Punkt B, z. B. dem Urlaubsort, zu finden. Hier möchte man zum einen einen kurzen Weg wählen. Zum anderen soll dieser aber nicht nur kurz, sondern auch schnell sein. Für dieses Problem sind mehrere gute Lösungen denkbar:

- Der kürzeste Weg,
- der schnellste Weg unabhängig von der zurückgelegten Strecke,
- eine Menge von Wegen, die zwar nicht die kürzesten Wege sind, dafür aber schneller als der kürzeste Weg sind.

Auf der Suche nach der richtigen Entscheidung hat der Entscheidungsträger dabei meistens mehrere Ziele (wie im Beispiel Streckenlänge und Reisedauer) zu bedenken. In der Regel handelt es sich bei diesen Zielen um gegensätzliche Ziele. Es gibt also keine Entscheidung, die bezüglich aller Kriterien bzw. Ziele optimal ist. Um trotzdem eine Entscheidung zu treffen, muss man zwischen den einzelnen Zielen abwägen. Es existieren dafür einige gängige Strategien, um die relative Wichtigkeit der Ziele in den Optimierungs- und Entscheidungsfindungsprozess einfließen zu lassen:

Ohne Präferenzen Man akzeptiert die Entscheidung, die die Methode nach unbekannten Präferenzen ausgewählt hat.

A Priori festgelegte Präferenzen Vor der eigentlichen Optimierung werden die einzelnen Ziele gewichtet.

Progressiv festgelegte Präferenzen Nach jeder Iteration des Optimierungsprozesses werden die Ziele neu formuliert.

A Posteriori festgelegte Präferenzen Alle nichtdominierten (siehe Abschnitt 5.4) Punkte werden ermittelt und aus ihnen dann eine Lösung ausgewählt.

Eine ausführliche Beschreibung der einzelnen Strategien ist zu finden in [55]. Daneben sind auch beliebige Hybride denkbar, vgl. Abschnitt 9.11 für eine von uns verwendete Hybridstrategie.

In der heutigen Zeit kann sich der Entscheidungsträger bei der Suche nach der richtigen Entscheidung durch die Errungenschaften der Entscheidungstheorie, in der versucht wird, das Entscheidungsproblem mathematisch zu modellieren und zu lösen, unterstützen lassen. Eine dabei häufig verwendete Form der Modellierung des Entscheidungsproblems ist die Darstellung des Problems als Vektoroptimierungsproblem [14].

DEFINITION **5.1.** *Vektoroptimierungsproblem*

Gegeben seien n *Entscheidungsvariablen* $x_1, \ldots, x_n$ *mit* $x = (x_1, \ldots, x_n) \in \mathbb{R}$,
m *Zielfunktionen* $f_1, \ldots, f_m$ *mit* $f = (f_1, \ldots, f_m) : \mathbb{R}^n \to \mathbb{R}^m$ *und*
k *Nebenbedingungen* $g_1, \ldots, g_k$, *mit* $k, n, m \in \mathbb{N}$.
O. B. d. A.[1] *müssen alle* m *Zielfunktionen unter Einhaltung der Nebenbedingungen minimiert werden:*
minimiere $f(x) = (f_1(x), \ldots, f_m(x))$
unter den Nebenbedingungen $g_i \sim 0 \ \forall \ i \in \{1, \ldots, k\}$ *und* $\sim \in \{<, \leq, =, \geq, >\}$
Die Menge X *aller zulässigen Entscheidungsvektoren heißt Entscheidungsraum oder Suchraum und die ihr zugeordnete Menge* $F = f(X)$ *von Zielfunktionsvektoren heißt Lösungsraum.*

Beim Lösungsraum handelt es sich um eine Teilmenge des $\mathbb{R}^m$. Da auf dem $\mathbb{R}^m$ und damit auch auf dem Lösungsraum keine Totalordnung definiert ist, kann man gerade bei gegensätzlichen Zielen meistens kein eindeutiges Optimum identifizieren. Werden die Zielfunktionen zu einer einzelnen Zielfunktion zusammengefasst, so existiert ein Optimum. Hierfür wird jedoch eine Gewichtung der einzelnen Ziele benötigt. Das Zusammenfassen kann erschwert werden, wenn die einzelnen Ziele inkommensurabel (nicht in gleicher Einheit messbar) sind [18].

Um optimale Lösungen zu identifizieren ohne die einzelnen Ziele untereinander in Beziehung setzen zu müssen, definierte Francis Ysidro Edgeworth (1881) einen neuen Optimalitätsbegriff, der später von dem italienischen Wirtschaftswissenschaftler Vilfredo Pareto (1896) generalisiert wurde und seitdem als Paretooptimalttät bekannt ist. Das Konzept der Paretooptimalität basiert auf der Vektorhalbordnung:

DEFINITION **5.2.** *Vektorhalbordnung*

Gegeben seien zwei Vektoren $f' = (f'_1, \ldots, f'_m)$, $f'' = (f''_1, \ldots, f''_m) \in \mathbb{R}^m$, $m \in \mathbb{N}$. *Dann ist Vektor* $f' \leq f''$, *wenn gilt:*

$$f'_i \leq f''_i \ \forall i \in \{1, \ldots, m\}$$

Die Dominanzrelation „$\prec$“ lässt sich von der auf den Lösungsraum übertragenen Vektorhalbordnung ableiten:

[1] Aufgrund des Dualitätsprinzips lassen sich alle Maximierungsprobleme in äquivalente Minimierungsprobleme umwandeln.

DEFINITION **5.3.** *Dominanzrelation*

Gegeben seien zwei Entscheidungsvektoren eines Vektoroptimierungsproblems $x^{'} = (x_1^{'}, \ldots, x_n^{'})$, $x^{''} = (x_1^{''}, \ldots, x_n^{''}) \in \mathbb{R}^n$, $n \in \mathbb{N}$. $x^{'} \succ x^{''}$ *($x^{'}$ dominiert $x^{''}$), wenn für die Zielvektoren* $f(x^{'})$, $f(x^{''})$ *gilt:*

$$f_i(x^{'}) \leq f_i(x^{''}) \text{ mit } i = 1, \ldots, m,$$

$$f(x^{'}) \neq f(x^{''}).$$

Mit Hilfe der Dominanzrelation lassen sich dann die paretooptimalen Entscheidungsvektoren identifizieren:

DEFINITION **5.4.** *Paretooptimalität*

Ein Entscheidungsvektor x^* *eines Vektoroptimierungsproblems heißt paretooptimal, wenn gilt:*

$$\nexists\, x \in X \text{ mit } x \succ x^*$$

Zu den meisten Vektoroptimierungsproblemen existiert nicht nur ein paretooptimaler Vektor, sondern eine größere Anzahl von Vektoren, die so genannte Paretomenge:

DEFINITION **5.5.** *Paretomenge*

Die Menge aller paretooptimalen Entscheidungsvektoren eines Vektoroptimierungsproblems

$$X^* = \{x^* \in X | \nexists\, x \in X : x \succ x^*\}$$

heißt Paretomenge dieses Vektoroptimierungsproblems.

Wenn man also z. B. ein Auto kaufen möchte, sucht man ein Fahrzeug, das möglichst viel Komfort bietet, dabei aber möglichst billig ist. Da die komfortabelsten Autos jedoch auch die teuersten Autos sind und die billigsten Autos nie die komfortabelsten Autos sind, findet man kein Auto, das bezüglich beider Ziele optimal ist. Zu jeder Preiskategorie gibt es jedoch Autos, die, verglichen mit den anderen Autos dieser Preiskategorie, den meisten Komfort bieten. Die komfortabelsten Autos für einen bestimmten Preis bzw. die billigsten Autos mit einem bestimmten Grad an Komfort sind die interessanten Alternativen für den Autokauf. Sie entsprechen der Paretomenge dieses Optimierungsproblems. Die Entscheidungsvariablen sind in diesem Fall der Preis und der Grad an Komfort. Die Zielfunktionen sind einfache lineare Funktionen. Der Preis entspricht der einen Zielfunktion, der Komfort entspricht der anderen Zielfunktion.

5.2 Black Box-Szenario

Bei diesem Szenario handelt es sich um einen Spezialfall von Optimierungsproblemen. Auch bei diesen Problemen hat man einen Eingabevektor von Entscheidungsvariablen $\vec{x} = (x_1, \ldots, x_n)$, für die die verschiedenen Zielfunktionswerte $\vec{y} = (y_1, \ldots, y_m)$ ermittelt werden. Es hat die Form:

$$F(\vec{x}) = \vec{y}, \text{ dabei ist } F \text{ die Menge der Zielfunktionen.}$$

Die Vorgehensweise des Moduls, das die Bewertung der einzelnen Entscheidungsvektoren übernimmt, ist unbekannt. In der Praxis ist dies der Fall, wenn man die der Optimierung zugrunde liegenden Begebenheiten nicht genau kennt und deswegen z. B. durch Experimente die Fitness einzelner Entscheidungsvektoren ermitteln muss. Es ist auch möglich, dass die Funktion zwar bekannt aber zu komplex ist und sich so einer Analyse entzieht. Ein solches Szenario kann man sich wie in Abbildung 5.1 dargestellt vorstellen.

In diesem Szenario kann das Bewertungsmodul lediglich mit einer Eingabe versorgt werden, evtl. einige Parameter eingestellt werden und im Anschluss die Ausgabe beobachtet werden.

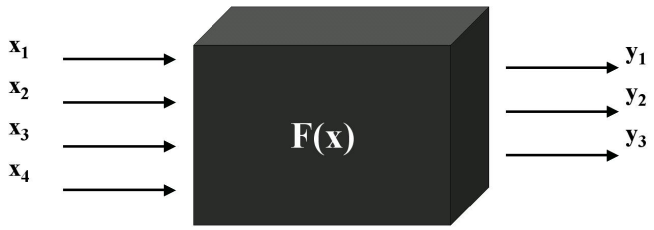


Abbildung 5.1: Schematische Darstellung einer Black Box

Hier kommen die randomisierten Algorithmen zum Einsatz. Auf der Suche nach dem geeigneten Algorithmus stellt sich die Frage, ob es ein Verfahren in der Menge der randomisierten Suchheuristiken gibt, das alle Probleme sehr gut löst. Das No Free Lunch Theorem von Wolpert und Macready [29] macht dazu eine erste Aussage.

THEOREM **5.1.** *No Free Lunch Theorem*

Für die Menge der Probleme F über einem endlichen Suchraum S und einem vollständig geordneten Wertebereich W, bei denen nur die Funktionsauswertung Kosten verursacht und kein Punkt zweimal ausgewertet wird, gilt:

$$\textit{Für zwei Algorithmen } A \textit{ und } A' \textit{ gilt } A_{S,W} = A'_{S,W},$$

wobei $A_{S,W}$ ($A'_{S,W}$) die durchschnittliche Anzahl an Zielfunktionsauswertungen über allen Problemen $f \in F$ für den Algorithmus A (A') bezeichnet.

Im Schnitt über alle Funktionen sind alle Algorithmen gleich gut. Ein sehr großer Anteil aller Probleme wird jedoch in der Praxis nicht betrachtet. Auch für die Klasse der tatsächlich zu lösenden Probleme gibt es eine ähnlich schlechte Prognose. Droste, Jansen und Wegener stellen in ihrem Almost No Free Lunch Theorem [89] fest, dass es für jedes von einem Verfahren gut gelöste Problem doppelt exponentiell viele sehr ähnliche Probleme gibt, bei denen das angewendete Verfahren sehr schlechte Ergebnisse erzielt.

Für unser Black Box Szenario kann es kein einzelnes bestes Verfahren geben. Es bleibt jedoch die Hoffnung, dass das Feld der randomisierten Suchheuristiken in seiner Gesamtheit ein gutes Reservoir ist, um zu vielen relevanten Problemen einen passenden Algorithmus zu finden.

5.3 Monte Carlo-Strategien

Verfahren dieser Gruppe werden oft für die Optimierung von Problemstellungen der verschiedensten Anwendungen genutzt [53, 52, 73]. Bei diesen Algorithmen handelt es sich um eine einfache Erweiterung einer randomisierten Suche.

Die randomisierte lokale Suche (auch hill climbing genannt) arbeitet mit einem einzelnen Suchpunkt, in dessen Nachbarschaft (z. B. alle Punkte, die durch eine Mutation des Suchpunkts erreichbar sind) nach mindestens genauso guten oder besseren Punkten gesucht wird. Sollte ein solcher Punkt gefunden werden, wird er als neuer Suchpunkt akzeptiert. Dieses Vorgehen macht deutlich, dass es bei einer randomisierten Suche das Problem gibt, dass der Suchprozess in lokalen Minimas stoppt, da ein solches Minimum nicht mehr verlassen werden kann. Sonst müssten zwischenzeitlich auch schlechtere Punkte akzeptiert werden.

Um das Problem der lokalen Minima zu umgehen, akzeptiert der Metropolis Algorithmus unter bestimmten Umständen auch einen schlechteren Punkt als neuen Suchpunkt. Dabei ist ausschlaggebend, wieviel schlechter der neue Punkt x' gegenüber dem aktuellen Punkt x ist. Die Metropolisfunktion

$$M(x, x', T) = e^{\frac{-(f(x')-f(x))}{T}} \text{ mit } T > 0$$

gibt dabei an, mit welcher Wahrscheinlichkeit ein solcher Punkt akzeptiert wird. $f(x)$ und $f(x')$ sind dabei die Zielfunktionswerte von x und x'. T ist eine normierende Größe, wobei aus $T \to 0$, $M \to 0$ und aus $T \to \infty$, $M \to 1$ folgt. Wenn man T zu klein wählt, hat man das gleiche Problem wie bei einer randomisierten lokalen Suche, wenn man T jedoch zu groß wählt, verhält sich der Metropolis Algorithmus wie eine ungesteuerte Suche.

Da man davon ausgeht, dass ein dynamischer, adaptiver Parameter T das Suchverhalten positiv beeinflusst, wurde Simulated Annealing von Kirkpatrick, Gelatt und Vecchi [65] entwickelt. Man nimmt dabei an, dass es sinnvoll sei, zu Anfang ein größeres T zu verwenden, um den vorhandenen Suchraum möglichst gut zu explorieren und dann in einer vielversprechenden Gegend sukzessive T zu senken, um dort das Minimum zu finden. Um T anzupassen, gibt es verschiedene Strategien. Meistens wird T durch eine von der Zeit abhängige Funktion verändert. Hierfür werden z. B. lineare, logarithmische, exponentielle oder sigmoide Funktionen verwendet. Um einen Metropolis Algorithmus zu erhalten, würde man eine konstante Funktion für T benutzen. Zum Teil werden adaptive Verfahren eingesetzt [58], bei denen T auch wieder erhöht werden kann, um lokale Minima wieder zu verlassen. In manchen Fällen wird T nicht in Abhängigkeit von der Iterationszahl des Algorithmusses sondern in Abhängigkeit von der Dauer seit der letzten Senkung oder des letzten akzeptierten Schritts verändert. Bis jetzt wurde kein nicht konstruiertes Problem gefunden, das die Überlegenheit von Simulated Annealing gegenüber dem Metropolis Algorithmus belegt [58].

Bei der Namensgebung ließen sich die Entwickler durch die Analogie zu einem Annealing Verfahren (ausglühen oder kühlen) inspirieren. Bei diesen Verfahren werden besonders reine Kristalle durch Abkühlen aus heißen Rohmaterialien gewonnen. T steht dabei für die Temperatur. Wenn man das Rohmaterial optimal abkühlt, haben die einzelnen Moleküle genug Zeit, um sich an die richtigen Gitterpositionen im Kristallgitter zu bewegen, werden dann aber durch die mittlerweile zu niedrige Temperatur daran gehindert, sich weiterhin zu bewegen.

Bei Verwendung von Simulated Annealing oder dem Metropolis Algorithmus ist es ratsam, die beste gefunde Lösung zu speichern, da diese Algorithmen, nachdem sie das globale Minimum gefunden haben, dieses auch wieder verlassen können [58]. Um diese Verfahren weiter zu verbessern, empfiehlt es sich, eine Multistartstrategie zu verfolgen, da man auf diesem Weg die Möglichkeit hat, mehr Minima zu besuchen und so die Wahrscheinlichkeit zu erhöhen, das globale Minimum zu finden.

5.4 Optimierung durch Evolution

Nachdem sich der Mensch beim Simulated Annealing durch natürliche Vorgänge hat inspirieren lassen, und die Prinzipien, die in der Natur beobachtet wurden auf andere Bereiche übertrug, war es naheliegend, sich nach weiteren Mechanismen umzusehen, die in der Natur erstaunliche Ergebnisse hervorbringen und auch diese für den Menschen direkt nutzbar zu machen.

Auf der Suche nach solchen Mechanismen fällt der Blick unausweichlich auf die Ergebnisse der Evolution. Sie hat während ihrer 4,5 Millarden Jahre andauernden Geschichte nicht nur eine unwahrscheinliche Artenvielfalt erzeugt, sondern auch technische Meisterleistungen vollbracht:

- Charakteristisch für viele Landvögel wie Geier, Störche oder Milane ist die Aufspreizung der Flügelenden (siehe Abbildung 5.2). Mit Hilfe dieses Prinzips können sie ihren Strömungswiederstand erheblich reduzieren, da dank der Aufspreizung der Flügelenden anstatt einem großen Wirbel mit hohem Widerstand viele kleine Wirbel, deren Widerstände aufsummiert deutlich geringer sind als der des großen Wirbels, entstehen.

Abbildung 5.2: Aufspreizung der Flügelenden beim Rabengeier (aus [56])

- Ein weiteres bemerkenswertes Merkmal des Vogelflügels sind seine Deckfedern. Beim Ablösen der Strömung richten sie sich auf (siehe Abbildung 5.3) und verhindern so das Abreißen der Strömung.

Abbildung 5.3: Aufrichten der Deckfedern am linken Flügel einer Skua (aus [56])

- Fischschleim kann dank seiner Zusammensetzung Mikrowirbel einfangen und so den Wasserwiderstand senken (siehe Abbildung 5.4).

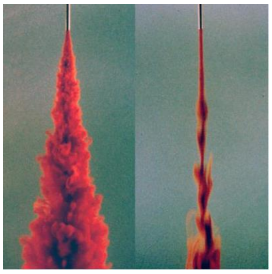


Abbildung 5.4: Darstellung der wasserwiderstandssenkenden Wirkung von Fischschleim (aus [56]) rechts mit Schleim, links ohne Schleim

Die Natur hat nicht nur viele Optimierungsprobleme mit nahezu bester Lösung gelöst, sondern auch häufig das gleiche Optimierungsproblem bei verschiedenen Arten ähnlich gelöst. So ähneln sich die Augen der Menschen und der Kraken sowohl in Aufbau als auch in Funktion. Das zugrunde liegende Optimierungsverfahren scheint reproduzierbar gut zu sein.

So ist es nicht weiter erstaunlich, dass nicht nur im Bereich der Bionik versucht wird, von der Natur gute Ideen zu kopieren, sondern auch in anderen Forschungsgebieten. Seit den sechziger Jahren wird verstärkt versucht, das Prinzip der Evolution auf die numerische Optimierung zu übertragen. In den sechziger Jahren wurde die Evolutionsstrategie von Rechenberg [57] und Schwefel [50] entwickelt, und Anfang der siebziger Jahre veröffentlichte Holland [59] den genetischen Algorithmus. Die Klasse der genetischen Programmierung wurde erst später populär. Sie wurde entscheidend von Koza [61] geprägt.

5.5 Evolutionäre Algorithmen

Um die Funktionsweise der Evolution, wie sie von Darwin [23] dargelegt wurde, auf ein Optimierverfahren zu übertragen, muss das entsprechende Verfahren ähnliche Strukturen verwenden um die Daten zu verwalten wie das Vorbild, die Natur.

5.5.1 Mechanismen und Strukturen natürlicher Evolution

In der Natur werden die (Erb-)Informationen in langen Nukleinsäureketten, der DNS (Desoxyribonukleinsäure), gespeichert. Bestimmte Sequenzabschnitte können dabei zu Genen zusammengefasst werden. Ein solches Gen bestimmt die Ausprägung körperlicher Merkmale, wie z. B. der Augenfarbe. Die einzelnen Gene können dabei verschiedene Ausprägungen, die Allele, zeigen. Eine bestimmte Anzahl Gene wird in einem Chromosom zusammengefasst. Während die einzelnen Gene eines Chromosoms eine zusammenhängende Kette bilden, handelt es sich bei verschiedenen Chromosomen um verschiedene Ketten. Im menschlichen Genom, der Summe aller Erbinformationen eines Menschen, ist jedes Chromosom doppelt vorhanden, man spricht von einem diploiden Chromosomensatz, lediglich die menschlichen Keimzellen tragen einen einfachen (haploiden) Chromosomensatz. Von den beiden Varianten eines Gens im doppelten Chromosomensatz, kommt nur ein Allel zur Ausprägung. Es wird dabei zwischen dominanten Allelen, die zur Ausprägung kommen, und rezessiven Allelen, die nicht zur Ausprägung kommen, unterschieden. Anhand der sichtbaren Merkmale, dem Phänotyp, kann man deshalb keine abschließende Aussage über den Genotyp treffen. Ein Mensch mit seinem kompletten Genom wird als Individuum bezeichnet.

Der eigentliche Evolutionsprozess findet in einer Gruppe, genannt Population, von Individuen einer Art statt und vollzieht sich in Generationen. Die Beschränkung auf Individuen einer Art existiert, da die genetische Distanz zwischen Individuen verschiedener Arten so groß ist, dass sie keine lebensfähigen oder zumindest keine zeugungsfähigen Nachkommen produzieren können (und deshalb sofort wieder aus dem Evolutionsprozess verschwinden). Auch die Reproduktion von Individuen verschiedener Generationen kommt nur selten vor, da die einzelne Individuen nur während einer bestimmten Phase ihres Lebens zeugungsfähig sind bzw. sterben, bevor die Generation der Nachkommen zeugungsfähig ist.

Der Evolution liegen einige grundsätzliche Mechanismen zu Grunde:

Fitnessfunktion Bei dieser Funktion handelt es sich um ein Maß, die Güte eines Individuums einzuschätzen. In der Natur gibt es im eigentlichen Sinn keine Fitnessfunktion. Sie drückt sich lediglich dadurch aus, dass nur bestimmte Individuen das zeugungsfähige Alter erreichen bzw. sich auf der Suche nach einem Partner gegenüber ihren Rivalen durchsetzen und gesunde Nachkommen erzeugen. Es zeigt sich also erst in der Selektion, wie gut ein einzelnes Individuum ist.

Selektionsmechanismus Die natürliche Selektion, oft auch „survival of the fittest“ (von Spencer geprägt und von Darwin übernommen) genannt, ist ein komplexer Prozess in dem nicht alle Größen und deren Zusammenhänge bekannt sind. Rückblickend auf die Evolution lässt sich jedoch feststellen, dass in Analogie zu Spencers Worten sich in den meisten Fällen die Individuen mit den besten Eigenschaften reproduzieren und so ihre genetische Information dem Genpool, der Summe aller in der Population vorhandenen Gene, erhalten bleibt. Der Selektionsdruck auf die Population wird dabei größer, wenn die Zahl der Nachkommen so groß ist, dass die äußeren Bedingungen lediglich das Überleben eines kleinen Teils der Nachkommen gewährleisten.

Reproduktionsoperatoren Unter diesen Punkt fallen zwei Operatoren, Crossover und Mutation: Crossover dient dazu, mehrere Individuen miteinander zu rekombinieren. Es findet beim Menschen zwischen zwei sich entsprechenden Chromosomen aus verschiedenen haploiden Sätzen der Keimzellen von Vater und Mutter statt. Die beiden Chromosome drehen sich ineinander. Danach brechen die Chromosomen an mehreren Stellen auf und verbinden sich neu, wobei einzelne Chromosomteile von einem Chromosom zum anderen Chromosom

wechseln. Die korrespondierenden Chromosomteile haben dabei gleiche Länge, so dass die Chromosomlänge erhalten bleibt. Der Crossovervorgang ist in Abbildung 5.5 illustriert.

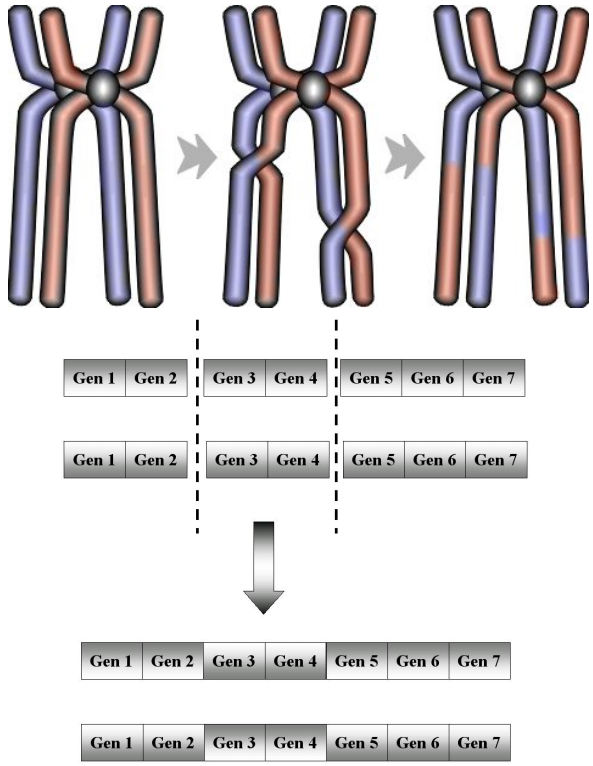


Abbildung 5.5: Darstellung des Crossovers auf Chromosomebene

Bei der Mutation handelt es sich um einen Kopierfehler der Erbinformation bei der Erzeugung der Keimzellen. Diese Fehler müssen dabei nicht unbedingt die Information eines Gens feststellbar verändern. Häufig sind erst dann Veränderungen phänotypisch sichtbar, wenn mehrere Mutationen innerhalb eines Gens stattgefunden haben. Im Unterschied zum Crossover, das auf Genebene arbeitet, finden Mutationen auf Ebene der Basenpaare, aus denen die DNS aufgebaut ist, statt. Hierbei können Basenpaare verschwinden, neue hinzukommen oder ein vorhandenes Paar wird durch ein anderes Paar ersetzt (siehe Abbildung 5.6).

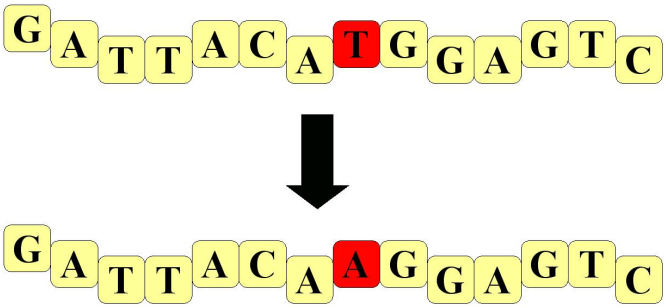


Abbildung 5.6: Darstellung der verschiedenen Mutationen auf DNS-Ebene

5.5.2 Umsetzung natürlicher Mechanismen und Strukturen in evolutionäre Algorithmen

Auch die EAs arbeiten auf Populationen von Individuen. Die Startpopulation wird dabei meist mit zufälligen Individuen initialisert, hier kann Hintergrundwissen über das Optimierungsproblem eingebracht werden, indem z. B. Individuen, die eine vielversprechende Gegend im Entscheidungsraum abdecken, initial gewählt werden. Im wesentlichen folgen EAs dabei dem in Abbildung 5.7 dargestellten Schema.

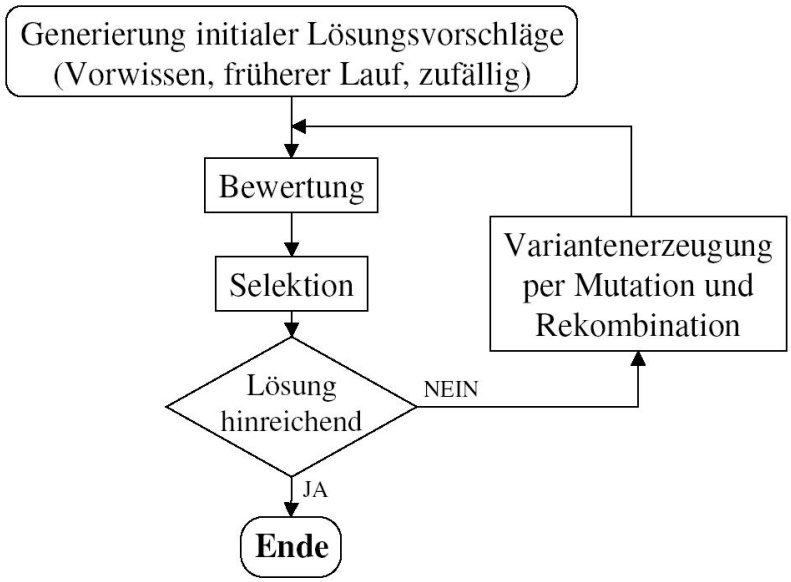


Abbildung 5.7: Grundalgorithmus der evolutionären Algorithmen (aus [96])

Repräsentation

Zunächst wird eine der DNS ähnliche genotypische Repräsentation der Individuen benötigt. Bei einem Optimierungsproblem, das bereits die Form eines Vektoroptimierungsproblems (vgl. Abschnitt 5.1) hat, können die einzelnen Entscheidungsvariablen der Entscheidungsvektoren als Gene aufgefasst werden. Die Summe der Gene wird dann in Analogie zur Genetik als Genom oder Chromosom bezeichnet. Bei den meisten realen Optimierungsproblemen hat man jedoch das Problem, dass nicht alle Entscheidungsvariablen Elemente der gleichen Menge sind. Der Entscheidungsraum, der mit dem genotypischen Raum gleichzusetzen ist, ist das karthesische Produkt aller beteiligten Mengen. Für diesen Fall müssen jedoch die Mutations- und Crossoveroperatoren angepasst werden. Da dies jedoch ein sehr aufwendiger Prozess ist und es die ansonsten sehr einfache Struktur der EAs deutlich komplexer werden lässt, bildet man die nicht einheitlichen Mengen auf andere Mengen ab. Jede dieser Konstellationen lässt sich auf diese Weise auf einen Raum der Form $\mathbb{R}^n$ oder den (besonders bei genetischen Algorithmen häufig verwendeten) Raum der Bitstrings $\{0,1\}^m$ abbilden. Durch diese Abbildung geht meistens die über der alten Menge bestehende Semantik verloren [58] oder es wird eine falsche Semantik impliziert, da z. B. alte Nachbarschaftsbeziehungen auf den neuen Räumen nicht mehr gelten oder durch neue Nachbarn erweitert wurden. Beispielsweise bei der Darstellung einer Teilmenge von $\mathbb{N}$ durch binäre Zahlen verändert sich der Hammingabstand einzelner Individuen. Die Zahlen „7 “ und „8 “ sind in $\mathbb{N}$ direkt benachbart, während sie im Raum der binären Zahlen vier Mutationen voneinander entfernt sind (0111=7 und 1000=8). Diesem Problem kann man durch Veränderungen der Kodierung über der binären Darstellung entgegen wirken, z. B. dem Gray Code (siehe Definition 5.6). Durch diese spezielle Form

der binären Codierung haben in $\mathbb{N}$ benachbarte Zahlen wieder einen Hammingabstand von eins. Es kommen aber weitere Zahlen hinzu, die ebenfalls einen Hammingabstand von eins haben, in $\mathbb{N}$ aber nicht benachbart sind. Die Wahl der Repräsentation ist also in hohem Maße ausschlaggebend für die spätere Güte des Verfahrens.

DEFINITION **5.6.** *Gray Code*

Sei eine Zahl $b = b_1 \dots b_k$ in binärer Darstellung gegeben, so ist die Gray Code-Darstellung $GC(b) = g = g_1 \dots g_k$ gegeben durch:

$$g_1 = b_1$$

$$g_i = b_{i-1}\bar{b}_i + \bar{b}_{i-1}b_i \text{ mit } i = 1, \dots, k$$

Bei dieser Darstellung unterscheiden sich in $\mathbb{N}$ benachbarte Zahlen nur in einem Bit. Sie haben einen Hammingabstand von eins.

Initialisierung

Nachdem die Art der Darstellung festgelegt wurde, kann die Startpopulation initialisiert werden. Üblicherweise werden hierbei zufällige Individuen generiert. Bei EAs, die Individuen variabler Länge zulassen, wird zusätzlich mit Hilfe einer Zufallszahl die Länge festgelegt. Die erlaubten Längen werden dabei meistens in Form eines Intervalls angegeben.

Fitnessbewertung

Als nächster Schritt wird eine Fitnessfunktion F_{EA} benötigt, die jedem Individuum einen Wert zuweist, der angibt, wie die Überlebensaussichten der einzelnen Individuen im Selektionsprozess sind. Die Funktion sollte effizient auswertbar sein [58] und die die Realität gut widerspiegeln [38]. Häufig ist es jedoch nicht möglich, eine bestimmte Funktion explizit anzugeben, z. B. im Black Box Szenario (siehe Abschnitt 5.2) oder wenn es mehrere Ziele zu optimieren gilt. Häufig existiert auch keine absolute Fitness, sondern eine Fitness relativ zu den restlichen Individuen in der Population (siehe [58]).

Selektionsverfahren

Anhand der so ermittelten Fitnesswerte erfolgt die Selektion, um diejenigen Individuen zu ermitteln, die sich durch Reproduktion in die nächste Generation einbringen sollen. Ähnlich wie in der Natur soll besseren Individuen eine größere Chance eingeräumt werden als schlechteren. Dabei wird üblicherweise eines der folgenden Selektionsverfahren verwendet:

zufällige Selektion Bei diesem Verfahren handelt es sich um das einfachste Selektionsverfahren. Aus der alten Population werden zufällig Individuen gewählt, die dann miteinander rekombiniert werden. Bei diesem Vorgehen vernachlässigt man jedoch komplett die Informationen, die aus der Fitness ableitbar sind. Deswegen ist dieses Verfahren nicht zu empfehlen.

Turnierselektion Für dieses Verfahren werden k Individuen zufällig aus der aktuellen Population gezogen. In einem Turnier wird das Individuum mit der besten Fitness der gewählten k Individuen ausgewählt. Diese Methode hat den Vorteil, dass durch die zufällige Auswahl der Individuen für die Turniere die besten Individuen nicht mehr so leicht den Selektionsprozess dominieren. Dadurch bleibt mehr Diversität innerhalb der Population erhalten.

fitnessproportionale Selektion Bei diesem Verfahren soll die Wahrscheinlichkeit $P(x_i)$, ein bestimmtes Individuum x_i zu wählen, proportional zum Fitnesswert dieses Individuums sein:

$$P(x_i) = \frac{F_{EA}(x_i)}{\sum_{n=1}^{N} F_{EA}(x_n)} \text{ mit } N = |Population|$$

Häufig werden die einzelnen Fitnesswerte mit der Summe $\sum_{n=1}^{N} F_{EA}(x_n)$ aller Fitnesswerte normiert und so auf das Intervall $[0, 1]$ abgebildet. Ihre so modifizierte Fitness entspricht dann direkt ihrer Auswahlwahrscheinlichkeit. Bei dieser Methode kann es passieren, dass ein Individuum mit sehr guter Fitness den gesamten Auswahlprozess dominiert und damit die Diversität in der Population verringert. Im schlimmsten Fall wird nur noch ein Individuum wiederholt ausgewählt, so dass man durch Crossover keine neuen Individuen mehr erzeugen kann.

Dieses Selektionsverfahren wird auch als Rouletteradselektion bezeichnet. Man kann sich den Auswahlprozess dabei als wiederholtes Drehen eines (in Abbildung 5.8 dargestellten) Rades vorstellen, auf dem jedem Individuum ein Anteil am Rad entsprechend seiner Fitness zugeordnet ist.

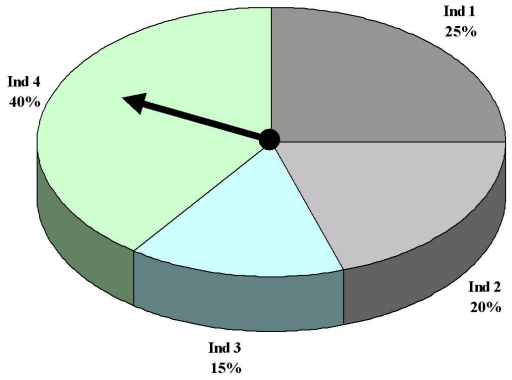


Abbildung 5.8: Darstellung eines bei der Rouletteradselektion verwendeten Rades

Rangbasierte Selektion Dieses Verfahren ist sehr gut geeignet, um die Dominanz eines Individuums mit sehr guter Fitness zu verhindern. Die Individuen werden ihrer Fitness entsprechend in eine Rangliste eingeordnet. Die Fitness wird also neu skaliert. Wenn es mehrere Fitnessfunktionen gibt, kann an dieser Stelle das Dominanzkriterium (vgl. Abschnitt 5.3) verwendet werden, um die einzelnen Individuen in die Rangliste einzuordnen. Im Anschluss wird z. B. mit der Rouletteradselektion anhand der neuen Fitnesswerte die Selektion durchgeführt.

Bei diesen probabilistischen Selektionsmethoden kann es vorkommen, dass das beste Individuum dem Genpool verloren geht. Um dies zu verhindern, verwendet man Elitismus. Die besten k Individuen der aktuellen Generation werden ausgewählt und direkt unverändert in die nächste Generation übernommen [38].

Operatoren

Um den Optimierungsprozess voran zu treiben, gibt es auch bei den EAs ein Pendant zu den Reproduktionsoperatoren der Natur. Zum einen existiert ein unärer Mutationsoperator, der genutzt wird, um neue Bereiche des Entscheidungsvariablenraums zu explorieren. Zum anderen gibt es den n-ären Crossoveroperator, mit dessen Hilfe Teile eines Individuums mit Teilen anderer Individuen kombiniert werden, um so bessere Individuen zu erzeugen oder wenigstens gute Teile in der Population zu verbreiten. Während Mutation nötig ist, damit jeder Punkt im Entscheidungsraum exploriert werden kann, hofft man bei bei den Crossoveroperatoren darauf, dass erfolgreiches

genetisches Material in der Population verbreitet wird. Diese Hoffnung basiert auf der Building Block Hypothese, die in Abschnitt 5.5.2 beschrieben ist. Die Operatoren werden auf den durch die Selektion ausgewählten Lösungen angewendet.

Da die Implementierung der Operatoren stark von der gewählten Repräsentation der Individuen abhängt, beschreiben wir an dieser Stelle lediglich gängige Varianten für binäre und reelle Entscheidungsräume.

Im Wesentlichen unterscheidet man beim Crossover auf binären Strings zwischen drei Varianten, die sich als maskenbasierte Auswahlverfahren mit Masken der Form $m \in \{0,1\}^n$ beschreiben lassen. Das eine Kind erhält dabei an der Stelle i die Information des ersten Elters, wenn $m_i = 0$ gilt, ansonsten erhält es die Information des zweiten Elters. Beim zweiten Kind ist dies umgekehrt. Wir beschränken uns auf die Rekombination von zwei Individuen, da Rekombination von mehr als zwei Individuen nur selten angewendet wird und die Varianten leicht auf Situationen mit mehr als zwei Individuen zu übertragen sind.

One-Point-Crossover Bei dieser Variante wird ein Kreuzungspunkt k benötigt. Es werden Masken der Form $m = 0^i 1^{n-i}$ mit $1 \leq i \leq n-1$ benutzt.

Multi-Point-Crossover Für diese Variante benötigt man h Kreuzungspunkte k_i mit $i = 1, \ldots, h$ und $k_i \neq k_j$ für $i \neq j$, die o.B.d.A. sortiert sind. Die verwendeten Masken haben für gerade h die Form: $m = 0^{k_1} 1^{k_2 - k_1} \ldots \ 1^{k_h - k_{h-1}}$

Und für ungerade h die Form: $m = 0^{k_1} 1^{k_2 - k_1} \ldots \ 0^{k_h - k_{h-1}}$

Uniform-Crossover Die Masken entstehen durch Gleichverteilung über allen möglichen Masken [38, 58].

Anschaulich bedeutet dies, dass beim Crossover von binären Strings die Strings in mehrere Teilstrings unterteilt werden. Die Nachfahren entstehen, indem die Teilstrings neu zusammengesetzt werden. Es wird im Wechsel ein Teilstring der verschiedenen Eltern gewählt (siehe Abbildung 5.9).

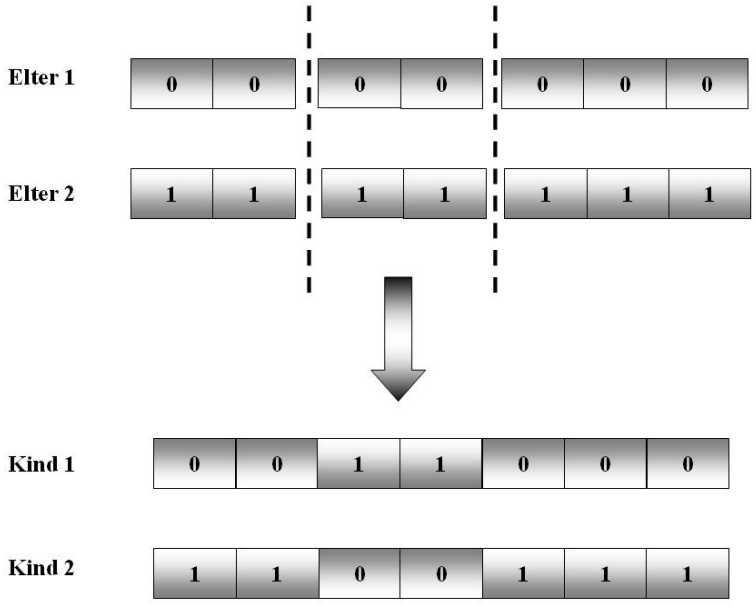


Abbildung 5.9: Darstellung des Multi-Point-Crossovers für Binärstrings

Bei der Optimierung auf Strings aus reellen Zahlen wird zwischen zwei gängigen Varianten unterschieden:

Diskrete Rekombination Für jede Stringposition wird gleichverteilt ein Elter ausgewählt, von dem der entsprechende Wert übernommen wird.

Intermediäre Rekombination An einer Position i wird über die Werte aller Eltern gemittelt und so der neue Wert festgelegt [38]. Häufig werden jedoch auch die bei den Eltern gefundenen Werte als Grenzen für ein Intervall benutzt, aus dem ein Wert gezogen wird. Hierbei kann eine beliebige Wahrscheinlichkeitsverteilung über dem Intervall benutzt werden (vgl. [24]).

Bei binären Strings wird meistens Punktmutation verwendet, die nur ein Bit verändert. Mit Hilfe der Mutationswahrscheinlichkeit werden die Positionen im String ermittelt, die mutiert werden sollen. An den entsprechenden Positionen wird das Bit geflippt. Es sind jedoch auch andere Mutationen denkbar, z.B. das Flippen einer ganzen Bitgruppe. Bei reellen Zahlen wird anstelle des Bitflipps ein reeller Term addiert und der alte Wert auf diese Weise verändert.

Crossover und Mutation werden jeweils nur mit einer bestimmten Wahrscheinlichkeit ausgeführt, der Crossoverwahrscheinlichkeit p_c bzw. der Mutationswahrscheinlichkeit p_m.

Building Block Hypothese

In diesem Abschnitt werden kurz die Mechanismen beleuchtet, die GAs mit Bitstringrepräsentation auf ihrer Suche nach einem Optimum treiben. Entgegen dem ersten Eindruck, dass in den Individuen Strings unabhängig voneinander optimiert werden, wird in der Building Block Hypothese vermutet, dass Schemata evolviert werden [47]. GAs wären nicht in der Lage, große Teile des Suchraums zu explorieren, wenn jeder String nur einen Punkt im Suchraum repräsentieren würde. Stattdessen stehen einzelne Strings für ganze Gruppen von Strings, die alle einem bestimmten Schema folgen. Diese Schemata sind Strings der Länge l (wobei l der Länge der Individuen entspricht) über dem Alphabet $\{0, 1, *\}$, wobei $*$ ein Platzhalter ist. Anstelle des Platzhalters kann ein beliebiges Zeichen stehen. Das Schema $H = 11 * *01$ repräsentiert z. B. die Strings $\{110001, 110101, 111001, 111101\}$. Solche Schemata zeichnen sich durch folgende Charakteristika aus:

Ordnung $o(H)$ Die Ordnung eines Schemas ist die Anzahl fest definierter Positionen (keine Platzhalter) im Schema.

Definierende Länge $\delta(H)$ Die definierende Länge ist der Abstand zwischen der ersten und der letzten fest definierten Posotion.

Schemata mit kleiner Ordnung, kleiner definierender Länge und hoher durchschnittlicher Fitness der durch sie repräsentierten Strings heißen Building Blocks (vgl. [47]). Im Schema Theorem (siehe Theorem 5.2) werden die Einflüsse von Mutation und Crossover auf den Erhalt der Schemata untersucht. Es zeigt, dass Schemata hoher Ordnung besonders anfällig gegenüber Punktmutationen und Schemata hoher definierender Länge besonders anfällig gegenüber Crossover sind. Das Schema Theorem geht dabei jedoch von einer stark vereinfachten Situation aus. Es werden lediglich destruktive Einflüsse von Mutation und Crossover betrachtet, weshalb es nur eine untere Schranke für die erwartete Häufigkeit eines Schemas in einer Nachfolgegeneration angibt.

THEOREM **5.2.** *Schema Theorem*

Für einen GA mit Bitstringrepräsentation und fitnessproportionaler Selektion gilt:
Sei $m(H,t)$ die Anzahl der Individuen in Generation t, die das Schema H enthalten. Die erwartete Anzahl $E[m(H,t+1)]$ der Individuen in Generation $t+1$, die das Schema H enthalten ist dann:

$$E[m(H,t+1)] \geq \; m(H,t) \cdot \; \frac{F_{EA}(H)}{\bar{F}_{EA}} \cdot \; (1 - p_c \frac{\delta(H)}{l-1} - o(H)p_m)$$

Dabei ist $F_{EA}(H)$ die durchschnittliche Fitness der Individuen, die das Schema H enthalten, und $\bar{F}_{EA}$ die durchschnittliche Fitness der Population in Generation t. p_c ist die Crossoverwahrscheinlichkeit und p_m ist die Mutationswahrscheinlichkeit.

Während seiner detaillierten Untersuchung des Schema Theorems entwickelte Goldberg die Building Block Hypothese (vgl. [47]). Sie besagt, dass Building Blocks durch Rekombination gute Individuen formen.

Generell bleibt zum Schema Theorem zu sagen, dass es nur bei linear separablen Funktionen sinnvolle Aussagen treffen kann (vgl. [58]). Auch die Building Block Hypothese gilt nur für separable Funktionen. Trotz Bedenken zeigen diese beiden Aussagen, dass es bei der Arbeitsweise der evolutionären Verfahren einige Mechanismen gibt, die nicht direkt erkennbar und nur schwer beweisbar bzw. falsifizierbar sind.

5.6 Genetische Programmierung

> „Genetic programming addresses the problem of automatic programming, namely, the problem of how to enable a computer to do useful things without instructing it, step by step, on how to do it.“ John R. Koza

Wie man den Worten Kozas entnehmen kann, soll mit Hilfe der GP Quelltext von Programmen evolviert werden. Anstelle der bei anderen evolutionären Algorithmen häufig verwendeten Bitstrings oder reellwertigen Vektoren werden bei der genetischen Programmierung ausführbare Programme als Individuen dargestellt.

5.6.1 Repräsentation

Ein Programm setzt sich aus folgenden Elementen zusammen.

Menge der Terminale Diese Menge umfasst alle Eingaben und Konstanten, die an ein GP-System übergeben werden. Anders ausgedrückt handelt es sich bei den Terminalen um Funktionen, deren Ausgabe von keiner Eingabe abhängt [12].

Menge der Funktionale In dieser Menge sind alle Anweisungen, Operatoren und Funktionen vereinigt, die im GP-System verwendet werden dürfen. Einige typische Funktionsmengen sind.

- boolesche Funktionen
- arithmetische Funktionen
- Schleifen
- bedingte Anweisungen
- Subroutinen

Einen vollständigeren Überblick findet man in [12].

Bei der Wahl dieser beiden Mengen gilt es zu beachten, dass sie mächtig genug sein müssen, um das gegebene Problem zu lösen. Sie sollten jedoch nicht zu mächtig sein, da der daraus resultierende Suchraum vergrößert wird und dies in manchen Fällen die Suche erschweren kann[2].

Für den Aufbau eines Programms aus Funktionalen und Terminalen gibt es drei gebräuchliche Arten:

Bäume Ein einfaches Individuum einer Baum-GP ist in Abbildung 5.10 dargestellt. Bei Baumstrukturen kommen Terminale nur in den Blättern vor, bei den inneren Knoten handelt es sich immer um Funktionale. Wenn das Programm ausgewertet wurde, steht die Ausgabe des Programms in der Wurzel. Für die Auswertung der Programme sind verschiedene Strategien denkbar. Eine Darstellung der verschiedenen Strategien findet man in [12].

Bei der Auswertung der Programme wird nur lokaler Speicher benötigt. Jeder Knoten besitzt seinen eigenen lokalen Speicher, in dem die Werte seiner Operanden gespeichert sind. Nur in diesem Knoten sind die Werte der Operanden zugreifbar.

[2] Diese Aussage nimmt Bezug auf das Ökonomieprinzip von Ockham („For it is pointless to do with more what can be done with less.“), das später u. a. von Einstein, Dirac und Moody aufgegriffen wurde.

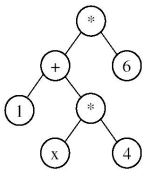


Abbildung 5.10: Individuum einer Baum-GP (aus [96])

Lineare Listen Abbildung 5.11 zeigt die Darstellung eines Individuums eines GP-Systems, dessen Individuen in Form linearer Listen verwaltet werden. Es handelt sich bei diesen Programmen um eine Liste von Befehlen, die der Reihe nach von links nach rechts ausgewertet werden. Im Gegensatz zur Baum GP benötigt man bei linearen Listen globalen Speicher, in dem die Operanden gespeichert werden. Das Ergebnis eines Programmdurchlaufs wird in einem vorher festgelegten Speicherplatz abgelegt. Diese Programme werden meistens als Mehradressregistermaschine realisiert.

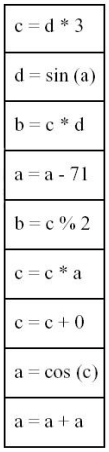


Abbildung 5.11: Individuum einer GP mit Programmen in Form linearer Listen (aus [96])

Graphen Eine andere Art der Repräsentation ist ein Graph (wie z. B. ein in Abbildung 5.12 dargestelltes PADO Programm). In dieser Variante folgt man von einem ausgezeichneten Startknoten aus den Kanten bis zu einem ausgezeichneten Stopknoten. Da in einem solchen System jedoch Schleifen und Rekursion erlaubt sind, ist es nicht abzuschätzen, wie lange ein Programm läuft. Deshalb wird die Auswertung eines solchen Programms nach einer bestimmten Zeit abgebrochen. PADO verwendet einen Stack und Speicherregister, auf die jeweils mit bestimmten Befehlen zugegriffen werden kann. Details zu PADO findet man in [91].

5.6.2 Fitnessbewertung

Die Güte eines Programms ist nur in Abhängigkeit von einer bestimmten Eingabe feststellbar. Es kann überprüft werden, ob es die richtige Ausgabe oder die falsche Ausgabe erzeugt bzw. wieweit die Ausgabe von dem erwarteten Wert abweicht. Zum trainieren der Programme und zum bewerten ihrer Güte werden Supervised Learning Strategien (siehe [38]) benutzt. Beim Supervised Learning verwendet man Beispieleingaben, zu denen man die gewünschte Ausgabe kennt. Mit Hilfe einer

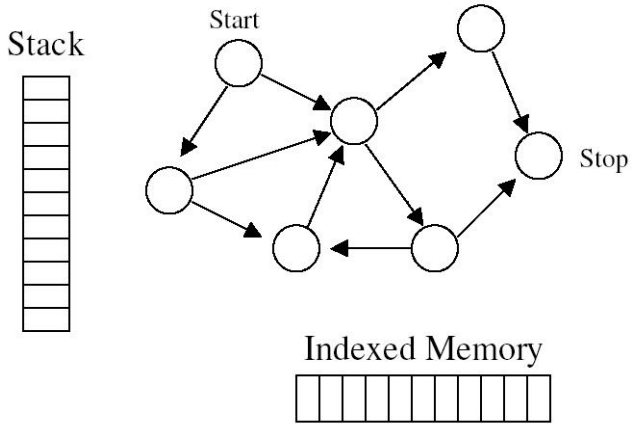


Abbildung 5.12: Individuum einer Graph-GP (aus [96])

Fehlerfunktion über der Menge der Beispieleingaben N wird die Fitness

$$F_{EA}(P) = \sum_{i=1}^{|N|} |p_i - o_i| \text{ bzw. } F_{EA}(P) = \sum_{i=1}^{|N|} (|p_i - o_i|)^2$$

der einzelnen Programme p ermittelt. Dabei ist p_i die Ausgabe des Programms P für die Beispieleingabe i und o_i die erwünschte Ausgabe.

5.6.3 Operatoren

Für die Evolution der Programme werden wieder die bekannten Operatoren Mutation und Crossover verwendet. Da der Aufbau der GP-Individuen sich jedoch stark von dem in Abschnitt 5.5.2 beschriebenen Aufbau der Individuen unterscheidet, benötigt man an dieser Stelle dafür angepasste Varianten dieser Operatoren.

Mutation

Die Mutation ist für die angesprochenen Repräsentationen der Programme verschieden realisiert:

- Bei Individuen einer Baum-GP wird zufällig ein Knoten aus dem Baum ausgewählt. Der an diesem Knoten hängende Teilbaum wird durch einen neuen zufällig erzeugten Teilbaum ersetzt.
- Bei Programmen in Form linearer Listen kann man verwendete Konstanten, Operatoren oder Register verändern.
- Im Falle von PADO wurde auch die Mutation durch separat evolvierte Programme durchgeführt. Genauere Angaben zu diesem Prozess findet man in [9].

Crossover

Mit Hilfe eines der in Abschnitt 5.5.2 beschriebenen Verfahren werden Programme ausgewählt, die miteinander rekombiniert werden sollen.

- Bei einer Baum-GP kann in den gewählten Eltern z. B. jeweils ein Teilbaum ausgewählt werden. Diese werden dann wie in Abbildung 5.13 dargestellt vertauscht.

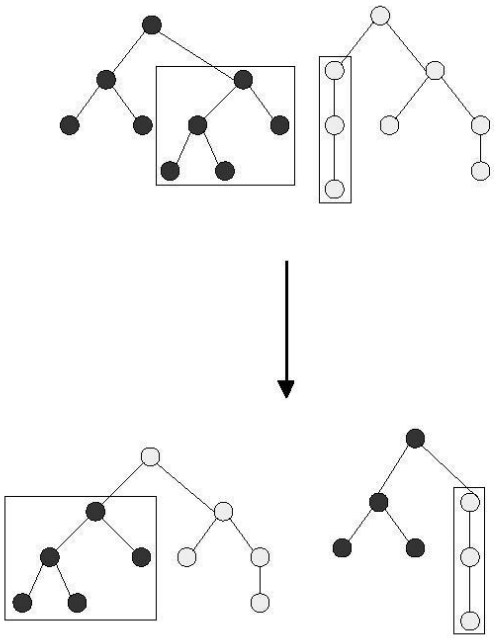


Abbildung 5.13: Crossover bei einer Baum-GP (aus [96])

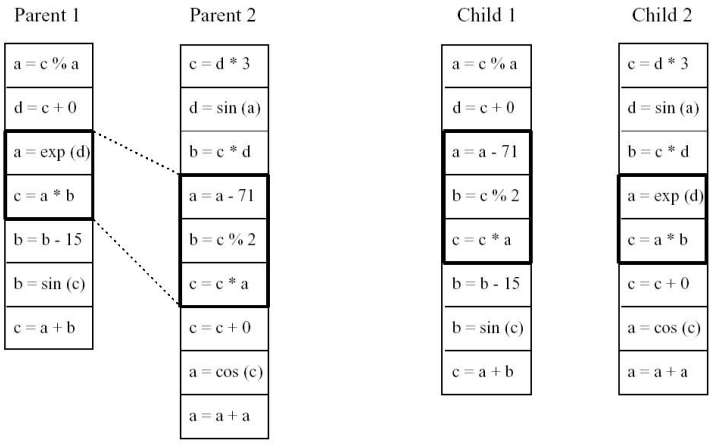


Abbildung 5.14: Crossover bei einer linearen GP (aus [75])

- Bei Individuen in Form linearer Listen wird in jedem der Elterindividuen eine Folge von Befehlen ausgewählt. Diese Befehlsfolgen werden untereinander vertauscht. Dieser Vorgang ist in Abbildung 5.14 dargestellt.

- In GP-Systemen, die eine Graphrepräsentation der Individuen verwenden, werden je nach GP verschiedene Rekombinationsoperatoren verwendet. PADO z. B. evolviert in einer separaten Population Programme mit, die die Rekombination der Graphen realisieren. Eine detailliertere Darstellung dieser Metaevolution findet man in [8].

Im Vergleich zu dem in der Natur angewendeten Crossover (homologes Crossover) würde das bei der GP angewendete Crossover in der Natur meist zu nicht lebensfähigen Organismen führen. Die Natur erzielt eine bessere Quote beim Crossover, weil in der Natur nur Individuen der selben Art miteinander rekombiniert werden und die genetische Information gut modularisiert in Genen vorliegt. Dadurch werden bei der Rekombination in der Regel nur Allele eines Gens getauscht, also bestimmte semantische Blöcke oder Module.

Das Auffinden guter Crossoverpunkte ist daher ein guter Ansatzpunkt, um ein GP-System zu verbessern. So ist es auch auffällig, dass ab einem bestimmten Punkt in der Evolution eines GP-Systems die Länge der Individuen exponentiell zunimmt. Bei diesem Phänomen spricht man von Bloat. Bei genauerer Untersuchung der Individuen fällt auf, dass im Wesentlichen so genannte Introns hinzukommen (vgl. [12]), Befehle, die keinen Einfluss auf die Ausgabe des Programms haben. Hierunter fallen z. B.:

- Multiplikation oder Division mit 1,

- Addition oder Subtraktion von 0,

- Manipulation eines Registers, das nicht verwendet wird.

Als Ursache hierfür wird vermutet, dass sich die Individuen ab dem Einsetzen des Bloats nicht mehr durch Mutation oder Crossover verbessern können. Stattdessen wird die Fitness durch Anwenden der Operatoren häufig sogar verschlechtert. Durch Einfügen von Introns kommen in den Individuen jedoch Stellen hinzu, an denen Mutation und Crossover keinen Schaden anrichten können. Sie dienen zum Schutz gegen Crossover innerhalb funktioneller Einheiten des Programmcodes.

Explicitly Defined Introns

Es wurde versucht, aus dieser Beobachtung zu lernen und GP zu verbessern. Um Modularisierung zu fördern und Individuen vor schädlichen Crossovereinflüssen zu schützen, wurden EDIs (Explicitly Defined Introns) eingeführt. Man kann sie sich z. B. als Crossoverwahrscheinlichkeit an den einzelnen Crossoverpunkten vorstellen (vgl. [12]). Zu Beginn der Evolution hat dabei jeder Punkt die gleiche Wahrscheinlichkeit, ein guter Crossoverpunkt zu sein. Im Lauf der Evolution wird dann untersucht, ob Crossover an bestimmten Punkten zu tendenziell besseren Individuen führt. Die Crossoverwahrscheinlichkeit dieser Punkte wird an diesen Stellen sukzessive erhöht. An Crossoverpunkten, die tendenziell zu schlechteren Individuen führen wird die Crossoverwahrscheinlichkeit gesenkt (siehe Abbildung 5.15).

Automatically Defined Functions

Eine weitere Möglichkeit, um die Programme vor negativem Crossover zu schützen bzw. gut modularisierte Programme zu erzeugen ist die Verwendung von ADFs (Automatically Defined Functions, vgl. [12, 60, 70]). Bei dieser Form der Baum-GP wird das Programm in zwei Zweige unterteilt (siehe Abbildung 5.16). Zum einen den ADF Definition Branch, in dem Module bzw. Funktionen evolviert werden und zum anderen der Result Producing Branch, in dem das eigentliche Programm evolviert wird. Die Menge der Funktionale des Result Producing Branch wird um die im ADF Definition Branch definierten Funktionen erweitert. Auf diesem Weg haben Crossover und

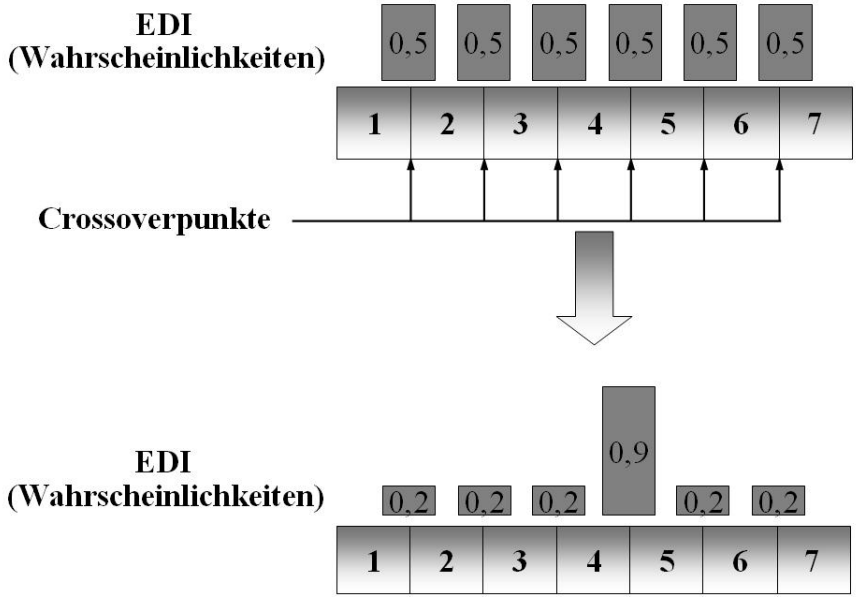


Abbildung 5.15: Entwicklung der Crossoverwahrscheinlichkeiten im Lauf der Evolution eines Programms

Mutation, die auf dem Result Producing Branch angewendet werden, keinen Einfluss auf die im ADF Definition Branch definierten Funktionen bzw. Module.

5.6.4 Abschließende Bemerkungen

Dieser Abschnitt über die genetische Programmierung soll in der vorliegenden Form lediglich einen ersten Überblick über das Forschungsgebiet geben. Es werden zusätzlich einige weitergehende Ideen und Konzepte angesprochen, die in die von uns implementierten Algorithmen eingeflossen sind. Weitere und tiefergehende Informationen zu GP kann man z. B. in [12, 60, 61, 70, 75] finden.

5.7 Genetische Algorithmen

Bei den genetischen Algorithmen (GA) handelt es sich um die bekannteste Klasse der evolutionären Algorithmen. Sie folgen dem in Abbildung 5.7 dargestellten Ablauf und verwenden die in Abschnitt 5.5.2 beschriebenen Selektionsverfahren und die in Abschnitt 5.5.2 beschriebenen Crossover- und Mutationsoperatoren. Üblicherweise arbeiten die GAs mit Individuen in Bitstringrepräsentation, aber inzwischen werden immer häufiger anstelle der Bitstrings Strings reeller Zahlen verwendet (z. B. im Bereich der Tragflügeloptimierung u. a. in [1]). Der bei der Mutation verwendete, reellwertige Summand (vgl. Abschnitt 5.5.2) wird dabei meist zufällig mit Hilfe einer Normalverteilung $N(m, \sigma^2)$ um den Mittelwert m mit der Standardabweichung σ ermittelt.

Im Bereich der genetischen Algorithmen wurden über die hier erwähnten Begebenheiten hinaus viele weitere Untersuchungen gemacht bzw. neue Varianten oder Verbesserungen entwickelt. Wir wollen an dieser Stelle nur auf die von uns verwendeten Mechanismen eingehen und stellen deshalb im Anschluss den Niching-Operator vor. Ausführliche Darstellungen zum Thema GAs findet man u. a. in [26, 18, 31, 38, 47, 58, 59, 85].

Niching

Ein Problem bei der Optimierung mit GAs ist, dass die Population mit Fortschreiten der Evolution dazu neigt, von einem Individuum dominiert zu werden. Bei multimodalen Fitnessfunktionen oder

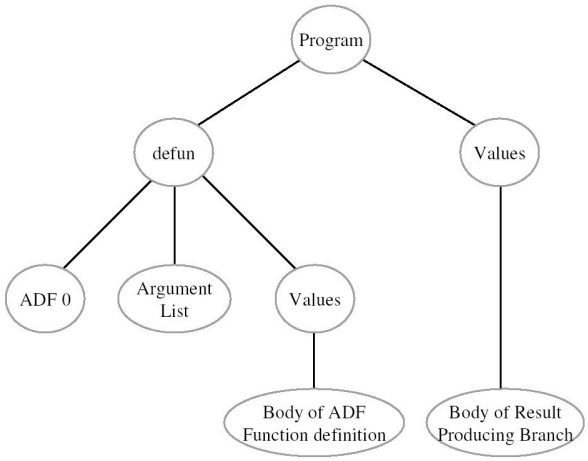


Abbildung 5.16: GP-Individuum mit ADFs (aus [96])

in der Mehrzieloptimierung kann es jedoch interessant sein, auch andere Optima zu kennen, selbst wenn sie nur lokal gut sein sollten. Um dieses Ziel zu erreichen, wurden einige Strategien entwickelt, die unter der Bezeichnung Nichingstrategien zusammengefasst sind.

Bei den Nichingstrategien wird zwischen Strategien, die mehrere Nischen parallel verwalten, und denen, die mehrere Nischen im Laufe ihrer Evolution besuchen, unterschieden. Bei dieser Unterscheidung ließ man sich von der Natur inspirieren. Auch die Evolution findet in geographisch getrennten Räumen statt, so dass Pinguine in der Natur nicht auf Eisbären treffen, oder durch die Zeit getrennt, so dass bislang noch kein Mensch einen lebendigen Dinosaurier gesehen hat.

Zu den letzteren Verfahren zählt das Sequential Niching, dass von Beasley, Bull und Martin (vgl. [25]) entwickelt wurde. Dieses Verfahren basiert auf einer sequentiellen Multistartstrategie, wobei vor jedem neuen GA-Start die Fitnessfunktion verändert wird. Alle Punkte im Radius um das gefundene Optimum bekommen eine schlechtere Fitness. Auf diese Weise soll erreicht werden, dass ein bereits gefundenes Optimum nicht erneut gefunden wird.

Zu den weitaus häufiger verwendeten Verfahren des parallelen Nichings gehört unter anderem die Idee, anstelle eines haploiden Chromosomensatzes einen diploiden oder polyploiden Chromosomensatz zu verwenden. Dieses Verfahren hat zum Teil bei dynamischen Zielfunktionen mit wechselnden Optima gute Erfolge erzielt (vgl. [28]).

De Jong entwickelte ein Verfahren, das er Crowding Factor Model (vgl. [30]) nannte. Bei diesem Verfahren wird in jeder Iteration des GAs versucht, alle bis dahin gefundenen Nischen zu erhalten. In jeder Generation wird mit Hilfe fitnessproportionaler Selektion eine Gruppe von Individuen ausgewählt, die mutiert und rekombiniert werden sollen. Diese Gruppe wird Generation Gap (GG) genannt. Um alle ursprünglichen Nischen zu erhalten, ersetzen die Individuen der GG jeweils ein ihnen möglichst ähnliches Individuum. Für jedes Individuum der GG wird ein Generationssample gezogen. Diese Generationssamples heißen Crowding Factor (CF). Für jedes Individuum des CF wird mit Hilfe des Hammingabstands die Ähnlichkeit zu dem entsprechenden Individuum der GG ermittelt. Das Individuum der GG ersetzt dann das Individuum des CF, dass ihm am ähnlichsten ist, und wird so in die nächste Generation übernommen.

Eine der bis jetzt meistverwendeten Nichingstrategien ist das Fitness Sharing (vgl. [27]). Hierbei werden Individuen, die in dicht besiedelten Bereichen des Lösungsraums liegen, bestraft. Die neue Fitness $F'_{EA}(i)$ eines Individuums i wird dabei häufig nach der folgenden Formel berechnet:

$$F'_{EA}(i) = \frac{F_{EA}(i)}{\sum_{j=1}^{n} sh(d(i,j))}$$

mit $d(i,j) =$ Abstand der Individuen i und j, gemessen mit einer Metrik.

Dabei wird als Sharing Function sh häufig die folgende Funktion verwendet:

$$sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha} & falls\ d < \sigma_{share}, \\ 0 & sonst. \end{cases}$$

mit $\alpha =$ konst., üblicherweise $\alpha = 1$,

σ_{share}, die Größe der zu betrachtenden Nachbarschaft um das Individuum i.

Diese Aufzählung soll nur einen ersten Eindruck von diesem Gebiet vermitteln. Eine detailiertere Abhandlung findet man in [85].

Kapitel 6

Implementierung

> „Das Denken für sich allein bewegt nichts, sondern nur das auf einen Zweck gerichtete und praktische Denken.“
>
> Aristoteles (384-322 v. Chr.), griech. Philosoph

Nachdem wir in den vorangegangenen Kapiteln einen Blick auf die Arbeiten Anderer geworfen haben, werden wir nun unsere eigenen Ziele und Wege beschreiben. Zunächst klären wir in einer Anforderungsdefinition, was unser Programm leisten soll. Hieraus leiten wir die Struktur ab und beschreiben die einzelnen Komponenten. Im letzten Abschnitt gehen wir detailliert auf die praktische Umsetzung dieser Struktur ein. Wir beschreiben die verwendeten Vorgehensweisen und Tools.

6.1 Ziel

Unsere Applikation soll für spezielle Probleme der Biochemie maßgeschneiderte Peptide und Proteine entwickeln helfen. Dieses Ziel können wir natürlich nur in einer groben Näherung erreichen, da all die verwendeten computergestützen Verfahren ungenaue Beschreibungen der Natur verwenden und damit auch ungenaue Ergebnisse erzielen. Deswegen soll unser Verfahren nicht die experimentellen Methoden (wie in Kapitel 2 dargestellt) ersetzen, sondern unterstützen. Viele Verfahren, wie z. B. Phage Display und Peptidarrays benötigen Bibliotheken von Proteinsequenzen, die bezüglich einer bestimmten Eigenschaft getestet werden (siehe Kapitel 2). Solche Bibliotheken werden meist zufällig erzeugt. Da der Raum aller möglichen Proteinsequenzen sehr groß ist, können solche Bibliotheken nur einen Bruchteil dieser enormen Menge möglicher Sequenzen untersuchen. Unser Programm soll eine Verbesserung dieser Methode bieten. Die erste Stufe des Sequenzscreenings soll nicht länger experimentell, sondern „in silico“ durchgeführt werden (siehe Abbildung 6.1). Dies könnte Ressourcen sparen und die Qualität der verwendeten Bibliotheken aufwerten. Um unser Programm für den Anwender nutzbar zu machen, ist es wichtig, die Problemstellung des Anwenders zu erfassen. Probleme, die in der Praxis häufig auftreten, werden im Folgenden erläutert.

Erzeugung stabiler Proteine Wenn Proteine als Wirkstoff eingesetzt werden sollen, ist es wichtig, dass sie möglichst stabil sind. Proteine sind dann sehr stabil, wenn die dazugehörige Energielandschaft sehr glatt ist und es ein ausgeprägt „trichterförmiges“ Optimum gibt. Im Idealfall läuft der Trichter nach unten hin spitz zu, damit das Protein nicht zwischen ähnlich energetisch günstigen Faltungen wechseln kann (siehe Abbildung 6.2).

Affinitätsselektion Die meisten Wirkstoffe sollen an bestimmte Rezeptoren im Körper docken. Die Rezeptoraffinität ist ein wichtiges Kriterium, um die Qualität eines neuen Wirkstoffs

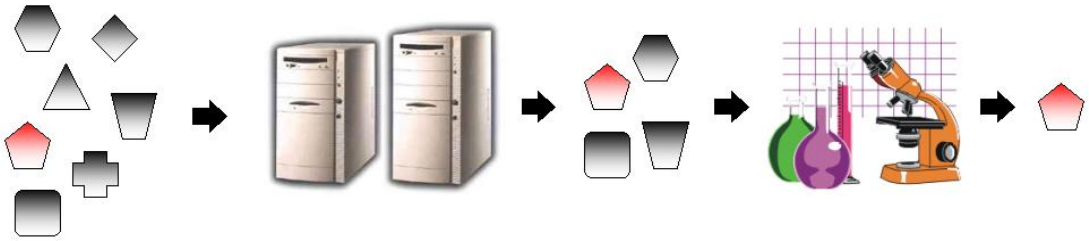


Abbildung 6.1: Computergestützter Screeningprozess: Im ersten Schritt werden durch Simulation eine Menge vielversprechender Proteine ausgewählt. Im zweiten Schritt werden diese experimentell untersucht. Mit der gleichen Zahl von Experimenten kann so eine große Menge von Kandidaten untersucht werden.

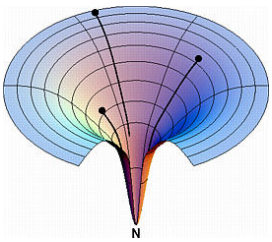


Abbildung 6.2: Idealisierte Energielandschaft (aus [33])

einzuschätzen. Hierbei ist jedoch nicht nur die hohe Affinität, sondern auch eine hohe Spezifität wichtig. Durch eine möglichst hohe Rezeptorspezifität werden die Nebenwirkungen eines Medikaments gesenkt, da man nur den Zielrezeptor beeinflusst und nicht an unerwünschten Rezeptoren dockt.

Epitopimitation Bei der Entwicklung bestimmter neuer Medikamente kann es darum gehen, bekannte Epitope zu imitieren. Auf diesem Weg kann z. B. ein Überangebot an bestimmten Bindungsmöglichkeiten erzeugt werden, um eine ungewollte Immunantwort auf körperfremde Stoffe zu schwächen.

6.2 Struktur

Unser System gliedert sich in drei Stufen (siehe Abbildung 6.3):

Sequenzfindungsmodul Versucht, optimale Peptidsequenzen bezüglich eines anwenderspezifizierten Ziels zu finden.

Faltungsmodul Versucht, die optimale Faltung einer gegebenen Sequenz zu errechnen.

Modell Errechnet anhand eines Modells die Energie einer gegebenen Faltung.

Jede einzelne Stufe ruft die darunterliegende in einer Schichtarchitektur auf. Hier gibt es einen Trade-Off. Erhöhen wir den Durchsatz innerhalb einer Stufe, verlieren wir an Genauigkeit. Durch die erhöhte Geschwindigkeit sind wir in der Lage, in der darüberliegenden Stufe mehr Auswertungen durchzuführen. Dadurch kann ein besseres, aber auch ein schlechteres Ergebnis erzielt werden. Die richtige Gewichtung dieser drei Komponenten ist eine schwierige Frage, die wir nur in Ansätzen untersuchen können, da uns die Validierung durch Experimente in vitro leider nicht möglich ist (vgl. [94]).

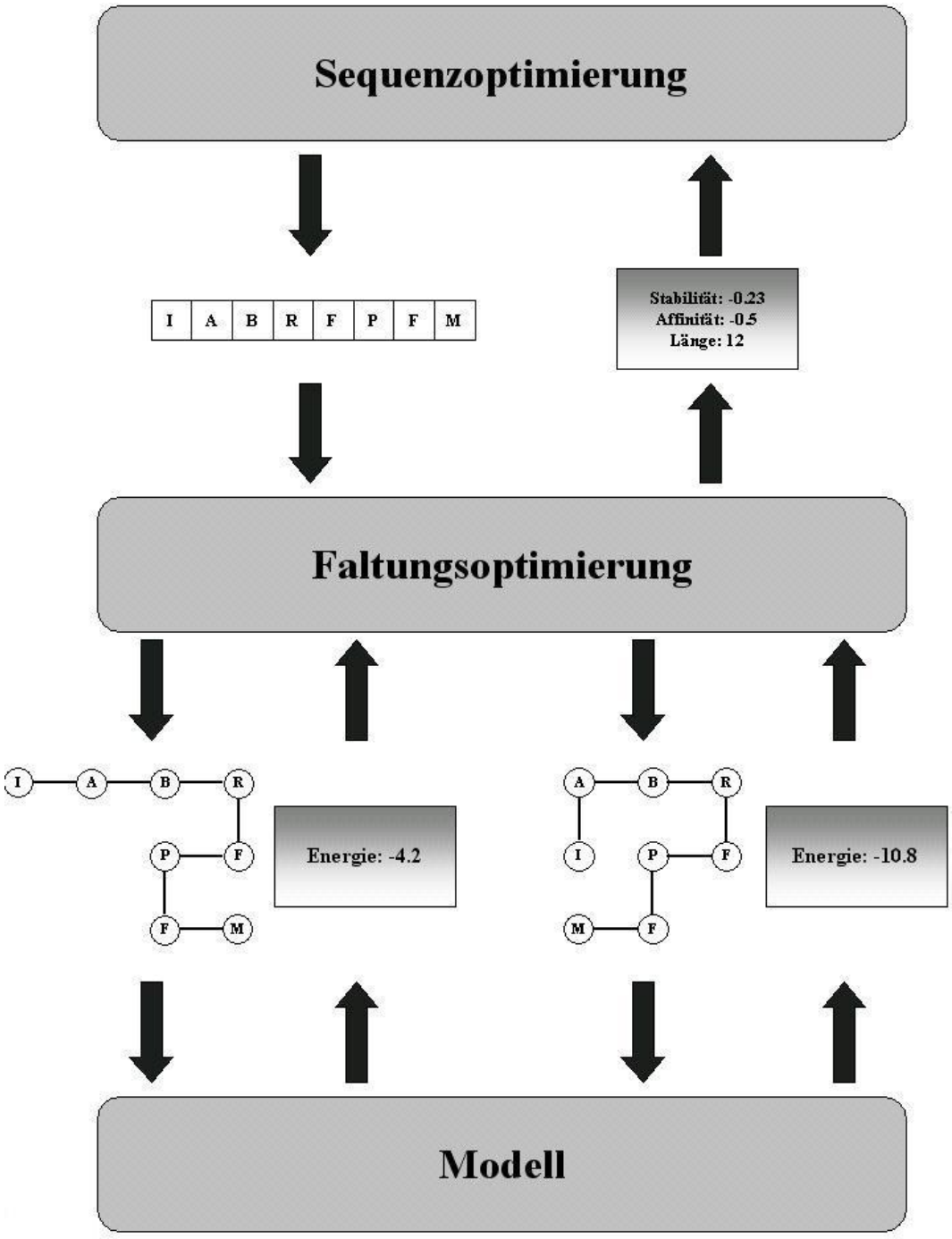


Abbildung 6.3: Modularer Aufbau des Programms

6.2.1 Modell

Das Modell ist die physikalische Grundlage, auf der die gesamte Faltungsberechnung basiert. Der Wahl des Modells kommt deshalb eine enorme Bedeutung zu, es ist das Fundament der Architektur.

An dieser Stelle hat man die Auswahl zwischen exakten sehr komplexen Modellen und ungenauren Modellen, die dafür weniger komplex sind. In den Kapiteln 3 und 4 werden einige dieser Modelle beschrieben. Wir haben uns für ungenauere Modelle entschieden, da sie den Vorteil haben, dass sie dank ihrer geringeren Komplexität mehr Auswertungen in der gleichen Zeit ermöglichen. Die an dieser Stelle eingesparte Rechenzeit kann benutzt werden, um mit dem Sequenzfindungsmodul weitere Sequenzen zu untersuchen.

6.2.2 Faltungsmodul

Im Faltungsmodul wird versucht, innerhalb des Modells eine bestimmte Peptidsequenz optimal zu falten. Hierzu verwenden wir in der Literatur häufig angewandte Heuristiken, sowie ein fehlerfreies Verfahren der vollständigen Enumeration. Ist die vermutlich optimale Faltung gefunden, können bezüglich dieser die verschiedenen übrigen Zielfunktionswerte (z. B. Ähnlichkeit) berechnet werden. Dem Benutzer soll die Möglichkeit gegeben werden, Rezeptorgegenden anzugeben, für die Peptidbinder gefunden werden sollen. Zusätzlich können andere Rezeptoren spezifiziert werden, an die das Peptid möglichst nicht binden soll. Auf diese Weise kann die Spezifität der Bindung des Peptids an bestimmte Rezeptoren erhöht werden. Falls das Peptid ein bestimmtes Epitop enthalten soll, kann dieses ebenfalls angegeben werden. In dieser Schicht sollten ebenfalls einfache, effiziente Verfahren genutzt werden, da innerhalb kurzer Zeit möglichst viele Faltungen getestet werden sollen.

6.2.3 Sequenzfindungsmodul

Das Sequenzfindungsmodul ist die oberste Schicht der Architektur. Hier sollen verschiedene Sequenzen hinsichtlich benutzerdefinierter Kriterien untersucht werden. Da meist mehrere Kriterien verwendet werden, ist hier ein multikriterieller Ansatz gefragt. Das Modul sollte auch das Ergebnis erzeugen: Eine Menge von Sequenzen für das gegebene Problem. Wie groß diese Menge sein soll, hängt von den Bedürfnissen des Benutzers ab und muss deswegen einstellbar sein. Außerdem ist es wahrscheinlich, dass verschiedene Benutzer verschiedene Bewertungen bezüglich der Zielprioritäten haben. Es muss also eine Möglichkeit geben, diese Wünsche in die Optimierung zu integrieren.

Da die Zielfunktionsauswertungen für die Laufzeit der entscheidende Faktor sind, können wir im Bereich des Sequenzfindungsmoduls auch komplexere Verfahren implementieren ohne die Laufzeit wesentlich zu erhöhen. Wenn durch diese komplexen Verfahren Zielfunktionsauswertungen vermieden werden, kann die Laufzeit sogar erheblich verkürzt werden.

6.3 Umsetzung

Wir haben uns entschieden, unser Projekt an die Kea-Software, die von der PG419 des Informatik-Lehrstuhls 11 an der Universität Dortmund entwickelt wurde [81], anzubinden. So kann ein langwieriger Prozess der Softwareentwicklung vermieden werden, der keinerlei wissenschaftliche Bedeutung hat. Kea bietet dem Benutzer ein Framework, um multikriterielle Optimierungsprobleme mittels verschiedener Heuristiken zu lösen. Es stehen eine graphische Benutzeroberfläche sowie ein Kommandozeilenmodus zur Verfügung, was das Programm interaktiv sowie in einem Batchsystem einsatzfähig macht. Zusätzlich bietet Kea eine Reihe von Verfahren, um Ergebnisse zu vergleichen und optisch aufzubereiten. Kea arbeitet nach dem Black-Box-Prinzip. Dank seines modularen Aufbaus (siehe Abbildung 6.4) bietet es dem Entwickler die Möglichkeit, eigene Probleme, Heuristiken, Auswertungsverfahren und graphische Anzeigen einzubinden. Hierfür stehen klar definierte Schnittstellen zur Verfügung [34]. Kea steht unter der GNU General Public License

(GNU GPL[1]) und kann daher für akademische Zwecke kostenlos genutzt werden. Der Entwurf un-

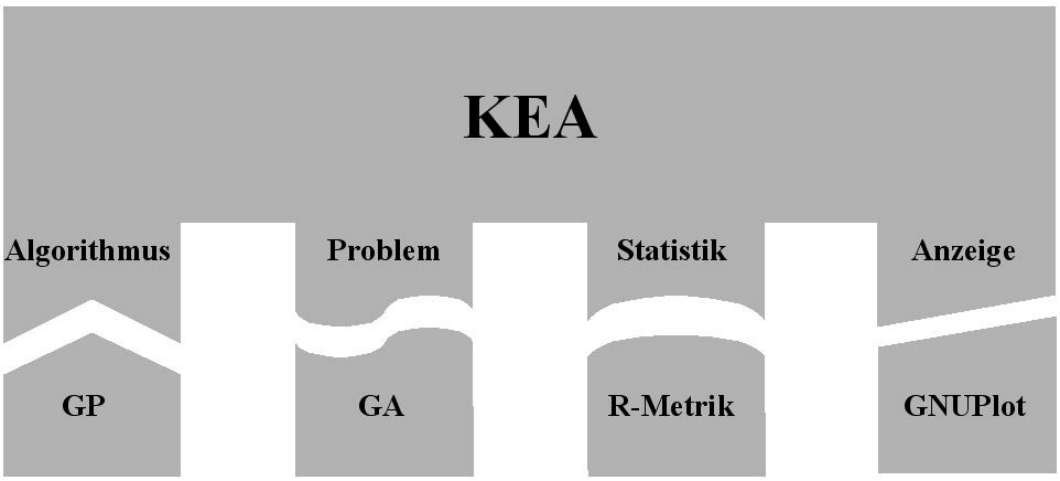


Abbildung 6.4: Schnittstellen der KEA-Software (Auswahl)

seres Programms fand unter Verwendung der UML-Entwicklungsumgebung Together statt. Da das Projekt erweiterbar und auch für andere Nutzer verwendbar sein soll, haben wir unser Design klar strukturiert. Als Programmiersprache wählten wir JAVA, da unser Projekt plattformunabhängig nutzbar sein muss. Dies ist vor allem dann wichtig, wenn auf einem großen Rechencluster mit unterschiedlicher Betriebssystemausstattung parallel optimiert werden soll.

Um den verteilten Entwicklungsprozess zu unterstützen, benutzten wir das bekannte Versionsmangementsystem CVS[2]. Unsere statistischen Auswertungen wurden mit Hilfe der Mathematiksoftware MATLAB gemacht.

6.3.1 Modell

Wie wir bereits erwähnten, ist die Faltungsberechnung der für die Laufzeit entscheidende Faktor. Für die Faltungsberechnung werden die Modelle benötigt. Um die Faltungsberechnung effizient zu gestalten haben wir uns dafür entschieden, vereinfachende Modelle zu verwenden. Um den Vorteil der vereinfachenden Modelle in vollem Umfang nutzen zu können, ist es wichtig, dass die Modelle sehr effizient implementiert werden. Wir haben uns deshalb entschieden, an dieser Stelle hinsichtlich der Übersichtlichkeit Abstriche zu machen, um an Effizienz zu gewinnen.

6.3.2 Faltungsmodul

Da es eine ungeklärte Frage ist, welche Heuristik sich für die Faltung von Peptidsequenzen am besten eignet, untersuchten wir mehrere verschiedene Heuristiken. Nach eingehender Literaturrecherche entschieden wir uns einen genetischen Algorithmus, eine Simulated Annealing-Strategie und einen Monte Carlo-Algorithmus zu implementieren. Dadurch können wir Vergleiche mit anderen, herkömmlichen Strategien bieten. Außerdem wird noch die Alternative der vollständigen Enumeration geboten, um bei kurzen Sequenzen einen Vergleich zum Optimum bieten zu können.

6.3.3 Sequenzfindungsmodul

Um der Anforderung, verschieden lange Sequenzen erzeugen zu können, Rechnung zu tragen, haben wir uns in dieser Schicht für ein GP-System entschieden. Auch andere Verfahren können so

[1] Alles wissenswerte zur GNU GPL findet man unter http://www.gnu.org/licenses/licenses.html.
[2] Siehe http://www.cvshome.org/, weitere Informationen zum Einsatz von CVS finden sich in [42].

angepasst werden, dass sie verschieden lange Individuen verwenden, bei GP-Systemen ist dies allerdings fest vorgesehen. Aus diesem Grund sind die Operatoren so konstruiert, dass sie verschieden lange Individuen erzeugen und handhaben können.

Wie bereits in Abschnitt 6.2.3 angesprochen, haben wir uns dabei für eine multikriterielle Variante entschieden, die ein Archiv fester Größe verwendet, um nichtdominierte Individuen zu speichern (Dominanz siehe 5.1). Um eine Gewichtung der einzelnen Kriterien zu ermöglichen, benutzen wir eine spezielle Strategie zur Aktualisierung des Archivs. Es können hierfür Prioritäten der einzelnen Zielfunktionen angegeben werden. Wenn keine Prioritäten angegeben werden, versucht die Strategie eine möglichst gute Streuung der nichtdominierten Individuen im Lösungsraum zu erzielen. Durch Angabe von Prioritäten werden anstelle einer gleichmäßigen Streuung vermehrt Individuen in der Nähe der Optima hochpriorisierter Kriterien ins Archiv aufgenommen. Eine detailliertere Darstellung dieses Verfahrens findet man in Abschnitt 9.5.

Als Ergebnis dieser Stufe kann der Benutzer sich das Archiv, die letzte Generation oder beides ausgeben lassen.

Bei einem Einsatz unseres Tools über die Diplomarbeit hinaus könnte es interessant sein, vom Anwender festgelegte Sequenzteile in die erzeugten Peptide einzubinden. Um dies zu ermöglichen, haben wir ein Verfahren implementiert, solche Sequenzteile in den Optimierungsprozess einzufügen. Sie können weder verändert werden noch aus der Sequenz verschwinden. Lediglich die Position in der Gesamtsequenz kann sich verändern.

6.3.4 Parameter

Wie von KEA vorgesehen, wird das Verhalten der einzelnen Schichten über Parameterdateien gesteuert. Eine Beschreibung der einstellbaren Parameter findet sich im Anhang C.

6.3.5 Ausgabe

Ein KEA-Optimierungslauf erzeugt als Ausgabe eine „.ast“-Datei. Ein Ausschnitt aus einer Ergebnisdatei ist in Abbildung 6.5 dargestellt. Im Kopf dieser Dateien werden die verwendeten Parameter gespeichert. Der Rumpf der Dateien enthält das Laufergebnis. Da KEA in erster Linie für Heuristiken aus dem Bereich der evolutionären Algorithmen entwickelt wurde, wird das Ergebnis in einzelnen Generationen abgespeichert. Diese Blöcke werden durch Statuszeilen voneinander getrennt. Innerhalb dieser Blöcke finden sich zeilenweise die einzelnen Individuen. Die Zeilen sind in Entscheidungsvektor und Zielfunktionsvektor unterteilt. Im ersten Teil steht im Fall der Optimierung von Peptiden bzw. Proteinen die Darstellung der Sequenz als String. Der zweite enthält die Ergebnisse der einzelnen gewählten Zielfunktionen (vgl. Abschnitt 8).

Neben dieser Art der Ausgabe haben wir eine graphische Ausgabe vorgesehen. Mit ihrer Hilfe können die Faltungen in das „.pdb“-Format[3] exportiert werden. Bei diesem Format handelt es sich um das Standardformat, das auch für Einträge in der Brookhaven Protein Data Bank, einer der größten Datenbanken für Tertiärstrukturen von Proteinen, verwendet wird. Der große Vorteil dieses Formats ist, dass es hierfür bereits diverse gängige Werkzeuge zur Visualisierung(vgl Abbildung 6.6) gibt. Die Programme RasMol und ein Chime-Viewer stehen auf der Webseite `http://www.umass.edu/microbio/rasmol` zum freien Download bereit.

[3]Eine dataillierte Beschreibung dieses Formats findet sich auf den Webseiten der Protein Data Bank unter `http://www.rcsb.org/pdb/info.html#File_Formats_and_Standards`.

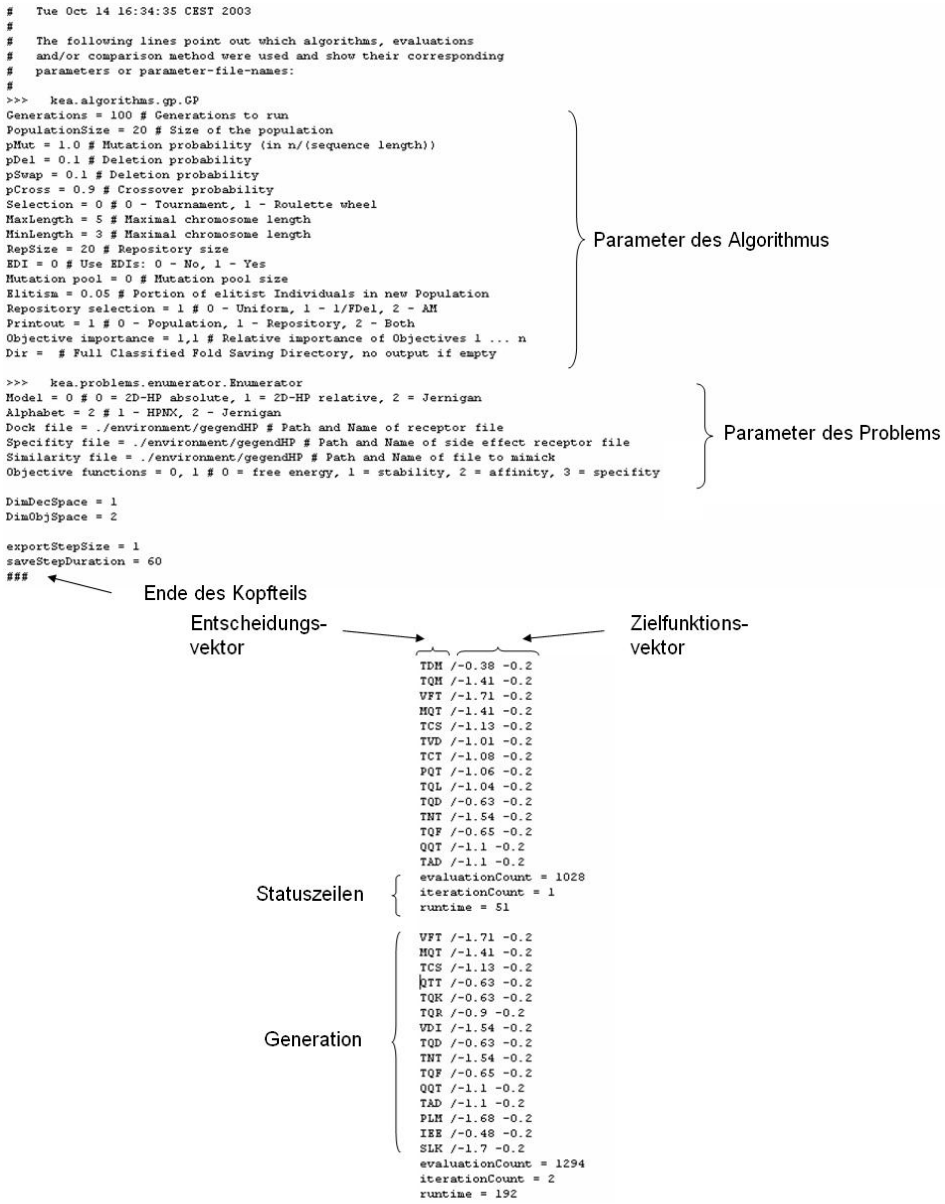


Abbildung 6.5: Ausschnitt aus einer KEA-Ergebnisdatei

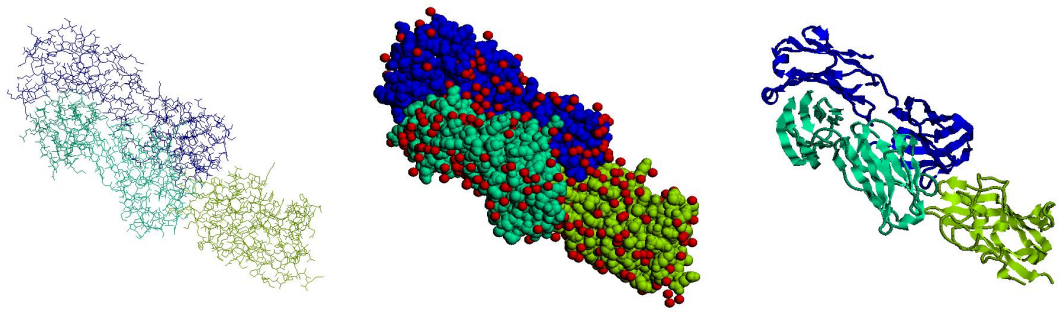


Abbildung 6.6: Drei verschiedene Darstellungsarten von Proteinen mit dem Programm Rasmol am Beispiel eines Komplexes aus einem Teil des Proteins Faktor VIII (grün) mit Teilen eines Antikörpers (blau und türkis).

6.3.6 Kleine Helfer

Um später Untersuchungen zur Epitopimitation durchführen zu können, ist es notwendig, die zu imitierenden Epitope im Format der von uns verwendeten Gitter anzugeben. Da die meisten bekannten Strukturen im „.pdb“-Format gespeichert sind, haben wir ein Programm entwickelt, das die Konvertierung in das von uns verwendete Format vornimmt. Da die im physikalischen Experiment ermittelten Strukturen nicht in Gitterform vorliegen, müssen bei der Konvertierungen die Positionen angepasst werden. Aufgrund der verwendeten Modelle (vereinfachende residuenbasierte Modelle) werden nur die $C\alpha$-Atome betrachtet, die natürliche Bezugspunkte für Residuen sind.

Im ersten Schritt der Konvertierung werden alle Kanten zwischen den einzelnen $C\alpha$-Atomen auf die im Jernigangitter übliche Länge normiert. Im zweiten Schritt wird versucht, die einzelnen $C\alpha$-Atome in das Gitter zu legen.

Für das Einpassen in das Gitter wird ein Fehlervektor $\vec{v}_{err}$ verwendet, der zu Beginn mit einem Vektor der Länge 0 initialisiert wird. Zur späteren Berechnung des Fehlervektors wird der Vektor $\vec{v}_{dir}$, der zwischen der alten normierten Position $Pos_{norm_{old}}$ und der aktuellen normierten Position $Pos_{norm_{act}}$ liegt und um den Fehlervektor verschoben wurde, benötigt. Es wird jeweils die Gitterposition P_{lat} gewählt, die den geringsten Abstand zu der erwünschten Position, der letzten gewählten Gitterposition $P_{lat_{old}}$ um den Vektor $\vec{v}_{dir}$ bereinigt, hat. Der neue Fehlervektor ergibt sich aus der Differenz des Richtungsvektors zwischen alter Gitterposition $P_{lat_{old}}$ und neuer Gitterposition $P_{lat_{act}}$ und des Vektors $\vec{v}_{dir}$.

Dieser Prozess wird durch den folgenden Pseudocode-Algorithmus und die Abbildung 6.7 verdeutlicht.

1. Normiere die Abstände zwischen den $C\alpha$-Atomen.
2. Intialisiere einen Fehlervektor $\vec{v}_{err} = \vec{0}$.
3. Solange nicht alle $C\alpha$-Atome im Gitter platziert sind:
   (a) Errechne Vektor $\vec{v}_{dir} = (Pos_{norm_{act}} - Pos_{norm_{old}}) - \vec{v}_{err}$.
   (b) Errechne erwünschte Position $P_{comp} = P_{lat_{old}} + \vec{v}_{dir}$.
   (c) Ermittle eine gültige Gitterposition P_{lat} mit minimalem Abstand von der erwünschten Position P_{comp}.
   (d) Errechne neuen Fehlervektor $\vec{v}_{err} = (P_{lat_{act}} - P_{lat_{old}}) - \vec{v}_{dir}$.

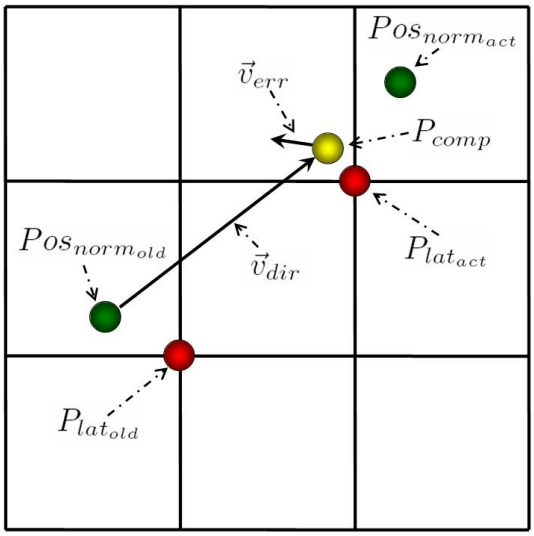


Abbildung 6.7: Einordnung experimentell ermittelter Atompositionen in ein Gitter

Kapitel 7

Faltungsoptimierung

> „Wenn du ein Problem lösen kannst, warum solltest du dich sorgen? Wenn du es nicht lösen kannst, was bringt es, sich zu sorgen?“
>
> Shantideva

In diesem Kapitel befassen wir uns mit dem Peptidfaltungsproblem (siehe Abschnitt 2.2) für eine feste Sequenz.

Im ersten Abschnitt werden die für die Faltungsberechnung verwendeten Gittermodelle und Energiefunktionen vorgestellt. Wenn die Sequenz von probabilistischen Verfahren in diesen Modellen gefaltet wird, kann es zu Überlagerungen an einzelnen Gitterpositionen kommen. Da in der Natur keine Überlagerungen vorkommen können, haben wir für diesen Fall eine Constraint Handling-Methode entwickelt, die ebenfalls in diesem Abschnitt behandelt wird.

Im nachfolgenden Abschnitt stellen wir die von uns eingesetzten Algorithmen (Enumerator, GA, SA) vor, mit deren Hilfe wir gute Faltungen ermitteln wollen.

7.1 Modelle

Im Rahmen der Diplomarbeit haben wir uns für zwei verschiedene Modelle entschieden. Zunächst wurde ein einfacheres, zweidimensionales, rechtwinkliges Gittermodell (vgl. Abschnitt 4.2) implementiert. Anhand dieses einfachen Modells haben wir die verwendeten Algorithmen auf ihre Korrektheit hin überprüft und danach sichergestellt, dass sie zur Lösung von Problemen dieser Problemklasse geeignet sind. Als zweite Variante haben wir uns für das dreidimensionale Gitter von Raghunathan und Jernigan (siehe Abschnitt 4.2.3) entschieden. Es ist weitaus komplexer, aber auch realitätsnäher, und vor allem empirisch besser belegt, als andere vorgestellte Gitter. Wir entschieden uns gegen ein kontinuierliches Modell, da solche Modelle nicht vollständig enumerierbar sind und wir daher keinen Vergleich mit dem Optimum bieten können. Ein weiterer, sehr wichtiger Grund ist der, dass Sequenzen in verhältnismäßig kurzer Zeit gefaltet werden müssen, um die darüberliegende Schicht nicht zu laufzeitaufwendig zu gestalten. Kontinuierliche Modelle sind hierfür unserer Ansicht nach zu komplex.

Für beide Modelle stehen die HPNX-Energiefunktion (siehe Abschnitt 4.3.2) und die Jernigan-Energiefunktion (siehe Abschnitt 4.3.3) zur Verfügung. Sie unterscheiden sich in der Größe des verwendeten Alphabets und den Potentialfunktionen. In den meisten Untersuchungen verwenden wir o.B.d.A. die Kombinationen zweidimensionales Gitter und HPNX-Energiefunktion (im Folgenden 2D-HPNX-Modell bezeichnet) und dreidimensionales Gitter nach Raghunathan und Jernigan und Jernigan-Energiefunktion (im Folgenden 3D-Jernigan). Die wichtigste Frage beim Entwurf des Modells ist die Codierung der Eingabe. Die dreidimensionale Form der Faltung muss durch eine Menge von Inputvariablen beschrieben werden. Da wir in Gittern mit konstanter Nachbarschaftsgröße arbeiten, ist es möglich, die Nachbarn eines Punktes zu indizieren. Wir können eine

Faltung w der Länge $|w|$ in einem Gitter mit einer Nachbarschaft der Größe n durch folgenden Eingabevektor beschreiben:

$$x_{abs} = \begin{pmatrix} x_1 \\ \vdots \\ x_{|w|-1} \end{pmatrix}, x_i \in \mathbb{N}, 1 \leq x_i \leq n$$

Hier beschreibt jede Koordinate x_i die absolute Position des Nachbarn von Residuum w_i, auf der sich Residuum w_{i+1} befindet. Jede mögliche Koordinate steht also für einen Richtungsvektor, der auf die letzte Residuenposition addiert wird.

Die zweite verwendete Codierungsvariante ist die relative Codierung. Richtungen werden nun nicht mehr nach dem Schema „Nord-Süd-Ost-West“, sondern relativ codiert (also „Links-Geradeaus-Rechts“). Das Koordinatensystem wird während des Faltungsprozess mitgedreht. Formal definiert sieht der Eingabevektor dann folgendermaßen aus:

$$x_{rel} = \begin{pmatrix} x_1 \\ \vdots \\ x_{|w|-1} \end{pmatrix}, x_i \in \mathbb{N}, 1 \leq x_i \leq n-1,$$

wobei jede Position x_i eine Rotationsmatrix darstellt, um die das Koordinatensystem gedreht wird. Die nächste Residuenposition wird dann durch die Addition des gedrehten Einheitsvektors auf die Vorgängerposition ermittelt. In den Tabellen 7.1 und 7.2 werden die von uns verwendeten Codierungen für das zweidimensionale Modell dargestellt. Die absolute Codierung zeigt als Cha-

Tabelle 7.1: 2-Dimensionale Codierung des Eingabevektors (Absolut)

Eingabe	Richtung	Richtungsvektor
1	Nord	$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$
2	Ost	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$
3	Süd	$\begin{pmatrix} 0 \\ -1 \end{pmatrix}$
4	West	$\begin{pmatrix} -1 \\ 0 \end{pmatrix}$

rakteristikum, dass eine Punktmutation[1] an einer Vektorposition nur geringe Auswirkungen auf die Form der Faltung hat. Bei der relativen Codierung kann eine veränderte Rotation die Form des Proteins maßgeblich ändern (siehe Abbildungen 7.2 und 7.3). Ob sich dies auf die Performance auswirkt, also ob die Energiefunktion dadurch schwieriger zu optimieren wird, wird in Kapitel 10 erörtert.

Ein großes Problem für Heuristiken ist die Erzeugung ungültiger Lösungen, die eine oder mehrere Nebenbedingungen verletzen. Bezogen auf das Faltungsproblem heißt das: Die Algorithmen erzeugen Faltungen, die physikalisch nicht möglich sind. In der von uns verwendeten Eingabecodierung ist nur eine Verletzung dieser Nebenbedingungen möglich, nämlich eine Überschneidung

[1] Auf dem Gebiet der evolutionären Algorithmen spricht man bei Änderungen der zu optimierenden Größe von Mutationen. Im Unterschied zu der biologischen Verwendung des Begriffs Mutation hat man es an dieser Stelle nicht mit einer Mutation in der Sequenz sondern mit einer Veränderung der Faltung zu tun.

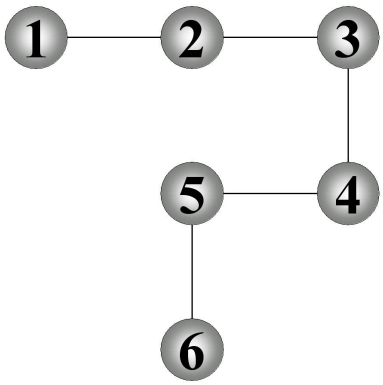


Abbildung 7.1: Vergleich absolute und relative Codierung
Absolut: 1,2,2,3,4,3
Relativ: 2,3,2,3,3,1

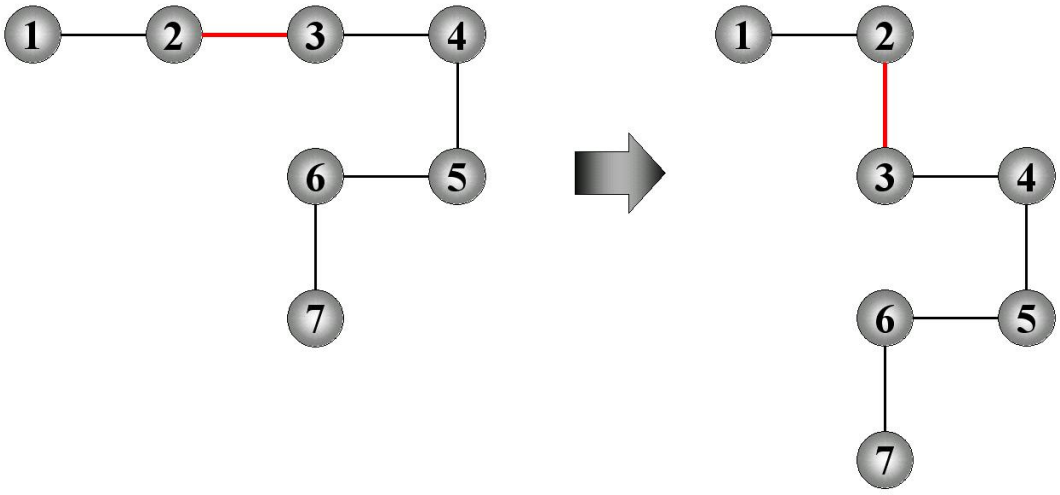


Abbildung 7.2: Mutation (Absolute Codierung)

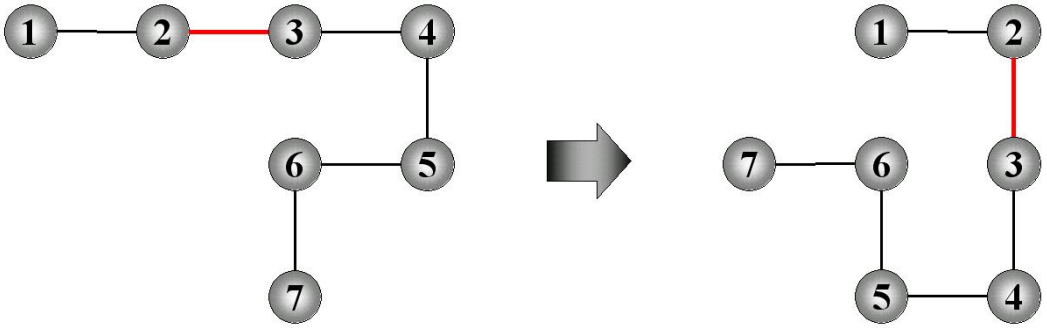


Abbildung 7.3: Mutation (Relative Codierung)

Tabelle 7.2: -Dimensionale Codierung des Eingabevektors (Relativ)

Eingabe	Richtung	Richtungsvektor
1	Links	$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$
2	Geradeaus	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
3	Rechts	$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$

innerhalb der Sequenz. Überschneidungen sind Faltungen, bei denen einer Gitterposition mehrere Aminosäuren zugeordnet werden. Da viele Sequenzen ungültig sein können, ist es durchaus möglich, dass ein Großteil, wenn nicht sogar die gesamte Population aus ungültigen Individuen besteht. Hier gibt es keine Suchrichtung und die Heuristik verkommt zu einer randomisierten Suche. Ist auch die Nachbarschaft ungültig, ist es unwahrscheinlich, wieder gültige Lösungen zu finden. Wenn man dies verhindern will, ist eine Constraint Handling-Methode vonnöten.

Die Energiefunktion ist jedoch nur für gültige Lösungen definiert. Da jedoch auch ungültige Lösungen erzeugt werden können, überlegten wir uns eine Modifikation, mittels derer wir die Suche weg von ungültigen hin zu überschneidungsfreien Faltungen leiten können. Zuerst einmal muss sichergestellt werden, dass ungültige Lösungen schlechter bewertet werden, als gültige. Dies lässt sich dadurch garantieren, dass wir sie mit einem positiven Fitnesswert ausstatten. Sollten nun alle ungültigen Faltungen den gleichen Fitnesswert erhalten? Dies würde wie oben beschrieben keinen Sinn machen, denn uns würde erneut die Suchrichtung fehlen. Wir brauchen eine Funktion, um die "Nähe" einer ungültigen Lösung zu gültigen Lösungen zu bewerten. Zusätzlich sollte diese Funktion effizient zu berechnen sein. Zwei Größen, begründet durch die folgenden Überlegungen, können maßgeblich für die Funktion sein:

1. C, die Anzahl der Überschneidungen innerhalb der Faltungen. Ein Individuum sollte schlechter bewertet werden, wenn es mehrfache Überschneidungen enthält.

2. P, die Überschneidungsposition, die am nächsten an der Sequenzmitte liegt. Ein Individuum, das eine Überschneidung in der Mitte enthält, ist schwerer in ein gültiges zu verwandeln, als eines, das eine Überschneidung am Rande der Sequenz enthält.

Daraus entwickelten wir folgende Fitnessfunktion für ungültige Individuen:

$$F_{crash}(w) = C|w| - r\sqrt{(P - \frac{|w|}{2})^2}$$

Der Faktor r spiegelt wider, wie sehr wir Überlegung 2 bezüglich Überlegung 1 gewichten. Für den genetischen Algorithmus wählten wir $r = 1$, da wir durch ein Crossover die Möglichkeit haben, eine Überschneidung gleich welcher Position zu entfernen. Bei zwei Überschneidungen gestaltet sich dies schwieriger. Für das Simulated Annealing wählten wir $r = 4$, wir gewichteten also die Position der Überschneidung stärker. Uns steht in diesem Verfahren kein Crossover zur Verfügung, und für Mutationen ist eine Überschneidung in der Mitte der Sequenz äußerst schwer zu korrigieren.

7.2 Algorithmen

In diesem Abschnitt erläutern wir die Verfahren, mit denen wir die native Konformationen von Peptidsequenzen und die freie Energie dieser Konformationen berechnen wollen. Für die Faltung

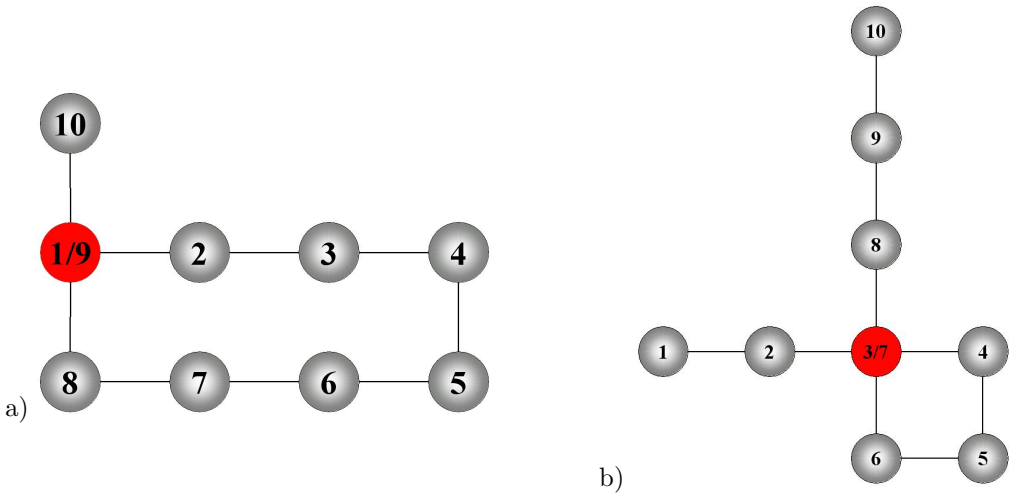


Abbildung 7.4: Überschneidungen: a) Einfach zu bewältigen b) Schwierig zu bewältigen

der Peptide benutzen wir die in Abschnitt 7.1 beschriebenen Gitter und die von uns ausgewählten Energiefunktionen.

Wir haben uns an dieser Stelle für drei verschiedenen Verfahren entschieden.Als erstes haben wir ein deterministisches Verfahren implementiert. Es hat den Vorteil, dass es immer das Optimum findet. Bei diesem Verfahren ergibt sich jedoch das Problem, dass es aufgrund der Größe des Suchraums zu sehr langen Laufzeiten kommt. Bei den anderen beiden Verfahren handelt es sich um randomisierte Suchheuristiken. Sie haben den Vorteil, dass sie im Vergleich zum deterministischen Verfahren eine deutlich niedrigere Laufzeiten haben. Sie haben jedoch den Nachteil, dass sie nicht immer das Optimum finden. Für diese randomisierten Verfahren kann keine Garantie für die Güte der gefundenen Lösungen gegeben werden. Wir haben uns für einen genetischen Algorithmus und Simulated Annealing entschieden.

7.2.1 Enumerator – Vollständige Suche

Der Enumerator ist ein deterministischer, iterativer Algorithmus, der durch vollständige Enumeration des Faltungsraumes eine garantiert optimale Lösung des Faltungsproblems für eine gegebene Sequenz bietet. Durch die NP-Vollständigkeit dieses Problems ist die Laufzeit des Enumerators leider nur exponentiell beschränkt. Wir haben uns aber bemüht, durch das Weglassen ungültiger oder symmetrischer Sequenzen die Laufzeit etwas zu verkürzen. Der Enumerator arbeitet alle möglichen Faltungen der Reihe nach ab. Durch vorgegebene Regeln erzeugt er keine deckungsgleichen Faltungen. Erzeugt er an einer Position einen „Crash“, eine Kreuzung innerhalb der Sequenz, so schließt er alle Sequenzen dieses Typs von der weiteren Berechnung aus. Da der Enumerator alle Sequenzen berechnet, kann auch die Stabilitätsfunktion fehlerfrei berechnet werden.

Eine Liste der beim Enumerator einstellbaren Parameter findet sich im Anhang C.

7.2.2 GA – Populationsbasierte randomisierte Suchheuristik

Als ersten stochastischen Optimierungsalgorithmus haben wir einen GA implementiert. Er folgt dem in Abbildung 7.5 beschriebenen Schema. Die einzelnen darin vorkommenden Elemente wurden wie folgt von uns ausgefüllt:

Bewertung Als Fitnessfunktion benutzen wir die für eine bestimmte Faltung berechnete freie Energie (siehe Abschnitt 7.1).

Selektion An dieser Stelle bieten wir die in Abschnitt 5.5.2 beschriebenen Verfahren Rouletteradselektion und Turnierselektion an.

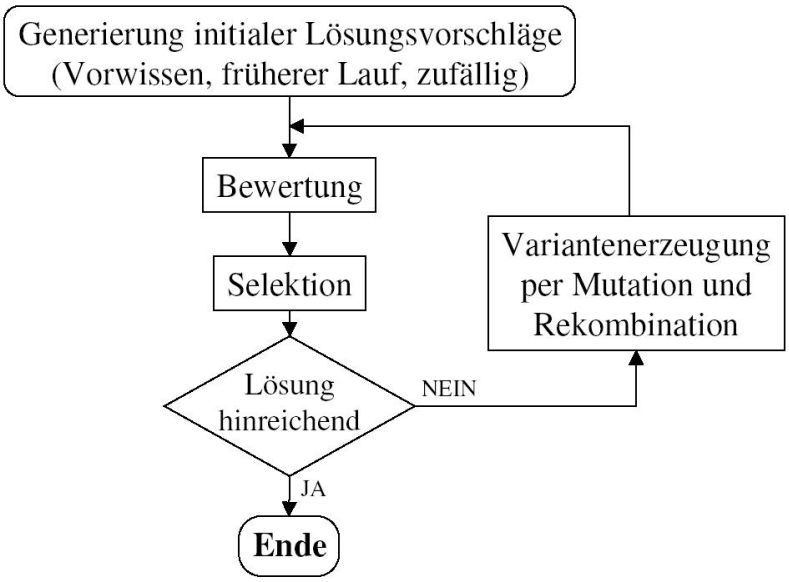


Abbildung 7.5: Grundalgorithmus der evolutionären Algorithmen (aus [96])

Abbruchkriterium Der Optimierungsprozess wird nach einer vorher festgelegten Anzahl Iterationen abgebrochen.

Mutation Als Mutationsverfahren haben wir uns für die in Abschnitt 5.5.2 vorgestellte Punktmutation entschieden.

Rekombination Die Rekombination haben wir durch das in Abschnitt 5.5.2 beschriebene One-Point-Crossover realisiert.

Diese gebräuchliche Variante haben wir durch ein paar Erweiterungen ergänzt:

Um die Faltungsdiversität in den Populationen zu erhalten, wird der Selektionsprozess nicht nur nach den Fitnesswerten der beteiligten Individuen entschieden. Die bei der Selektion verwendeten Fitnesswerte werden durch einen neu eingeführten Nichingwert (vgl. Abschnitt 5.7) modifiziert. Für eine gegebene Population wird ermittelt, wie häufig einzelne Faltungen vorkommen. Häufig vorkommende Faltungen werden in der Selektion benachteiligt, damit eine gute Faltung nicht die ganze Population dominiert. Um das Ermitteln der Faltungshäufigkeiten zu beschleunigen, betrachten wir anstelle der Faltungen die Energiewerte. Wir gehen davon aus, dass gleiche Energiewerte bedeuten, dass gleiche Faltungen vorliegen. Obwohl diese Annahme nicht immer richtig ist, zeigen die Versuchsergebnisse (vgl. Kapitel 10), dass durch die Einführung des Nichings der GA signifikant verbessert wurde. Das Niching ist für Turnierselektion und Rouletteradselektion verschieden umgesetzt:

Turnierselektion Während bei der Turnierselektion üblicherweise das Individuum x' mit der besten Fitness das Turnier gewinnt, gewinnt es bei dieser Variante nur mit einer bestimmten Wahrscheinlichkeit $P_{Sel}(x')$ das Turnier. Die Wahrscheinlichkeit hängt von der Anzahl $|x'|$ Individuen mit gleicher Fitness[2] wie das Individuum x', der Anzahl der Individuen $|x''|$ mit der gleichen Fitness wie das Individuum x'', der Populationsgröße $|Pop|$ und dem vorher festgelegten Nichinggewicht G ab. Wie man ersehen kann, wird die Überlebenswahrschein-

[2]Wie bereits gesagt gehen wir an dieser Stelle davon aus, dass gleiche Energiewerte gleichen Faltungen entsprechen. Der Energiewert entspricht dem Fitnesswert.

lichkeit kleiner, falls $|x'|$ größer als $|x''|$ ist, also x' häufig in der Population vorkommt.

$$P_{Sel}(x') = 1 - \frac{\frac{|x'|}{|x''|} * G}{|Pop|}$$

Rouletteradselektion Für die Einteilung des Rouletterads werden die umgekehrten Ränge der Individuen $\overline{Rang(x)}$ der Population verwendet. Das schlechteste Individuum einer Population erhält den Rang $\overline{Rang(x)} = 1$. Das beste Individuum einer Population erhält bei der Ppopulationsgröße $|Pop| = n$ den Rang $\overline{Rang(x)} = n$. Die Größe des Anteils von Individuum x am Rouletterad $B_{Rou}(x)$ wird anschließend in Abhängigkeit von der Anzahl der Individuen $|x|$ mit der gleichen Fitness wie das Individuum x und dem vorher festgelegten Nichinggewicht G angepasst.

$$B_{Rou}(x) = \left(\frac{\overline{Rang(x)}}{|x|^G} \right)^2$$

Um zu verhindern, dass durch die Selektion oder die Anwendung der Operatoren ein vormals bestes Individuum verloren geht, benutzen wir Elitismus. Hierfür werden die besten fünf Prozent[3] der Vorgängerpopulation unverändert in die nächste Generation übernommen. Das beste gefundene Individuum kann sich deshalb nie verschlechtern.

Eine Liste der beim GA einstellbaren Parameter findet sich im Anhang C.

7.2.3 SA – Nicht-populationsbasierte randomisierte Suchheuristik

Um mit bereits erzielten Ergebnissen vergleichbar zu sein, haben wir Simulated Annealing implementiert. Beim SA handelt sich um ein gebräuchliches Verfahren für die Berechnung der Faltung mit minimaler Energie auf gitterbasierten Modellen. Der SA folgt dem in Abschnitt 5.3 beschriebenen Ablauf.

Für dieses Verfahren stellen wir verschiedene Abkühlstrategien zur Verfügung. Mit Hilfe dieser Strategien wird jeweils eine neue Temperatur T_{i+1} aus der alten Temperatur T_i, der Temperatur zu Beginn der Optimierung T_{max}, der Temperatur am Ende der Optimierung T_{min}, der maximalen Anzahl an durchzuführenden Iterationen i_{max} und der aktuellen Iteration i_{cur} berechnet. Die Temperatur zu Beginn der Optimierung T_{max} und die Temperatur am Ende der Optimierung T_{min} werden aus der Akzeptanzwahrscheinlichkeit schlechterer Punkte zu Beginn der Optimierung p_{max}, der Akzeptanzwahrscheinlichkeit schlechterer Punkte am Ende der Optimierung p_{min} und der maximalen Anzahl durchzuführender Iterationen i_{max} berechnet. Die Akzeptanzwahrscheinlichkeiten zu Beginn der Optimierung p_{max} und am Ende der Optimierung p_{min} werden vom Anwender in den Parametern festgelgt. Die Temperatur wird zur Berechnung der Metropolisfunktion (vgl. Abschnitt 5.3) benötigt, mit der ermittelt wird, ob ein neuer Suchpunkt akzeptiert wird, wenn er schlechter ist als der aktuelle Suchpunkt. Die verschiedenen Strategien werden im Folgenden dargestellt.

Eine Liste der beim SA einstellbaren Parameter findet sich im Anhang C.

1 - linear

$$T_{i+1} = T_{max} - i_{cur} \frac{T_{max} - T_{min}}{i_{max}}$$

2 - exponentiell

$$T_{i+1} = T_i * (-\frac{1}{\log_2 p_{min}})^{\frac{1}{i_{max}}}$$

[3] Die Anzahl der zu übernehmenden Individuen wird abgerundet. Es wird aber immer mindestens das beste Individuum übernommen.

3 - hyperbolisch

$$T_{i+1} = \frac{A}{i_{cur} + 1} + T_{max} - A$$

$$A = \frac{((T_{max} - T_{min}) * (i_{max} + 1))}{i_{max}}$$

4 - logarithmisch

$$T_{i+1} = T_{max} - {i_{cur}}^{A}$$

$$A = \frac{\ln (T_{max} - T_{min})}{\ln i_{max}}$$

5 - sigmoid Typ A

$$T_{i+1} = \frac{T_{max} - T_{min}}{1 + e^{0,3(i_{cur} - \frac{i_{max}}{2})}} + T_{min}$$

6 - sigmoid Typ B

$$T_{i+1} = T_{max} * e^{(-A * {i_{cur}}^2)}$$

$$A = \frac{1}{i_{max}^2} * \ln \frac{T_{max}}{T_{min}}$$

7 - Metropolis Monte Carlo

$$T_{i+1} = T_i$$

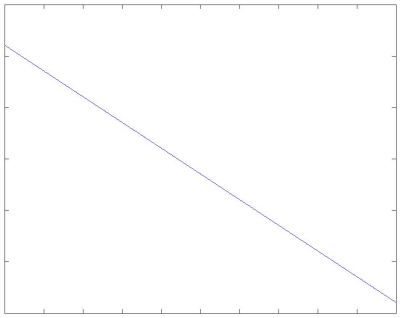


Abbildung 7.6: Schematische Darstellung der linearen Funktion

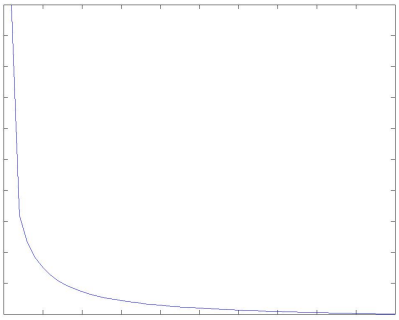


Abbildung 7.7: Schematische Darstellung der hyperbolischen Funktion

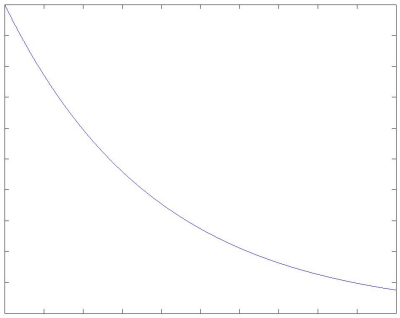


Abbildung 7.8: Schematische Darstellung der exponentiellen Funktion

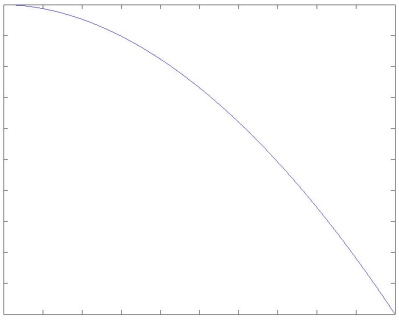


Abbildung 7.9: Schematische Darstellung der logarithmischen Funktion

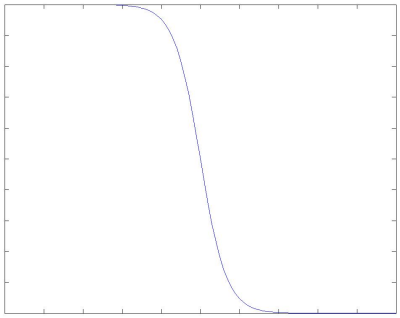


Abbildung 7.10: Schematische Darstellung der sigmoiden Funktion (Typ A)

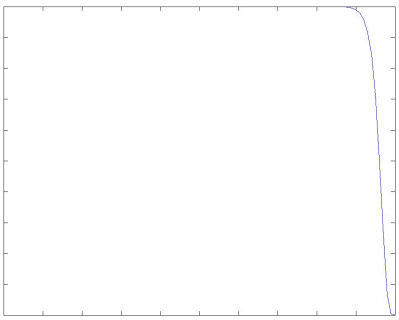


Abbildung 7.11: Schematische Darstellung der sigmoiden Funktion (Typ B)

Kapitel 8

Zielfunktionen

> „Man muß sich einfache Ziele setzen, dann kann man sich komplizierte Umwege erlauben.“
>
> Charles de Gaulle (1890-1970), frz. General u. Politiker

Im folgenden Kapitel stellen wir unsere Ansätze dar, um anhand der in Kapitel 7 vorgestellten Proteinfaltungssysteme möglichst optimale Sequenzen für ein gegebenes Problem zu finden. Das System arbeitet nach dem Prinzip der Black-Box-Optimierung (siehe Abschnitt 5.2). Eine unbekannte Funktion der Form

$$F(w) = y_1, ..., y_n,$$

wobei w die Proteinsequenz sei, soll optimiert werden. In diesem Kapitel werden verschiedene Zielfunktionen $y_1, ..., y_n$ motiviert und entwickelt.

8.1 Einleitung

Es gibt eine Menge sinnvoller Kriterien für die Fitness eines Peptids. Bei der Auswahl kommt es maßgeblich darauf an, welche Eigenschaften wir von dem gewünschten Peptid fordern und wo es eingesetzt werden soll. Es sind viele verschiedene Einsatzgebiete denkbar, wie in Abschnitt 6.1 bereits dargelegt wurde. Es kann sein, dass nicht immer alle Kriterien für den Nutzer von Bedeutung sind. Daher halten wir es in unserem Design offen, welche Zielfunktionen der Benutzer verwenden möchte und welche er nicht benötigt. Auch eine Gewichtung einzelner Zielfunktionen wird ermöglicht.

8.2 Potentialfunktionen

Das Kriterium, welches als erstes auf der Hand liegt, ist die Energiefunktion, die Bestandteil des Molekülmodells ist (vgl. Kapitel 4). Mit Hilfe dieser Energiefunktion können andere relevante Größen abgeleitet werden, wie z. B. die Stabilität der nativen Konformation. Wir bieten dem Benutzer die Möglichkeit, diese Funktion zu nutzen, da in der Literatur (z. B. [93, 88]) fast ausschließlich die Energiefunktion verwendet wird. Sie kennzeichnet die native Faltung und dient dazu, diese im Prozess der Faltungsoptimierung zu finden. Es ist wichtig, zu bemerken, dass wir an dieser Stelle die Zielfunktionen für die Sequenzoptimierung vorstellen. Die Potentialfunktion wird jedoch in erster Linie in der Faltungsoptimierung verwendet. Ob sie auch als Kriterium für eine gute Sequenz gewählt wird, ist davon unabhängig (siehe Abb. 6.3).

8.3 Stabilität

In der klassischen Mechanik gilt ein Zustand als stabil, wenn die Energiefunktion dort ein Minimum erreicht hat. In der Statistischen Mechanik und Thermodynamik (siehe Abschnitt 2.2) wird dieses

Konzept durch statistische Elemente ergänzt und verallgemeinert. Durch Interaktionen mit umliegenden Molekülen, wie z. B. dem Lösungsmittel, kann das Protein zufällig eine höhere Energie erhalten und seinen Zustand ändern. Ein Molekül befindet sich also mit einer gewissen Wahrscheinlichkeit in einem Zustand $X_0(E_0)$ mit Energie E_0. Diese Wahrscheinlichkeit $p(X_0)$ ist abhängig von der Energie und der Temperatur:

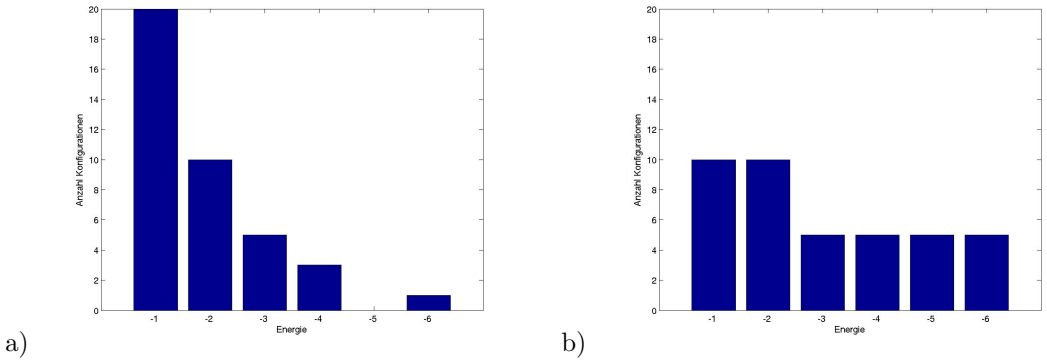
$$p(X_0) = \frac{e^{-\frac{E_0}{k_b T}}}{Z}$$

$$Z = \sum_{i=0}^{|X|} e^{-\frac{E_i}{k_b T}}$$

Eine Faltung gilt als besonders stabil, wenn $p(E_0)$ maximal wird. Nun ist es leider nicht möglich, die Zustandssumme Z über alle Zustände $|X|$ des Moleküls zu berechnen. In einer Gitterstruktur ist dies zwar per vollständiger Enumeration für kurze Sequenzen möglich, jedoch wächst die Zustandszahl wie in Kapitel 4 beschrieben exponentiell. Wir approximieren die Zustandssumme, indem wir bei der Faltungsoptimierung für jedes Energieniveau E_i die Anzahl gefundener Faltungen $|X(E_i)|$ protokollieren. Hierbei ist zu bedenken, dass wir heuristisch arbeiten. Der Wert $|X(E_i)|$ ist die Anzahl der von der Heuristik gefundenen Faltungen auf dem Energieniveau E_i. Dieser Wert kann vom tatsächlichen Wert stark abweichen.

$$Z' = \sum_{i=0}^{|E_i|} |X(E_i)| e^{-\frac{E_i}{k_b T}}$$

Das Ziel ist es, einen möglichst steilen Abfall in der relativen Häufigkeit der Energieniveaus zu haben, so dass im Idealfall auf dem untersten Level nur noch ein optimaler Zustand existiert (siehe Abbildung 8.1). Mit unseren Faltungsoptimierungsheuristiken nehmen wir ein ungleichmäßig verteiltes Sample aus dem Konformationsraum. Die Sampling-Dichte ist gerade im Bereich niedriger Energien besonders groß. Dieser Effekt erklärt sich durch die hohe Qualität der Heuristiken bei der Auffindung von lokalen und globalen Minima. Ob er sich bemerkbar macht, wird in Kapitel 10 untersucht. Wir hoffen, hiermit die Stabilität des Peptids weitaus realistischer darzustellen, als es die Energiefunktion kann.



a) b)

Abbildung 8.1: Histogramm: a) Stabile Faltung b) Instabile Faltung

8.4 Affinität

Oft ist es das Ziel einer Simulation, ein Peptid zu entwickeln, das an einen vorgegebenen Rezeptor bindet. Man spricht hier von Rezeptoraffinität. Wir bieten dem Benutzer die Möglichkeit, seinen Rezeptor R im verwendeten Gitterformat anzugeben (siehe Abschnitt 6.3.6). Das aktive Zentrum

kann in Form eines Rechtecks angegeben werden. Das Programm berechnet mittels den in Kapitel 7 vorgestellten Methoden mögliche Faltungen. Diese werden mit allen möglichen Dockingpositionen eines Rezeptors in Verbindung gebracht (Brute-Force-Strategie). Diese Brute-Force-Methode besitzt natürlich den Nachteil eines großen rechnerischen Aufwands, allerdings garantiert sie die optimale Dockingposition. Hier wird nun mittels der Energiefunktion die Bindungsenergie innerhalb des Proteins und zwischen Protein und Rezeptor ermittelt und summiert (siehe Abbildung 8.2). In ersten Versuchen stellten wir jedoch fest, dass die Optimierung der Peptid-Rezeptor-Energie zu

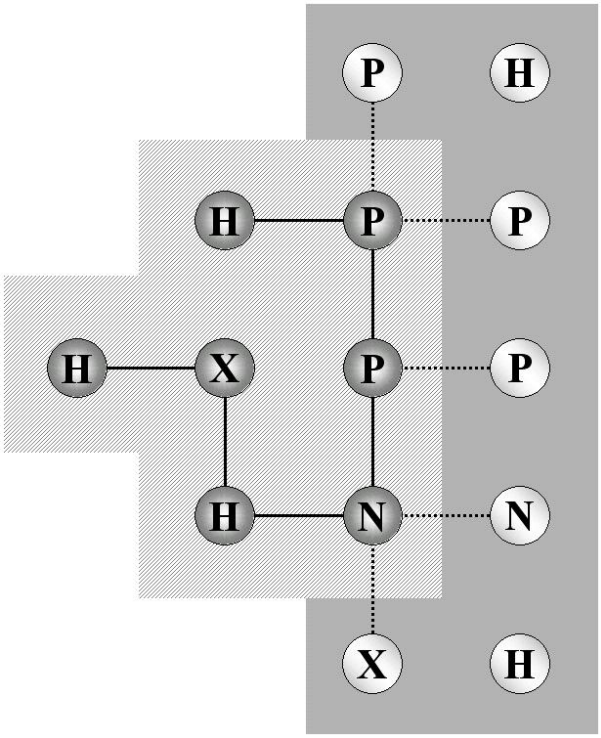


Abbildung 8.2: Affinitätsmessung

extrem hydrophoben Sequenzen (z. B. HHHHPH) führte. Die H-H-Bindungen brachten hohe Affinitätsscores, obwohl die Sequenzen in ihrer nativen Faltung ein fast genauso tiefes Energiepotential und daher in Wirklichkeit eine geringe Affinität haben. Deswegen haben wir uns entschieden, den Affinitätsscore um die Energie der nativen Faltung zu bereinigen:

$$E_{Aff}(w, R) = E_{Prot-Prot} + E_{Prot-Rez} - E_{Nat}$$

Es sei bemerkt, dass hier zwischen $E_{Prot-Prot}$ und E_{Nat} unterschieden wird. $E_{Prot-Prot}$ stellt das Potential dar, welches innerhalb des Proteins in der Dockingform vorliegt. E_{Nat} stellt die minimale Energie in nativer Konformation (siehe 8.2) dar.

8.5 Spezifität

Wir modellieren Spezifität durch die Affinität des Proteins an einen dem Zielrezeptor sehr ähnlichen Rezeptor $\overline{R}$, den Antirezeptor. Auch hier verwenden wir wieder den in Abschnitt 8.4 vorgestellten Algorithmus. Da wir minimieren, ist die Spezifität folgendermaßen gegeben:

$$E_{Spez}(w, R, \overline{R}) = E_{Aff}(w, R) - E_{Aff}(w, \overline{R})$$

Je spezifischer die Bindung des Proteins an den Rezeptor R ist, umso tiefer liegt E_{Spez}.

8.6 Ähnlichkeit

Ein anderer denkbarer Anwendungsfall ist die Epitopimitation (siehe Abschnitt 6.1). Die Zielfunktion $E_{Ähn}$ soll die räumliche Ähnlichkeit einer nativen Faltung zu einem vorgegebenen Epitop E messen. Hier wird ebenfalls eine Brute-Force-Methode verwendet. Im Gegensatz zum Affinitätsalgorithmus bringt dieser Ansatz hier keinen gravierenden Zeitnachteil, da der Algorithmus nur einmal pro Sequenz ausgeführt wird. Die Ähnlichkeit wird nur an der durch den Faltungsalgorithmus bestimmten (vermutlich nativen) Faltung gemessen. Das Epitop wird in X- und Y-Richtung Ausschnitt für Ausschnitt über die Oberfläche des gefalteten Proteins geführt. Dann wird bestimmt, auf welchem Niveau in Z-Richtung das Protein das Epitop aufweisen soll. Hier gibt es zwei verschiedene Möglichkeiten: Entweder es wird zugelassen, dass das Protein an Stellen Residuen aufweist, an denen das Epitop Lücken enthält (tolerante Ähnlichkeit) oder es wird verboten (harte Ähnlichkeit). Beide Methoden können sinnvoll sein. Wird eine Lücke innerhalb des Epitops geschlossen, so kann es sein, dass der Rezeptor nicht mehr daran binden kann. Also wäre ein sehr schlechter Score anzunehmen. Werden Lücken zugelassen, so wird das Niveau gewählt, an dem sich am meisten Residuen überschneiden. Es muss aber mindestens noch ein Residuum des Epitops an der Proteinoberfläche liegen. Verbieten wir Lücken, so wird das maximale Niveau des Ausschnitts gewählt. Die Scorefunktion für die harte Ähnlichkeit für ein Epitop E und eine Sequenz w sieht

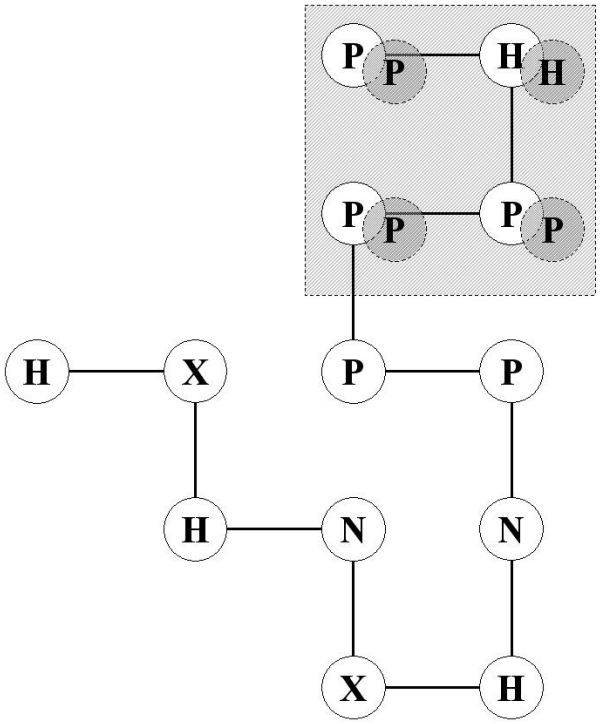


Abbildung 8.3: Tiefenähnlichkeit (2D-HPNX-Modell)

folgendermaßen aus:

$$E_{H-Ähn}(w,E) = \sum_{e_i \in E} \rho(e_i), \quad \rho(e_i) = \begin{cases} -1 & \text{Native Faltung von } w \text{ und Epitop enthalten} \\ & \text{an Stelle } e_i \text{ das gleiche Residuum.} \\ 0 & \text{sonst} \end{cases}$$

Die Scorefunktion der toleranten Ähnlichkeit enthält zusätzlich einen Strafterm für die bei der harten Ähnlichkeit nicht auftretenden „verschlossenen Lücken“ des Epitops:

$$E_{Tol-Ähn}(w,E) = \sum_{e_i \in E} \rho(e_i), \quad \rho(e_i) = \begin{cases} -1 & \text{Native Faltung von } w \text{ und Epitop enthalten} \\ & \text{an Stelle } e_i \text{ das gleiche Residuum.} \\ c & \text{Das Epitop enthält an Stelle } e_i \text{ eine Lücke,} \\ & \text{die native Faltung von } w \text{ nicht.} \\ 0 & \text{sonst} \end{cases}$$

$c > 0$ ist eine Konstante, die die Stärke der Bestrafung einer verschlossenen Lücke festlegt. Wir verwenden $c = 2$.

Zusätzlich zu der toleranten Ähnlichkeit und der harten Ähnlichkeit, die beide nur die Oberfläche des Proteins nach dem Epitop absuchen, haben wir ein drittes Verfahren entwickelt, das die gesamte Proteinstruktur in die Suche miteinbezieht. Diese Form der Ähnlichkeit ist interessant, wenn es sich bei dem Epitop nicht um einen Rezeptor o. ä. handelt, sondern z. B. um einen Teil der Struktur, der für eine bestimmte Eigenschaft des Proteins verantwortlich ist. Solche Strukturteile müssen nicht zwingend an der Molekülоberfläche liegen, sondern können auch im Kern des Proteins vorkommen. Diese Form der Ähnlichkeit bezeichnen wir als Tiefenähnlichkeit E_{Tief}. Bei der Untersuchung der Tiefenähnlichkeit wird das Protein in einem iterativen Prozess nach und nach auf seiner gesamten räumlichen Struktur mit dem Epitop überlagert (siehe Abbildung 8.3). Für diese Überlagerungen werden mit Hilfe der Funktion

$$E_{Tief}(w,E) = \sum_{e_i \in E} \rho(e_i), \quad \rho(e_i) = \begin{cases} -1 & \text{Native Faltung von } w \text{ und Epitop enthalten} \\ & \text{an Stelle } e_i \text{ das gleiche Residuum.} \\ 0 & \text{sonst} \end{cases}$$

Ähnlichkeitswerte der einzelnen Überlagerungen berechnet. Danach wir das Epitop gedreht und das Überlagern und Bewerten wird widerholt, bis für alle möglichen Orientierungen des Epitops die Überlagerungen berechnet wurden. Die Tiefenähnlickeit ist dann die Beste aller berechneten Überlagerungsähnlichkeiten.

8.7 Länge

Wie in Kapitel 6.1 erwähnt, sinkt mit der Größe des Peptids die Bioverfügbarkeit. Das Peptid kann sich nicht mehr so gut im Körper verteilen. Wünschenswert ist also ein kurzes Peptid. Hieraus folgt:

$$E_L(w) = |w|$$

Da die Länge durch die Vorgaben innerhalb der GP beschränkt ist, kann man hier direkt den Optimalwert angeben.

Kapitel 9

Sequenzoptimierung

> „Für jedes menschliche Problem gibt es immer eine einfache Lösung: klar, einleuchtend und falsch."
>
> Henry Louis Mencken (1880-1956), amerik. Journalist und Literaturkritiker

Bei dem von uns zur Sequenzfindung verwendeten Algorithmus handelt es sich um ein lineares GP-System. Wir wollen jedoch keinen Quellcode für Programme, sondern Peptidsequenzen evolvieren. Besonders wichtig ist für uns die Eigenschaft der GP, Individuen variabler Länge zuzulassen und durch Anwendung der einzelnen genetischen Operatoren erzeugen zu können. Wir erweiterten die GP um ein Archiv, so dass multikriterielle Probleme mittels eines Paretoansatzes (siehe Kapitel 5) optimiert werden können. Nachdem wir zuerst den grundlegenden Ablauf des von uns verwendeten GP-Systems algorithmisch darstellen, erläutern wir die Repräsentation der Individuen. Danach werden die Operatoren im einzelnen vorgestellt. Die multikriterielle Erweiterung wird eingehend erläutert. Abschließend werden mögliche Einflussnahmen des Benutzers auf den Optmierungsprozess dargestellt.

9.1 Algorithmus

Die von uns verwendete Sequenzoptimierungs-GP verläuft nach dem folgenden Pseudocodeschema:

Schematischer Ablauf des GP-Systems

1. Initialisiere Startpopulation mit zufälligen Individuen.
2. Solange Abbruchkriterium nicht erfüllt:
   (a) Bewerte alle Individuen.
   (b) Wähle beste Individuen aus dem Archiv und übernehme sie in die neue Population.
   (c) Solange neue Population nicht vollständig:
      i. Wähle zwei Individuen.
      ii. Rekombiniere sie.
      iii. Mutiere sie.
      iv. Übernehme sie in die neue Population.
   (d) Aktualisiere das Archiv.

Wir geben hier nur einen Überblick über den groben Ablauf der GP. Die Erläuterung der einzelnen Schritte folgt in den nächsten Abschnitten.

9.2 Repräsentation

Ein entscheidender Faktor für den Erfolg eines GP-Systems ist die gewählte Repräsentation der Individuen. Wir haben uns dafür entschieden, die einzelnen Individuen durch einen eindimensionalen String zu repräsentieren, der die Sequenz des entsprechenden Peptids darstellt. Für die Strings stehen zwei verschiedene Alphabete zur Verfügung. Zum einen das Alphabet $\sum = \{H, P, N, X\}$, das die Menge der Aminosäuren in die vier Klassen hydrophob, hydrophil positiv geladen, hydrophil neutral geladen und hydrophil negativ geladen einteilt (siehe Abschnitt 4.3.2), und zum anderen das Alphabet der 20 Aminosäuren.

9.3 Initialisierung

Die Initialisierung der Population geschieht zufällig. Die Länge eines Individuums wird innerhalb einer unteren und oberen Schranke uniform verteilt ausgewürfelt. Danach wird das Individuum mit einer zufälligen Basensequenz der entsprechenden Länge initialisiert (vgl. [12]).

9.4 Elitismus

Im Falle der Mehrzieloptimierung ist nicht klar, welche Individuen elitär sind und ohne Veränderung in die neue Generation übernommen werden können. Wählen wir das ganze Archiv paretooptimaler Punkte, so würde die Population bei weitem größer als sinnvoll. Es bedarf also einer Selektionsfunktion, die Individuen aus dem Archiv wählt, welche in die aktuelle Population übernommen werden (die verschiedenen Selektionsfunktionen werden in Abschnitt 9.10 vorgestellt). Die GP bietet einen Elitismusparameter p_e, der angibt, wie groß der Anteil der elitären Individuen an der Population sein darf. Die tatsächliche Anzahl elitärer Individuen errechnet sich folgendermaßen:

$$E = \begin{cases} \lfloor\, |Pop| * p_e \,\rfloor & , \; |Pop| * p_e > 1 \\ 1 & , \; sonst \end{cases}$$

Wir erzwingen einen minimalen Elitismus, da sonst keine angemessene Streuung im Archiv gewährleistet werden kann.

9.5 Selektion

Als Selektionsmechanismus haben wir uns für die in Abschnitt 5.5.2 beschriebene Turnierselektion mit Turniergröße zwei entschieden. Da wir mehrere Zielfunktionen betrachten, müssen wir aus den einzelnen Werten eine Fitness errechnen nach der die Turniere ausgetragen werden können. Wir benutzen hierfür das Konzept der Paretodominanz (siehe Abschnitt 5.3). Das bessere der beiden Individuen wird dabei für die Rekombination ausgewählt. Sollten beide Lösungen sich gegenseitig nicht dominieren, gewinnt die Lösung, in deren Nachbarschaft weniger Individuen liegen. Um diese Nachbarschaftsbetrachtung durchzuführen, wird der Lösungsraum in Hypercubes [13] eingeteilt, eine Gitterstruktur, die über ebensoviele Dimensionen verfügt, wie der Lösungsraum (siehe auch Abschnitt 9.10). Die Anzahl der benachbarten Lösungen ist die Anzahl der Lösungen, die im gleichen Hypercube liegen. Je zwei Turniersieger werden miteinander rekombiniert und die dadurch entstehenden Lösungen werden mutiert.

9.6 Crossover

An dieser Stelle verwenden wir das in Abschnitt 5.5.2 beschriebenes One-Point-Crossover. In Erweiterung dazu bieten wir eine Variante an, die EDIs verwendet. In unserem Fall stellen die EDIs Auswahlwahrscheinlichkeiten für die einzelnen Crossoverpunkte dar. Jedem Crossoverpunkt

$i = 1, \ldots, n$ wird eine Crossoverwahrscheinlichkeit p_i zugewiesen, die am Anfang $p_i = \frac{1}{n}$ beträgt. Es gilt immer $\sum_{i=1,\ldots,n} p_i = 1$. Nach einem erfolgreichen Crossover wird das entsprechende EDI um den Wert $a = \frac{1}{n}$ erhöht und alle EDIs des Individuums werden erneut normiert. Durch die EDIs wollen wir erreichen, dass die Crossoverpunkte nicht innerhalb guter Module [12] liegen, sondern Crossover häufig zwischen Modulen durchgeführt wird.

9.7 Deletion

Unter der Annahme, dass mit Hilfe der EDIs die einzelnen Individuen weitestgehend modularisiert wurden, können in den Individuen zwischen den einzelnen fitnessverbessernden Modulen Teile vorkommen, die Junk (funktionslose Sequenzteile) enthalten. Mit Hilfe der Deletion sollen diese Teile identifiziert und entfernt werden. Es wird dabei ein zufälliger Punkt m im Individuum als Mutationspunkt ausgewählt. Von diesem Punkt ausgehend wird in beide Richtungen im String nach wahrscheinlichen Modulgrenzen g gesucht, bis folgende Ungleichung erfüllt ist:

$$EDI(g) > p, \; p \text{ sei uniform verteilt im Intervall } \left[0, \frac{2}{|Sequenz|}\right]$$

Dann gehen wir davon aus, eine Modulgrenze gefunden zu haben. Aus der Ungleichung folgt, dass eine Crossover-Wahrscheinlichkeit $EDI[i] > \frac{2}{|EDI|}$ immer als Modulgrenze angesehen wird. Der Sequenzteil, der zwischen den so gefundenen Grenzen liegt, wird entfernt (siehe Abbildung 9.1. Die Sequenzlänge darf dabei jedoch nie die Mindestlänge unterschreiten, ansonsten wird der Vorgang nicht durchgeführt.

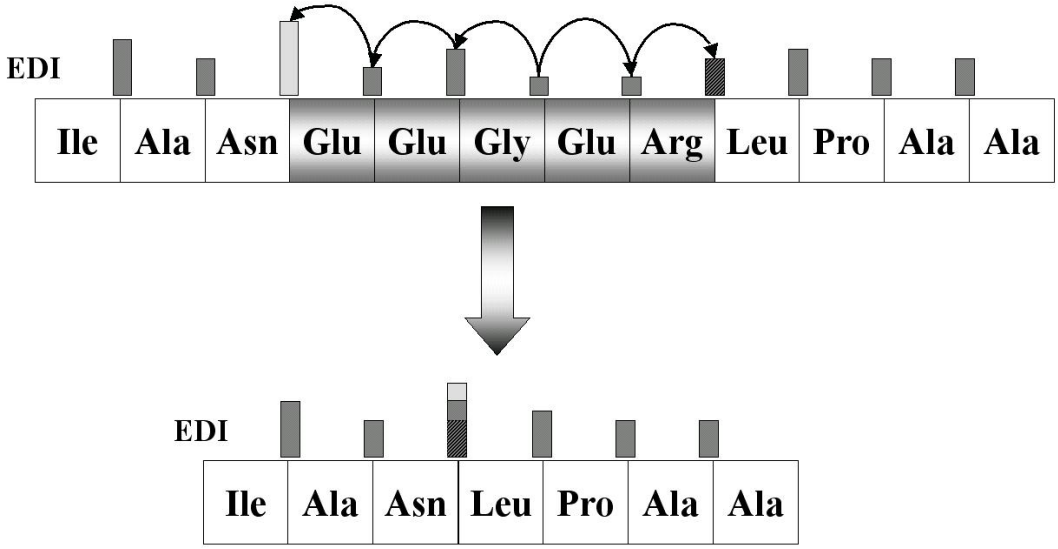


Abbildung 9.1: Deletion (Beispiel), der graue Sequenzteil wird ausgeschnitten, die EDIs an den Grenzen werden gemittelt.

9.8 Swapping

Mit Hilfe des Swapping wird die Anordnung der Module in einer Sequenz verändert, indem zwei Module ihre Plätze tauschen. Da diese Mutation nur bei voll modularisierten Sequenzen erfolgversprechend ist, werden zunächst die Modulgrenzen identifiziert. Dabei betrachten wir sowohl den Anfang der Sequenz als auch das Ende der Sequenz als Modulgrenze. Um die übrigen Modulgrenzen

zu finden, werden einmal die EDI-Wahrscheinlichkeiten durchlaufen. Alle zugehörigen Crossoverpunkte i für die $EDI[i] > \frac{1,5}{|EDI|}$ gilt, werden als Modulgrenze angesehen. Aus der Menge der so festgelegten Module werden zwei Module gewählt und vertauscht. Die EDI-Wahrscheinlichkeiten an den Grenzen der beiden vertauschten Module bleiben dabei gleich. Sollte keine Modulgrenze innerhalb der Sequenz gefunden werden, wir das Swapping nicht durchgeführt.

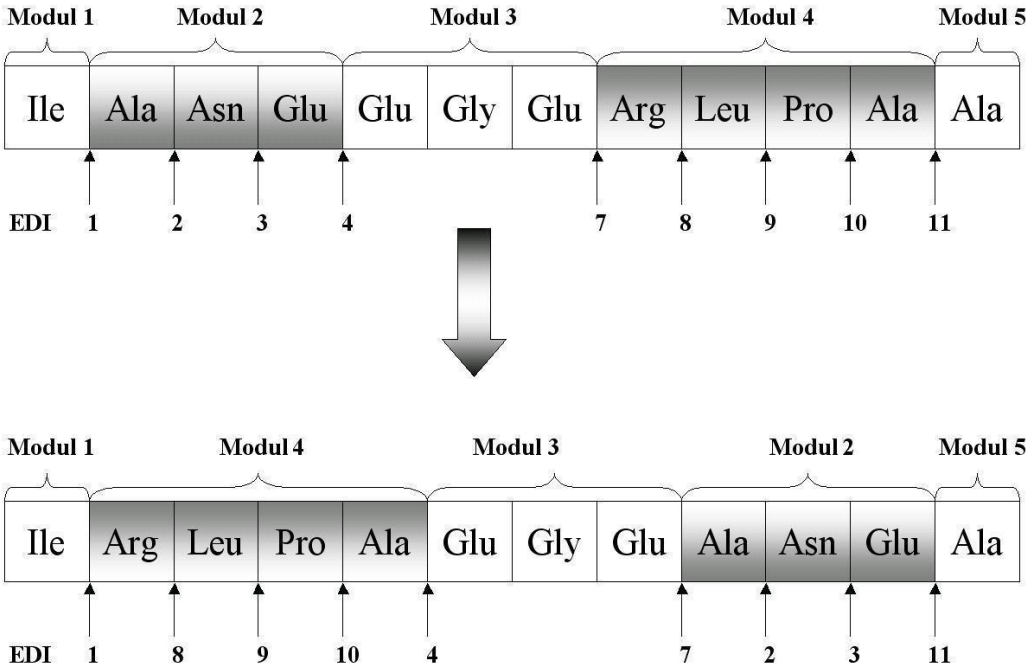


Abbildung 9.2: Swapping zwischen Modul 2 und 4

9.9 Insertion

Die in Abschnitt 5.5.2 beschriebene Punktmutation erweiternd haben wir die Insertion entwickelt. In Anlehnung an die Automatically Defined Functions (ADF siehe Abschnitt 5.6.3) benutzten wir einen Pool von Teilsequenzen, die den Function Definition Branches entsprechen. Die einzelnen Elemente des Pools werden mitevolviert, also mutiert und miteinander rekombiniert (siehe Abbildung 9.3). Die atomaren Elemente (z. B. bei der HPNX Energiefunktion die Elemente H, P, N, X) und der leere String bleiben dabei stets als mögliche Mutation erhalten. Die Fitness $F_{mut}(i)$ der einzelnen im Mutationspool enthaltenen Teilsequenzen ergibt sich aus ihrer Auswahlhäufigkeit $Cho(i)$ und ihrem Erfolg $Suc(i)$. Erfolg bedeutet, dass die Mutation zu einer Fitnessverbesserung geführt hat. Er wird mittels folgender Funktion berechnet:

$$F_{mut}(i) = \frac{Suc(i)}{Cho(i) + 0,01} + \frac{0,2}{Cho(i) + 1}$$

Hierbei sorgt der zweite Summand dafür, dass noch nicht gewählte Mutationen eine gewisse Chance haben, ausgewählt zu werden. Der erste Summand modifiziert die so gegebene Basiswahrscheinlichkeit mit der Erfolgsquote der entsprechenden Mutation. Die Konstanten 0.01 und 1 sind willkürlich. Sie dienen dazu, Divisionen durch Null zu verhindern. Ähnlich wie bei der Punktmutation wird ein Mutationspunkt im Individuum gewählt. Anstatt den dort gefundenen Buchstaben durch einen zufällig ausgewählten neuen Buchstaben zu ersetzen, wird ein zufällig aus dem Pool gezogenes Element an dieser Stelle eingefügt. Wenn die Poolgröße für den Teilsequenzenpool auf null gesetzt wird, erhält man eine normale Mutation einschließlich des Löschens einer Position.

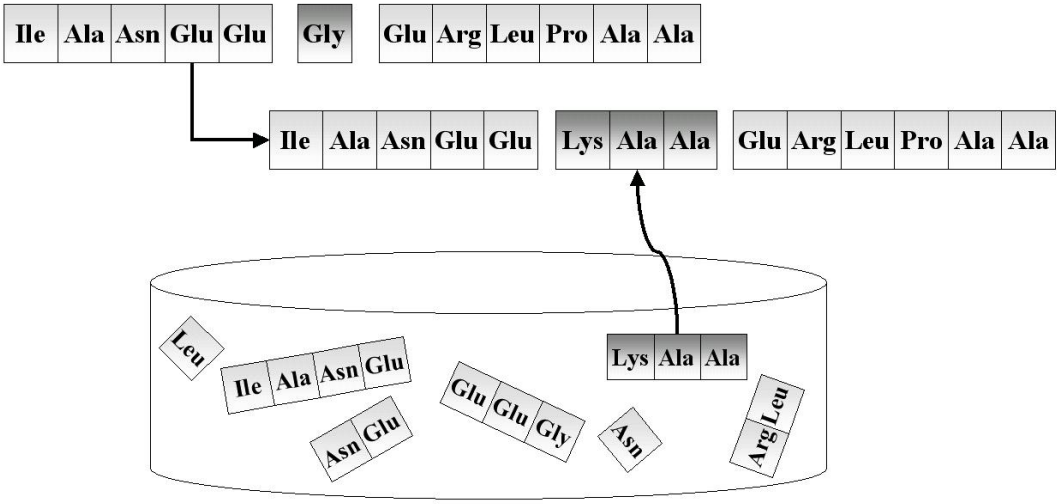


Abbildung 9.3: Insertion

9.10 Archivstrategien

Paretobasierte Algorithmen zeichnen sich meist durch die Verwendung eines Archivs zur Speicherung paretooptimaler Individuen aus. Diese in der Größe beschränkten Archive dienen als Approximationen an die tatsächliche Paretofront (die oft eine unendliche Zahl paretooptimaler Punkte enthalten kann). Wie gut dies gelingt, hängt nicht zuletzt von der Qualität des Archivs ab. Zum einen muss eine gute Streuung auf der vermeintlichen Paretofront erzielt werden. Die Streuung kann durch verschiedene Metriken berechnet werden, z. B. durch die Entropiefunktion (in [40]). Im Folgenden wird dieses Streuungsmaß als σ bezeichnet. Desweiteren wird das Archiv auch verwendet um z. B. durch den Elitismusoperator Einfluss auf die aktuelle Population zu nehmen (siehe Abbildung 9.4). Hieraus wird klar, welche Aufgaben ein Archiv zu erfüllen hat. Zum

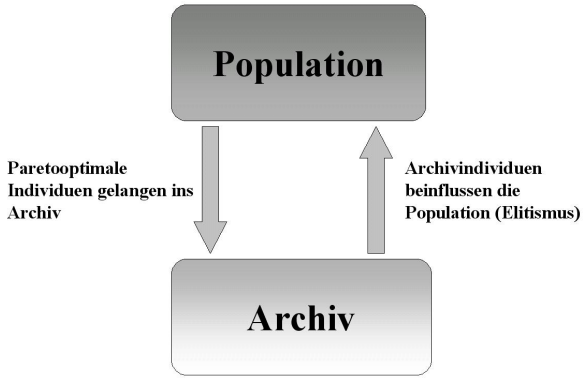


Abbildung 9.4: Interaktion: Archiv↔Population

einen muss zu jedem Zeitpunkt eine möglichst gleichmäßige Streuung des Archivinhalts gewährleistet sein. Zum anderen müssen Individuen, die im Archiv gespeichert sind, den Algorithmus in seiner Konvergenz unterstützen. Es kommt also nicht nur darauf an, die optimale Streuung zu erzielen, sondern auch, die „richtigen“ Punkte für den Optimierungsverlauf zu speichern. Dieses Maß für die Wahrscheinlichkeit des Archivs, Verbesserungen zu erzeugen, benennen wir ρ. Ebenso wie bei einer Heuristik ab und zu eine Verschlechterung in Kauf genommen wird, so sollte auch bei einem guten Archiv eine Lösung, die zwar paretooptimal ist, aber nicht in der idealen Verteilung liegt, Einfluss auf die Population nehmen können. Ein Archiv zeichnet sich durch zwei

Abbildung 9.5: a) Gute Streuung , b) Schlechte Streuung

Fitnessfunktionen aus: Die Selektionsfunktion $F_{sel}(Ind)$ und die Löschfunktion $F_{del}(Ind)$ (Ind sei ein Archivindividuum). $F_{sel}(Ind)$ gibt die Fitness an, nach der ein Individuum aus dem Archiv ausgewählt wird, um die Population zu beeinflussen (z. B. durch Elitismus). F_{del} ist die Löschfunktion. Sie dient zur Auswahl eines Individuums, welches aus dem Archiv gelöscht wird, wenn die maximale Archivgröße überschritten ist. In der Literatur wurden bisher viele Archivstrategien erörtert, die alle eine einigermaßen gleichmäßige Verteilung auf der Paretofront bei guter Konvergenz ermöglichen([41, 86, 97, 35, 62, 20], für eine Zusammenfassung auch [18]). Wir verwenden die in [13] vorgestellten Ansätze.

Löschfunktion: Adaptive Hypercubes

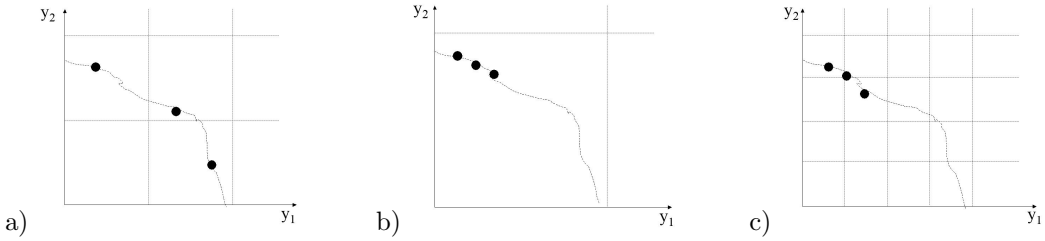


Abbildung 9.6: Hypercubes im Lösungsraum, a) Die Hypercubes haben die richtige Größe, b) die Hypercubes sind zu groß, c) die Hypercubes sind zu klein. Bei b) und c) wird keine gute Streuung erzielt.

Um eine gleichmäßige Streuung der Archivlösungen über den Suchraum zu erreichen, verwenden wir adaptive Hypercubes. Es wird ein Gitter über den Suchraum gelegt, wie in Abbildung 9.6 dargestellt. Die Löschfunktion für Individuum Ind_i berechnet sich aus der Anzahl d_i der Individuen im gleichen Hypercube:

$$F_{del}(Ind_i) = d_i^2$$

Es gibt nun verschiedene Arten der Hypercube-Erzeugung. Auf der Hand liegt hier zum einen die statische Methode: In jeder Dimension j des Lösungsraums wird eine Seitenlänge e_j fest definiert. Dieser Ansatz hat zwei gravierende Nachteile. Er erfordert Spezialwissen über die Form des Lösungsraums, das nicht immer gegeben ist. Zusätzlich verkleinert sich die Spannweite der gefundenen Paretomenge im Optimierungsverlauf, so dass statische Hypercubes die falsche Größe haben können. Wir verwenden dynamische Hypercubes, die ihre Größe der aktuellen Front anpassen. Hierzu berechnen wir e_j anhand der maximalen und minimalen Koordinaten der Individuen:

$$e_j = \frac{\max_{1,\ldots,|Ind|} y_{i,j} - \min_{1,\ldots,|Ind|} y_{i,j}}{c|Ind|}$$

$|Ind|$ sei die Populationsgröße, $y_{i,j}$ die j-te Koordinate des i-ten Individuums. c ist hierbei eine Form des Selektionsdrucks und bestimmt, wieviele Hypercubes über die Front gelegt werden, und wie stark daher die Bestrafung von Individuenclustern ist. Wir wählten für unsere Experimente $c = 0,9$.

Selektionsfunktionen

Die GP bietet dem Benutzer drei verschiedene Selektionsfunktionen zur Auswahl. Diese haben alle ihre Vorzüge und Nachteile. In Kapitel 10 werden wir untersuchen, welche Strategie die besten Ergebnisse verspricht.

Uniforme Selektion Diese Selektion wird unter der Annahme angewendet, dass alle Individuen des Archivs mit gleicher Wahrscheinlichkeit erfolgreiche Nachkommen erzeugen, die σ und ρ erhöhen.

$$F_{sel}(Ind_i) = 1$$

Inverse Selektion Hier gehen wir ebenfalls davon aus, dass alle Individuen ρ gleichwahrscheinlich verbessern. Wir hoffen jetzt allerdings, dass Nachkommen von Individuen in unterbesetzten Regionen des Lösungsraums zur Erhöhung von σ führen.

$$F_{sel}(Ind_i) = \frac{1}{F_{del}}$$

Alphamännchenselektion Bei der Alphamännchenselektion (AM-Selektion) haben wir uns von der Natur inspirieren lassen. Bei vielen Rudeltieren gibt es ein Alphamännchen, das bei der Fortpflanzung ein Vorrecht hat. Bei ihm handelt es sich in den meisten Fällen um ein körperlich überlegenes Tier, bei dem man annehmen kann, dass seine Nachkommen wieder starke Tiere sind und so eine hohe Überlebenswahrscheinlichkeit besitzen.

Diesen Gedanken übertragen wir auf unseren Selektionsmechanismus, indem wir Individuen, die schon vorher verhältnismäßig viele gute Nachkommen erzeugt haben, bevorzugt auswählen. Hierzu werden zwei Größen definiert: Cho_i gibt an, wie oft ein Individuum schon gewählt wurde. Suc_i zählt die Anzahl der erfolgreichen Nachkommen. Erfolgreich heißt hier nicht nur, ins Archiv zu gelangen. Ein erfolgreicher Nachkomme muss ein Mitglied des Archivs dominieren, also zu einer tatsächlichen Verbesserung der Front führen. Daher wählten wir als funktionale Repräsentation dieses Gedankens folgenden Ausdruck:

$$F_{sel}(Ind_i) = \frac{Suc_i}{1 + Cho_i} + \frac{\sum Suc_i}{1 + \sum Cho_i}$$

Es ist zu bemerken, dass ein Individuum mit $Cho_i = 0$ immer noch eine gewisse Grundfitness hat, so dass auch neue Individuen eine Chance haben, gewählt zu werden. Hierzu dient der zweite Summand der Formel, der für alle Individuen eine gewisse Grundfitness gewährt. Die Grundfitness ist die durchschnittliche Erfolgsrate des Archivs. Wir wählten an dieser Stelle bewusst keine Konstante, da im Evolutionsprozess die durchschnittliche Erfolgsrate stark variieren kann. Dadurch würden neue Individuen mit $Cho_i = 0$ entweder über- oder unterbewertet. Wählen wir die durchschnittliche Erfolgsrate, passt sich die Grundfitness dynamisch an.

9.11 Benutzerdefinierte Zielprioritäten

Herkömmliche multikriterielle Heuristiken sind entweder aggregierend oder paretobasiert (siehe Kapitel 5.1). Der Benutzer wird also entweder gezwungen, eine Gewichtung der Ziele anzugeben und erhält als Antwort nur einen optimalen Punkt. Oder er erhält eine Auswahl paretooptimaler Punkte, die allerdings normalerweise keine Zielgewichtung enthält, sondern idealerweise gleichverteilt ist (siehe Abschnitt 9.10). Was ist nun, wenn der Benutzer seine Prioritäten auf verschiedene Ziele legen möchte, auf andere jedoch nicht? In einem konkreten Fall könnte der Benutzer z. B. an der Optimierung von Stabilität und Länge interessiert sein. Allerdings ist ihm vor allem die Stabilität wichtig, die Länge ist nur von sekundärer Bedeutung. Hier könnte ihm keiner der herkömmlichen Ansätze weiterhelfen. Eine Synthese aus a priori und a posteriori-Verfahren ist daher notwendig (siehe Abschnitt 5.1). Die Front sollte in eine Richtung gedrängt werden.

Trotzdem dürfen andere Ziele nicht vernachlässigt werden, die Front sollte immer noch abgedeckt werden. Wir erreichen dies durch Verzerrung der adaptiven Hypercubestruktur, welche über den Lösungsraum gelegt wird. Zuerst werden die Kantenlängen der Cubes normiert (hierbei sei C_{ij} der Startpunkt des Cubes i in Dimension j, min_j und max_j die Ausdehnungen des Archives in Dimension j):

$$C'_{ij} = \frac{C_{ij} - min_j}{max_j - min_j}$$

Nun kann eine Verzerrung der Koordinaten stattfinden, die die Kantenlänge der einzelnen Hypercubes verändert. Es sind viele Abbildungen denkbar, wir wählten eine exponentielle Variante:

$$C''_{ij} = 1 - (1 - C'_{ij})^{p_j}$$

p_j ist hierbei der Prioritätsfaktor, der aus den Prioritäten des Benutzers errechnet wird. Nun müssen die neuen Startpunkte wieder auf das Intervall (min_j, max_j) abgebildet werden:

$$C'''_{ij} = C''_{ij}(max_j - min_j) + min_j$$

Dem Benutzer, der gegebenenfalls wenig Know-How in Optimierungsfragen besitzt, muss nun eine intuitiv verständliche Möglichkeit geboten werden, den Wert eines Zieles anzugeben. Im normalen Sprachgebrauch sind z. B. Ausdrücke wie doppelt so wichtig, halb so wichtig etc. üblich. Diese müssen wir in sinnvolle Prioritätsfaktoren umsetzen. Der Benutzer gibt für jedes Ziel eine Priorität $P_j \in \mathbb{R}^{>0}$ an. Die Prioritätsfaktoren ergeben sich folgendermaßen aus den Prioritäten:

$$p_j = \frac{P_j^2}{\prod_{j=1}^{d} P_j^{2/d}}$$

Diese Form der Skalierung hat die Eigenschaft, dass die absolute Größe der Prioritäten nicht maßgeblich für die Prioritätsfaktoren ist, sondern ausschließlich das Verhältnis dieser zueinander. Es gelten folgende Aussagen:

$$\forall j_1, j_2 \in 1, \ldots, d : P_{j_1}/P_{j_2} = \sqrt{p_{j_1}/p_{j_2}}$$

$$\prod_{j=1,\ldots,d} p_j = 1$$

Die Hypercubes werden quadratisch verzerrt.

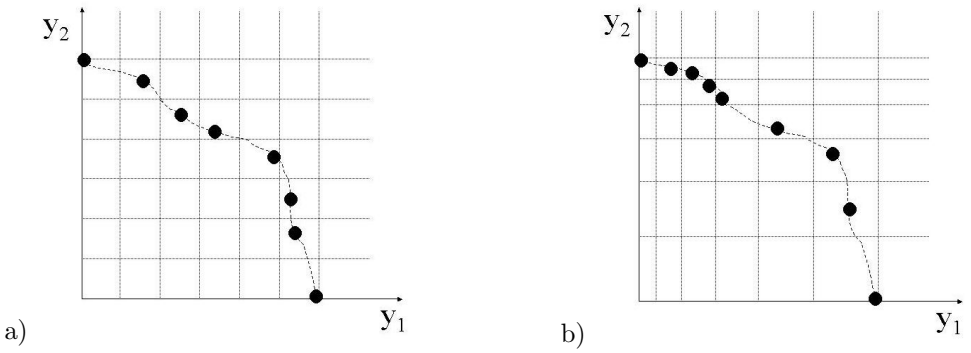


Abbildung 9.7: a) Gleiche Prioritäten, b) hohe Gewichtung von y_2

.

9.12 Sequenzsubstitution

Im Lauf unserer Arbeit haben wir festgestellt, dass es eine interessante Anwendung unseres Programms ist, nicht nur zufällige Sequenzen zu evolvieren. In vielen Fällen ist man daran interessiert, bestimmte funktionale Einheiten in die Sequenzen einzubinden.

Dieser Tatsache Rechnung tragend, haben wir die Sequenzsubstitution entwickelt. Mit ihr bieten wir dem Anwender die Möglichkeit, kurze Sequenzteile festzulegen. Diese Teile werden im evolutionären Prozess nicht mehr verändert oder entfernt, lediglich ihre Position in der Gesamtsequenz kann sich verändern. Zusätzlich stellen wir sicher, dass das angegebene Sequenzteil in jedem neuen Individuum enthalten ist.

Dieses Feature haben wir auf Grund der von uns gewählten Realisierung Sequenzsubstitution genannt. In der Sequenzoptimierungsstufe wird diese Teilsequenz nämlich nicht als Teilsequenz betrachtet, sondern durch eine Substitution ersetzt. Hierfür wird das Alphabet um zusätzliche Zeichen erweitert. Diese zusätzlichen Buchstaben dürfen im evolutionären Prozess nicht verändert oder entfernt werden. Also weder Mutation noch Insertion haben Einfluss auf sie und beim Crossover stellen wir sicher, dass in allen Nachkommen alle Substitutionen vorkommen. Erst wenn die Sequenzen an die faltungsoptimierende Stufe weitergegeben werden, wird diese Substitution rückgängig gemacht.

Kapitel 10

Versuche

„Nur ein Narr macht keine Experimente.“

Charles Darwin (1809-82), brit. Naturforscher

Aufgrund der hochgradigen Komplexität der Proteinfaltung und vor allem der Sequenzfindung entziehen sich diese einer vollständigen theoretischen Untersuchung. Daher ist ein statistische Analyse notwendig. Im Wissen, dass es niemals möglich sein wird, auf alle denkbaren Fragen eine Antwort zu geben, entschieden wir uns für die folgende Auswahl interessanter und aufschlussreicher Untersuchungen. Durch den mehrstufigen Aufbau des Tools wird sich auch die Versuchsplanung in mehrere Phasen unterteilen:

1. Parameteranalyse des genetischen Algorithmus und des Simulated Annealing
2. Auswahl einer der parameterisierten Heuristiken für die Sequenzanalyse
3. Konvergenzanalyse und Parameterfinetuning der gewählten Heuristik
4. Analyse der Zielfunktionen
5. Parameteranalyse „unkritischer“ Standardparameter der GP
6. Parameteranalyse neuer und sensibler Parameter der GP

Sämtliche Versuche wurden auf dem PC-Cluster des Lehrstuhls für Systemanalyse der Universität Dortmund ausgeführt. Für die Parallelisierung nutzten wir die Batchsystemsoftware LSF. Die folgende Hardware stand uns zur Verfügung:

- 5 x AMD Athlon 900, 256-384 MB RAM
- 1 x AMD Athlon 1200MP Doppelprozessor, 1 GB RAM

10.1 Statistische Methoden

Für sämtliche statistischen Auswertungen nutzten wir KEA (siehe Kapitel 6) und die Mathematiksoftware MATLAB. Im Folgenden werden die verwendeten Verfahren kurz angerissen.

10.1.1 Boxplots

Boxplots sind Diagrammtypen, die eine Darstellung wichtiger statistischer Kenngrößen über einer Wertemenge in einer Grafik ermöglichen (wie in Abbildung 10.1 dargestellt). Diese Kenngrößen sind im Einzelnen:

- Durchschnitt: Die Referenzlinie innerhalb der Box stellt den Durchschnitt der Wertemenge dar.

- Percentile: Die Unter- und Obergrenze der Box stellen das 25%- bzw. 75%-Percentil dar. Der Abstand zeigt die interquartile Distanz.

- Minimum und Maximum der Stichprobe (abgesehen von Ausreißern) werden durch Referenzlinien unter- und oberhalb der Boxen dargestellt.

- Ausreißer: Als Ausreißer werden standardmäßig Werte angesehen, die weiter als das 1,5-fache der interquartilen Distanz von den Boxgrenzen entfernt liegen. Diese werden durch Kreuze dargestellt.

Wir verwenden Boxplots, da sie hervorragend für den Vergleich von Stichproben geeignet sind. Sie sind aussagekräftig und anschaulich zugleich und der Leser kann sich direkt einen Überblick über eine Gruppe von Ergebnissen verschaffen.

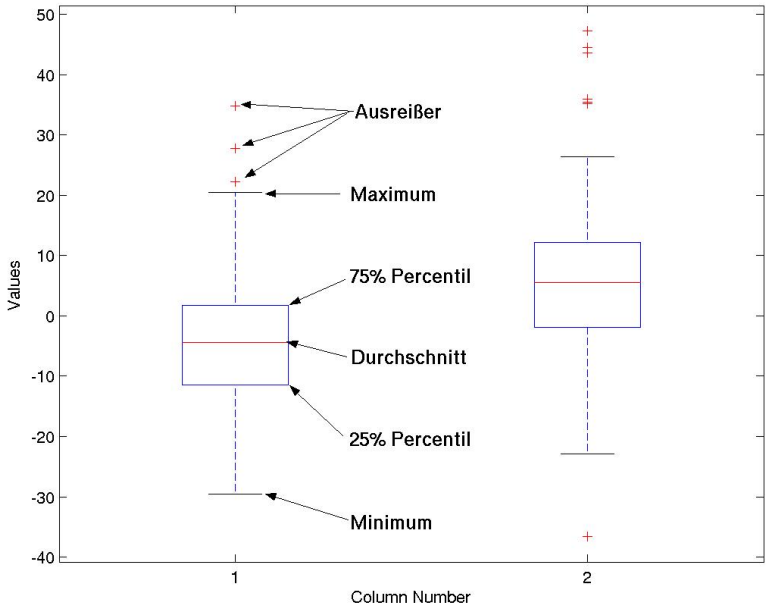


Abbildung 10.1: Erklärung: Boxplot

10.1.2 Metriken zum Vergleich multikriterieller Lösungsmengen

Im Gegensatz zu einkriteriellen Lösungsmengen fällt es sehr schwer, Lösungsmengen mit mehreren Zielen zu vergleichen. Naheliegend wäre es natürlich, einfach die Zahl nichtdominierter Punkte zu zählen, aber eine Aussage über die Güte der Lösungsmenge ergibt sich hieraus meist nicht. In der Literatur finden sich eine ganze Handvoll Verfahren, die sich mit diesem Problem beschäftigen. Eine gute Zusammenfassung findet sich in [69, 68] und in [34]. Wir verwenden die R2-Metrik von Hansen und Jaszkiewicz [51], welche im Folgenden kurz vorgestellt wird. Die R-Metriken vergleichen jeweils zwei Lösungsmengen A und B bezüglich einer Menge U so genannter Nützlichkeitsfunktionen $u(\vec{z})$, die jedem Punkt $\vec{z}$ im Lösungsraum ein Nützlichkeitsmaß zuordnen ($u \in U : \mathbb{R}^k \rightarrow \mathbb{R}$). Desweiteren wird eine Dichtefunktion $p(u)$ definiert, die den Nützlichkeitsfunktionen Wahrscheinlichkeiten zuordnet. $\hat{u}$ sei $\max_{\vec{z} \in A} u(\vec{z})$. Ein Problem der R-Metriken ist, dass sie nicht transitiv sind, d.h.: $A > B$ und $B > C \not\Rightarrow A > C$.

R2-Metrik

Die von uns verwendete R2-Metrik berechnet die erwartete Differenz der Lösungsmengen A und B bezüglich der Nützlichkeitsfunktionen:

$$R2(A, B) = E(\hat{u}(A)) - E(\hat{u}(B)).$$

Wenn $R2(A, B) > 0$ ist, gilt bezüglich der R2-Metrik $A > B$, analog gilt $B > A$, falls $R2(A, B) < 0$. Es gilt ebenfalls $R2(A, B) = -R2(B, A)$.

10.1.3 Prüfen von Hypothesen

In den folgenden Auswertungen erzeugen wir zwei verschiedene Arten von Ergebnisdaten. Zum einen vergleichen wir Stichproben verschiedener Algorithmen. Unterscheiden diese sich im Mittelwert, untersuchen wir, ob dieser Unterschied zufällig aufgetreten sein kann, oder ob ein Algorithmus tatsächlich dem anderen Algorithmus überlegen ist. Wir wollen die so genannte Nullhypothese H_0 zurückweisen, d. h. die Annahme, dass die Durchschnittsdifferenz der beiden Stichproben null ist. Gelingt uns dies, ist also die Wahrscheinlichkeit der Nullhypothese sehr klein (üblicherweise $p(H_0) < 0,05$), können wir entsprechend schließen, dass tatsächlich ein Qualitätsunterschied vorliegt. Zum anderen werten wir die Ergebnisse der R2-Metriken aus. Wir haben nicht mehr zwei Stichproben, die wir vergleichen, sondern nur eine Stichprobe, für die überprüft werden soll, ob sich der Durchschnitt signifikant von null unterscheidet. Wir müssen dafür eine andere Nullhypothese (Der Durchschnitt der Probe ist null.) zurückweisen.

Folgende Tests wurden verwendet:

t-Test Der t-Test ist der meistbenutzte Signifikanztest. Er liefert neben der Wahrscheinlichkeit der Nullhypothese $p_t(H_0)$ auch das so genannte Konfidenzintervall cf_t. Wir können nur signifikante Aussagen treffen, falls der Durchschnitt bzw. die Durchschnittsdifferenz außerhalb dieses Intervalls liegt. Der t-Test setzt jedoch voraus, dass die Stichproben normalverteilt sind. Da dies nicht immer der Fall ist, verwenden wir auch weitere Tests.

Kruskal-Wallis-Test Der Kruskal-Wallis-Test setzt im Gegensatz zum t-Test keine Normalverteilung voraus. Er nutzt ausschließlich die Ränge der Werte innerhalb der Stichprobe, um die Wahrscheinlichkeit der Nullhypothese $p_k(H_0)$ zu berechnen. Aufgrund dieser Tatsache ist es allerdings nicht möglich, ein Konfidenzintervall anzugeben. Wir nutzen diesen Test, um zwei Stichproben miteinander zu vergleichen.

Wilcoxon-Test Dieser Test ist wie der Kruskal-Wallis-Test parameterfrei und betrachtet ebenfalls die Rangverteilung innerhalb der Probe verglichen mit dem Durchschnittswert der zu prüfenden Hypothese. Wir nutzen ihn zur Analyse einer Stichprobe. Die Wahrscheinlichkeit der Nullhypothese bezeichnen wir als $p_w(H_0)$.

10.2 Parameteranalyse

Wollen wir das Verhalten der Faltungsheuristiken analysieren, lassen sich viele interessante Fragen stellen:

- Welche Auswirkungen hat die Wahl der Eingabecodierung?
- Ist die Wahl der Cooling-Strategie beim Simulated Annealing von Bedeutung?
- Kann eine große Population zu besseren Ergebnissen führen?
- Ist der genetische Algorithmus der Monte Carlo-Strategie überlegen, oder ist letztere die bessere Wahl?
- Können mit Hilfe des Nichings bessere Ergebnisse erzielt werden?

Um diese und andere Fragen zu beantworten, haben wir verschiedene Testreihen im Batchsystem LSF parallel durchgeführt. Um die Analyse nicht auf eine Sequenz zu beschränken, erzeugten wir eine Stichprobe von 100 Sequenzen der Länge 15. Diese wurden zufällig generiert. Wir berechneten mit dem Enumerator die minimale Energie jeder Stichprobe und generierten so einen Benchmark, an dem sich die Algorithmen messen konnten. Als Maß e wählten wir den durchschnittlichen Energiewert über alle 100 Sequenzen. Das Minimum liegt im 2D-HPNX-Modell bei einem Energiewert von -10,1. Die zu testenden Heuristiken sollten diesem Minimum möglichst nah kommen. Es ist uns leider nicht möglich, den Optimalwert für das 3D-Jernigan-Modell zu errechnen, da die Rechendauer zu groß wäre. Es stellt sich nun die Frage nach dem Vorgehen, um so verschiedene Heuristiken fair zu vergleichen. Die Laufzeit in Sekunden ist auszuschließen, da die Effizienz sehr stark von der Implementierung, der zugrundeliegenden Rechenkraft und der Auslastung der verwendeten Maschine abhängt. Deshalb entschieden wir uns, die Anzahl der bewerteten Faltungen, also der Zielfunktionsauswertungen bei allen Heuristiken gleichzusetzen. Wir wählten 8000 Zielfunktionsauswertungen als Grundlage für die Parameteranalyse. Sofern nicht anders angegeben, führten wir mit jedem Experiment, also mit jeder untersuchten Parametereinstellung, 50 unabhängige Läufe durch. Wir erreichten so 5000 Sequenzauswertungen, eine ausreichend hohe Zahl, um verlässliche Aussagen zu treffen. Die Standardparameter der Simulated Annealing-Strategie und des genetischen Algorithmus untersuchten wir in einem faktoriellen Versuchsdesign. Mit den hier ermittelten Basiswerten führten wir weitere Experimente durch, um die besten Einstellungen weiterer kritischer Parameter zu untersuchen. Die Ergebnisse stellten wir zum einen graphisch mittels Boxplots (Abschnitt 10.1.1) dar. Zum anderen wurden für jede getestete Parametereinstellung Mittelwert $\bar{e}$ und Varianz σ angegeben. Um unseren Ergebnissen statistisch verifizierte Aussagen folgen lassen zu können, führten wir bezüglich je zweier Stichproben Signifikanztests (siehe Abschnitt 10.1.3) durch. Da nicht alle Ergebnisse normalverteilt sind, setzten wir neben dem t-Test auch den parameterlosen Kruskal-Wallis-Test ein. In den Tabellen (siehe z. B. Tabellen in Abbildung 10.4 und 10.2) sind folgende Werte aufgeführt:

1. Die Parametereinstellungen der Stichproben A und B,
2. Die Differenz der Durchschnitte der Stichproben ($\bar{e}_A - \bar{e}_B$),
3. Die Wahrscheinlichkeit der Nullhypothese ($\bar{e}_A - \bar{e}_B = 0$, A ist im Mittel genauso gut wie B), des Kruskal-Wallis-Tests ($p_k(H_0)$) und des t-Tests ($p_t(H_0)$),
4. Das Konfidenzintervall des t-Tests (cf_t),
5. Die Anzahl der Freiheitsgrade df= $(|A| * |B|) - 2$ der Signifikanztests,
6. Die Folgerung, die geschlossen werden kann, falls die Nullhypothese verworfen wurde (bei $p_k(H_0) < 0,05$ und $p_t(H_0) < 0,05$),

10.2.1 Simulated Annealing – Faktorielles Design

Wir wählten ein faktorielles Design, um die in Tabelle 10.1 dargestellten Parameter mit verschiedenen Einstellungen zu untersuchen. Um statistische Signifikanz gewährleisten zu können,

Tabelle 10.1: Parameterisierung der Simulated Annealing-Heuristik

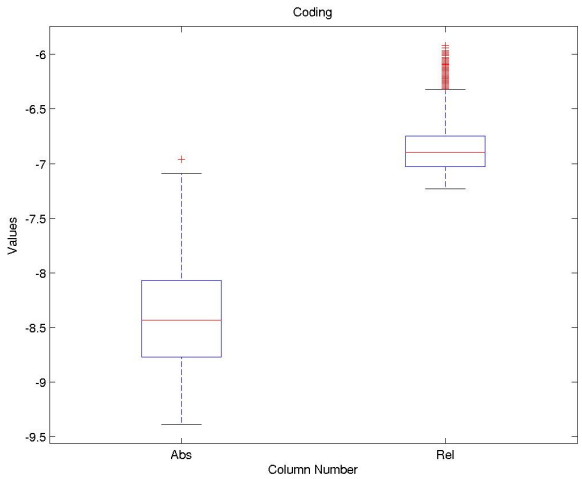
Parameter	Werte
Cooling-Strategie	lin., exp., log., sig., const.
Codierung	absolut, relativ
Wahrscheinlichkeit	0,1, 0,2, 0,4, 0,8
Mutationsstärke (in n/Sequenzlänge)	1, 2, 4

führten wir mit jeder der $5 * 2 * 4 * 3 = 120$ verschiedenen Einstellungen 50 unabhängige Läufe auf unserem Benchmark durch. Dies ergibt insgesamt 5000 Sequenzauswertungen jeder einzelnen Parametereinstellung. Die Ergebnisse (siehe Abbildungen 10.2 bis 10.10) sind äußerst aufschlussreich.

Betrachtet man die Eingabecodierung, so erkennt man äußerst große Qualitätsunterschiede (siehe Abbildungen 10.2, 10.3). Die absolute Codierung ist der relativen in beiden Modellen sehr deutlich überlegen. Der durchschnittliche Unterschied der Stichproben beträgt im 2D-HPNX-Modell 1,5354 Energiepunkte, im 3D-Jernigan-Modell 9,0305 Energiepunkte. Die Wahl der Codierung wirkt sich am stärksten auf die Ergebnisse aus. Die Mutationsstärke führt in einem ersten Versuch im zweidimensionalen Fall bei einem Wert von 4/Sequenzlänge zum besten Ergebnis (siehe Abbildung 10.8). Da hier eventuell ein Trend vorliegen könnte (noch höhere Mutationsstärken führen zu noch besseren Ergebnissen), untersuchten wir die Mutationsstärke in einem Folgeexperiment genauer (siehe Abbildung 10.10). Es stellte sich jedoch heraus, das auch hier eine Mutationsstärke von 4/Sequenzlänge die besten Ergebnisse erzielte. Im dreidimensionalen Fall ist 2/Sequenzlänge durchschnittlich geringfügig besser als 4/Sequenzlänge (siehe Abbildung 10.9). Allerdings lässt sich anhand der Ergebnisse keine signifikante Aussage darüber treffen, ob sich die Nullhypothese zurückweisen lässt, also ob dieser Unterschied zufällig ist. Eine Mutationsstärke von 1/Sequenzlänge führt zu signifikant schlechteren Ergebnissen. Die Cooling-Algorithmen sind ohne besonderen Einfluss (siehe Abbildungen 10.4, 10.5). konstante (Monte Carlo), sigmoide und lineare Kühlalgorithmen scheinen in beiden Modellen die besten Ergebnis zu liefern, während exponentielle und logarithmische Schemata nicht so gut abschneiden. Hinsichtlich des besten Kühlalgorithmus lässt sich keine signifikante Aussage treffen. Die durchschnittlich besten Ergebnisse erbrachten lineares (2D-HPNX) und konstantes (3D-Jernigan) Abkühlen. Als initiale Akzeptanzwahrscheinlichkeit sollte 0,4 (2D-HPNX) bzw. 0,8 (3D-Jernigan) gewählt werden (siehe Abbildungen 10.6, 10.7). Sie scheint sich ebenfalls nur moderat auf die Ergebnisse auszuwirken. Wir empfehlen daher die in Tabelle 10.2 zusammengefassten Parametereinstellungen für den SA.

Tabelle 10.2: Optimale Parameter der Simulated Annealing-Heuristik: Ergebnisse des faktoriellen Designs

Parameter	2D-HPNX	3D-Jernigan
Cooling-Strategie	Lin.	Const.
Codierung	absolut	absolut
Initiale Akzeptanzwahrscheinlichkeit	0,4	0,8
Mutationsstärke (in n/Sequenzlänge)	4	2

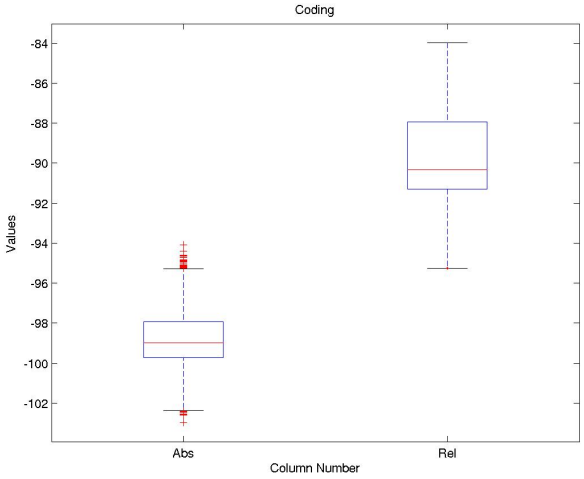


Absolute Codierung ist relativer Codierung deutlichst überlegen.

Wert	$\bar{e}$	σ
Abs	-8,3838	0,48435
Rel	-6,8484	0,24719

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
Abs	Rel	-1,5354	0	0	[-1,5549, -1,516]	5998	Abs schlägt Rel

Abbildung 10.2: Boxplot: SA - Relative und absolute Codierung der Faltung

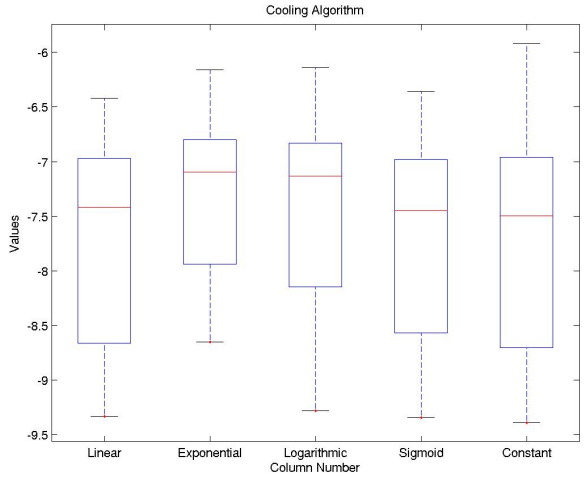


Wie auch im zweidimensionalen Modell ist der SA mit absoluter Eingabecodierung dem SA mit relativer Eingabecodierung deutlichst überlegen. Ein größerer Unterschied trat bei keinem anderen Parameter auf.

Wert	Avg.	σ
Abs	-98,8364	1,4621
Rel	-89,8059	2,2178

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
Abs	Rel	-9,0305	0	0	[-9,1255, -8,9354]	5998	Abs schlägt Rel

Abbildung 10.3: Boxplot: SA - Relative und absolute Codierung der Faltung, 3D-Jernigan

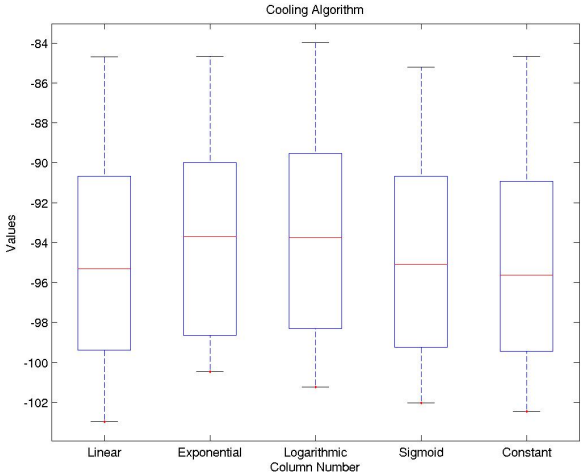


Zwischen linearem und sigmoidem Cooling sowie der Monte Carlo-Strategie (const.) ergaben sich keine signifikanten Unterschiede.
Exponentielles und logarithmisches Cooling scheinen nicht geeignet zu sein.

Wert	$\bar{e}$	σ
Lin.	-7,7839	0,87016
Exp.	-7,3313	0,65382
Log.	-7,4827	0,8009
Sig.	-7,7577	0,8433
Const.	-7,7248	0,99632

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
Lin.	Exp.	-0,45258	0	0	[-0,5142, -0,39097]	2398	Lin. schlägt Exp.
Lin.	Log.	-0,30124	0	0	[-0,36819, -0,2343]	2398	Lin. schlägt Log.
Lin.	Sig.	-0,026192	0,23921	0,45408	[-0,094786, 0,042403]	2398	Nicht mögl.
Lin.	Const.	-0,059058	0,71256	0,1221	[-0,13394, 0,015824]	2398	Nicht mögl.
Exp.	Log.	0,15134	2,5539e-05	4,2644e-07	[0,092816, 0,20987]	2398	Log. schlägt Exp.
Exp.	Sig.	0,42639	0	0	[0,36599, 0,4868]	2398	Sig. schlägt Exp.
Exp.	Const.	0,39353	0	0	[0,32607, 0,46098]	2398	Const. schlägt Exp.
Log.	Sig.	0,27505	0	4,4409e-16	[0,20921, 0,34089]	2398	Sig. schlägt Log.
Log.	Const.	0,24218	1,6653e-15	6,4455e-11	[0,16982, 0,31455]	2398	Const. schlägt Log.
Sig.	Const.	-0,032867	0,57931	0,38317	[-0,10676, 0,041024]	2398	Nicht mögl.

Abbildung 10.4: Boxplot: SA - Cooling-Algorithmen im Vergleich

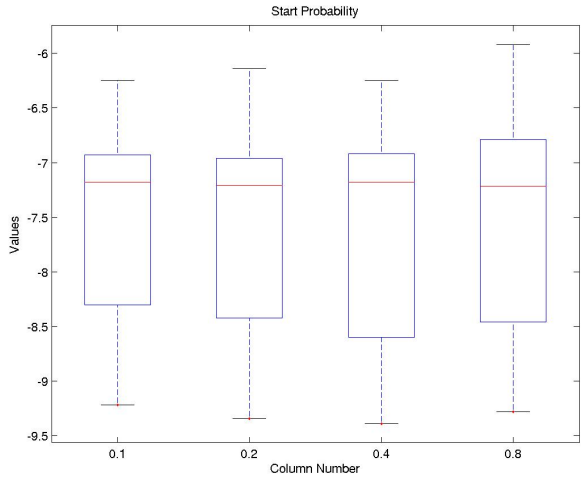


Es ist nicht möglich, einen signifikant besten Cooling-Algorithmus auszumachen. Lineares, exponentielles und sigmoides Cooling zeigen nur geringe Unterschiede. Deutlich schlechter schneiden logarithmisches und exponentielles Cooling ab.

Wert	Avg.	σ
Lin.	-94,7356	4,9356
Exp.	-93,7126	4,8309
Log.	-93,529	4,8927
Sig.	-94,6246	4,8096
Const.	-95,0039	4,8139

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
Lin.	Exp.	-1,023	9,5812e-14	3,1115e-07	[-1,4139, -0,63203]	2398	Lin. schlägt Exp.
Lin.	Log.	-1,2066	6,6613e-16	2,0853e-09	[-1,6, -0,81316]	2398	Lin. schlägt Log.
Lin.	Sig.	-0,11103	0,42599	0,57681	[-0,50115, 0,27908]	2398	Nicht mögl.
Lin.	Const.	0,2683	0,08407	0,17776	[-0,12198, 0,65858]	2398	Nicht mögl.
Exp.	Log.	-0,18358	0,071178	0,35512	[-0,5728, 0,20565]	2398	Nicht mögl.
Exp.	Sig.	0,91196	6,5081e-13	3,7748e-06	[0,52607, 1,2978]	2398	Sig. schlägt Exp.
Exp.	Const.	1,2913	0	6,6134e-11	[0,90523, 1,6774]	2398	Const. schlägt Exp.
Log.	Sig.	1,0955	2,6645e-15	3,5203e-08	[0,70716, 1,4839]	2398	Sig. schlägt Log.
Log.	Const.	1,4749	0	1,3589e-13	[1,0863, 1,8634]	2398	Const. schlägt Log.
Sig.	Const.	0,37934	0,0081471	0,053591	[-0,0058679, 0,76454]	2398	Nicht mögl.

Abbildung 10.5: Boxplot: SA - Cooling-Algorithmen im Vergleich, 3D-Jernigan

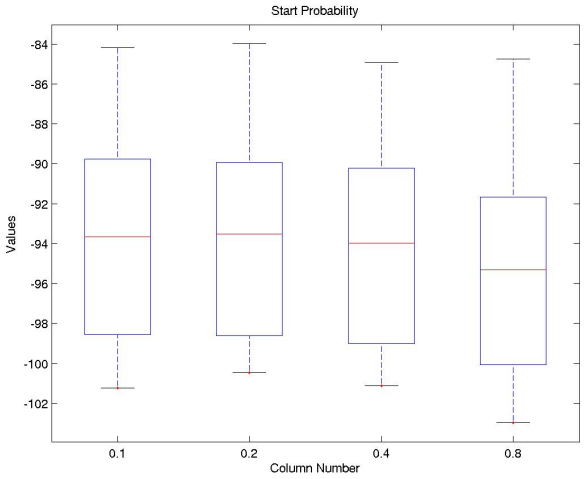


Die initiale Akzeptanzwahrscheinlichkeit scheint keinen großen Einfluss auf die Performanz des Algorithmus zu haben. Zwischen $p = 0,2$ und $p = 0,4$ ergaben sich keine signifikanten Unterschied. $p = 0,1$ und $p = 0,8$ schneiden schlechter ab.

Wert	$\bar{e}$	σ
0,1	-7,5717	0,7571
0,2	-7,6334	0,8082
0,4	-7,6911	0,9035
0,8	-7,5681	0,94753

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
0,1	0,2	0,061673	0,0077466	0,031093	[0,0056084, 0,11774]	2998	0,2 schlägt 0,1
0,1	0,4	0,11932	0,00044557	9,0385e-05	[0,059643, 0,179]	2998	0,4 schlägt 0,1
0,1	0,8	-0,0035867	0,043874	0,90882	[-0,064989, 0,057816]	2998	Nicht mögl.
0,2	0,4	0,057647	0,21884	0,065608	[-0,0037243, 0,11902]	2998	Nicht mögl.
0,2	0,8	-0,06526	1,0699e-05	0,042497	[-0,12831, -0,0022101]	2998	0,2 schlägt 0,8
0,4	0,8	-0,12291	5,0523e-10	0,00028178	[-0,18919, -0,056624]	2998	0,4 schlägt 0,8

Abbildung 10.6: Boxplot: SA - Initiale Akzeptanzwahrscheinlichkeit, 2D-HPNX

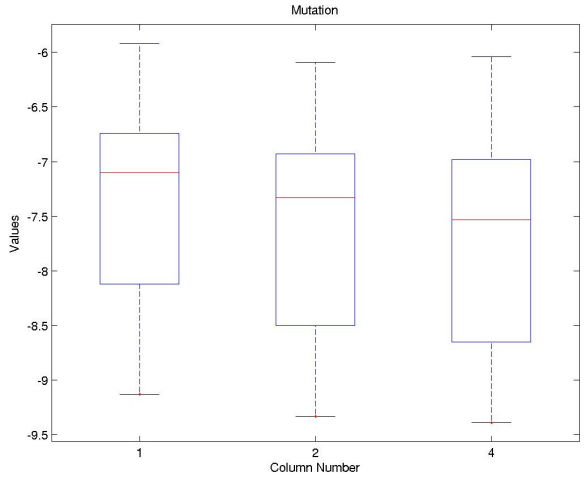


Eine hohe Startwahrscheinlichkeit von $p = 0,8$ liefert im 3D-Jernigan-Modell anders als im 2D-HPNX-Modell die signifikant besten Resultate aller Versuche.

Wert	Avg.	σ
0,1	-93,6562	4,8919
0,2	-93,7849	4,8816
0,4	-94,179	4,9092
0,8	-95,6645	4,6156

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
0,1	0,2	0,12866	0,23974	0,47096	[-0,22122, 0,47853]	2998	Nicht mögl.
0,1	0,4	0,52277	1,0922e-06	0,0035105	[0,1719, 0,87363]	2998	0,4 schlägt 0,1
0,1	0,8	2,0083	0	0	[1,6678, 2,3488]	2998	0,8 schlägt 0,1
0,2	0,4	0,39411	0,0001636	0,027547	[0,043616, 0,74461]	2998	0,4 schlägt 0,2
0,2	0,8	1,8796	0	0	[1,5395, 2,2197]	2998	0,8 schlägt 0,2
0,4	0,8	1,4855	0	0	[1,1444, 1,8266]	2998	0,8 schlägt 0,4

Abbildung 10.7: Boxplot: SA - Initiale Akzeptanzwahrscheinlichkeit, 3D-Jernigan

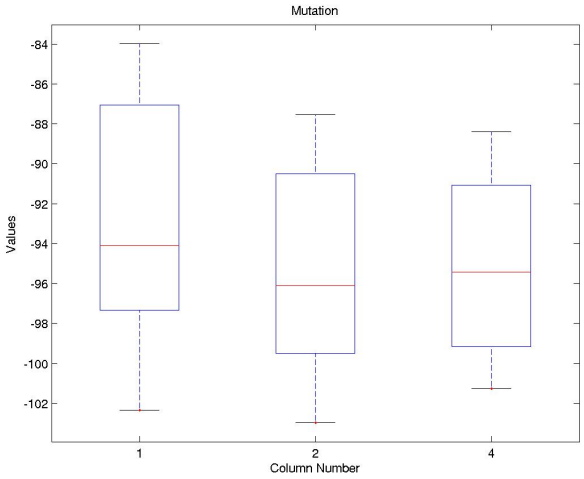


Die Mutationsrate hat großen Einfluss auf die Qualität der Ergebnisse. Die besten Ergebnisse erreichte 4/Sequenzlänge. Die Ergebnisse lassen vermuten, dass noch höhere Mutationsstärken sinnvoll sein könnten. In Abbildung 10.10 wird dieser Fragestellung nachgegangen.

Wert	$\bar{e}$	σ
1	-7,3843	0,78365
2	-7,6784	0,85301
4	-7,7855	0,88576

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
1	2	0,29412	0	0	[0,24334, 0,3449]	3998	2 schlägt 1
1	4	0,40119	0	0	[0,34934, 0,45304]	3998	4 schlägt 1
2	4	0,10707	1,9572e-07	0,00010028	[0,05316, 0,16098]	3998	4 schlägt 2

Abbildung 10.8: Boxplot: SA - Mutationsstärke (in x/Sequenzlänge), 2D-HPNX

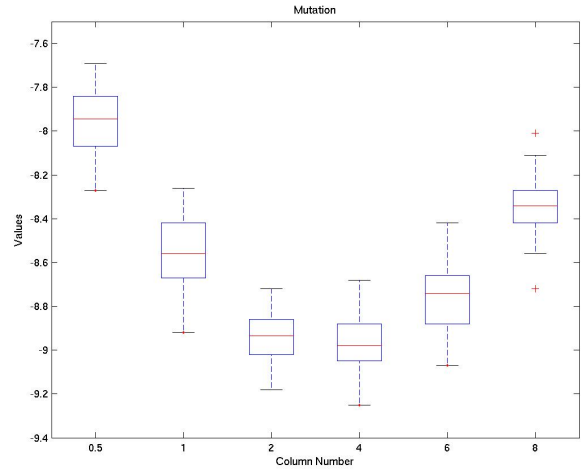


Eine Mutationsstärke von 2/Sequenzlänge scheint nur minimal, jedoch nicht signifikant besser zu sein als 4/Sequenzlänge. 1/Sequenzlänge ist signifikant schlechter als die beiden anderen Einstellungen.

Wert	Avg.	σ
1	-92,6344	5,4031
2	-95,2234	4,6207
4	-95,1055	4,1181

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
1	2	2,589	0	0	[2,2773, 2,9006]	3998	2 schlägt 1
1	4	2,4711	0	0	[2,1733, 2,7689]	3998	4 schlägt 1
2	4	-0,11786	0,92086	0,3945	[-0,3892, 0,15348]	3998	Nicht mögl.

Abbildung 10.9: Boxplot: SA - Mutationsstärke (in x/Sequenzlänge), 3D-Jernigan



Die Ergänzung der in Abbildung 10.8 dargestellten Analyse ergab, dass eine Mutationsstäke von 4/Sequenzlänge tatsächlich eine gute Wahl ist.

Wert	$\bar{e}$	σ
5	-7,9582	0,14737
1	-8,5456	0,15317
2	-8,7806	1,1263
4	-8,976	0,13197
6	-8,7578	0,15951
8	-8,3492	0,14515

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
0,5	1	0,5874	0	0	[0,52775, 0,64705]	98	1 schlaegt 0,5
0,5	2	0,8224	1,1102e-16	1,5298e-06	[0,50361, 1,1412]	98	2 schlaegt 0,5
0,5	4	1,0178	0	0	[0,96228, 1,0733]	98	4 schlaegt 0,5
0,5	6	0,7996	0	0	[0,73865, 0,86055]	98	6 schlaegt 0,5
0,5	8	0,391	2,2204e-16	0	[0,33295, 0,44905]	98	8 schlaegt 0,5
1	2	0,235	1,1102e-15	0,14697	[-0,084002, 0,554]	98	Nicht moegl,
1	4	0,4304	1,1102e-16	0	[0,37366, 0,48714]	98	4 schlaegt 1
1	6	0,2122	1,6144e-08	8,8874e-10	[0,15014, 0,27426]	98	6 schlaegt 1
1	8	-0,1964	2,0935e-08	2,3131e-09	[-0,25562, -0,13718]	98	1 schlaegt 8
2	4	0,1954	0,09105	0,22599	[-0,12285, 0,51365]	98	Nicht moegl,
2	6	-0,0228	1,3969e-07	0,88759	[-0,34205, 0,29645]	98	Nicht moegl,
2	8	-0,4314	1,1102e-16	0,0084903	[-0,75011, -0,11269]	98	2 schlaegt 8
4	6	-0,2182	2,2445e-09	3,6576e-11	[-0,2763, -0,1601]	98	4 schlaegt 6
4	8	-0,6268	0	0	[-0,68186, -0,57174]	98	4 schlaegt 8
6	8	-0,4086	3,3307e-16	0	[-0,46913, -0,34807]	98	6 schlaegt 8

Abbildung 10.10: Boxplot: SA - Mutationsstärke (in x/Sequenzlänge), erweiterte Analyse, 2D-HPNX

10.2.2 Genetischer Algorithmus - Faktorielles Design

Beim genetischen Algorithmus, dem zweiten Gegenstand unserer statistischen Erhebungen, untersuchten wir die in Tabelle 10.3 dargestellten Parameter mit den aufgeführten Einstellungen. Durch einen faktoriellen Versuchsaufbau ergaben sich $2^5 = 32$ Parametersätze. Die Ergebnisse

Tabelle 10.3: Parameterisierung des genetischen Algorithmus

Parameter	Werte
Crossoverwahrscheinlichkeit	0,7, 0,9
Codierung	absolut, relativ
Populationsgröße	50, 100
Selektion	Turnier, Rouletterad

sind analog zum vorangegangenen Kapitel dargestellt (siehe Abbildungen 10.11-10.17). Die Codierung zeigt auch beim genetischen Algorithmus bei geringer Varianz enorme Auswirkungen auf die Ergebnisse. Die relative Codierung ist der absoluten im zweidimensionalen Fall weit überlegen (siehe Abbildung 10.11). Dies kehrt sich im dreidimensionalen Fall interessanterweise um, hier ist die absolute Codierung der relativen ebenso klar überlegen (siehe Abbildung 10.12). Die Crossoverwahrscheinlichkeit ist bei beiden Modellen ein Parameter ohne signifikanten Einfluss (siehe Abbildungen 10.13, 10.14). Die Untersuchungen zur Populationsgröße zeigen einen schwachen, nichtsdestotrotz statistisch signifikanten Unterschied zwischen 100 Individuen und 50 Individuen (siehe Abbildungen 10.15, 10.16). Bei beiden Modellen sind 100 Individuen geringfügig besser. Trotz gleicher Zahl von Zielfunktionsauswertungen scheint eine große Population also von Vorteil zu sein. Bei der Analyse der Selektionsschemata stellte sich heraus, dass die Turnierselektion der Rouletteradselektion signifikant überlegen ist (siehe Abbildung 10.17). Abgesehen vom Laufzeitvorteil der Turnierselektion ($(O(n)$ im Vergleich zu $O(n^2)$) zeigt diese auch eine deutlich geringere Varianz der Ergebnisse. Sie führt zu konstant guten Resultaten.

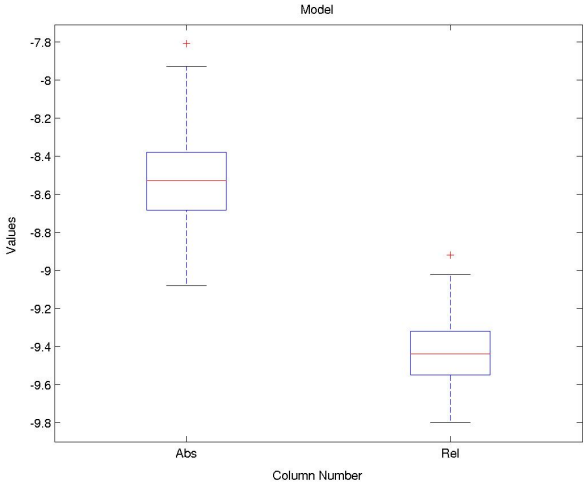
Wir ermittelten anhand dieser Daten wieder eine optimale Parameterkombination, welche in Tabelle 10.4 vorgestellt wird.

Tabelle 10.4: Optimale Parameter des genetischen Algorithmus

Parameter	2D-HPNX	3D-Jernigan
Crossoverwahrscheinlichkeit	0,9	0,9
Codierung	relativ	absolut
Mutationsstärke (in n/Sequenzlänge)	2	1
Populationsgröße	100	100
Selektion	Turnier	Turnier
Niching (Turniertyp)	1	1

10.2.3 Niching

Der neu eingefügte Niching-Parameter G (siehe Abschnitt 5.7) wurde unter Verwendung der besten ermittelten Einstellungen der Basisparameterenden ausführlich untersucht. Im Folgenden wird erörtert, ob eine größere Diversität der Population im Suchraum Auswirkungen auf die Qualität der Ergebnisse hat. Da sich das Niching für Turnierselektion und Rouletteradselektion unterscheidet, haben wir Experimente für beide Fälle durchgeführt (siehe Abbildungen 10.18-10.20). Wie man erkennen kann, ist der Einfluss des Nichings nur in der Rouletteradselektion signifikant (siehe Tabelle 10.18). Ein Niching-Parameter von $G = 0,5$ erbringt die besten Ergebnisse. Ohne Niching ($G = 0$) sind die Ergebnisse geringfügig schlechter. Die Einführung dieses Parameters ist also vorteilhaft. Das Niching der Turnierselektion hat in beiden Modellen keinen besonderen Einfluss auf die Ergebnisqualität, er wirkt sich lediglich auf die Varianz der Ergebnisse aus (siehe Abbildungen 10.19-10.20). Vermutlich ist die Diversitätpräservation bereits durch den Einsatz der Turnierselektion gegeben.

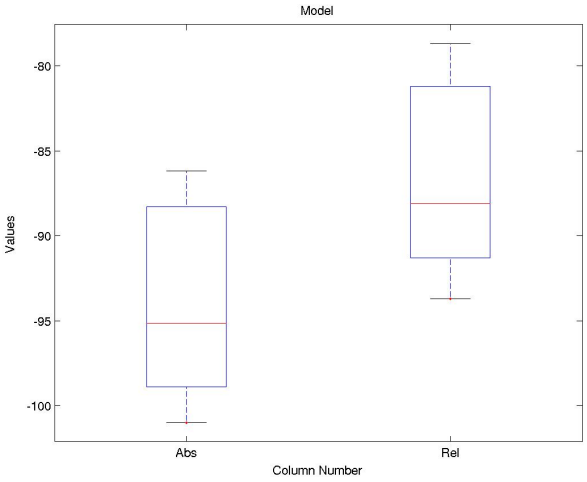


Der einflussreichste Parameter ist die Wahl der Codierung. Im zweidimensionalen Fall führt die relative Codierung zu weitaus besseren Ergebnissen als die absolute Codierung.

Wert	$\bar{e}$	σ
Abs	-9,2713	0,22543
Rel	-9,802	0,15597

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
Abs	Rel	0,5307	0	0	[0,51169, 0,54971]	1598	Rel schlägt Abs

Abbildung 10.11: Boxplot: GA - Relative und absolute Codierung der Faltung, 2D-HPNX

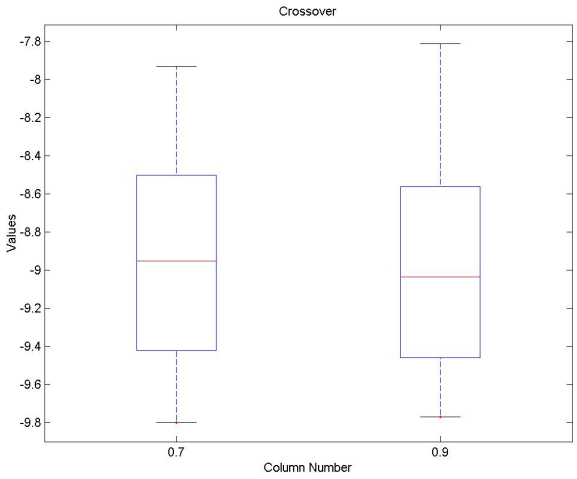


Im dreidimensionalen Fall ist die absolute Codierung weitaus erfolgreicher, als die relative Codierung. Das ist sehr interessant, da im zweidimensionalen Modell die relative Codierung überlegen war (siehe Abbildung 10.11).

Wert	$\bar{e}$	σ
Abs	-93,6823	4,674
Rel	-86,4515	4,6285

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
Abs	Rel	-7,2308	0	0	[-7,6609, -6,8008]	1798	Abs schlägt Rel

Abbildung 10.12: Boxplot: GA - Relative und absolute Codierung der Faltung, 3D-Jernigan

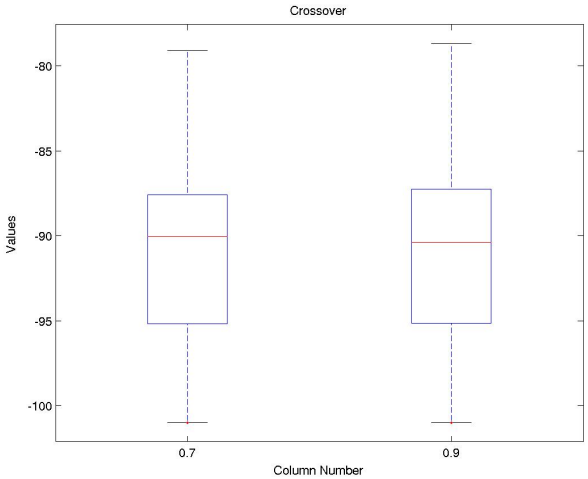


Crossover scheint ein Parameter ohne signifikanten Einfluss zu sein. Die durchschnittlich besseren Ergebnisse erzielte $p = 0,9$, jedoch lässt sich mit den verwendeten Tests die Nullhypothese nicht verwerfen. Der Unterschied könnte also auch zufälliger Natur sein.

Wert	$\bar{e}$	σ
0,7	-9,5287	0,33372
0,9	-9,5446	0,32349

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
0,7	0,9	0,015925	0,43866	0,33263	[-0,016306, 0,048156]	1598	Nicht mögl.

Abbildung 10.13: Boxplot: GA - Crossoverwahrscheinlichkeit

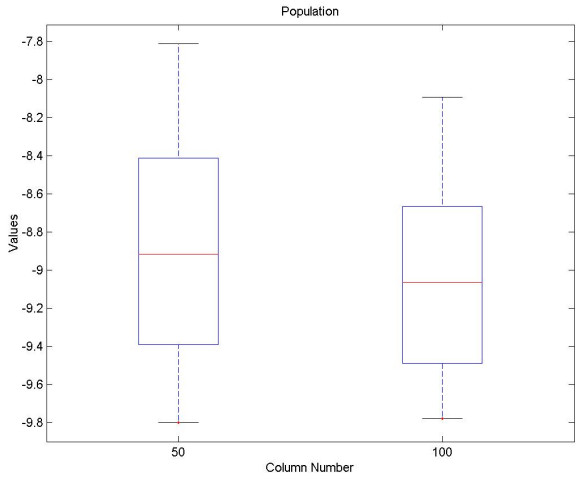


Verschiedene Crossoverwerte führen nicht zu signifikanten Veränderungen in der Ergebnisqualität. Die durchschnittlich besseren Ergebnisse wurden mit $p = 0.7$ erzielt. Allerdings lässt sich auch hier die Nullhypothese nicht verwerfen, die Unterschiede können also zufällig sein.

Wert	Avg.	σ
0,7	-90,0974	5,7935
0,9	-90,0365	5,9896

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
0,7	0,9	-0,060912	0,4106	0,82645	[-0,60569, 0,48387]	1798	Nicht mögl.

Abbildung 10.14: Boxplot: GA - Crossoverwahrscheinlichkeit, 3D-Jernigan

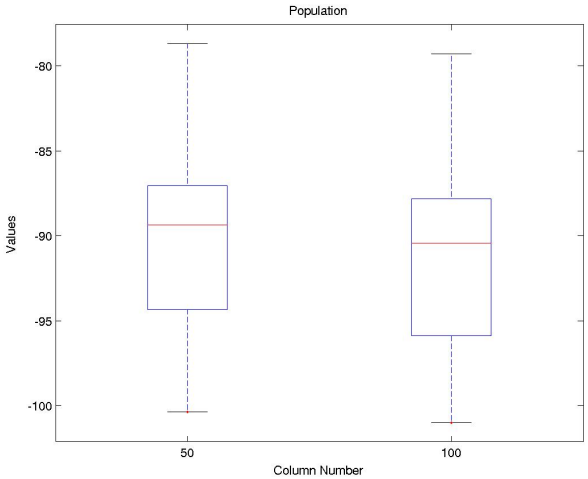


Die Populationsgröße hat geringen, jedoch signifikanten Einfluss auf die Ergebnisse. Die besten Versuchsergebnisse wurden mit einer Individuenzahl von 100 erzielt.

Wert	$\bar{e}$	σ
50	-9,5124	0,3418
100	-9,561	0,31326

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
50	100	0,04865	0,003427	0,003043	[0,016498, 0,080802]	1598	100 schlägt 50

Abbildung 10.15: Boxplot: GA - Populationsgröße, 2D-HPNX

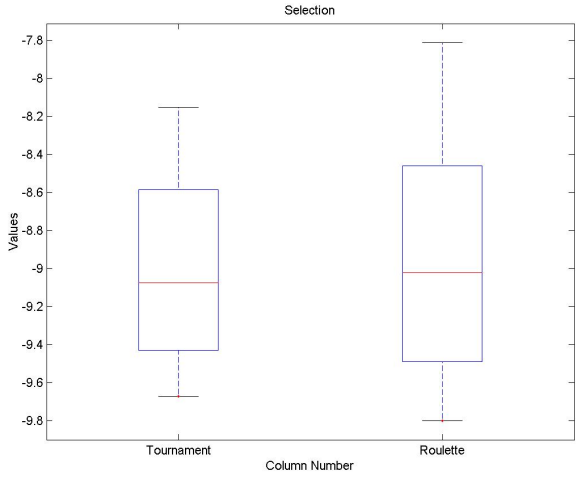


Eine Populationsgröße von 100 Individuen ist auch im dreidimensionalen Modell signifikant besser als eine Populationsgröße von 50 Individuen. Allerdings ist der durchschnittliche Unterschied äußerst gering (0,82428).

Wert	Avg.	σ
50	-90,0669	5,8908
100	-90,8912	5,9988

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
50	100	0,82428	3,0207e-12	3,2644e-05	[0,43574, 1,2128]	3598	100 schlägt 50

Abbildung 10.16: Boxplot: GA - Populationsgröße, 3D-Jernigan

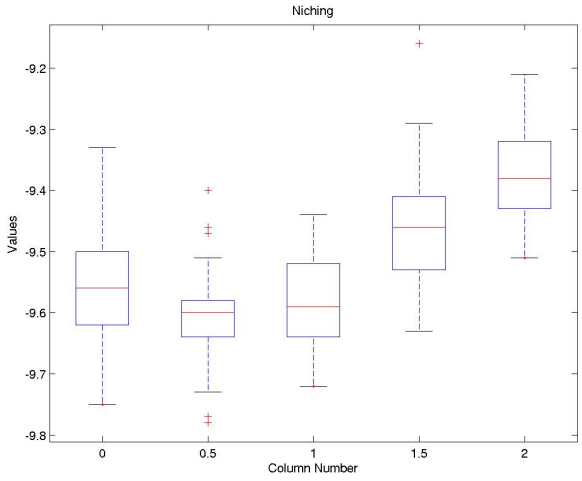


Turnierselektion ist effizienter und bietet signifikant bessere Ergebnisse als die Rouletteradselektion. Die Varianz der Ergebnisse ist deutlich geringer, die Ergebnisqualität ist also konstanter.

Wert	$\bar{e}$	σ
Tour.	-9,5902	0,26621
Roul.	-9,4832	0,37356

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
Tour.	Roul.	-0,10697	0,00019471	5,721e-11	[-0,13879, -0,075164]	1598	Tour. schlägt Roul.

Abbildung 10.17: Boxplot: GA - Selektion

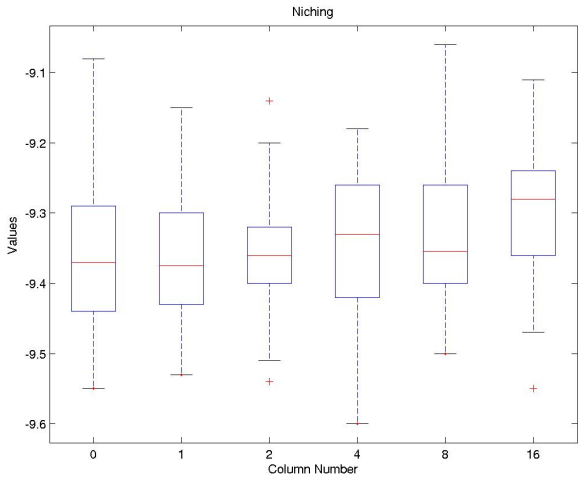


Die besten Ergebnisse erzielten wir mit einem Niching-Wert von $G = 0,5$. Der Einsatz der Niching-Strategie führt damit zu signifikant besseren Ergebnissen.

Wert	$\bar{e}$	σ
0	-9,558	0,089351
0,5	-9,6086	0,076185
1	-9,581	0,073796
1,5	-9,4528	0,093286
2	-9,3688	0,078132

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
0	0,5	0,0506	0,0055106	0,0029694	[0,017646, 0,083554]	98	0,5 schlägt 0
0	1	0,023	0,22991	0,16366	[-0,009523, 0,055523]	98	Nicht mögl.
0	1,5	-0,1052	4,7226e-07	9,7526e-08	[-0,14145, -0,068948]	98	0 schlägt 1,5
0	2	-0,1892	1,7653e-14	0	[-0,22251, -0,15589]	98	0 schlägt 2
0,5	1	-0,0276	0,12718	0,068795	[-0,057367, 0,0021671]	98	Nicht mögl.
0,5	1,5	-0,1558	3,0509e-13	8,6597e-15	[-0,1896, -0,122]	98	0,5 schlägt 1,5
0,5	2	-0,2398	0	0	[-0,27043, -0,20917]	98	0,5 schlägt 2
1	1,5	-0,1282	4,1005e-10	1,6163e-11	[-0,16158, -0,094818]	98	1 schlägt 1,5
1	2	-0,2122	1,1102e-16	0	[-0,24236, -0,18204]	98	1 schlägt 2
1,5	2	-0,084	6,616e-06	4,0916e-06	[-0,11815, -0,04985]	98	1,5 schlägt 2

Abbildung 10.18: Boxplot: GA - Niching (Rouletteradselektion), 2D-HPNX

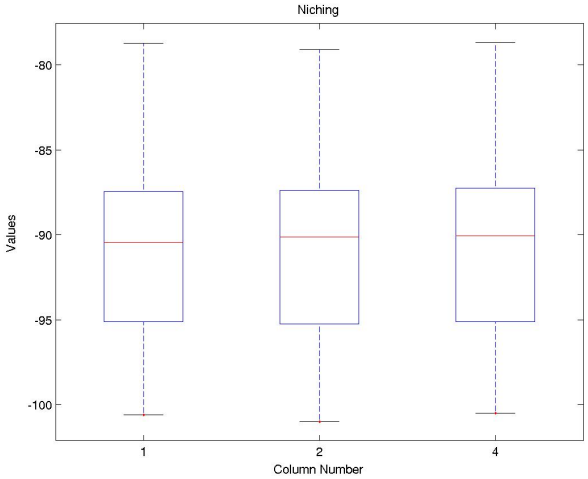


Bei der Turnierselektion ist keine klare Tendenz zu erkennen. Der Niching-Parameter scheint von geringer Bedeutung zu sein. Im Durchschnitt erzielten wir die besten Ergebnisse mit $G = 1$.

Wert	$\bar{e}$	σ
0	-9,3636	0,10575
1	-9,3662	0,088684
2	-9,3544	0,08132
4	-9,3424	0,097553
8	-9,3316	0,10106
16	-9,3022	0,096517

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
0	1	0,0026	0,95048	0,8943	[-0,036133, 0,041333]	98	Nicht mögl.
0	2	-0,0092	0,60002	0,62688	[-0,046638, 0,028238]	98	Nicht mögl.
0	4	-0,0212	0,18415	0,3	[-0,061577, 0,019177]	98	Nicht mögl.
0	8	-0,032	0,14742	0,12511	[-0,073052, 0,0090515]	98	Nicht mögl.
0	16	-0,0614	0,0021412	0,0031041	[-0,10158, -0,021219]	98	0 schlägt 16
1	2	-0,0118	0,521	0,48967	[-0,045568, 0,021968]	98	Nicht mögl.
1	4	-0,0238	0,1389	0,2048	[-0,0608, 0,0132]	98	Nicht mögl.
1	8	-0,0346	0,13161	0,071871	[-0,072335, 0,0031347]	98	Nicht mögl.
1	16	-0,064	0,00054722	0,00082115	[-0,10079, -0,027215]	98	1 schlägt 16
2	4	-0,012	0,34095	0,50563	[-0,047643, 0,023643]	98	Nicht mögl.
2	8	-0,0228	0,40954	0,21689	[-0,059205, 0,013605]	98	Nicht mögl.
2	16	-0,0522	0,0021877	0,0042846	[-0,08762, -0,01678]	98	2 schlägt 16
4	8	-0,0108	0,95048	0,5879	[-0,050221, 0,028621]	98	Nicht mögl.
4	16	-0,0402	0,073326	0,04095	[-0,078713, -0,001687]	98	Nicht mögl.
8	16	-0,0294	0,071722	0,14006	[-0,068619, 0,0098194]	98	Nicht mögl.

Abbildung 10.19: Boxplot: GA - Niching (Turnierselektion), 2D-HPNX



Im 3D-Jernigan-Modell zeigt das Niching innerhalb der Turnierselektion ebenfalls keinen besonderen Einfluss auf die Ergebnisqualität. Keine der Einstellungen scheint den anderen signifikant überlegen zu sein. Durchschnittlich beste Ergebnisse ergab auch hier $G = 1$.

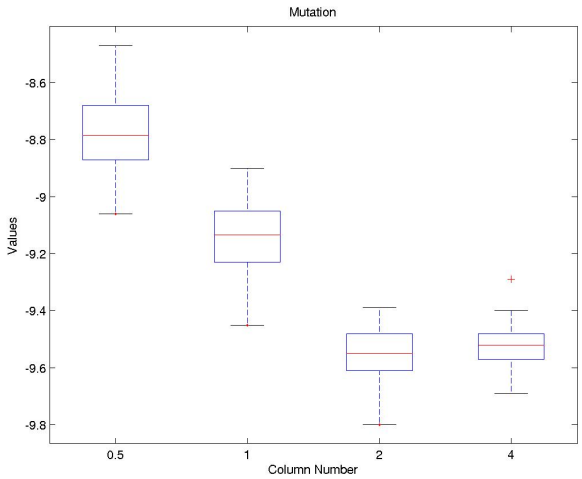
Wert	Avg.	σ
1,0	-90,5663	5,9659
2,0	-90,5115	5,9801
4,0	-90,3594	5,9324

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
1,0	2,0	-0,054784	0,70144	0,82226	[-0,53296, 0,42339]	2398	Nicht mögl.
1,0	4,0	-0,2069	0,17362	0,39437	[-0,68316, 0,26937]	2398	Nicht mögl.
2,0	4,0	-0,15211	0,31843	0,53166	[-0,62895, 0,32472]	2398	Nicht mögl.

Abbildung 10.20: Boxplot: GA - Niching (Turnierselektion), 3D-Jernigan

10.2.4 Mutation

Wir haben uns entschieden, die Mutation als besonders sensiblen Parameter zusätzlich separat zu untersuchen. Die Ergebnisse sind in den Abbildungen 10.21-10.23 aufgeführt. Diese zeigen, dass die Mutation neben der Codierung den stärksten Einfluss auf die Ergebnisse hat. Schon kleine Änderungen können starke Auswirkungen auf das Ergebnis haben. Im zweidimensionalen Modell zeigte sich, dass eine Mutationsstärke von 2/Sequenzlänge die besten Ergebnisse erbringt (siehe Abbildung 10.21). 4/Sequenzlänge ist nur geringfügig, jedoch nicht signifikant besser, zeigt dafür eine geringere Varianz der Ergebnisse. Geringere Mutationsstärken führen zu signifikant schlechteren Ergebnissen. Beim dreidimensionalen Modell zeigte sich in einem ersten Test, dass innerhalb der getesteten Werte die geringste Mutationsstärke von 1/Sequenzlänge die signifikant und mit großem Abstand besten Ergebnisse erzielte (siehe Abbildung 10.22). Die Auswertungen legen nahe, dass eventuell geringere Mutationsstärken zu noch besseren Resultaten führen könnte. In einem Folgeexperiment untersuchten wir diese Fragestellung und erhielten aufschlussreiche Ergebnisse. Eine Mutationsstärke von 0,5/Sequenzlänge ist noch besser als die im ersten Versuch getesteten Werte. Er ist gleichzeitig der beste gefundene Wer. Eine Mutationsstärke von 0,25/Sequenzlänge schneidet signifikant schlechter ab, als 0,5/Sequenzlänge. Damit kann man abschließend sagen, dass die Mutationsrate ein sehr wichtiger Parameter ist und modellabhängig angepasst werden muss, um gute Resultate zu erhalten.

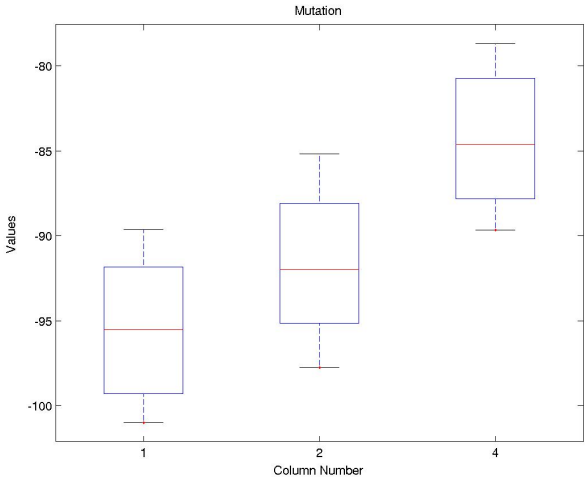


Im zweidimensionalen Fall erzielten wir mit 2/Sequenzlänge die besten Ergebnisse, 4/Sequenzlänge scheint nur marginal schlechter zu sein. 1/Sequenzlänge ist weniger zufriedenstellend.

Wert	$\bar{e}$	σ
0,5	-8,7836	0,14842
1	-9,1426	0,1261
2	-9,548	0,094782
4	-9,5238	0,07298

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
0,5	1	0,359	5,5511e-16	0	[0,30434, 0,41366]	98	1 schlägt 0,5
0,5	2	0,7644	0	0	[0,71498, 0,81382]	98	2 schlägt 0,5
0,5	4	0,7402	0	0	[0,69378, 0,78662]	98	4 schlägt 0,5
1	2	0,4054	0	0	[0,36113, 0,44967]	98	2 schlägt 1
1	4	0,3812	0	0	[0,34031, 0,42209]	98	4 schlägt 1
2	4	-0,0242	0,21039	0,15576	[-0,057772, 0,0093718]	98	Nicht mögl.

Abbildung 10.21: Boxplot: GA - Mutationsstärke (in x/Sequenzlänge), 2D-HPNX

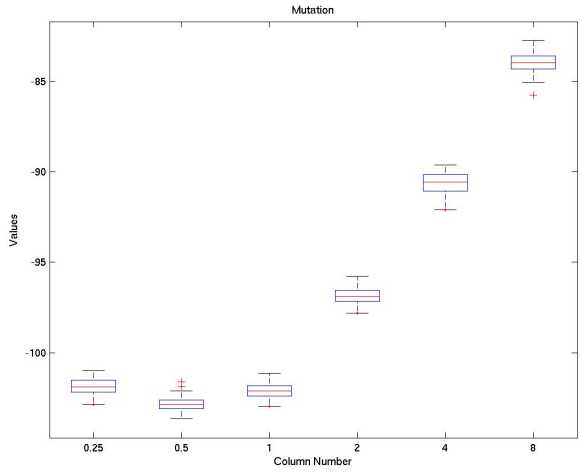


Im dreidimensionalen Fall ist eine schwache Mutation (1/Sequenzlänge) gut, stärkere Mutationen führen zu deutlich schlechteren Ergebnisse. Die Auswertungen zeigen eventuell einen Trennd: Geringere Mutationsstärken könnten zu noch besseren Ergebnissen führen. Dies haben wir in einer Folgeuntersuchung getestet (vgl. Abbildung 10.23).

Wert	$\bar{e}$	σ
1,0	-95,2841	3,7579
2,0	-90,8357	3,5956
4,0	-84,081	3,6636

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
1,0	2,0	-4,4484	0	0	[-4,865, -4,0318]	1198	1,0 schlägt 2,0
1,0	4,0	-11,2032	0	0	[-11,6235, -10,7828]	1198	1,0 schlägt 4,0
2,0	4,0	-6,7547	0	0	[-7,1659, -6,3436]	1198	2,0 schlägt 4,0

Abbildung 10.22: Boxplot: GA - Mutation (in x/Sequenzlänge), 3D-Jernigan



Wie in Abbildung 10.22 bereits vermutet, zeigt sich tatsächlich, dass eine Mutationsstärke von 1/Sequenzlänge immernoch zu hoch gewählt ist. In diesem erweiterten Versuch wurden die signifikant besten Ergebnisse mit einer Mutationsstärke von 0,5/Sequenzlänge erzielt. Niedrigere und höhere Werte führen zu Verschlechterungen.

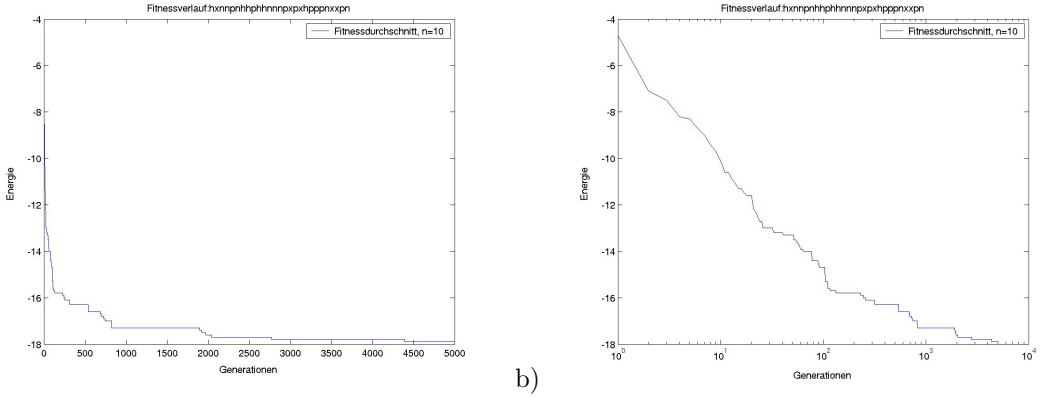
Wert	Avg.	σ
0,25	-101,891	0,44874
0,5	-102,8509	0,40893
1	-102,0827	0,40597
2	-96,9081	0,48142
4	-90,6189	0,58214
8	-84,0011	0,5763

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
0,25	0,5	0,95992	3,3751e-14	0	[0,78953, 1,1303]	98	0,5 schlaegt 0,25
0,25	1	0,19172	0,028363	0,027331	[0,021892, 0,36155]	98	1 schlaegt 0,25
0,25	2	-4,9829	0	0	[-5,1676, -4,7982]	98	0,25 schlaegt 2
0,25	4	-11,2721	0	0	[-11,4784, -11,0658]	98	0,25 schlaegt 4
0,25	8	-17,8899	0	0	[-18,0949, -17,6849]	98	0,25 schlaegt 8
0,5	1	-0,7682	1,0169e-12	2,2204e-15	[-0,92991, -0,60648]	98	0,5 schlaegt 1
0,5	2	-5,9428	0	0	[-6,1201, -5,7655]	98	0,5 schlaegt 2
0,5	4	-12,232	0	0	[-12,4317, -12,0323]	98	0,5 schlaegt 4
0,5	8	-18,8498	0	0	[-19,0481, -18,6515]	98	0,5 schlaegt 8
1	2	-5,1746	0	0	[-5,3513, -4,9978]	98	1 schlaegt 2
1	4	-11,4638	0	0	[-11,663, -11,2646]	98	1 schlaegt 4
1	8	-18,0816	0	0	[-18,2794, -17,8837]	98	1 schlaegt 8
2	4	-6,2892	0	0	[-6,5012, -6,0772]	98	2 schlaegt 4
2	8	-12,907	0	0	[-13,1177, -12,6963]	98	2 schlaegt 8
4	8	-6,6178	0	0	[-6,8477, -6,3879]	98	4 schlaegt 8

Abbildung 10.23: Boxplot: GA - Erweiterte Analyse der Mutation, 3D-Jernigan

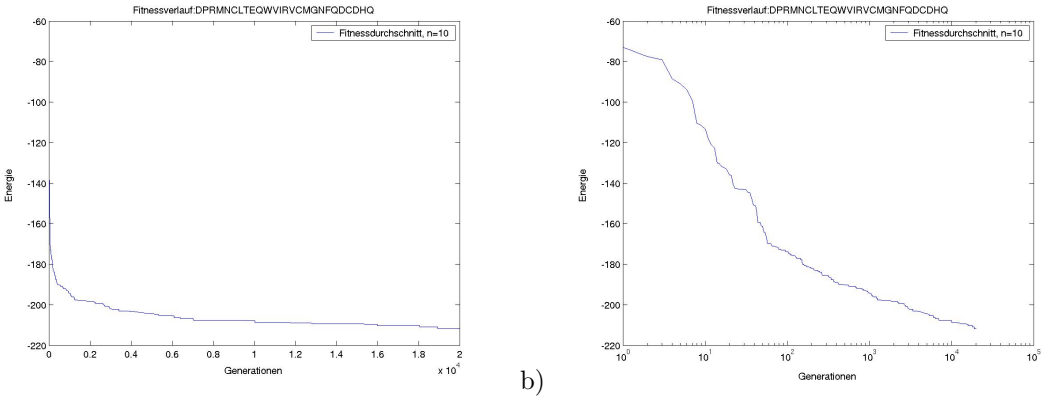
10.3 Konvergenz

Wir können durch alleiniges Betrachten der Konvergenzgeschwindigkeit natürlich nicht direkt die Effizienz des Algorithmus verbessern. Die Ergebnisse können uns allerdings Einblick in das Konvergenzverhalten verschaffen und es uns so erleichtern, ein Abbruchkriterium festzulegen. Wir haben an zwei zufälligen Sequenzen den Verlauf der Fitness des besten Individuums verfolgt. Dazu mittelten wir die Laufdaten von zehn unabhängig durchgeführten Läufen. Die Ergebnisse sind in den Abbildungen 10.24 (2D-HPNX) und 10.25 (3D-Jernigan) als linearer und semilogarithmischer Plot dargestellt. Man erkennt in beiden Modellen, dass die Konvergenzkurve eine näherungsweise logarithmische Form hat. Es scheint eine exponentielle Abhängigkeit zwischen Güte und Laufzeit zu bestehen. Das Problem, welches sich hieraus ergibt, ist die Qualität der Ergebnisse. Es zeigt sich wie bei jedem Approximationsalgorithmus, dass Resultate ab einer gewissen Güte meist mit unpraktikablen Laufzeiten verbunden sind.



a) b)

Abbildung 10.24: Beispielhafter Fitnessverlauf (2D-HPNX-Modell), Durchschnitt von zehn Läufen. a) linearer Plot b) semilogarithmischer Plot

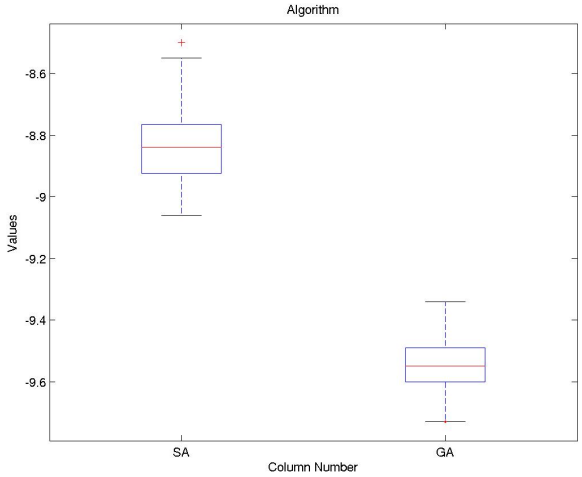


a) b)

Abbildung 10.25: Beispielhafter Fitnessverlauf (3D-Jernigan-Modell), Durchschnitt von zehn Läufen. a) linearer Plot b) semilogarithmischer Plot

10.4 Wahl der Faltungsheuristik

In diesem Abschnitt wird endgültig für die nun parameterisierten Heuristiken (SA/GA) entschieden, welche für die Sequenzoptimierung gewählt wird. Hierzu führten wir 100 Versuchsläufe durch und stellten die Ergebnisse in Abbildung 10.26 dar. Im 2D-HPNX-Modell ist der genetische Algorithmus dem Simulated Annealing signifikant überlegen. Bei geringer Varianz ist der GA um durchschnittlich 0,5 Energiepunkte besser, was einer Verbesserung um 7,4% entspricht. Im 3D-Jernigan-Modell ist genau das Gegenteil der Fall (siehe Abbildung 10.27). Hier ist das Simulated Annealing um 0,5 Energiepunkte besser, was 0,5% entspricht. Diee logische Konsequenz hieraus ist, für das 2D-HPNX-Modell den GA und für das 3D-Jernigan-Modell das Simulated Annealing zu verwenden. In den folgenden Untersuchungen werden die entsprechenden Heuristiken verwendet.

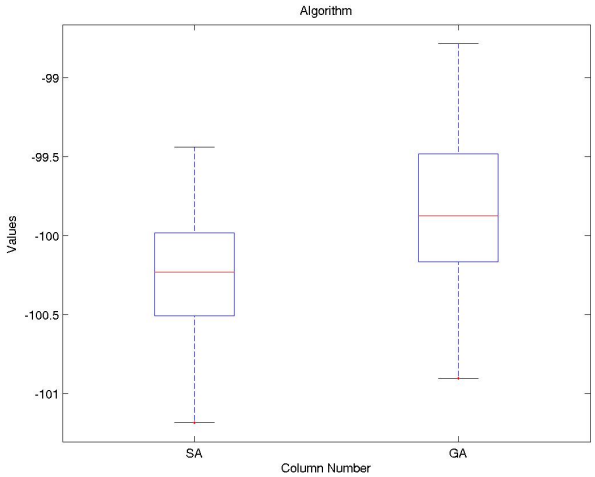


Im zweidimensionalen Fall ist der GA deutlich besser als der SA.

Wert	$\bar{e}$	σ
SA	-8,8365	0,11647
GA	-9,5473	0,082055

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
SA	GA	0,7108	0	0	[0,6827, 0,7389]	198	GA schlägt SA

Abbildung 10.26: Vergleich von Simulated Annealing und GA, 2D-HPNX



Im dreidimensionalen Fall hat sich die Simulated Annealing-Strategie als besser herausgestellt.

Wert	$\bar{e}$	σ
SA	-100,2419	0,3898
GA	-99,8267	0,45221

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
SA	GA	-0,41517	6,6394e-10	5,0439e-11	[-0,5329, -0,29744]	198	SA schlägt GA

Abbildung 10.27: Vergleich von Simulated Annealing und GA, 3D-Jernigan

10.5 Stabilitätsanalyse

Es stellt sich die Frage, wie verlässlich die Heuristiken die Stabilität des gefalteten Peptids vorhersagen können. Wie verlässlich ist die aus der Energielandschaft gezogene Stichprobe (siehe Abbildung 10.28), um die Wahrscheinlichkeit der nativen Konformation zu berechnen (siehe Abschnitt 2.2). Um dies festzustellen, erzeugten wir Referenzstichproben der Größe 100. Sie enthielten HPNX-Strings mit Längen zwischen 5 und 15. Wir berechneten die tatsächliche Stabilität mittels des Enumerators und verglichen die Ergebnisse mit der Simulated Annealing-Strategie, dem genetischen Algorithmus und einer randomisierten Suche. Zusätzlich verglichen wir die Heuristiken untereinander und maßen die Reproduzierbarkeit der Ergebnisse bei zwei unabhängigen Läufen der gleichen Heuristk. Die randomisierte Suche zieht ein gleichmäßig verteiltes Sample aus dem Konformationsraum, während sämtliche Heuristiken erwarten lassen, dass eine höhere Samplingdichte in der Nähe von lokalen Optima erzielt wird. Wir maßen die Korrelation und die Rangkorrelation

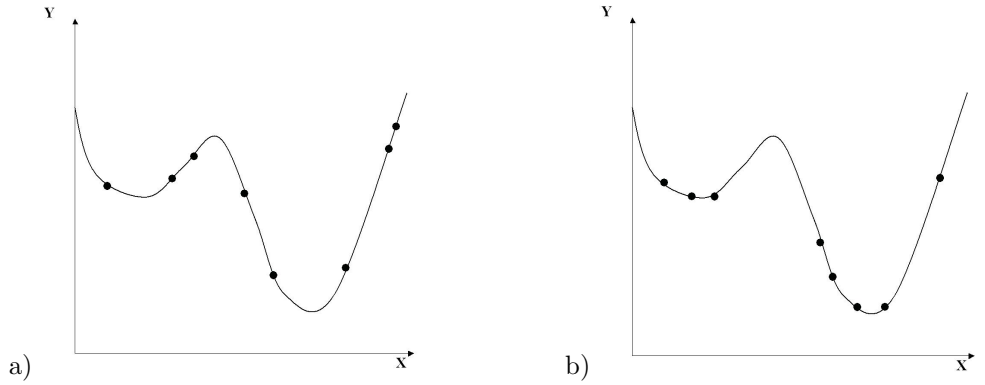


Abbildung 10.28: Samplingdichte (idealisiert), a) zufälliges Sample, b) Heuristik (GA/SA)

zwischen all diesen Stichproben. Unseres Erachtens ist die Rangkorrelation aussagekräftiger für die Qualität der Ergebnisse, da die Sequenzfindungsheuristiken rangbasiert arbeiten. Wichtig ist also nicht der absolute Wert der Stabilität, sondern vielmehr die Ordnung zwischen diesen Werten. Wir untersuchten auch, ob die Qualität mit größerer Samplingdichte stark zunimmt, denn diese Ergebnisse sind maßgeblich, um die korrekte Laufzeit für die Faltungsheuristiken festzustellen. Die Nullhypothese kann bei dieser Stichprobengröße bereits bei einer Korrelation von $r = 0,32$ mit 99,9%-iger Sicherheit verworfen werden. Als weiteres Maß wählten wir die Wahrscheinlichkeit, dass ein Vergleich zwischen Sequenzen s_1 und s_2, welcher mittels Algorithmus A ausgeführt wird, in Algorithmus B das gleiche Ergebnis erzielt:

$$p_v(B(s_1) > B(s_2) | A(s_1) > A(s_2)) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} Comp(s_i, s_j)}{n^2}$$

$$Comp(s_i, s_j) = \begin{cases} 0 & A(s_i) > B(s_i) \text{ und } A(s_j) \leq B(s_j) \\ 0 & A(s_i) < B(s_i) \text{ und } A(s_j) \geq B(s_j) \\ 0 & A(s_i) = B(s_i) \text{ und } A(s_j) \neq B(s_j) \\ 1 & \text{sonst} \end{cases}$$

Wie man aus Abbildung 10.7 ersehen kann, korrelieren die tatsächlichen Größen zwischen der deterministischen Methode des Enumerators und den einzelnen Heuristiken nicht sonderlich stark. Die Heuristiken unterscheiden sich sehr in der Qualität der Stabilitätsvorhersage. Ein anderes Bild zeigt sich bei der Rangkorrelation und der Wahrscheinlichkeit des korrekten Vergleichs (siehe Tabellen 10.5 und 10.6). Hier korrelieren die Algorithmen stärker. Die korrekte Vorhersagewahrscheinlichkeit ist nur marginal verschieden. Alle Heuristiken erreichen eine Wahrscheinlichkeit eines korrekten Vergleichs von über 75%. Dies zeigt, dass die Heuristiken durchaus in der Lage sind, mit einer hohen Güte die Stabilität eines Proteins zu errechnen. Interessanterweise korrelieren

zwei unabhängige Stichproben ein und derselben Heuristik sehr stark miteinander. Das heißt, dass unabhängig von Startpunkt und Zufallseinwirkung die Heuristiken an ähnlichen Stellen ähnliche Fehler machen. Zusätzlich korrelieren der genetische Algorithmus und das Simulated Annealing-Verfahren ebenfalls stark untereinander. Wir vermuten deshalb, dass verschiedene Heuristiken gerne die gleichen Vorhersagefehler machen, und daher die Stabilität ab einem gewissen Grad schwer vorhersagbar wird. Diese Schlussfolgerungen treffen auch auf den dreidimensionalen Fall zu (siehe Tabellen 10.8, 10.9 und 10.10). Hier ist die Wahrscheinlichkeit eines korrekten Vergleichs sogar noch höher, sie liegt zwischen 84,5% und 86,7% (Tabelle 10.10).

Wir untersuchten, inwiefern eine höhere Zahl an Zielfunktionsauswertungen der Algorithmen die Qualität der Stabilitätsvorhersage beeinflusst. Hierzu wählten wir den gleichen Versuchsaufbau, gewährten jedoch allen Algorithmen die zehnfache Anzahl Zielfunktionsauswertungen. Da wir nicht implizit die Stabilität optimieren, ist nicht unbedingt zu erwarten, dass eine Erhöhung der Auswertungszahl keine große Verbesserung bringt. Und tatsächlich ist keine Steigerung der Vorhersagequalität bei genetischem Algorithmus und Simulated Annealing zu erkennen. Was jedoch auf den ersten Blick überraschend scheint, ist, dass selbst die randomisierte Suche keine Verbesserung zeigt. Dies kann sich eventuell dadurch erklären lassen, dass die falsch vorhergesagten Sequenzen so genannte „Needle in the Haystack“ (Nadel im Heuhaufen)-Funktionen sind, mit einem sehr schwer zu findenden Optimum, wie z. B. in Abbildung 2.9, c).

Tabelle 10.5: Rangkorrelation der Stabilität (2D-HPNX-Modell)

r_R	Enumerator	GA 1	GA 2	SA 1	SA 2	Random 1	Random 2
Enumerator	1,0000	0,4047	0,4168	0,5010	0,4166	0,5229	0,5126
GA 1	-	1,0000	0,9732	0,8777	0,8717	0,6990	0,6272
GA 2	-	-	1,0000	0,8569	0,8576	0,6794	0,6211
SA 1	-	-	-	1,0000	0,9277	0,7443	0,6302
SA 2	-	-	-	-	1,0000	0,7268	0,6091
Random 1	-	-	-	-	-	1,0000	0,7756
Random 2	-	-	-	-	-	-	1,0000

Tabelle 10.6: Wahrscheinlichkeit eines korrekten Vergleichs (2D-HPNX-Modell)

p_v	Enumerator	GA 1	GA 2	SA 1	SA 2	Random 1	Random 2
Enumerator	1,0000	0,7590	0,7641	0,7873	0,7617	0,7901	0,7811
GA 1	-	1,0000	0,9712	0,9030	0,9042	0,8320	0,8296
GA 2	-	-	1,0000	0,8992	0,9046	0,8254	0,8182
SA 1	-	-	-	1,0000	0,9401	0,8396	0,8225
SA 2	-	-	-	-	1,0000	0,8407	0,8153
Random 1	-	-	-	-	-	1,0000	0,8666
Random 2	-	-	-	-	-	-	1,0000

Tabelle 10.7: Korrelation der Stabilität (2D-HPNX-Modell)

r	Enumerator	GA 1	GA 2	SA 1	SA 2	Random 1	Random 2
Enumerator	1,0000	0,4638	0,4020	0,3812	0,3737	0,5668	0,4509
GA 1	-	1,0000	0,7686	0,8120	0,7081	0,5193	0,4893
GA 2	-	-	1,0000	0,6194	0,7657	0,5939	0,6123
SA 1	-	-	-	1,0000	0,5753	0,3488	0,3128
SA 2	-	-	-	-	1,0000	0,5104	0,5079
Random 1	-	-	-	-	-	1,0000	0,5434
Random 2	-	-	-	-	-	-	1,0000

Tabelle 10.8: Rangkorrelation der Stabilität (3D-Jernigan-Modell)

r_R	Enumerator	GA 1	GA 2	SA 1	SA 2	Random 1	Random 2
Enumerator	1,0000	0,7828	0,7896	0,8110	0,8319	0,8099	0,7915
GA 1	-	1,0000	0,9821	0,9329	0,9412	0,9798	0,9799
GA 2	-	-	1,0000	0,9315	0,9361	0,9851	0,9751
SA 1	-	-	-	1,0000	0,9474	0,9351	0,9262
SA 2	-	-	-	-	1,0000	0,9456	0,9317
Random 1	-	-	-	-	-	1,0000	0,9803
Random 2	-	-	-	-	-	-	1,0000

Tabelle 10.9: Korrelation der Stabilität (3D-Jernigan-Modell)

r	Enumerator	GA 1	GA 2	SA 1	SA 2	Random 1	Random 2
Enumerator	1,0000	0,5517	0,6354	0,6155	0,6982	0,7439	0,7091
GA 1	-	1,0000	0,7894	0,7497	0,7292	0,8614	0,7476
GA 2	-	-	1,0000	0,7636	0,8107	0,8531	0,7821
SA 1	-	-	-	1,0000	0,7349	0,7454	0,6744
SA 2	-	-	-	-	1,0000	0,8388	0,7404
Random 1	-	-	-	-	-	1,0000	0,8362
Random 2	-	-	-	-	-	-	1,0000

Tabelle 10.10: Wahrscheinlichkeit eines korrekten Vergleichs (3D-Jernigan-Modell)

p_v	Enumerator	GA 1	GA 2	SA 1	SA 2	Random 1	Random 2
Enumerator	1,0000	0,8452	0,8453	0,8572	0,8671	0,8560	0,8506
GA 1	-	1,0000	0,9731	0,9277	0,9313	0,9715	0,9687
GA 2	-	-	1,0000	0,9279	0,9291	0,9746	0,9636
SA 1	-	-	-	1,0000	0,9464	0,9302	0,9273
SA 2	-	-	-	-	1,0000	0,9361	0,9302
Random 1	-	-	-	-	-	1,0000	0,9719
Random 2	-	-	-	-	-	-	1,0000

Tabelle 10.11: Wahrscheinlichkeit eines korrekten Vergleichs bei zehnfacher Auswertungszahl (2D-HPNX-Modell)

p_v	Enumerator	GA 1	GA 2	SA 1	SA 2	Random 1	Random 2
Enumerator	1,0000	0,7537	0,7531	0,7614	0,7762	0,7613	0,7792
GA 1	-	1,0000	0,9867	0,9366	0,9351	0,8004	0,8351
GA 2	-	-	1,0000	0,9365	0,9351	0,8042	0,8338
SA 1	-	-	-	1,0000	0,9730	0,8414	0,8673
SA 2	-	-	-	-	1,0000	0,8297	0,8618
Random 1	-	-	-	-	-	1.0000	0,8818
Random 2	-	-	-	-	-	-	1.0000

Tabelle 10.12: Wahrscheinlichkeit eines korrekten Vergleichs bei zehnfacher Auswertungszahl (3D-Jernigan-Modell)

p_v	Enumerator	GA 1	GA 2	SA 1	SA 2	Random 1	Random 2
Enumerator	1,0000	0,7485	0,7467	0,7699	0,7650	0,7639	0,7613
GA 1	-	1,0000	0,9904	0,9495	0,9572	0,9757	0,9753
GA 2	-	-	1,0000	0,9531	0,9607	0,9745	0,9731
SA 1	-	-	-	1,0000	0,9792	0,9565	0,9555
SA 2	-	-	-	-	1,0000	0,9569	0,9547
Random 1	-	-	-	-	-	1.0000	0,9918
Random 2	-	-	-	-	-	-	1.0000

10.6 Sequenzauswahl

Bei der Analyse der Sequenzauswahl ergibt sich das Problem, dass ein Versuchslauf einen mitunter sehr langen Zeitraum in Anspruch nehmen kann. Ein Optimierungslauf mit den unten angegebenen Parametern benötigt durchschnittlich 40 Minuten CPU-Zeit. Dadurch ist es nicht mehr möglich, wie in Abschnitt 10.2 beschriebene ausführliche Tests durchzuführen. Ein weiteres, großes Problem für den Vergleich der Qualität ist die multidimensionale Form der Ergebnisse. Man kann nicht mehr einfach auf herkömmliche statistische Verfahren zurückgreifen. Wir verwenden daher eine spezielle Metrik zum Vergleich multikriterieller Ergebnisse, die R2-Metrik von Hansen und Jaszciewicz [51], welche in Abschnitt 10.1.2 eingeführt wird. Auch hier gehen wir ähnlich wie in den vorangegangenen Kapiteln vor. Zuerst wird der Versuchsaufbau mittels eines faktoriellen Designs geplant. Mittels der Testergebnisse sollen die „unkritischen“ Parameter auf einen guten Wert festlegt werden. Zusätzliche Features der GP werden hier zuerst abgeschaltet und erst später untersucht. Diese Methode spart im Vergleich zu einem faktoriellen experimentellen Design zeitintensive Versuchsauswertungen. Allerdings kann unter Umständen die Genauigkeit darunter leiden. Wir wählten die in Tabelle 10.13 zusammengefassten Parameter und Werte für diese erste Stufe. Wir erhalten dadurch $2 * 3 * 2 = 12$ verschiedene Parameterkombinationen, die wir acht Mal wie-

Tabelle 10.13: Parameterisierung der Sequenzoptimierungs-GP

Parameter	Werte
Crossoverwahrscheinlichkeit	0,7, 0,9
Mutationsstärke	$1/\|w\|$, $2/\|w\|$, $4/\|w\|$
Populationsgröße	10, 20

derholten. Getestet wurde mittels der parameterisierten Heuristiken auf dem 2D-HPNX-Modell sowie dem 3D-Jernigan-Modell. Es wurden 1000 Sequenzauswertungen zugelassen. Wir verglichen die Ergebnisse paarweise mittels der in Abschnitt 10.1.2 vorgestellten R2-Metrik. Die Ergebnisse werden im ersten Teil der Tabellen 10.16 (2D-HPNX-Modell) und 10.17 (3D-Jernigan-Modell) präsentiert. Aufgeführt sind (ähnlich wie in Abschnitt 10.2):

1. Die Parametereinstellungen der Stichproben A und B,
2. Der Durchschnitt ($\bar{R}2$) und Standardabweichung (σ) der R2-Metrik,
3. Die Wahrscheinlichkeit der Nullhypothese ($\bar{R}2 = 0$, A ist im Mittel genauso gut wie B) des Wilcoxon-Tests ($p_w(H_0)$) und des t-Tests ($p_t(H_0)$),
4. Das Konfidenzintervall des t-Tests (cf_t),
5. Die Anzahl der Freiheitsgrade df= $(|A| * |B|) - 1$,
6. Die Folgerung bei Verwurf der Nullhypothese (bei $p_w(H_0) < 0,05$ und $p_t(H_0) < 0,05$).

Wir führten neben dem t-Test auch den Wilcoxon-Test durch, da nicht alle Stichproben normalverteilt waren (Lilliefors-Test, Signifikanzwert 5%). Im Gegensatz zum t-Test geht der Wilcoxon-Test nicht von einer normalverteilten Stichprobe aus (siehe Abschnitt 10.1.3). Die Untersuchungen ergaben, dass es im zweidimensionalen Fall sinnvoll zu sein scheint, eine kleine Mutationswahrscheinlichkeit zu wählen ($1/|w|$ schlägt $2/|w|$ und $4/|w|$). Im dreidimensionalen Modell führt eine Mutationswahrscheinlichkeit von $4/|w|$ zu den besten Ergebnissen. Die Crossoverwahrscheinlichkeit sollte innerhalb des 2D-HPNX-Modells recht hoch gewählt werden (0,9 schlägt 0,7). Innerhalb des 3D-Jernigan-Modells kann man mit den verwendeten Tests keine Aussage über die Crossoverwahrscheinlichkeit treffen. Eine Wahrscheinlichkeit von 0,7 scheint zwar besser zu sein, aber nicht statistisch signifikant. Im Gegensatz zu den Faltungsheuristiken zeigt sich, dass eine kleine Population (10 Individuen bei 100 Generationen) einer großen Population (20 Individuen bei 50 Generationen) in beiden Modellen signifikant überlegen ist.

Nachdem wir diese Parameter auf die ermittelten Werte festgelegt hatten, gingen wir zum zweiten Schritt über. Wir testeten die neu eingeführten Parameter der Sequenzoptimierung (Deletion, Swapping, EDIs, Archivstrategien und eine erweiterte Mutation, siehe auch Kapitel 9) auf ihre Wirksamkeit. Im Gegensatz zu den herkömmlichen Parametern ist hier keineswegs gesichert, dass sie einen positiven (oder überhaupt einen erkennbaren) Einfluss auf die Ergebnisse haben. Getestet wurden die in Tabelle 10.14 aufgeführten Einstellungen. Wir führten in dieser zweiten

Tabelle 10.14: Neue Parameter - Getestete Einstellungen

Parameter	Werte
Deletion	An,Aus
Swapping	An,Aus
EDIs	An,Aus
Archivstrategien	Uniform,Invers,AM
Mutationspoolgröße	0,8,16

Testphase kein faktorielles Design durch, sondern testeten die Parameter unabhängig voneinander. Die Ergebnisse sind im unteren Teil der Tabellen 10.16 und 10.17 dargestellt. Es zeigt sich, dass die Deletion im zweidimensionalen Modell zu signifikanten Ergebnisverbesserungen führt, während dies im dreidimensionalen Modell nicht der Fall ist. Das Swapping bringt sogar in beiden Modellen signifikante Ergebnisverbesserungen. EDIs ohne spezielle Zusatzfunktionen sind im zweidimensionalen Fall jedoch der herkömmlichen Methode unterlegen. Im dreidimensionalen Fall führen sie zu leicht besseren Ergebnissen, allerdings sind diese nicht statistisch signifikant, der Unterschied könnte also zufällig sein. Die Untersuchung der Archivstrategien zeigte, dass die inverse Auswahlstrategie die beste Wahl ist. Im zweidimensionalen Modell schneidet die uniforme Strategie signifikant schlechter ab. Die AM-Strategie wird von beiden anderen Strategien deutlichst übertroffen. Im dreidimensionalen Modell deuten die Auswertungen auf ähnliche Resultate hin, allerdings scheinen die Unterschiede hier nur sehr gering zu sein. Es wird in keinem Fall statistische Signifikanz erreicht. Der Mutationspool der Insertion scheint im 2D-HPNX-Modell keine Vorteile zu bringen. Wählt man die Poolgröße 0 (keine Insertion), sind die Ergebnisse besser, als bei größerem Pool. Zwischen Poolgröße 8 und 16 zeigen sich jedoch keine signifikanten Unterschide. Im 3D-Jernigan-Modell scheint eine Poolgröße von 8 jedoch einer Poolgröße von 0 und 16 signifikant überlegen zu sein. Zwischen Poolgrößen von 0 und 16 konnten wir keine signifikanten Unterschiede feststellen.

Aus diesen Resultaten ergaben sich die in Tabelle 10.15 aufgeführten Parametereinstellungen, die wir in Kapitel 13 verwenden.

Tabelle 10.15: Optimale Parameter der Sequenzoptimierung

Parameter	2D-HPNX	3D-Jernigan
Crossoverwahrscheinlichkeit	0,9	0,7
Mutationsstärke	$1/\|w\|$	$4/\|w\|$
Populationsgröße	10	10
Deletion	An	Aus
Swapping	An	An
EDIs	Aus	An
Archivstrategien	Invers	Invers
Mutationspoolgröße	0 (keine Insertion)	8

Tabelle 10.16: Ergebnisse der GP-Parameterisierung, 2D-HPNX-Modell (R2-Metrik)

A	B	$\bar{R}2$	σ	$p_w(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
Mutation								
$1/\|w\|$	$2/\|w\|$	0,039428	0,066646	2,1949e-55	0	[0,035139, 0,043717]	1023	$1/\|w\|$ schlägt $2/\|w\|$
$1/\|w\|$	$4/\|w\|$	0,02537	0,066601	5,9426e-25	0	[0,021084, 0,029656]	1023	$1/\|w\|$ schlägt $4/\|w\|$
$2/\|w\|$	$4/\|w\|$	-0,013249	0,073995	3,3434e-10	3,6847e-08	[-0,017933, -0,0085646]	1023	$4/\|w\|$ schlägt $2/\|w\|$
Population								
10	20	0,023548	0,071337	9,665e-70	0	[0,021264, 0,025832]	2303	10 schlägt 20
Crossover								
0,7	0,9	-0,013856	0,072569	1,1122e-16	3,2263e-18	[-0,016949, -0,010762]	2303	0,9 schlägt 0,7
Deletion								
Aus	An	-0,0068403	0,03723	0,0015428	3,9793e-05	[-0,010082, -0,0035982]	399	An schlägt Aus
Swapping								
Aus	An	-0,0050002	0,043928	0,029674	0,023344	[-0,0093182, -0,00068218]	399	An schlägt Aus
EDIs								
Aus	An	0,0096803	0,037302	1,3574e-06	3,3519e-07	[0,0060137, 0,013347]	399	Aus schlägt An
Archivstrategien								
Uni.	Inv.	-0,0039141	0,033315	0,024626	0,019273	[-0,0071889, -0,00063934]	399	Inv. schlägt Uni.
Uni.	AM	0,050091	0,041366	3,5584e-59	0	[0,046025, 0,054157]	399	Uni. schlägt AM
Inv.	AM	0,050091	0,041366	3,5584e-59	0	[0,046025, 0,054157]	399	Inv. schlägt AM
Mutationspoolgröße								
0	8	0,0071707	0,040957	0,0007902	0,00071199	[0,0030396, 0,011302]	399	0 schlägt 8
0	16	0,0091869	0,039048	2,0187e-05	6,1396e-06	[0,0052483, 0,013126]	399	0 schlägt 16
8	16	0,0018825	0,044005	0,43176	0,39274	[-0,002443, 0,006208]	399	Nicht mögl.

Tabelle 10.17: Ergebnisse der GP-Parameterisierung, 3D-Jernigan-Modell (R2-Metrik)

A	B	$\bar{R}2$	σ	$p_w(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
Mutation								
$1/\|w\|$	$2/\|w\|$	-0,0030411	0,062296	0,16273	0,11857	[-0,0068612, 0,00077902]	1023	Nicht mögl.
$1/\|w\|$	$4/\|w\|$	-0,011027	0,05678	1,147e-09	7,4777e-10	[-0,014509, -0,0075452]	1023	$4/\|w\|$ schlägt $1/\|w\|$
$2/\|w\|$	$4/\|w\|$	-0,0080938	0,051661	5,7746e-07	6,2969e-07	[-0,011262, -0,0049259]	1023	$4/\|w\|$ schlägt $2/\|w\|$
Population								
10	20	0,017879	0,055922	2,7771e-47	0	[0,015595, 0,020164]	2303	10 schlägt 20
Crossover								
0,7	0,9	0,0018988	0,057477	0,90171	0,11286	[-0,00044882, 0,0042465]	2303	Nicht mögl.
Deletion								
Aus	An	0,0066908	0,043126	0,0047201	0,0026613	[0,0023409, 0,011041]	399	Aus schlaegt An
Swapping								
Aus	An	-0,015375	0,045443	9,11e-10	4,3354e-10	[-0,020085, -0,010665]	399	An schlaegt Aus
EDIs								
Aus	An	-0,0055284	0,041818	0,106	0,0085155	[-0,0096389, -0,0014178]	399	Nicht mögl.
Archivstrategien								
Uni.	Inv.	-0,0018868	0,053473	0,14892	0,49198	[-0,0072804, 0,0035068]	399	Nicht mögl.
Uni.	AM	0,0024864	0,054125	0,89171	0,384	[-0,0031235, 0,0080964]	399	Nicht mögl.
Inv.	AM	0,0044284	0,047437	0,1419	0,085179	[-0,00061698, 0,0094737]	399	Nicht mögl.
Mutationspoolgröße								
0	8	-0,0080723	0,047782	0,0024794	0,0007997	[-0,012769, -0,0033755]	399	8 schlaegt 0
0	16	0,00243	0,048714	0,12578	0,34455	[-0,0026191, 0,0074791]	399	Nicht mögl.
8	16	0,010359	0,052295	0,00013565	0,00019943	[0,0049389, 0,01578]	399	8 schlaegt 16

Kapitel 11

Offene Fragen

> „Fragen zu stellen lohnt sich immer – wenn es sich auch nicht immer lohnt, sie zu beantworten."
>
> Oscar Wilde (1854-1900), ir. Schriftsteller

In diesem Kapitel wollen wir einige interessante Beobachtungen und daraus abgeleitete Hypothesen zum Einsatz evolutionärer Verfahren im Bereich der Peptidfaltung beschreiben. Wir haben im Rahmen dieser Diplomarbeit begonnen, diese Hypothesen zu untersuchen. Eine endgültige Untersuchung kann jedoch erst in ergänzenden Arbeiten erfolgen.

11.1 Metaevolution als Phänomen zweistufiger Optimierung

Während der Stabilitätsuntersuchungen kam die Frage auf, welche Eigenschaften eines Peptids für die Stabilität der Faltung ausschlaggebend sind. Es wurde deutlich, dass die Stabilität im wesentlichen von dem Aussehen der Energielandschaft des Faltungsraums abhängt. Eine Konformation ist sehr stabil, wenn sie – bildlich betrachtet – in einem breiten, nach unten spitz zulaufenden, tiefen Trichter liegt.

In einer so gestalteten Energielandschaft fällt es evolutionären Verfahren leicht, das globale Optimum zu finden. Dagegen liefern solche Heuristiken bei rauhen Energielandschaften häufig schlechte Ergebnisse. Da auf der Faltungsoptimierungsebene die Energie als einziges Kriterium optimiert wird, werden Sequenzen mit rauher Energielandschaft häufig zu schlecht bewertet. Dies bedeutet jedoch, dass auf der Sequenzoptimierungsebene diese Sequenzen in der Selektion seltener ausgewählt und verdrängt werden.

Auf Dauer könnte dies dazu führen, dass sich in den Populationen des sequenzoptimierenden Verfahrens vermehrt Sequenzen finden, die sich durch leicht zu optimierende Energielandschaften auszeichnen. Wie wir eingangs bereits erwähnt haben, handelt es sich bei solchen Konformationen um besonders stabile Konformationen.

Wir vermuten deshalb, dass aufgrund des zweistufigen Designs nicht nur die freie Energie, sondern implizit auch die Stabilität mitevolviert wird bzw. die besseren Individuen der Sequenzoptimierung glatte Energielandschaften im Bereich der Faltungsoptimierung haben.

Auf ein allgemeines zweistufiges Optimierungsproblem übertragen würde es bedeuten, dass der Einsatz heuristischer Verfahren sich auf das Aussehen der Energielandschaften auswirkt. In der unteren bewertenden Stufe fänden sich überwiegend leicht zu optimierende Landschaften, da in ihnen häufiger das Optimum gefunden wird. Dadurch erhöht sich die Wahrscheinlichkeit, dass die entsprechenden Individuen in der übergeordneten Stufe eine gute Fitness haben. Sie hätten somit einem Selektionsverfahren in der übergeordneten Stufe einen Vorteil.

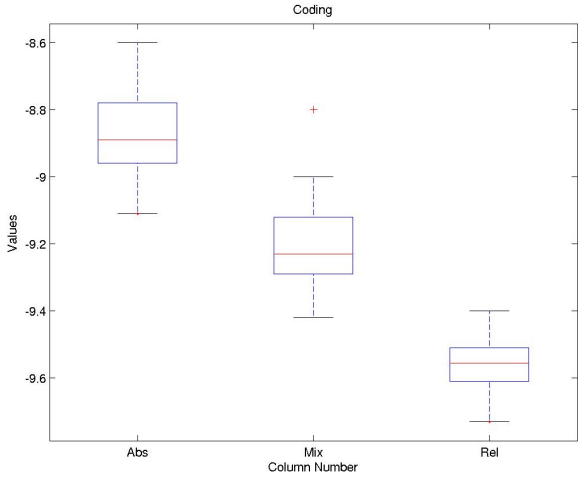
11.2 Codierungshypothese

Unsere Untersuchungen zur Codierung bei der Faltungsoptimierung im 2D-HPNX Modell haben ergeben, dass bei Simulated Annealing die absolute Codierung der relativen Codierung überlegen ist (vgl. Abbildung 10.2). Bei dem genetischen Algorithmus ist es genau umgekehrt (vgl. Abbildung 10.11). Dieser Umstand lässt sich möglicherweise dadurch erklären, dass für das Crossover die relative Codierung Vorteile bringt, während die absolute Codierung besser für die Mutation geeignet ist. SA verwendet ausschließlich Mutation als Operator, während die treibende Kraft des GAs häufig das Crossover ist.

Um diese Vermutung zu überprüfen, haben wir den GA erweitert, so dass er in einem Lauf parallel relative und absolute Codierung verwenden kann. Für die Mutation wird die absolute Codierung verwendet, während das Crossover auf der relativen Codierung arbeitet.

In einer ersten Untersuchung dieser Hypothese haben wir mit den verschiedenen Codierungen jeweils 50 Läufe mit dem GA durchgeführt. Als Parameter haben wir die in Abschnitt 10.2 ermittelten GA Parameter verwendet. Für das 2D-HPNX-Modell hat sich dabei ergeben, dass die relative Codierung der Mischcodierung eindeutig überlegen ist. Die Mischcodierung ist jedoch besser als die Absolute Codierung (vgl. Abbildung 11.1). Für das 2D-HPNX-Modell scheint damit die Codierungshypothese widerlegt zu sein.

Im Fall des Jernigan-Modell ist die absolute Codierung der Mischcodierung deutlich überlegen, während die relative Codierung am schlechtesten abschneidet (vgl. Abbildung 11.2). Nach diesen beiden kurzen Untersuchungen scheint die Codierungshypothese widerlegt zu sein. Es bleiben jedoch die Fragen, warum im 2D-HPNX-Modell beim SA die absolute Codierung der relativen Codierung überlegen ist, beim GA jedoch die relative Codierung deutlich besser ist als die absolute Codierung. Desweiteren bleibt unklar, warum beim GA im 2D-HPNX-Modell die relative Codierung die beste Codierung ist, während im Jernigan-Modell die absolute Codierung die überlegene Codierung ist.

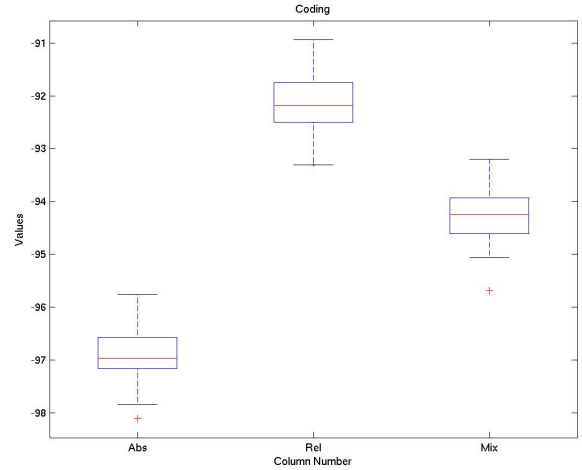


Die Vermutung, dass bei dem GA unterschiedliche Codierungen für Mutation und Crossover einen Vorteil bringen, bestätigte sich nicht. Relative Codierung setzt sich auch gegenüber der Mischcodierung durch. Allerdings ist die Mischcodierung signifikant besser als die absolute Codierung.

Wert	$\bar{e}$	σ
Abs	-8,8804	0,119879
Mix	-9,2162	0,116761
Rel	-9,5621	0,071685

A	B	$\bar{e}_A - \bar{e}_B$	$p_k(H_0)$	$p_t(H_0)$	cf_t	df	Folgerung
Abs	Mix	0,3358	1,1102e-16	0	[0,28884, 0,38276]	98	Mix schlägt Abs
Abs	Rel	0,6816	0	0	[0,64241, 0,72081]	98	Rel schlägt Abs
Mix	Rel	0,3458	0	0	[0,30735, 0,38425]	98	Rel schlägt Mix

Abbildung 11.1: Vergleich zwischen absoluter, relativer und Mischcodierung des GAs im 2D-HPNX-Modell



Auch diese Untersuchung bestätigt die Annahme, dass eine Mischcodierung besser ist als absolute Codierung und relative Codierung, nicht. Die Mischcodierung ist zwar signifikant besser als die relative Codierung, ist jedoch genauso signifikant schlechter als die absolute Codierung.

Wert	Avg.	σ
Abs	-96,9161	0,46574
Rel	-92,1463	0,53906
Mix	-94,2598	0,50074

A	B	$\bar{x}_A - \bar{x}_B$	$p_k(0)$	$p_t(0)$	cf_t	df	Folgerung
Abs	Rel	-4,7697	0	0	[-4,9696, -4,5697]	98	Abs schlaegt Rel
Abs	Mix	-2,6562	0	0	[-2,8481, -2,4643]	98	Abs schlaegt Mix
Rel	Mix	2,1135	0	0	[1,907, 2,3202]	98	Mix schlaegt Rel

Abbildung 11.2: Boxplot: Vergleich zwischen absoluter, relativer und Mischcodierung des GAs im 3D-Jernigan-Modell

Kapitel 12

Zusammenfassung

„Wir müssen das Spiel erst noch einmal Paroli ziehen lassen.“

Horst Hrubesch, Fußballtrainer

Bevor wir den Blick nach vorne richten (Kapitel 14), fassen wir im folgenden Kapitel die erzielten Ergebnisse zusammen. Das Ziel der Diplomarbeit war es, neue Methoden der computergestützten Peptidentwicklung zu entwickeln und bestehende zu verbessern. Das Ergebnis ist ein Programm, welches für unterschiedliche biochemische Anwendungen mittels computerisierter Modelle möglichst gute Peptide entwickelt. Dem Programm liegt eine Schichtarchitektur zu Grunde.

Die unterste Schicht, die Modellebene, berechnet zu einer gegebenen Proteinkonformation den zugehörigen energetischen Zustand. Hierzu wählten wir Modelle, die die biochemische Realität in die simplifizierende Sichtweise der Informatik geeignet abbilden. Die Modelle wurden um verschiedene Eingabecodierungen und eine Constraint-Handling-Methode erweitert.

Die darüberliegenden Schicht, die Faltungsoptimierung, besteht aus einer einkriteriellen Optimierungsheuristik, welche zu einer gegebenen Peptidsequenz die native Faltung errechnet. Die Modellschicht wird hierbei als eine Bewertungs-Black-Box benutzt. Die verwendeten Heuristiken sind zum einen ein genetischen Algorithmus und zum anderen ein Simulated Annealing-Verfahren. Desweiteren implementierten wir einen exakten deterministischen Algorithmus, der allerdings mit exponentieller Laufzeit verbunden ist. Der genetische Algorithmus erhielt zusätzlich eine Niching-Strategie zur Diversitätspräservation. Die implementierten Heuristiken, so konnten wir zeigen, erzielten Ergebnisse, die durchaus nah an die im Modell möglichen Optimalwerte gelangten. Eine tiefergehende Analyse erbrachte außerdem aufschlussreiche Ergebnisse hinsichtlich der einzelnen Parameter. Die Codierung und die Mutationsstärke schienen z. B. eine enorme Auswirkung auf die Qualität der Ergebnisse zu haben, Crossover und Populationsgröße scheinen weitaus weniger einflussreich zu sein. Interessanterweise war die absolute Codierung der relativen Codierung in manchen Fällen klar überlegen, in anderen genauo klar unterlegen. Anhand dieser experimentellen Resultate verfeinerten wir die Heuristiken, so dass wir sie mit hinlänglicher Genauigkeit zur Faltungsberechnung nutzen konnten. Die vermutlich native Faltung wird dann bezüglich verschiedener Kriterien bewertet. Diese sind in Kapitel 8 festgehalten. Es sollte zum Beispiel möglich sein, Peptide zu entwickeln, die bestimmte Funktionsgruppen aufweisen, eine hohe Stabilität besitzen, an einen Rezeptor binden und besonders kurz sind. Wir stellten die Ziele durch berechenbare Funktionen dar, um sie für den Computer fassbar zu gestalten. Die interessanteste hierbei zu nennende Zielfunktion ist die Stabilität. Die Heuristiken der Faltungsoptimierung bewerten nicht nur die native Konformation, sondern errechnen aus dem Verlauf der Optimierung die Wahrscheinlichkeit, mit der sich das Protein in dieser Konformation befindet. Wir stellten fest, dass die Qualität der Stabilitätsvorhersage unabhängig von der verwendeten Faltungsheuristik ungefähr gleich ist und zogen aus dieser Beobachtung Rückschlüsse über die Energielandschaft der zu faltenden Peptide (siehe Kapitel 10.5).

In der obersten Schicht, der Sequenzoptimierung, fassen wir nicht mehr die Proteinfaltung als

Optimierungsproblem auf, sondern die Findung optimaler Proteinsequenzen bezüglich oben genannter Kriterien. Unser Ansatz der Sequenzoptimierung ist ein evolutionäres Verfahren ähnlich bekannter Systeme der genetischen Programmierung. Dieses erweiterten wir zu einem multikriteriellen Algorithmus. Zusätzlich zu den gängigen Operatoren Crossover und Mutation entwickelten wir neue Operatoren auf Basis der Modularisierungsidee (EDIs). Zum einen das Swapping, mit dem versucht wird, funktionelle Einheiten des Proteins zu vertauschen und somit eine Verbesserung zu erzielen. Zum anderen die Deletion, die versucht, unnütze Module zu finden und zu entfernen. Beide Methoden führten zu Verbesserungen der Ergebnisse. Als Erweiterung der herkömmlichen Mutation führten wir einen evolvierbaren Mutationspool ein. Es sollten sich kleine Building Blocks entwickeln, die mit der Zeit zu immer effektiveren Mutationen führen sollten. Diese Erwartungen wurden in den Experimenten leider nicht erfüllt, so dass dieser Ansatz noch einer Überarbeitung bedarf. Ein häufig unterschätztes Thema bei der Entwickung multikriterieller Heuristiken ist die Bedeutung des Archivs zur Speicherung nichtdominierter Punkte. Hier entwickelten und evaluierten wir verschiedene Methoden der Selektion sowie eine adaptive Löschfunktion. Es zeigte sich, dass die Selektionsfunktion starken Einfluss auf die Ergebnisse hat. Die inverse Selektion führt im Vergleich zur herkömmlichen Selektion zu Verbesserungen, während die Alphamännchen-Selektion negativen Einfluss auf die Ergebnisse hat.

Um dem Nutzer die Möglichkeit zu geben, verschiedene Ziele unterschiedlich zu gewichten, entwickelten wir ein Priorisierungssystem, welches die Lösungen in bestimmten Teilen des Lösungsraums bevorzugt behandelt. Außerdem wird erlaubt, verschiedene Teilsequenzvorgaben durch die Sequenzsubstitution in den Optimierungsprozess einfließen zu lassen.

Abschließend wurden Hypothesen und Ideen vorgestellt, die sich während der Arbeit ergaben, aber leider im Rahmen der Diplomarbeit nicht weiterverfolgt werden konnten. Zu nennen ist hier vor allem die Idee der Metaevolution durch zweistufige Optimierung. Wir vermuten, dass die Sequenzoptimierung Einfluss auf die Energielandschaft der zu faltenden Peptide und somit auf die Qualität der Faltungsergebnisse nimmt. In Kurzform: Je länger der Sequenzoptimierungsprozess, desto genauer die Faltungsergebnisse.

Am Ende dieser Diplomarbeit steht ein fertiges Programm, das als Entwicklungshilfe für eine Vielzahl biochemischer und pharmazeutischer Anwendungen dienen kann. Außerdem hoffen wir, einige wissenschaftliche Fragen beantwortet zu haben und vielleicht die eine oder andere Anregung für weitere Forschungsarbeiten gegeben zu haben.

Kapitel 13

Ein erster praktischer Versuch

„In der Theorie gibt es keinen Unterschied zwischen
Theorie und Praxis. In der Praxis schon ...“

Chuck Reid

An dieser Stelle wollen wir als kleinen Vorgriff auf den Ausblick (siehe Kapitel 14) die Möglichkeiten des Programms an einer ersten praktischen Anwendung darstellen. Wir versuchen, ein kurzes Peptid zu entwickeln, dass die Rezeptorgegend eines natürlichen Proteins aufweist. Um für diese Aufgabenstellung eine für die Natur gültige Aussage zu treffen, fehlt uns zum einen die Möglichkeit zur experimentellen Validierung und zum anderen sind die von uns verwendeten Modelle nicht exakt genug. Für die von uns verwendete Modellwelt können wir jedoch eine gültige Aussagen treffen.

Für diesen Versuch wählten wir das Protein Thrombin, das Teil der Gerinnungskaskade[1] ist. Es ist maßgeblich beteiligt an der Bildung von Fibrin, dem Endprodukt der Gerinnungskaskade und hat überdies einen steuernden Einfluss auf die Stärke der Gerinnungsreaktion. Dadurch ist das Thrombin lebenswichtig für den menschlichen Organismus.

Thrombin ist es ein kleines Protein, was seine Untersuchung vereinfacht. Mit Hilfe des H/D-Austausches[2] wurden von Baerga-Ortiz et al. 2002 (vgl. [10]) einige Epitope des Thrombins lokalisiert.

Ziel unseres Experiments ist es, ein Peptid zu finden, das ein strukturell ähnliches Epitop aufweist. Ein solches Peptid ist interessant, da in manchen Fällen das Thrombin durch eine unerwünschte Immunreaktion angegriffen wird und so die Gerinnungskaskade empfindlich gestört werden kann. Durch ein Überangebot künstlich erzeugter Epitope könnte diese Immunreaktion geschwächt werden und die Gerinnung in voller Stärke ablaufen.

Zunächst müssen wir das Epitop (Residuen 77, 77A, 78, 79, 80 des Proteins 1A2C aus der Brookhaven Protein Data Bank) in das Jernigangitter übertragen. Die Epitopgegend ist in Abbildung 13.1 dargestellt. Hierbei kommt es beim Übertragen bereits zu gewissen Abweichungen im Vergleich zur nativer Konformation (siehe Abbildungen 13.2 und 13.3). Im eigentlichen Optimierungsschritt suchen wir ein Peptid, das eine starke Ähnlichkeit mit dem Epitop aufweist. Da es sich im Fall des Thrombins um ein lineares Epitop handelt, können wir den entsprechenden Sequenzabschnitt mit Hilfe der Sequenzsubstitution vorgeben. Durch die Sequenzsubstitution muss die Sequenz des Epitop nicht neu entwickelt werden, sondern lediglich ein Peptid, welches das Epitop in der gewünschten Konformation stabilisiert. Für die Faltungsoptimierung verwendeten wir den SA. Für die Sequenzoptimierung wählten wir die Kriterien Stabilität, Ähnlichkeit und Länge. Zwei der nichtdominierten Individuen, die in diesem GP-Lauf gefunden wurden, sind in

[1] Die Gerinnungskaskade ist ein Prozess, der die Blutgerinnung steuert.

[2] Der H/D-Austausch ist eine Methode zur Strukturanalyse, bei der gewisse an der Molekülоberfläche liegende Wasserstoffatome durch Deuterium ersetzt werden. Der Austausch wird separat für unkomplexierte Moleküle und für Molekülkomplexe durchgeführt. Im Komplex können die Wasserstoffatome im Bereich des Epitops nicht ausgetauscht werden, da sie nicht mehr an der Oberfläche liegen. Dieser Bereich und damit das Epitop lässt sich mit Hilfe der Massenspektroskopie ermitteln.

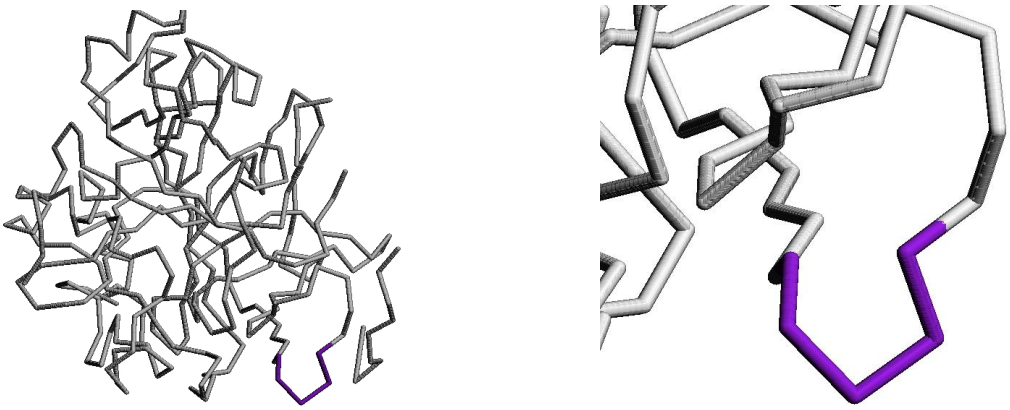


Abbildung 13.1: Das Epitop des Thrombins

Abbildung 13.2: Das Epitop in natürlicher Konformation aus zwei verschiedenen Blickwinkeln

Abbildung 13.3: Das Epitop im Jernigangitter aus zwei verschiedenen Blickwinkeln

den Abbildungen 13.4 und 13.5 dargestellt. Beide Peptide enthalten exakt das Epitop. Das erste dargestellte Peptid ist ein sehr kurzes und hat eine helixförmige Struktur. Mit einer Stabilität von 0,34395 ist es jedoch relativ instabil. Das zweite Peptid ist deutlich länger. Mit einer Stabilität von 0,97841 ist es dafür aber sehr stabil.

Mit diesem Versuch haben wir gezeigt, dass man mit unserem Programm Peptide erzeugen kann, die ein bestimmtes Epitop enthalten und dabei stabil sind. Es wurde aber auch deutlich, dass die von uns verwendeten Modelle noch nicht sehr realitätsnah sind. Bereits bei der Konvertierung des nur fünf Residuen umfassenden Epitops in unser Gitter entfernten wir uns von der in der Natur vorkommenden Anordnung der Residuen. Es ist deshalb zu befürchten, dass die von uns gefundenen Peptide in vitro nicht stabil sind oder eine gänzlich andere Konformation einnehmen.

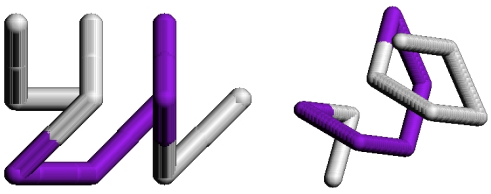


Abbildung 13.4: Ein Peptid mit der Sequenz NAERNIEFYKSS und einer Stabilität von 0,34395 aus zwei verschiedenen Blickwinkeln

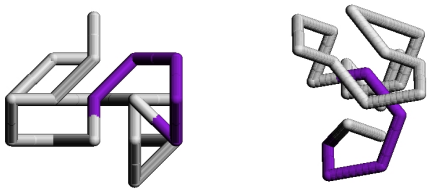


Abbildung 13.5: Ein Peptid mit der Sequenz WAERNIEQPGNFWGHWHERFLKIS und einer Stabilität von 0,97841 aus zwei verschiedenen Blickwinkeln

Kapitel 14

Ausblick

> „Das Leben ist voller Möglichkeiten, aber du kriegst nie eine.“
>
> Charlie Brown, Baseballspieler und Comicfigur

Wir befassten uns in dieser Arbeit mit den Grundlagen des in silico Peptiddesigns und implementierten und untersuchten ein Programm, mit dem man den Prozess des Sequenzdesigns und die Ermittlung der nativen Konformation der Sequenzen simulieren kann. Für die Faltungssimulation benutzen wir einfache Gittermodelle. Für die Sequenzsuche verwenden wir zufällig erzeugte Sequenzen. Dieses einfache Vorgehen legt eine Vielzahl von Verbesserungen nahe.

Durch die Weiterentwicklung unseres Programms im Hinblick auf die verwendeten Modelle können eventuell Ergebnisse erzeugt werden, die den in vivo Faltungsprozess besser imitieren und deshalb von größerer praktischen Relevanz wären. Auf den in silico errechneten Ergebnissen aufbauend könnte langfristig die Anzahl aufwendiger in vitro durchgeführter Experimente zur Bestimmung der Faltung reduziert werden. So könnte man die Suche nach neuen Wirkstoffen auf Peptidbasis erheblich kürzer und kostengünstiger gestalten. Der Aufwand der Entwicklung besserer Modelle ist gerechtfertigt, da wir gezeigt haben, dass unser algorithmischer Ansatz generell für Probleme aus dem Bereich des Peptiddesigns geeignet ist.

Die Sequenzsuche könnte z. B. verbessert werden, indem man die Sequenzen nicht zufällig erzeugt, sondern auf eine Bibliothek stabil faltender Motive zurückgreift. Mit Hilfe von in vitro-Experimenten können kurze Sequenzteile ermittelt werden, die stabil falten. Die neuen Sequenzen könnten aus diesen Motiven zusammengesetzt werden. Wir hoffen, dass so mehrheitlich Sequenzen erzeugt werden, die stabile Konformationen haben. Auf diesem Weg würde zusätzlich sichergestellt, dass neu erzeugten Sequenzen gut modularisierbar sind, da sie aus einzelnen stabilen Modulen aufgebaut sind.

Mittelfristig könnten diese oder ähnliche Methoden zur Erzeugung von Sequenzbibliotheken eingesetzt werden, die für biochemisches Screening verwendet werden können. Dabei wird in silico eine große Anzahl von Peptidsequenzen berechnet, die bestimmte Eigenschaften besitzen sollen. Diese Sequenzen können anschließend in vitro darauf untersucht werden, ob sie die erhofften Eigenschaften besitzen.

Desweiteren sind Fragestellungen denkbar, bei deren Beantwortung unser Programm unterstützend eingesetzt werden kann. So zeigt sich, dass viele Proteine, die das Produkt natürlicher Evolution sind, mehrere Funktionen haben. Dies legt den Verdacht nahe, dass es kürzere Proteine geben müsste, die nur eine einzelne dieser Funktionen besitzen. Es besteht die Hoffnung, dass es möglich ist, diese Funktionen auf andere Proteine zu übertragen, indem die funktionellen Sequenzabschnitte mit ihrer spezifischen Konformation eingebunden und stabilisiert werden. In diesem Zusammenhang gibt es eine Vielzahl von Proteinen, die Funktionen besitzen, die hohe wissenschaftliche Bedeutung haben.

Ein interessantes Beispiel ist Green Fluorescent Protein (GFP). Im Gegensatz zu anderen fluoreszierenden Proteinen schafft sich GFP in einem autokatalytischen Prozess sein Chromophor aus

den eigenen Aminosäuren selbst. Die Fluoreszenz von GFP könnte auf andere Proteine übertragen werden, die dann nicht nur ihre normale biologische Funktion ausführen würden, sondern durch ihre Fluoreszenz auch lokalisiert werden könnten. Für die medizinische und biotechnologische Forschung wären solche Abkömmlinge von GFP sehr nützliche Werkzeuge.

Für die in silico-Untersuchung dieser Aufgabenstellung ist besonders die Sequenzsubstitution interessant. Mit Hilfe der Sequenzsubstitution lassen sich funktionale Einheiten in den Prozess der Sequenzevolution einbauen. Dabei kann bereits während des Entwurfs des Proteins dank des Ähnlichkeitstests überprüft werden, ob das Protein einen Faltungsblock enthält, der dem des ursprünglichen Proteins (an dieser Stelle der fluoreszierende Teil des GFP) ähnlich ist. Die Ähnlichkeit fließt in Form einer Zielfunktion in die Sequenzevolution ein. Ein erster Versuch, der dem gleichen Versuchsaufbau folgt, ist in Kapitel 13 beschrieben.

Mit diesem kurzen Ausblick beenden wir unsere Diplomarbeit. Das von uns entwickelte Programm ist eine Basis, auf der aufbauend Untersuchungen im Bereich des Peptiddesigns durchgeführt werden können. Dieses Kapitel soll dabei nur einen ersten Eindruck vermitteln, wie unser Programm weiterentwickelt werden kann und welche interessanten und vielfältigen Möglichkeiten sich im Bereich der Biotechnologie durch den Einsatz rechnergestützter Simulation bieten ...

Anhang A

Experimentelle Methoden der Strukturbestimmung

In diesem Kapitel geben wir eine ausführlichere Beschreibung der bereits in Abschnitt 2.3 vorgestellten experimentellen Verfahren der Strukturbestimmung von Proteinen und Peptiden. Hierbei erklären wir zunächst das ältere Verfahren der Röntgenkristallographie und im Anschluss die Kernmagnetische Resonanzspektroskopie.

A.1 Röntgenkristallographie

Bei der Röntgenkristallographie handelt es sich um ein Verfahren zur Untersuchung der Tertiärstruktur von Kristallen. Für die Kristallographie werden eine Röntgenstrahlenquelle, der zu untersuchende Kristall und ein Empfänger benötigt. Der Aufbau eines Kristallographen ist in Abbildung A.1 schematisch dargestellt.

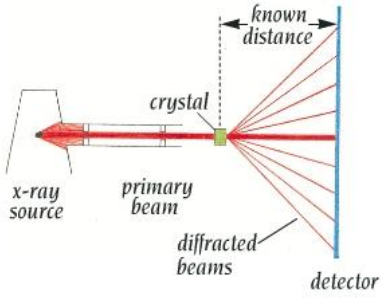


Abbildung A.1: Röntgenkristallograph (aus [17])

In ihrer Arbeitsweise sind die Kristallograhen einem herkömmlichen Lichtmikroskop sehr ähnlich. Bei einem Mikroskop wird mit einer Lichtquelle ein Objekt bestrahlt, das das Licht streut. Mit Hilfe einer optischen Linse werden die Strahlen erneut gebündelt und ergeben ein für den Betrachter sichtbares Bild. Bei der Röntgenkristallographie wird anstelle einer Lichtquelle eine Röntgenstrahlenquelle verwendet. Die Röntgenstrahlen werden durch das zu analysierende Modell gebeugt. Da es jedoch kein Äquivalent zur optischen Linse gibt, um die gestreuten Röntgenstrahlen zu bündeln, kann man sie lediglich auffangen und die dabei entstehenden Muster (siehe Abbildung A.2) auswerten. Anhand der so gewonnenen Daten kann ein Modell der Struktur des zu untersuchenden Objekts konstruiert werden.

Normales Licht hat eine Wellenlänge von 300 bis 700 nm. Da die Abstände zwischen den einzel-

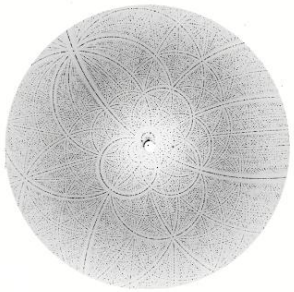


Abbildung A.2: Diffraktionsmuster (aus [17])

nen Atomen jedoch im Bereich von 1 bis 3 Å[1] liegen, sind sie lichtmikroskopischen Untersuchungen nicht mehr zugänglich. Dank ihrer dreidimensional geordneten, periodischen, gitterartigen Strukturen sind Interferenzen zu erwarten, wenn man Kristalle mit Wellen der Wellenlänge ähnlich den interatomaren Abständen bestrahlt.

Um Kristalle zu beschreiben, reicht es, wenn man die kleinste, sich wiederholende Einheit sowie die Größe des einbettenden Raumes kennt. Der einbettende Raum wird durch drei Vektoren $\vec{a}$, $\vec{b}$, $\vec{c}$, die Basisvektoren, festgelegt. Durch ihre Aneinanderreihung im Raum wird das Translationsgitter aufgespannt. Die kleinste Masche des Gitters heißt Elementarzelle. Sie wird durch die Gitterkonstanten a, b, c (die Beträge der Basisvektoren) und die Winkel zwischen den Basisvektoren α (zwischen $\vec{b}$ und $\vec{c}$), β (zwischen $\vec{a}$ und $\vec{c}$), γ (zwischen $\vec{a}$ und $\vec{b}$) festgelegt (siehe Abbildung A.3). Um die räumliche Lage eines Atoms anzugeben, wird das Translationsgitter benutzt. Als Einheit dienen die Gitterkonstanten. Das Translationsgitter ist dabei als abstraktes mathematisches Konstrukt zu verstehen, dessen Ursprung beliebig auf der Fläche der Einheitszelle bewegt werden kann.

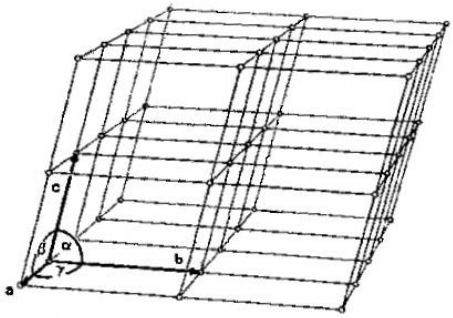


Abbildung A.3: Darstellung eines Translationsgitters (aus [95])

Bestrahlt man ein eindimensionales Gitter, z. B. ein optisches Strichgitter mit Strichabstand d mit Röntgenstrahlen, kommt es zu Interferenzerscheinungen. Diese entstehen, da von jedem Punkt des Gitters eine kugelförmige Streuwelle gleicher Wellenlänge ausgeht. In Abhängigkeit vom Betrachtungswinkel θ und dem Punktabstand entsteht dabei eine Phasendifferenz (siehe Abbildung A.4). Ist die Phasendifferenz $\Delta = n\lambda$ ein ganzzahliges Vielfaches von λ, sind die Streuwellen in Phase und es kommt zu einer amplitudenverstärkenden, konstruktiven Interferenz. Beträgt die Phasendifferenz $\lambda/2$, löschen sich benachbarte Wellen aus. Es kommt zu destruktiver Interferenz.

Um das Verständnis der Interferenzbedingugen für einen dreidimensionalen Kristall zu vereinfachen, können dank der Periodizität Kristalle in mehrere, einander überlagernde Gitter eingeteilt werden, die jeweils ein Atom aus der Einheitszelle und dessen periodische Äquivalente repräsentie-

[1] Ein Ångström entspricht einem Abstand von $10^{-10}m$.

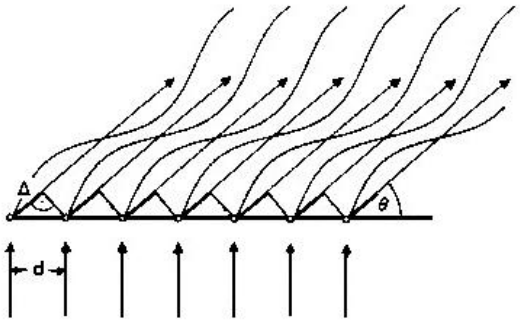


Abbildung A.4: Phasendiferenz bei Beugung im eindimensionalen Gitter (aus [95])

ren. Aus einem dieser „Einatom-Translationsgitter“ kann man zunächst eine Reihe betrachten, bei der es an den Punkten im Gitter punktförmige Streuzentren gibt. Die Phasendifferenz zwischen zwei benachbarten Punkten lässt sich dann aus dem Einfallswinkel μ und dem Ausfallswinkel ν bei bekanntem Abstand a der Streuzentren mit der Laue-Gleichung berechnen:

$$a\cos\mu_a + a\cos\nu_a = n_1\lambda$$

Es lassen sich so Ausfallswinkel berechnen, bei denen konstruktive Interferenz stattfindet. Sie liegen auf dem Kegelmantel eines so genannten Lauekegels (siehe Abbildung A.5). Wenn man nun eine zweite, nicht parallele Reihe hinzunimmt, zeigt sich, dass es auf dieser einen Punkt mit gleicher Phasendifferenz gibt. Wenn man eine Gerade durch diese beiden Punkte zieht, lässt sich der gesamte Beugungsvorgang als Spiegelung der einfallenden Strahlen an dieser Achse auffassen. Für den realen, dreidimensionalen Fall muss jedoch ein System aus drei Lauegleichungen

$$\begin{aligned} a\cos\mu_a + a\cos\nu_a &= n_1\lambda \\ b\cos\mu_b + b\cos\nu_b &= n_2\lambda \\ c\cos\mu_c + c\cos\nu_c &= n_3\lambda \end{aligned}$$

betrachtet werden. Die ausfallenden Strahlen müssen auf den Kegelmänteln dreier verschiedener Kegel liegen, um eine Reflexion beobachten zu können. Man erhält eine Reflexionsebene (Netzebene), die durch die Miller-Indizes (hkl) charakterisiert wird, wobei (hkl) die Ebene und hkl die Reflexion bezeichnet. Jede Netzebene gehört zu einer Ebenenschar paralleler Ebenen, so dass jedes Atom des Kristalls auf einer dieser Ebenen liegt. Die Ebene der Schar, die am nächsten am Ursprung liegt, ihn jedoch nicht schneidet, teilt die a-, b- und c-Achse in Achsenabschnitte $1/h$, $1/k$ und $1/l$. Die Millerindizes ergeben sich als Kehrwert dieser Achsenabschnitte. Um eine Reflexion zu erzeugen, muss der Ein- und Ausfallswinkel θ einen bestimmten Wert annehmen. Mit Hilfe der Bragg-Gleichung lässt sich dieser Winkel in Abhängigkeit vom Abstand d zwischen den Ebenen der Ebenenschar und der Wellenlänge der eingestrahlten Strahlung bestimmen.

$$2d\sin\theta = n\lambda \qquad (n = 1, 2, 3, \ldots)$$

Bisher wurden bei allen Berechnungen idealisierte Strukturen betrachtet. Es wurde nicht berücksichtigt, dass die Röntgenstrahlen von den Elektronen reflektiert werden. Die Elektronenhülle reicht jedoch in die Größenordnung der Wellenlänge der Röntgenstrahlen und man hat es deshalb nicht mit punktförmigen Streuzentren zu tun. Es kommt so zu einer Phasenverschiebung. Dieses Phänomen ist mit zunehmendem Beugungswinkel θ zunehmend stärker ausgeprägt. Zusätzlich dazu liegen die einzelnen Atome nicht fest auf ihren Gittern, sondern oszillieren um diese Positionen. Auch dies führt zu kleinen Abweichungen im Vergleich zu den erwarteten Werten. Diese Abweichungen müssen bei der Berechnung berücksichtigt werden.

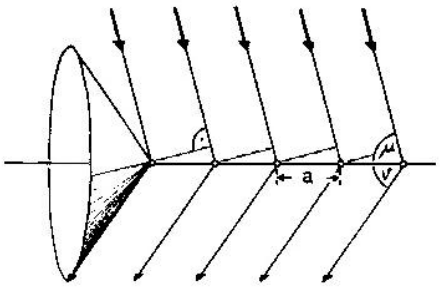


Abbildung A.5: Darstellung der Lauekegel (aus [95])

Nachdem bis jetzt nur einatomare Strukturen betrachtet wurden, müssen jetzt noch die Auswirkungen anderer Atome in der Nachbarschaft berücksichtigt werden. Für sie können Phasenverschiebungen berechnet werden, die zusammen mit den Amplituden der einzelnen reflektierten Wellen die Intensität der Reflexion bestimmen. Bei bekannten Phasenverschiebungen durch die einzelnen Atome und Kenntnis aller einzelnen Wellen kann so mit Hilfe von Fouriertransformationen die Elektronendichte des Kristalls berechnet werden und so auch dessen Tertiärstruktur. Da aber nur Intensitäten gemessen werden, kann nur noch der Betrag der Amplitude errechnet werden, die Phaseninformationen gehen verloren. Da jedoch die Sequenzen, aus denen der Kristall besteht, bekannt sind, können Modelle der Anordnung der Sequenzteile im Raum erstellt werden. Für diese werden die Reflexionsintensitäten berechnet, die für einen so aufgebauten Kristall zu erwarten sind. Bei zu großer Differenz zwischen dem gemessenen und dem errechneten Wert wird in einem iterativen Prozess das Model solange verändert, bis die errechneten Werte zu den gemessenen Werten passen. Bei der Anwendung der Röntgenkristallographie für die Untersuchung von Proteinen ist allerdings zu beachten,

- dass die Röntgenstrahlen die Kristalle schädigen können. Sie sind deshalb stark tiefzukühlen.
- Dass die Anordnung der einzelnen Moleküle im Kristall nicht in vollem Ausmaß das dynamische Faltungsverhalten der Proteine widerspiegelt,
- und dass es für einige Proteine keine Möglichkeit der Kristallzüchtung gibt [17].

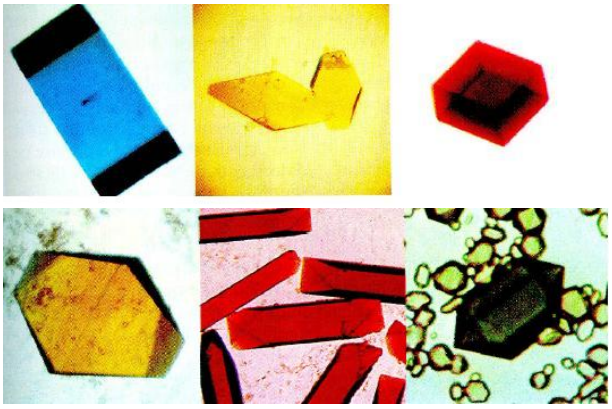


Abbildung A.6: Bilder von Proteinkristallen (aus [17])

A.2 Kernmagnetische Resonanzspektroskopie

Die NMR-Spektroskopie (Nuclear Magnetic Resonance) ist ein Verfahren, das auf dem Verhalten von Atomkernen in einem starken Magnetfeld basiert. Die Spektroskope bestehen aus einem starken Magneten, einem Sender für elektromagnetische Wellen und einem Empfänger (für eine schematische Darstellung eines NMR-Spektroskops siehe Abbildung A.7).

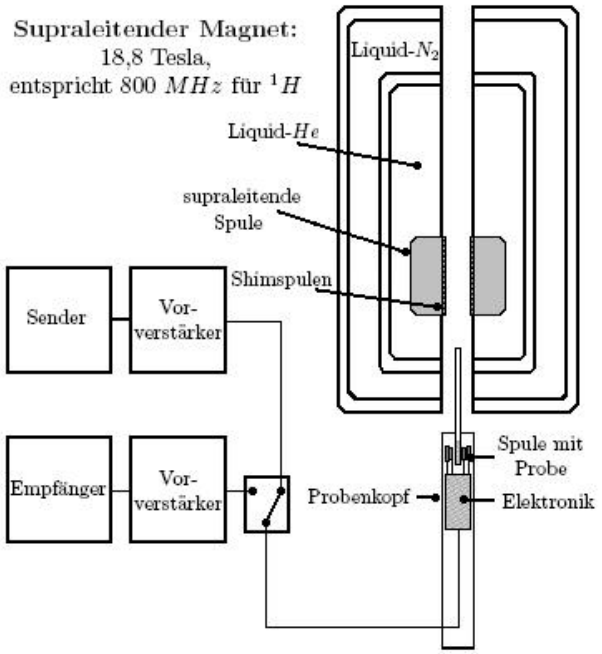


Abbildung A.7: NMR-Spektroskop

Die NMR-Spektroskopie beruht auf der Quantentheorie. Die ersten NMR-Untersuchungen wurden 1945 durchgeführt, in denen die Absorption von Radiowellen durch Paraffinwachs bzw. Wasser beobachtet wurde. Aus diesen Absorptionsmustern lassen sich bei bekannter Primärstruktur Rückschlüsse auf die Sekundär- und Tertiärstruktur treffen. Nachdem es sich bei den verwendeten Spektroskopen noch um Modelle im Experimentierstadium handelte, waren in den späten 1950ern die ersten kommerziellen Varianten erhältlich.

Weite Verbreitung fand diese Technik allerdings erst in den 1980ern, was vor allem darauf zurückzuführen ist, dass die Leistung der Spektroskope erst seit dieser Zeit einen sinnvollen Einsatz ermöglicht. Heutige Geräte haben eine ^1H-Resonanzfrequenz von bis zu 900 MHz unter Verwendung von Magneten der Stärke $21,1\ Tesla$.

Wie bereits erwähnt, wird bei dieser Art der Spektroskopie die Absorptions- bzw. Emissionsfähigkeit von elektromagnetischen Wellen durch bestimmte Atomkerne bestimmt. Diese Absorptionsfähigkeit beruht auf dem Eigendrehimpuls (Spin) um die Polarachse des Kerns und dem daraus resultierenden magnetischen Moment. Die Anordnung dieser Spinachsen ist zufällig und ungeordnet, so dass es im natürlichen Zustand keine messbare Energiedifferenz zwischen den einzelnen Spinzuständen gibt (siehe Abbildung A.8). Der Spin der Kerne wird als diskrete Größe I angegeben, wobei nur Kerne mit Spin $\neq\ 0$ Energie absorbieren bzw. emittieren. I ist dabei

- 0 bei Kernen mit gerader Neutronen- und Protonenzahl,
- ganzzahlig bei Kernen mit gerader Neutronen- und ungerader Protonenzahl,
- halbzahlig bei Kernen mit ungerader Neutronen- und Protonenzahl.

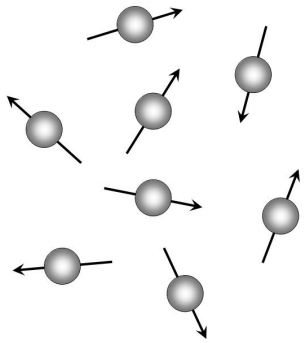


Abbildung A.8: Zufällige Verteilung der Spinachsen

Aus I ergibt sich eine bestimmte Anzahl an Spinzuständen

$$m = I, (I-1), (I-2), \ldots, -I.$$

Für die Spektroskopie sind die Kerne mit $I = 1/2$ oder $m = 2$ interessant, also diejenigen Kerne, die zwei verschiedene Zustände bzw. Energielevel einnehmen können. Ihr magnetisches Moment μ ist dann

$$\vec{\mu} = \frac{\gamma I h}{2\pi},$$

wobei h die Plancksche Konstante und γ das kernspezifische gyromagnetische Verhältnis ist. Letzteres hat (wie in Abbildung A.9 dargestellt) Einfluss auf die Spinrichtung.

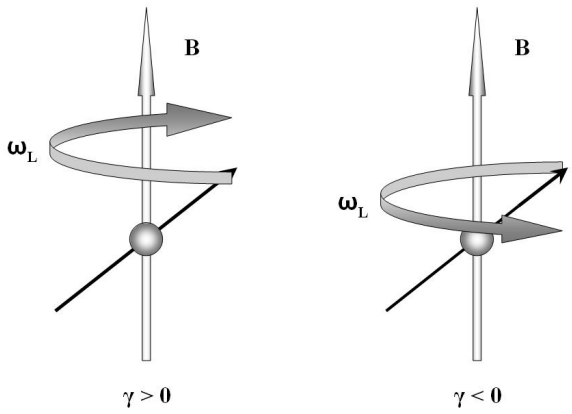


Abbildung A.9: Spinrichtung in Abhängigkeit vom gyromagnetischen Verhältnis in Lamorfrequenz

Die Frequenz des Spins bzw. die Präzision oder Lamorfrequenz ω_L (Darstellung der Rotation mit ω_L in Abbildung A.9) hängt vom gyromagnetischen Verhältnis und der Flussdichte ab:

$$\omega_L = \gamma B_0$$

Die Kenndaten einiger in der NMR häufig verwendeter Atome kann man der Tabelle A.1 entnehmen. In einem starken Magnetfeld ordnen sich die Spinachsen parallel bzw. antiparallel zu den Feldlinien an. Hierbei treffen zwei Kräfte aufeinander. Zum einen hat der Kern Spin um seine Spinachse, was ihn einem Kreisel gleich stabilisiert, zum anderen neigt der Kern dazu, sich nach der Orientierung der Feldlinien auszurichten, der Kreisel will also der Gravitation folgend umfallen. Das Gegeneinanderwirken der Kräfte ist in Abbildung A.10 dargestellt. Die Energiedifferenz ΔE zwischen zwei benachbarten Energiezuständen ist deshalb von der Flussdichte B_0 des

magnetischen Felds und dem Moment γ abhängig (siehe Abbildung A.11).

$$\Delta E = h\nu \quad mit \quad \nu = \frac{\gamma B_0}{2\pi} \quad und \quad v = \frac{\gamma B_0}{2\pi}$$

Tabelle A.1: Wichtige Atome mit Spin, gyromagnetischem Verhältnis und Resonanzfrequenz

Kernisotop	Spin I	gyromagnetisches Verhältnis $10^7 MHz/T$	NMR Frequenz in MHz magnetische Flussdichte $B_0 = 2,349T$
^1H	1/2	26,7519	100
^2H	1	4,1066	15,351
^3H	1/2	28,5350	106,664
^{12}C	0	-	-
^{13}C	1/2	6,7283	25,144
^{14}N	1	1,9338	7,224
^{15}N	1/2	-2,7126	10,133
^{31}P	1/2	10,8394	40,481

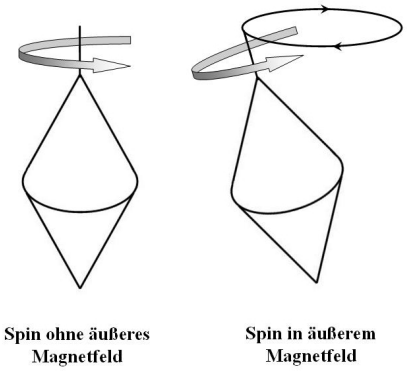


Abbildung A.10: Darstellung der aus Spin und Magnetfeld resultierenden Kräfte

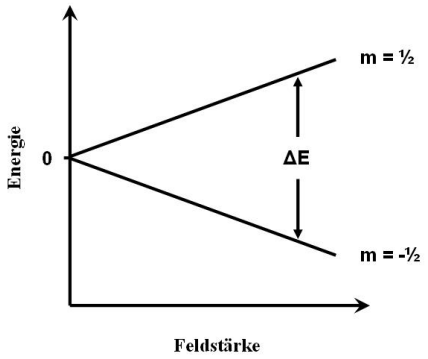


Abbildung A.11: Abhängigkeit der Energiedifferenz zwischen Energiezuständen von Moment und magnetischer Flussdichte

Wenn man die Kerne mit elektromagnetischen Wellen bestrahlt, können sie im Falle einer Resonanz der Frequenzen Energie aufnehmen und so ihren Zustand ändern. Auch bei leichter Frequenzabweichung von der Resonanzfrequenz kann noch Energie absorbiert werden. Bei Kernen mit zwei Zuständen würde die Zustandsänderung zu einem Wechsel der Spinrichtung führen, die Spinachse würde also um 180 Grad gedreht. Bei magnetischen Feldern ab 2 $Tesla$ muss die Wellenlänge dabei zwischen Zentimetern und Metern liegen, es werden Radiowellen benutzt.

Da die Frequenz des Spins vom gyromagnetischen Verhältnis abhängt, haben die einzelnen Atome in einem Magnetfeld fester Stärke verschiedene Resonanzfrequenzen (siehe Tabelle A.1). Man kann also immer nur eine Art Atome analysieren. Da die Umgebung der einzelnen Atome, also benachbarte Moleküle und Bindungen (z. B. H-Brücken), sich auf die Anordung der Elektronen auswirken, kann es hier durch Abschirmung zu einer lokalen Beeinflussung des Magnetfeldes kommen. Die Resonanzfrequenz des zu stimulierenden Atoms verschiebt sich leicht. Dieser Effekt nennt sich chemische Verschiebung. Desweiteren wird das Magnetfeld noch durch das Phänomen der Spinkopplung beeinflusst. Der Kernspin einzelner Atome wird durch den Kernspin benachbarter, chemisch gebundener Atome beeinflusst. Das Feld kann dadurch sowohl gestärkt als auch

Tabelle A.2: Typische Frequenzen und Wellenlängen elektromagnetischer Wellen

Wellentyp	Typische Frequenz in Hz	Typische Wellenlängen
Radio	10^8	3 m
Mikrowellen	10^{10}	3 cm
Infrarot	3 x 10^{13}	10 μm
Ultraviolett	3 x 10^{16}	10 nm
Röntgenstrahlen	3 x 10^{18}	0,1 nm
γ-Strahlen	3 x 10^{20}	10 pm

geschwächt werden. Es gibt eine Reihe möglicher Konstellationen, abhängig davon, in welchen Zuständen sich die gekoppelten Kerne gerade befinden und mit wie vielen Atomen sie gekoppelt sind. Die Kopplungskonstante J (in Hz) gibt an, wie breit das Frequenzintervall ist, in dem die unterschiedlichen Konstellationen liegen. J wird dabei nicht durch die Feldstärke des Magnetfelds beeinflusst. Der dritte Effekt, der sich auf das einwirkende Magnetfeld auswirkt, ist der Nuklear-Overhauser-Effekt (NOE). Im Gegensatz zur Spinkopplung wirkt sich der NOE nicht über chemische Bindungen, sondern über die räumliche Distanz aus. Der Kern eines Atoms reagiert dabei mit dipolaren Wechselwirkungen auf die Magnetisierungsänderung benachbarter Kerne. Dieser Effekt nimmt mit zunehmender Distanz in der sechsten Potenz ab und wirkt sich deshalb nur bei Abständen kleiner als 5 nm aus.

Die hierbei absorbierte Energie wird im Anschluss an die Anregung durch Wellen wieder abgegeben. Bei niederfrequenten Wellen (Radiowellen, Mikrowellen) findet jedoch fast keine spontane Emission statt. Durch Relaxation gelangen die angeregten Kerne wieder in ihre Ausgangssituation. Zum einen gibt es die Spin-Gitter-Relaxation, die in Richtung der Feldlinien erfolgt und durch Abgabe von Energie in Form von Wärme an das umliegende Gitter (z. B. Lösungsmittel) erfolgt. Zum anderen gibt es die Spin-Spin-Relaxation, die senkrecht zu den Feldlinien erfolgt. Bei ihr gibt ein angeregter Kern seine Energie an einen nicht angeregten Kern ab. Diese Mechanismen laufen jedoch sehr langsam ab.

Die bestrahlten Atome reagieren deshalb messbar anders, als bei der eingestellten Frequenz erwartet wird. Diese Abweichung lässt einen Rückschluss auf die Umgebung des betrachteten Atoms zu. Da die Abweichung jedoch nur minimal ist (einige hundert Hz), die absoluten Werte jedoch sehr groß sind (einige hundert MHZ), wird eine relative Skala benutzt. Als Referenzwert dient hierbei Tetramethylsilan (TMS $C_4H_{12}Si$). Es wird verwendet, da es gut löslich und sehr edel ist. Zudem enthält es viel ^1H und ^{13}C.

In der modernen NMR-Spektroskopie werden alle Kerne einer bestimmten Art (z. B. ^1H) durch einen kurzen Radiofrequenz-Impuls gleichzeitig angeregt. Bei einer Pulsdauer τ_r entsteht dabei ein kontinuierliches Anregungsfrequenzband der Breite $1/\tau_r$, das das gesamte Spektrum der chemischen Verschiebung abdeckt. Die angeregten Kerne geben daraufhin einen schwachen Wechselstrom ab, dessen Frequenz gemessen werden kann. Mit Hilfe der Fourier-Transformation wird dieses Signal dann in ein NMR-Spektrum transformiert. Ein Beispiel für ein dabei entstehendes NMR-Spektrum ist in Abbildung A.12 zu sehen. Wenn alle angeregten Kerne die gleiche chemische Verschiebung haben, so ergibt sich ein einziges, sinusförmiges Signal, im Falle verschiedener chemischer Verschiebungen überlagern sich mehrere Signale.

In einem zweidimensionalen Experiment werden alle ^1H Kerne gleichzeitig angeregt, um dann nach dem Magnetisierungstransfer zwischen zwei Kernen zu suchen. Hierbei kann nach Paaren, die entweder über NOE oder eine Spin-Spin-Kopplung Magnetisierung austauschen, gesucht werden. Experimente zur NOE Datenerhebung nennt man NOESY-Experimente, während Kopplungsexperimente COSY-Experimente genannt werden.

Bei großen Biopolymeren tritt jedoch das Problem auf, dass sich auch im zweidimensionalen Fall viele Signale überlagern und deshalb eine Auswertung verhindern. Dem kann man entgegenwirken, wenn man zusätzlich zu den ^1H Atomen auch die ^{13}C und ^{15}N Atome mituntersucht. Da ^{13}C und ^{15}N jedoch nur ein geringes natürliches Vorkommen haben, müssen zunächst Iso-

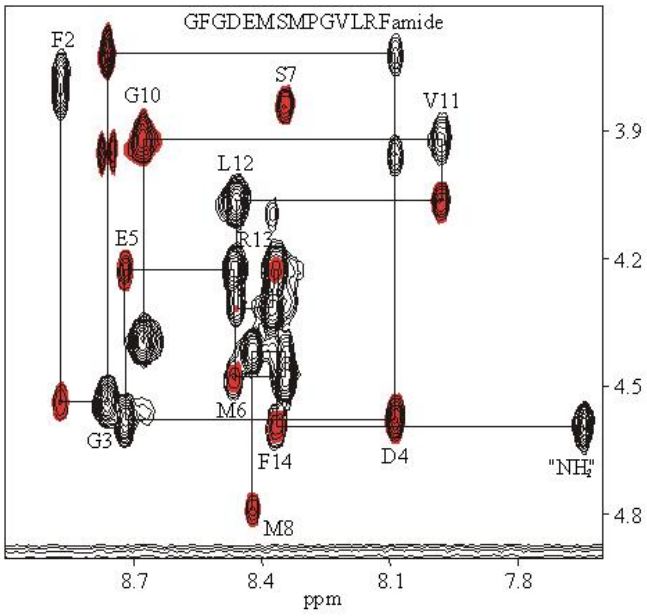


Abbildung A.12: Ein zweidimensionales NMR-Spektrum (aus [7])

tope gezüchtet werden, die eine bestenfalls hundertprozentige Sättigung mit den entsprechenden Atomen aufweisen. Dies erreicht man, wenn man die Moleküle z. B. in Bakterien züchtet, die auf einem ^{13}C-^{15}N Nährboden leben. Die Gewichtsgrenze für zu untersuchenden Molküle liegt dabei bei 200 kDalton[2], wobei bei diesen Versuchen die Kenntnis der Sekundärstruktur vorausgesetzt wird.

Die NMR-Spektroskopie ist derzeit das einzige Verfahren, um die Struktur von Biopolymeren in nicht kristallinem Zustand mit atomarer Auflösung zu bestimmen. Es ist dabei jedoch notwendig, hoch konzentrierte Proben zu verwenden. Abbildung A.13 zeigt ein durch NMR-Spektroskopie errechnetes Model eines Biopolymers.

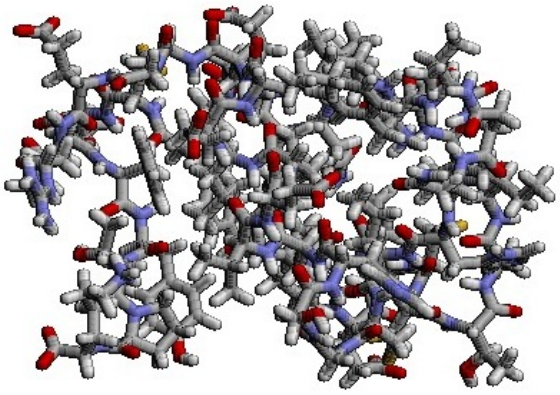


Abbildung A.13: Aus NMR-Spektrum errechnetes Modell des Insulins

[2]Ein Dalton entspricht der Masse eines Wasserstoffatoms: $3,32 * 10^{-24} g$.

Anhang B

Atomare Potentialfelder

Die in Wissenschaft und Industrie am häufigsten verwendeten Potentialfelder sollen im Weiteren vorgestellt werden. Eine gute Übersicht über die Modelle findet sich bei [71].

B.1 AMBER 4.1

Das AMBER-Potential (Cornell et al. 1995, [21]) wurde ursprünglich an einer geringen Anzahl organischer Moleküle parameterisiert und wird heute sehr häufig bei der Faltungsberechung verwendet. Das AMBER-Modell ist ein Klasse-I-Feld von niedriger Komplexität, da es z. B. nicht polare H-Atome nicht explizit modelliert. Es ist daher in der Berechnung signifikant schneller als andere Modelle. Allerdings ist die Genauigkeit im Vergleich zu den anderen Modellen ebenfalls geringer. In der Energieberechnung unterscheidet es sich zum folgenden CHARMM-Feld außer in den Konstanten nur in einem zusätzlichen Term für die H-Brücken. AMBER lässt deren Anziehungskraft bei hoher Entfernung stärker abfallen, als die normale van der Waals-Kraft.

$$E_{Bindungslänge} = \sum_{Bindungen} \frac{1}{2} k^b_{ij} (r_{ij} - b^0_{ij})^2$$

$$E_{Bindungswinkel} = \sum_{Bindungswinkel} \frac{1}{2} k^\theta_{ijk} (\theta_{ijk} - \theta^0_{ijk})^2$$

$$E_{Dihedral} = \sum_{Dihedralwinkel} k^\phi (1 + \cos(n(\phi - \phi^0)))$$

$$E_{vanderWaals} = \sum_{i,j} \frac{A_{ij}}{r^{12}_{ij}} - \frac{B_{ij}}{r^6_{ij}}$$

$$E_{elektrostatisch} = \sum_{i<j} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$$

$$E_{H-Bindung} = \sum_{\text{H}-\text{Bindungen}} \left[\frac{C_{ij}}{r^{12}_{ij}} - \frac{D_{ij}}{r^{10}_{ij}} \right]$$

ϵ_0 sei hier die elektrostatische Konstante.

B.2 CHARMM

Das CHARMM-Feld (Chemistry at HARvard Macromolecular Mechanics) von Mackerel und Karplus (Mackerel et al. 1995, [74]) ist wie AMBER ein Klasse-I-Feld. CHARMM wurde experimentell parameterisiert und war lange Zeit das meistverwendete Modell der MD, bis es von der

MMFF-Serie abgelöst wurde. Wasserstoffbrücken werden im Gegensatz zu AMBER nicht explizit modelliert, sondern sie gehen implizit durch die Lennard-Jones-Faktoren A_{ij} und B_{ij} in die Energiefunktion mit ein.

$$E_{Bindungslänge} = \sum_{Bindungen} \frac{1}{2} k_{ij}^{b} (r_{ij} - b_{ij}^{0})^2$$

$$E_{Bindungswinkel} = \sum_{Bindungswinkel} \frac{1}{2} k_{ijk}^{\theta} (\theta_{ijk} - \theta_{ijk}^{0})^2$$

$$E_{Dihedral} = \sum_{Dihedralwinkel} |k^{\phi}| - k^{\phi}(\cos(n\phi))$$

$$E_{vanderWaals} = \sum_{i,j} \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}$$

$$E_{elektrostatisch} = \sum_{i<j} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$$

B.3 MMFF VII

MMFF (Merck Molecular Force Field) ist ein hochkomplexes Klasse-II-Feld und daher deutlich genauer als AMBER und CHARMM. Entwickelt wurde es von Thomas Halgreen [49] und ist heute das populärste verwendete Feld.

$$E_{Bindungslänge} = \sum_{Bindungen} \frac{1}{2} k_{ij}^{b1} (r_{ij} - b_{ij}^{0})^2 (1 + k_{ij}^{b2}(r_{ij} - b_{ij}^{0}) + \frac{7}{12}(k_{ij}^{b2}(r_{ij} - b_{ij}^{0})^2))$$

$$E_{Bindungswinkel} = \sum_{Bindungswinkel} \frac{1}{2} k_{ijk}^{\theta} (\theta_{ijk} - \theta_{ijk}^{0})^2 * (1 + k_{ijk}^{\theta}(\theta_{ijk} - \theta_{ijk}^{0}))$$

$$E_{Dihedral} = \sum_{Dihedralwinkel} k_1^{\phi}(1 + \cos\phi) + k_2^{\phi}(1 + \cos 2\phi) + k_3^{\phi}(1 + \cos 3\phi)$$

$$E_{Bindungswinkel+Bindungslänge} = \sum_{Bindungswinkel} (k_{ijk}^{b\theta}(r_{ij} - b_{ij}^{0}) + k_{kji}^{b\theta}(r_{kj} - b_{kj}^{0}))(\theta_{ijk} - \theta_{ijk}^{0})$$

$$E_{vanderWaals} = \sum_{i,j} \frac{1,07 B_{ij}}{r_{ij} + 0,07 B_{ij}}^{7} - \frac{1,12 B_{ij}^{7}}{r_{ij}^{7} + 0,07 B_{ij}^{7}}$$

$$E_{elektrostatisch} = \sum_{i<j} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$$

B.4 MM

Die MM-Familie MM2, MM3 (Allinger et al. 1989, [2, 4]) und MM4 (Allinger et al. 1996, [3]) ist momentan der „State of the Art“ der molekularen Dynamik. Zur Berechnung von kleinen Molekülen wird es häufig genutzt. Es beschreibt das Verhalten der Proteine weitaus genauer als die harmonischen Modelle AMBER und CHRAMM. Die Bindungslängenenergie wird durch eine Approximation des Hook'schen Gesetz berechnet. Im Gegensatz zu fast allen anderen Modellen werden die Ladungen hier nicht als punktförmig aufgefasst, sondern als eine Menge von Dipolen. Dies kann häufig zu weitaus besseren Ergebnissen führen, als für andere Modelle erreichbar.

$$E_{Bindungslänge} = \sum_{Bindungen} \frac{1}{2} k_{ij}^{b} (r_{ij} - b_{ij}^{0})^2 (1 - 2(r_{ij} - b_{ij}^{0}))$$

$$E_{Bindungswinkel} = \sum_{Bindungswinkel} \frac{1}{2} k_{ijk}^{\theta} (\theta_{ijk} - \theta_{ijk}^{0})^2 (1 - 7 * 10^{-8} (\theta_{ijk} - \theta_{ijk}^{0})^4)$$

$$E_{Dihedral} = \sum_{Dihedralwinkel} k_1^{\phi}(1 + \cos\phi) + k_2^{\phi}(1 - \cos 2\phi) + k_3^{\phi}(1 + \cos 3\phi)$$

$$E_{Torsion+Bindungslänge} = \sum_{Torsionswinkel} k^{b\theta} (\theta_{ijk} - \theta_{ijk}^{0})(r_{ij} - b_{ij}^{0}) + (r_{jk} - b_{jk}^{0})$$

$$E_{vanderWaals} = \sum_{i,j} A_{ij} (2,9 * 10^5 e^{-12,5 \frac{r}{R_{vdW}}} - 2,25 (\frac{R_{vdW}}{r}^6))$$

$$E_{elektrostatisch} = \sum_{i<j} \frac{(\mu_i \mu_j)}{\epsilon_0 r_{ij}^3} (\cos\xi - 3\cos\alpha_i \cos\alpha_j)$$

Anhang C

Parameter

Einstellbare Parameter für die Sequenzoptimierung

Generations [Typ=Int, Range=1–∞, Default=100] Zahl der Generationen

PopulationSize [Typ=Int, Range=1–∞, Default=20] Populationsgröße

pMut - [Typ=Double, Range=0–∞, Default=1.0] Mutationswahrscheinlichkeit in 1/Sequenzlänge

pDel [Typ=Double, Range=0–1, Default=0.1] Wahrscheinlichkeit der Deletion (nur bei EDI=1)

pSwap [Typ=Double, Range=0–1, Default=0.9] Wahrscheinlichkeit des Swapings (nur bei EDI=1)

pCross [Typ=Double, Range=0–1, Default=0.9] Crossoverwahrscheinlichkeit

Substitution [Typ=String, Default ""] Vorgegebene Teilsequenzen, mehrere Teilsequenzen werden durch Kommata getrennt. Falls keine Eingabe erfolgt, werden keine Substitutionen verwendet.

MinLength [Typ=Int, Range=3–(Maxlength-1), Default=3] Minimale Sequenzlänge

MaxLength [Typ=Int, Range=4–∞, Default=3] Maximale Sequenzlänge

RepSize [Typ=Int, Range=1–∞, Default=100] Größe des Archivs

EDI [Typ=Int, Range=0–1, Default=1] Verwendung von EDIs: 0 - Aus, 1 - An

Mutation pool [Typ=Int, Range=0–∞, Defaults=20] Anzahl zusätzlich zu den Standardmutationen verwendeter Mutationsindividuen

Elitism [Typ=Double, Range=0–1, Defaults=0.05] Anteil elitärer Individuen in der Population

Repository selection [Typ=Int, Range=0–2, Default=1] Selektionsfunktion des Archivs: 0 - Uniform, 1 - Fitnessinvers, 2 - AM-Selektion

Printout [Typ=Int, Range=0–2, Default=1] Ausgabe des Algorithmus: 0 - Nur die aktuelle Generation, 1 - Nur das Archiv, 2 - Beides

Objective importance [Typ=Int[], Range=0–∞, Default=1,1] Prioritäten der einzelnen Zielfunktionen in der Form „P_1, P_2, P_3, ..."

Dir [Typ=String, Default=""] Ausgabeverzeichnis für die .pdb-Dateien. Falls leer, erfolgt keine Ausgabe.

Einstellbare Parameter für den genetischen Algorithmus

Individuals [Typ=Int, Range=1–∞, Default=20] Populationsgröße

Niching [Typ=Double, Range=0–∞, Default=0.5] Grad der Auswirkung des Nichings

Iterations [Typ=Int, Range=1–∞, Default=100] Zahl der Generationen

Selection method [Typ=Int, Range=1–2, Default=1] Selektionsschema: 1 - Turnierselektion, 2 - Rouletteradselektion

Tournament size [Typ=Int, Range=1–∞, Default=2] Turniergröße (nur bedeutsam bei Turnierselektion)

Crossover rate [Typ=Double, Range=0–1, Default=0.9] Crossoverwahrscheinlichkeit

Mutation rate - [Typ=Double, Range=0–∞, Default=2.0] Mutationswahrscheinlichkeit von (Mutation rate)/(Sequenzlänge)

Sequence [Typ=String, Default=""] zu faltende Sequenz (wird von der sequenzoptimierenden Stufe automatisch gesetzt)

Model [Typ=Int, Range=1–6, Default=2] Verwendete Eingabecodierung und Modell: 1 - 2D absolut codiert, 2 - 2D relativ codiert, 3 - 3D absolut codiert, 4 - 3D relativ codiert, 5 - 2D wobei Crossover in relativer Codierung und Mutation in absoluter Codierung durchgeführt wird, 5 - 3D wobei Crossover in relativer Codierung und Mutation in absoluter Codierung durchgeführt wird

Alphabet [Typ=Int, Range=1–2, Default=1] Verwendetes Alphabet und Energiefunktion: 1 - HPNX, 2 - Jernigan

Dock file [Typ=String, Default=""] Rezeptorfile, nur von Bedeutung, wenn als Zielfunktion „Affinität“ oder „Spezifität“ gewählt ist.

Specifity file [Typ=String, Default=""] Antirezeptorfile, nur von Bedeutung, wenn als Zielfunktion „Spezifität“ gewählt ist.

Similarity file [Typ=String, Default=""] Epitopfile, nur von Bedeutung, wenn als Zielfunktion „harte Ähnlichkeit“, „tolerante Ähnlichkeit“ oder „dreidimensionale Ähnlichkeit“ gewählt ist.

Objective functions [Typ=Int, Range=0–7, Default=0,1] Verwendete Zielfunktionen: 0 - Energiefunktion, 1 - Stabilität, 2 - Affinität, 3 - Spezifität, 4 - harte Ähnlichkeit, 5 - tolerante Ähnlichkeit, 6 - dreidimensionale Ähnlichkeit, 7 - Sequenzlänge

Einstellbare Parameter für das Simulated Annealing

Rounds [Typ=Int, Range=1–∞, Default=100] Zahl der Iterationen

Round length [Typ=Int, Range=1–∞, Default=10] Zahl der Faltungsauswertungen pro Iteration

Start probability [Typ=Double, Range=0–1, Default=0.95] Startwert für Akzeptanzwahrscheinlichkeit

Min probability - [Typ=Double, Range=0–1, Default=0.0001] Minimale Akzeptanzwahrscheinlichkeit

Mutation rate - [Typ=Double, Range=0–∞, Default=2.0] Mutationswahrscheinlichkeit in 1/Sequenzlänge

Sequence [Typ=String, Default=""] zu faltende Sequenz (wird von der sequenzoptimierenden Stufe automatisch gesetzt)

Model [Typ=Int, Range=1–4, Default=2] Verwendete Eingabecodierung und Modell: 1 - 2D absolut codiert, 2 - 2D relativ codiert, 3 - 3D absolut codiert, 4 - 3D relativ codiert

Cooling algorithm [Typ=Int, Range=1–7, Default=1] Abkühlschema: 1 - linear, 2 - exponentiell, 3 - hyperbolisch, 4 - logarithmisch, 5 - sigmoid (Typ A), 6 - sigmoid (Typ B), 7 - Monte Carlo

Alphabet [Typ=Int, Range=1–2, Default=1] Verwendetes Alphabet und Energiefunktion: 1 - HPNX, 2 - Jernigan

Dock file [Typ=String, Default=""] Rezeptorfile, nur von Bedeutung, wenn als Zielfunktion „Affinität“ oder „Spezifität“ gewählt ist.

Specifity file [Typ=String, Default=""] Antirezeptorfile, nur von Bedeutung, wenn als Zielfunktion „Spezifität“ gewählt ist.

Similarity file [Typ=String, Default=""] Epitopfile, nur von Bedeutung, wenn als Zielfunktion „harte Ähnlichkeit“, „tolerante Ähnlichkeit“ oder „dreidimensionale Ähnlichkeit “ gewählt ist.

Objective functions [Typ=Int, Range=0–7, Default=0,1] Verwendete Zielfunktionen: 0 - Energiefunktion, 1 - Stabilität, 2 - Affinität, 3 - Spezifität, 4 - harte Ähnlichkeit, 5 - tolerante Ähnlichkeit, 6 - dreidimensionale Ähnlichkeit, 7 - Sequenzlänge

Einstellbare Parameter für den Enumerator

Model [Typ=Int, Range=1–4, Default=2] Verwendete Eingabecodierung und Modell: 1 - 2D absolut codiert, 2 - 2D relativ codiert, 3 - 3D absolut codiert, 4 - 3D relativ codiert

Alphabet [Typ=Int, Range=1–2, Default=1] Verwendetes Alphabet und Energiefunktion: 1 - HPNX, 2 - Jernigan

Dock file [Typ=String, Default=""] Rezeptorfile, nur von Bedeutung wenn als Zielfunktion „Affinität“ oder „Spezifität“ gewählt ist.

Specifity file [Typ=String, Default=""] Antirezeptorfile, nur von Bedeutung wenn als Zielfunktion „Spezifität“ gewählt ist.

Similarity file [Typ=String, Default=""] Epitopfile, nur von Bedeutung wenn als Zielfunktion „harte Ähnlichkeit“, „tolerante Ähnlichkeit“ oder „dreidimensionale Ähnlichkeit“ gewählt ist

Objective functions [Typ=Int, Range=0–7, Default=0,1] Verwendete Zielfunktionen: 0 - Energiefunktion, 1 - Stabilität, 2 - Affinität, 3 - Spezifität, 4 - harte Ähnlichkeit, 5 - tolerante Ähnlichkeit, 6 - dreidimensionale Ähnlichkeit, 7 - Sequenzlänge

Literaturverzeichnis

[1] Akira Oyama. *Wing Design Using Evolutionary Algorithms*. PhD thesis, Tohoku University, Tohoku, Japan, 2000.

[2] N. L. Allinger. Conformational Analysis 130. MM2. A Hydrocarbon Force Field Utilizing V1 and V2 Torsional Terms. *J. Am. Chem. Soc.*, 99:8127–8134, 1977.

[3] N. L. Allinger and Y. Fan. Molecular Mechanics. The MM3 Force Field for Hydrocarbons. *J. Comp. Chem.*, 18:1827–1847, 1997.

[4] N. L. Allinger, Y. H. Yuh, and J-H. Lii. Molecular mechanics. The MM3 Force Field for hydrocarbons. *J. Am. Chem. Soc.*, 111:8551–8565, 1989.

[5] C. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[6] Arnold Neumaier. Molecular Modeling of Proteins and Mathematical Prediction of Protein Structure. *SIAM Review*, 39(3):407–460, 1997.

[7] Arthur S. Edison. Skriptum Molecular Structure and Dynamics by NMR Spectroscopy. http://ascaris.health.ufl.edu/~art/, 2002.

[8] Astro Teller. Evolving Programmers: The Co-evolution of Intelligent recombination operators. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, pages 45–68. MIT Press, Cambridge, 1996.

[9] Astro Teller and Manuela Veloso. PADO: A New Learning Architecture for Object Recognition. In Katsushi Ikeuchi and Manuela Veloso, editors, *Symbolic Visual Learning*, pages 81–116. Oxford University Press, 1996.

[10] Abel Baerga Ortiz, Carrie Hughes, Jeffrey Mandell, and Elizabeth Komives. Epitope mapping of a monoclonal antibody against human thrombin by H/D-exchange mass spectrometry reveals selection of a diverse sequence in a highly conserved protein. *Protein Science*, 11:1300–1308, 2002.

[11] I. Bahar, A. R. Atilgan, and B. Erman. Direct evaluation of thermal fluctuations in proteins using a single-parameter harmonic potential. *Fold Des.*, 2(3):173–181, 1997.

[12] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank Francone. *Genetic Programming - An Introduction; On Automatic Evolution of Computer Programs and its Applications.* dpunkt-Verlag, Heidelberg. Morgan Kaufmann, San Francisco, 1998.

[13] Thomas Bartz Beielstein, Philipp Limbourg, Jörg Mehnen, and Karlheinz Schmitt. Particle swarm optimizers for pareto optimization with enhanced archiving techniques. Interner Bericht des Sonderforschungsbereichs 531 *Computational Intelligence CI–153/03, University of Dortmund, 2003.*

[14] D.E. Bell, R.L. Keeney, and H. Raiffa. Conflicting objectives in decision International Series on Applied Systems Analysis 1. *Wiley, Chichester, 1977.*

[15] *S.D. Black and D.R. Mould. Development of hydrophobicity parameters to analyze proteins which bear post- or cotranslational modifications.* Anal. Biochem., *193:72–82, 1991.*

[16] *B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, Minimization, and Dynamics Calculations.* J. Computational Chemistry, *4(2):187 – 217, 1983.*

[17] *Carl Branden and John Tooze.* Introduction to Protein Structure. *Garland Publishing, New York, 2 edition, 1999.*

[18] *Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont.* Evolutionary Algorithms for Solving Multi-Objective Problems. *GENA - Genetic Algorithms and Evolutionary Computation. Kluwer Academic/Plenum Publishers, New York, 1 edition, 2002.*

[19] *Clare Sansom. School of Crystallography of Birkbeck Uuniversity of London.* http://www.cryst.bbk.ac.uk.

[20] *C. A. Coello Coello and M. Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In* Congress on Evolutionary Computation (CEC'2002), *volume 2, pages 1051–1056, Piscataway, New Jersey, 2002. IEEE Service Center.*

[21] *W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Jr. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman. A second generation force field for the simulation of proteins, nucleic acids and organic molecules.* J. Am. Chem. Soc., *117:5179–5197, 1995.*

[22] *Pierluigi Crescenzi, Deborah Goldman, Christos H. Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the Complexity of Protein Folding.* Journal of Computational Biology, *5(3):423–466, 1998.*

[23] *Charles Darwin.* Die Entstehung der Arten. *Reclam, Stuttgart, 1974.*

[24] *David B. Fogel. An Introduction to Simulated Evolutionary Optimization.* IEEE Transactions on Neural Networks, *5:3–14, 1994.*

[25] *David Beasley, David R. Bull, and Ralph R. Martin. A sequential niche technique for multimodal function optimization.* Evolutionary Computation, *1(2):101–125, 1993.*

[26] *David E. Goldberg.* Simple genetic algorithms and the minimal, deceptive problem. *Pitman, London, 1987.*

[27] *David E. Goldberg and Jon Richardson. In John J. Grefenstette, editor,* Proceedings of the Second International Conference on Genetic Algorithms, *pages 41–49, Hillsdale, 1987. Lawrence Erlbaum Associates.*

[28] *David E. Goldberg and Robert E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In John J. Grefenstette, editor,* Proceedings of the Second International Conference on Genetic Algorithms, *Hillsdale, 1987. Lawrence Erlbaum Associates.*

[29] *David H. Wolpert and William G. Macready. No Free Lunch Theorems for Optimization.* IEEE Transactions on Evolutionary Computation, *1(1):67–82, 1997.*

[30] *K. A. De Jong.* An Analysis of the Behavior of a Class of Genetic Adaptive Systems. *PhD thesis, University of Michigan, Ann Arbor, MI, 1975. Dissertation Abstracts International 36(10), 5140B, University Microfilms Number 76-9381.*

[31] *Kalyanmoy Deb.* Multi-Objective Optimization using Evolutionary Algorithms. *Wiley-Interscience Series in Systems and Optimization. John Wiley and Sons, Ltd., Baffins Lane, Chichester, West Sussex, England, 1 edition, 2001.*

[32] K. Dill. Principles of protein folding - A perspective from simple exact models. Protein Science*, 4:561–602, 1995.*

[33] Ken A. Dill and Hue Sun Chan. From levinthal to pathways to funnels. Nature Structural Biology*, 4(1), 1997.*

[34] M. Dragan, T. Hohm, T. Kohlen, D. Krämer, S. Kusper, P. Limbourg, H. Prothmann, M. Scheel, P. Senft, S. Sigg, N. Welp, and D. Zibold. Mehrzieloptimierung mittels evolutionärer Algorithmen*. Dortmund, 2003.*

[35] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[36] Edo Kussell and Eugene I. Shakhnovich. Glassy Dynamics of Side-Chain Ordering in a Simple Model of Protein Folding. Phys. Rev. Letters*, 89(16):168101–168104, 2002.*

[37] Edo Kussell, Jun Shimada, and Eugene I. Shakhnovich. Side-chain dynamics and protein folding. PROTEINS: Structure, Function, and Genetics*, 52:303–321, 2003.*

[38] A Engelbrecht. Computational Intelligence*. Wiley, New York, 2002.*

[39] Eric J. Sorin, Young Min Rhee, Bradley J. Nakatani, and Vijay S. Pande. Insights into nucleic acid conformational dynamics from massively parallel stochastic simulations. Biophys. J.*, 85:790–803, 2003.*

[40] Ali Farhang Mehr and Shapour Azarm. Diversity Assessment of Pareto Optimal Solution Sets: An Entropy Approach. In Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 723–728, Piscataway, New Jersey, 2002. IEEE Service Center.*

[41] Jonathan E. Fieldsend and Sameer Singh. On the Selection of Gbest, Lbest and Pbest Individuals, the Use of Turbulence and the Impact of Inertia in Multi-Objective PSO. IEEE Transactions on Evolutionary Computation*, 2002. under submission.*

[42] Karl Fogel. Open Source Development with CVS*. CoriolisOpen Press, Scottsdale, Arizona, 1999.* http://cvsbook.red-bean.com.

[43] C.R. G Woese, D.H. Dugre, S.A. Dugre, M. Kondo, and W.C. Saxinger. On the fundamental nature and evolution of the genetic code. In Cold Spring Harb Symp Quant Biol.*, volume 31, pages 723–736, 1966.*

[44] George P. Smith. Filamentous fusion phage: novel expression vectors that display cloned antigens on the virion surface. Science*, 228:1315–1317, 1985.*

[45] Gerald D. Fasman, editor. Prediction of protein structure and the principles of protein conformation*. Plenum Press, New York, 2 edition, 1990.*

[46] H.M. Geysen, H.M. Rodda, and T.J. Mason. The delineation of peptides able to mimick assembled epitopes. In Synthetic Peptides as Antigens. Ciba Foundations Symposium 119*, pages 131–149, New York, 1986. Wiley.*

[47] D. E. Goldberg. Genetic Algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.*

[48] O. Gursky and D. Atkinson. Thermal unfolding of human high-density apolipoprotein A-1: implications for a lipid-free molten globular state. Proc. Natl. Acad. Sci. USA*, 93(7):2991–2995, 1996.*

[49] *T. A. Halgren. MMFF VII. Characterization of MMFF94, MMFF94s, and Other Widely Available Force fields for conformational energies and for intermolecular- Interaction Energies and Geometries.* J. Comp. Chem., *20:730–748, 1999.*

[50] *Hans-Paul Schwefel.* Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. *Birkhäuser Verlag, Basel, 1977.*

[51] *M. P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, 1998.*

[52] *D. Hoffmann and E. Knapp. Protein dynamics with off-lattice Monte Carlo moves, 1996.*

[53] *D. Hoffmann, T. Washio, J. Jacob, and K. Gessler. Tackling concrete problems in molecular biophysics using monte carlo and Related Methods: Glycosylation, Folding, Solvation.*

[54] *Daniel Hoffmann and Holger Flörke. A structural role for glycosylation. Lessons from the hp-model.*

[55] *Ching-Lai Hwang and Abu Syed Md. Masud.* Multiple Objective decision making - Methods and applications. *Springer, Berlin, 1979.*

[56] *Ingo Rechenberg. Lehrstuhl für Bionik und Evolutionstechnik am Institut für Anlagentechnik, Prozesstechnik und Technische Akustik der TU Berlin.* http://www.bionik.tu-berlin.de.

[57] *Ingo Rechenberg.* Evolutionsstrategie: Optimierung der technischer Systeme nach Prinzipien der biologischen Evolution. *Frommann-Holzboog-Verlag, Stuttgart, 1973.*

[58] *Ingo Wegener. Skriptum Evolutionäre Algorithmen. 2002.*

[59] *John H. Holland.* Adaption in Natural and Artificial Systems. *University of Michigan Press, Ann Arbor, 1975.*

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

[61] *John R. Koza.* Genetic Programming: On the Programming of Computers by the Means of Natural Selection. *The MIT Press, Cambridge, MA, 1992.*

[62] *Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, and Hans-Paul Schwefel, editors,* Proceedings of the Parallel Problem Solving from Nature VI Conference, *pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.*

[63] *Keith D. Ball, Burak Erman, and Ken A. Dill. The elastic net algorithm and protein structure prediction.* J. Comp. Chem., *23(1):77–83, 2002.*

[64] *M. Khimasia and P. Coveney. Protein structure prediction as a hard optimization problem: The genetic algorithm approach.* Molecular Simulation, *19:205–226, 1997.*

[65] *S. Kirkpatrick, C.P. Gelatt, and M.P. Vecchi.* Optimization by Simulated Annealing. *IBM, New York, 1982.*

[66] *Klaus P. Schäfer. Lehrstuhl für Molekularbiologie am Fachbereich Biologie, der Universität Konstanz.* http://www.uni-konstanz.de/FuF/Bio/Bioinformatik.

[67] *D.K. Klimov and D. Thirumalai. Cooperativity in protein folding: from lattice models with sidechains to real proteins.* Fold. Des., *3(2):127–139, 1998.*

[68] *J. Knowles and D. Corne. On metrics for comparing non-dominated sets, 2002. In Congress on Evolutionary Computation (CEC 2002).*

[69] *J. D. Knowles.* Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization. *PhD thesis, The University of Reading, Department of Computer Science, Reading, UK, 2002.*

[70] *J.R. Koza, D. Andre, F.H. Bennet, and M.A. Keane.* Genetic Programming 3: Automatic Programming and Automatic Circuit Synthesis. *Morgan Kaufmann Publishers, Inc., San Francisco CA, 1 edition, 1997.*

[71] *A.R Leach.* Molecular Modeling Principles and Applications. *Prentice Hall Publicatons, New York, 2 edition, 2001.*

[72] *H. Li, R. Helling, C. Tang, and N. Wingreen. Emergence of preferred structures in a simple model of protein folding.* Science, *273:666–669, 1996.*

[73] *A. V. Lukashin, J. Engelbrecht, and S. Brunak. Multiple alignment using simulated annealing: Branch point definition in human mrna splicing.* Nucleic Acids Res., *20:2511–2516, 1992.*

[74] *A. D. Mackerell, J. Wiorkiewicz Kuczera, and M. Karplus. An all-atom empirical energy function for the simulation of nucleic acids.* J. Am. Chem. Soc., *117:11946–11975, 1995.*

[75] *Markus Brameier, Peter Dittrich, Wolfgang Kantschik, and Wolfgang Banzhaf. SYSGP - A C++ library of different GP variants. Technical report, Internal Report of SFB 531, Dortmund, 1998.*

[76] *S. Miller, J. Janin, Lesk. A. M., and C. Chothia. Interior and surface of monomeric proteins.* J Mol Biol., *196(3):641–656, 1987.*

[77] *S. Miyazawa and R. L. Jernigan. Estimation of effective inter-residue contact energies from protein crystal structures: quasi-chemical approximation.* Macromolecules, *18:534–552, 1985.*

[78] *S. Miyazawa and R. L. Jernigan. Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading.* J. Mol. Biol., *256:623–644, 1996.*

[79] *National Health Museum. Internetpräsenz von Access Execellence.* http://www.accessexcellence.org.

[80] *J.N. Onuchic, P.G. Wolynes, Z. Luthey Schulten, and Socci. N.D. Toward an outline of the topography of a realistic protein-folding funnel.* Proc. Natl. Acad. Sci. USA, *92:3626–3630, 1995.*

[81] *Projekt Gruppe 419 at University of Dortmund. Java Doc of* KEA *and* KEAGUI. *Contact LS11 at University of Dortmund for more details., 2002/03.*

[82] *G. Raghunathan and R. Jernigan. Ideal architecture of residue packing and its observation in protein structures.* Protein Science, *6:2072–2083, 1997.*

[83] *Rolf Backofen and Sebastian Will. Structure Prediction in an HP-type Lattice with an Extended alphabet. In* Proc of German Conference on Bioinformatics (GCB'98), *1998.*

[84] *Rune B. Lyngsø and Christian N.S. Pedersen. Protein folding in the 2d hp model, 1999.*

[85] *Samir W. Mahfoud.* Niching Methods for Genetic Algorithms. *PhD thesis, University of Illinois at Urbana-Champaign, Illinois, 1995.*

[86] *Sanaz Mostaghim, Jürgen Teih, and Ambrish Tyagi. Comparison of Data Structures for Storing Pareto-sets in MOEAs. In* Congress on Evolutionary Computation (CEC'2002), *volume 2, Piscataway, New Jersey, 2002. IEEE Service Center.*

[87] *S. Schulze Kremer. Genetic algorithms and protein folding.* Max-Planck Institute for Molecular Genetics., *143:175–222, 2000.*

[88] *Alena Shmygelska, Rosalía Aguirre Hernández, and Holger H. Hoos. An Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem. page 40. LNCS, 2002.*

[89] *Stefan Droste, Thomas Jansen, and Ingo Wegener. Perhaps Not a Free Lunch But At Least a Free Appetizer. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors,* Proceedings of theFirstGenetic and Evolutionary Computation Conference(GECCO '99), *pages 833–839, San Francisco CA, 1999. Morgan Kaufmann Publishers, Inc.*

[90] *MASSE Team. Internetpräsenz von MASSE - Modular Assay for Solar System Exploration.* http://www.masse.co.uk.

[91] *A. Teller and M. Veloso. PADO: Learning tree structured algorithms for orchestration into an object recognition system. Technical report, Department of Computer Science, Carnegie Mellon University, Pittsburgh, 1995.*

[92] *D.P. Tieleman, S.J. Marrink, and H.J.C. Berendsen. A computer perspective of membranes: Molecular dynamics studies of lipid bilayers systems.* Biochimica et Biophysica Acta, *1331/3:235–270, 1997.*

[93] *R. Unger and J. Moult. Genetic algorithms for protein folding simulations.* J. of Mol. Biol. 231, *1:75–81, 1993.*

[94] *Volker Gruhn and Andreas Thiel. Addison Wesley, München, 2000.*

[95] *Werner Massa.* Kristallstruktubestimmung. *Teubner Studienbücher: Chemie. Teubner, Stuttgart, 2 edition, 1996.*

[96] *Wolfgang Banzhaf. Skriptum Einführung in das Genetische Programmieren. 2002.*

[97] *E. Zitzler and L. Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute (ETH), Zurich, 1998.*

„Ich habe fertig."

Giovanni Trapattoni, Fußballtrainer