Prüfer:	Prof. Dr. Andreas Reuter
Betreuer:	Dr. Andreas Zell Dipl. Inf. Martin Reczko

begonnen am:	15. November 1991
beendet am:	12. Juni 1992
CR-Klassifikation:	C.1.2, I.2.6, I.2.8, J.3

Parallelisierung von Genetischen Algorithmen angewendet auf die Tertiärstrukturvorhersage von Proteinen

Frank Herrmann
Diplomarbeit Nr. 891

Institut für Parallele und Verteilte Höchstleistungsrechner
Universität Stuttgart
Breitwiesenstraße 20-22
D 7000 Stuttgart 80

Parallelisierung von Genetischen Algorithmen angewendet auf die Tertiärstrukturvorhersage von Proteinen

Frank Herrmann

Zusammenfassung

Proteine sind biologische Makromoleküle, die die Form von Ketten besitzen und in einer charakteristischen Weise gefaltet sind. Für die zahlreichen biologischen Funktionen der Proteine ist ihre Faltung von ausschlaggebender Bedeutung. Zur Vorhersage der Faltung wird die Energie der möglichen Faltungen minimiert, wobei als Optimierungsverfahren ein Genetischer Algorithmus verwendet wird. Genetische Algorithmen bilden Vererbungsmechanismen aus der Genetik nach. Der Genetische Algorithmus enthält eine Population von möglichen Faltungen, die in Form von Binärstrings kodiert sind. In einer Abfolge von Generationen werden die Vererbungsmechanismen so angewendet, daß Faltungen mit niedrigeren Energien entstehen. Die hierbei ausgeführten Berechnungen der Energien sind sehr aufwendig. Um die Optimierung zu beschleunigen, werden diese Berechnungen für die Faltungen einer Population parallel ausgeführt. Dazu ist der Genetische Algorithmus auf einem Parallelrechner *MASPAR-MP1* implementiert.

Inhaltsverzeichnis

Kapitel 1

Aufgabenstellung

Im Rahmen der Zusammenarbeit zwischen dem *IPVR* und dem *DKFZ* wurde in dieser Arbeit ein Genetischer Algorithmus auf dem *SIMD*-Parallelrechner *MASPAR MP1* implementiert und zur Optimierung der räumlichen Struktur von Proteinen eingesetzt.

Genetische Algorithmen sind Optimierungsverfahren, die Prozesse der natürlichen Evolution nachbilden. [1] enthält eine Einführung in die natürliche Genetik. Genetische Algorithmen werden zur Optimierung in Suchräumen eingesetzt, die aufgrund ihrer Eigenschaften für herkömmliche Verfahren schlecht geeignet sind. Die Punkte des Suchraums sind in Form von Strings kodiert. Durch diese Darstellung können Vererbungsmechanismen durch den Austausch von Teilen der Strings realisiert werden. Es ist eine Population mehrerer Strings vorhanden, in der die Vererbungsmechanismen in einer Folge von Generationen angewandt werden.

Proteine sind Kettenmoleküle, deren Aufbau und damit deren Funktion im genetischen Kode festgelegt ist. Dabei wird die Sequenz der Bausteine im Erbgut in eine Kette von Aminosäuren übersetzt. Die Abfolge der Aminosäuren stellt die Primärstruktur des Proteins dar. Sie bestimmt, wie sich das Molekül in die energetisch optimale räumliche Anordnung faltet, die als Tertärstruktur bezeichnet wird. Von vielen Proteinen kennt man die Primärstruktur, aber nur von etwa 350 Proteinen ist auch die Tertiärstruktur aufgeklärt. Die Primärstruktur läßt sich im Vergleich zur Tertiärstruktur relativ leicht bestimmen. Für die Erklärung der biochemischen Eigenschaften des Proteins muß aber seine Tertiärstruktur bekannt sein. Deshalb versucht man, die Tertiärstruktur aus der Primärstruktur vorherzusagen.

Der Raum der möglichen Faltungen eines Proteins ist sehr groß. Für jede Faltung kann eine Energie berechnet werden, die für die optimale Faltung ihren minimalen Wert annimmt. Der Genetische Algorithmus soll ausgehend von der Primärstruktur des Proteins eine energetisch günstige Faltung zur Vorhersage der Tertiärstruktur bestimmen.

Dazu muß aus der gegebenen Sequenz des Proteins eine Repräsentation des Moleküls erzeugt werden. In dieses Gerüst werden dann die Daten eingetragen, die die verschiedenen Faltungen unterscheiden. Die Daten zur Repräsentation der räumlichen Faltung erhält man durch die Entschlüsselung der in dem Genetischen Algorithmus verwendeten Strings. Die Kodierung der Faltungen in den Strings muß so gewählt werden, daß sie die Effekte unterstützt, denen Genetische Algorithmen ihre Leistungsfähigkeit verdanken. Dann muß die Funktion zur Berechnung der Energie einer in dieser Form gegebenen Faltung entwickelt werden.

Die Funktionen zum Aufbau der Repräsentation und zur Berechnung der Energie eines Moleküls wurden dann für die Durchführung der Optimierung auf dem Parallelrechner *MASPAR MP1* parallelisiert und in den Genetischen Algorithmus integriert. Der parallele Teil wurde in dem C-Derivat *MPL* implementiert. Zu seiner Einbindung wurden an dem Programm *GENESIS* die erforderlichen Änderungen vorgenommen.

Für die Testläufe wurde eine Menge von Proteinen bestimmt, deren Tertiärstruktur bekannt ist. So können die von dem Genetischen Algorithmus gemachten Vorhersagen überprüft werden. Die Proteine stammen aus der Brookhaven-Proteindatenbank. Die während der Optimierungen anfallenden Daten beschreiben den Verlauf der Optimierungen und repräsentieren die Vorhersagen in der vom genetischen Algorithmus verwendeten Form. Aus dieser werden durch eine serielle Version der entsprechenden Funktion wieder Repräsentationen der Moleküle erzeugt. Für die Beurteilung der Vorhersagen anhand der bekannten Strukturen wurden Ähnlichkeitsvergleiche durchgeführt.

Um die Leistung des Genetischen Algorithmus zu verbessern wurden Teile abgewandelt. Der in *GENESIS* verwendete Crossoveroperator wurde ersetzt. Der Selektionsmechanismus wurde den Erfordernissen der Energiefunktion angepaßt. Es wurde zusätzlich eine Version des Genetischen Algorithmus implementiert, bei der die Individuen der Population räumlich angeordnet werden. Dadurch werden die Vererbungsmechanismen lokal beschränkt und damit eine Nischenbildung ermöglicht.

Die vorliegende Arbeit ist in folgender Weise strukturiert: Das zweite Kapitel erklärt den Begriff des Genetischen Algorithmus. Es nennt die Teile, aus denen er aufgebaut ist, und führt in Überlegungen ein, die seine Funktionsweise erklären. Das dritte Kapitel beschreibt den chemischen Aufbau der Proteine. Es wird erläutert, wie dieser die große Vielfalt von Formen ermöglicht und welche Bedeutung die Proteine dadurch in der belebten Natur haben. Das vierte Kapitel beschreibt die Anwendung des Genetischen Algorithmus auf die Strukturvorhersage von Proteinen. Hier wird erläutert, wie die Kodierung der Faltung eines Proteins gewählt wurde. Im fünften Kapitel werden die Schritte erklärt, die zum Aufbau der Repräsentation des Moleküls aus der kodierten Form verwendet wurden. Hierbei werden Koordinatenmanipulationen mittels Verfahren aus der Vektoralgebra durchgeführt. Das sechste Kapitel beschäftigt sich mit der Berechnung der Energie eines Proteinmoleküls. Es werden zwei verschiedene Verfahren vorgestellt. Das gewählte Verfahren wird näher beschrieben. Im siebten Kapitel wird der verwendete Parallelrechner vorgestellt und die Aufteilung des Programms in einen seriellen und einen parallelen Teil vorgenommen. Im achten Kapitel werden die Eingangsdaten der Testläufe genannt. Im neunten Kapitel werden die Ergebnisse dargestellt und die Vorhersagen beurteilt. Im zehnten Kapitel werden die Schlußfolgerungen diskutiert, die aus den Ergebnissen gezogen wurden. Sie sind Anlaß zu Verbesserungs- und Erweiterungsvorschlägen.

Kapitel 2

Genetische Algorithmen

Zuerst wird der Begriff und die Arbeitsweise eines Genetischen Algorithmus erklärt. Dazu werden Definitionen aus [4] benutzt. Dann werden die Bestandteile eines Genetischen Algorithmus näher erläutert. Dem Vergleich mit herkömmlichen Optimierungsverfahren folgt die Einführung der Begriffe der Schemata und Building-Blocks. Mit ihrer Hilfe wird klar weshalb Genetische Algorithmen eine gerichtete Suche ausführen.

2.1 Begriff

Genetische Algorithmen sind Optimierungsalgorithmen. Sie optimieren das Verhalten eines Anwendungsprogramms, das durch eine Menge von Daten gesteuert wird. Bei der Manipulation der Steuerdaten werden Konzepte aus der Genetik verwendet. Steuerdaten können z.B. numerische Parameter einer Funktion sein, wobei der durch sie bestimmte Funktionswert dann beispielsweise maximiert werden soll. Beim klassischen Problem des Handlungsreisenden sind es symbolische Steuerdaten in Form der Reiseroute. Sie steuert die Berechnung der Weglänge. Das Anwendungsprogramm produziert in Abhängigkeit von den Steuerdaten Ergebnisse, die über sein Verhalten Aufschluß geben. Das Verhalten wird durch einen Fitneßwert beurteilt, von dem die Manipulation der Steuerdaten abhängt. (Vgl. Abbildung 2.1).

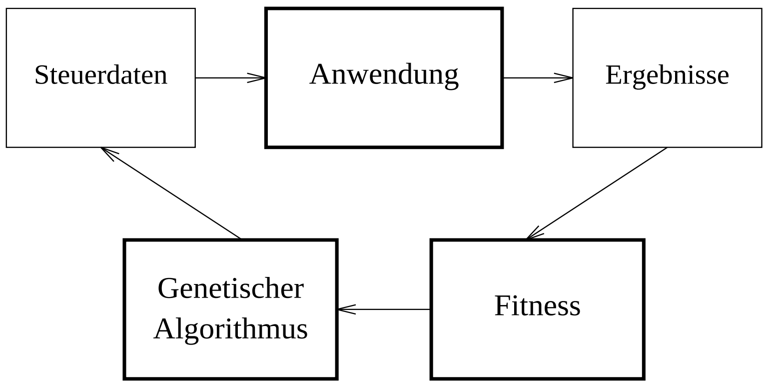


Abbildung 2.1: Genetischer Algorithmus und Umgebung

2.2 Arbeitsweise eines Genetischen Algorithmus

Die Menge aller möglichen Steuerdaten bildet einen Suchraum. Eine Menge von Punkten daraus stellt eine Population für den Genetischen Algorithmus dar. Diese Population besteht damit aus einer Sammlung möglicher Alternativen zur Steuerung des Anwendungsprogramms. Die Individuen in der Population sind in einem Alphabet kodiert. Die Strings entsprechen Chromosomen und ein Zeichen entspricht einem Gen.

Für jedes Mitglied der Population wird ein numerischer Wert bestimmt, der beschreibt, wie gut es das Anwendungsprogramm steuert. Jeder String wird dekodiert und der berechnete numerische Wert ist seine Fitneß. Entsprechend der Evolution wird nun eine Abfolge von Generationen erzeugt, wobei die Fitneß eines Individuums in Beziehung gesetzt zur Fitneß der anderen Individuen die Anzahl der Nachkommen bestimmt. Die Plätze der neuen Generation werden mit Nachkommen besetzt, indem Individuen der alten Generation, mit einer ihrer Fitneß entsprechenden Wahrscheinlichkeit, selektiert und reproduziert werden. Ein Individuum mit einer Fitneß, die über dem Durchschnitt der ganzen Population liegt, hat so eine große Selektionswahrscheinlichkeit, daß es mehr als einmal selektiert wird. Bei konstanter Populationsgröße werden schlechte durch bessere Individuen ersetzt. Die Individuen sammeln sich in den Teilen des Suchraums, denen die besten Fitneßwerte zugewiesen werden.

Nach diesem Schritt der Selektion und Reproduktion werden genetische Operatoren angewendet. Sie entsprechen der Vererbung, der Mutation oder anderen Vorgängen aus der Genetik. So entstehen neue Individuen aus Teilen der Individuen der Vorgängergeneration. Die neue Generation wird für die nächste Reproduktion wieder dekodiert und bewertet. Die Abfolge der Generationen wiederholt sich, bis ein Abbruchkriterium erfüllt ist oder bis mit einem vorläufig besten Ergebnis abgebrochen wird. (Vgl. Abbildung 2.2). Ein Abbruchkriterium kann z.B. das Erreichen eines bestimmten Fitneßwertes oder einer bestimmten Generationenzahl sein.

Im folgenden werden die einzelnen Teile genauer erklärt. Die Kodierung legt die den Steuerdaten entsprechenden Strings fest. Die genetischen Operatoren erzeugen neue Strings, und die Fitneß bewertet diese, um so ihre Selektionswahrscheinlichkeit zu bestimmen.

2.3 Die Kodierung

Um die Genetischen Operatoren möglichst einfach und einheitlich anwenden zu können, werden die Punkte als Strings über einem Alphabet kodiert. Im einfachsten Fall sind das Binärstrings einheitlicher Länge. Genetische Algorithmen manipulieren die Steuerdaten also nicht direkt.

Beispiel:

Eine Funktion mit dem Definitionsbereich [0..15] wird optimiert. Die Funktionsargumente sind binär kodiert. Die Menge der Strings ist: $\{0000, 0001, 0010, 0011, ..., 1110, 1111\}$. Ist die Kodierung festgelegt, dann ist die Anwendung der Operatoren problemunabhängig und reduziert sich auf einfache Stringmanipulationen.

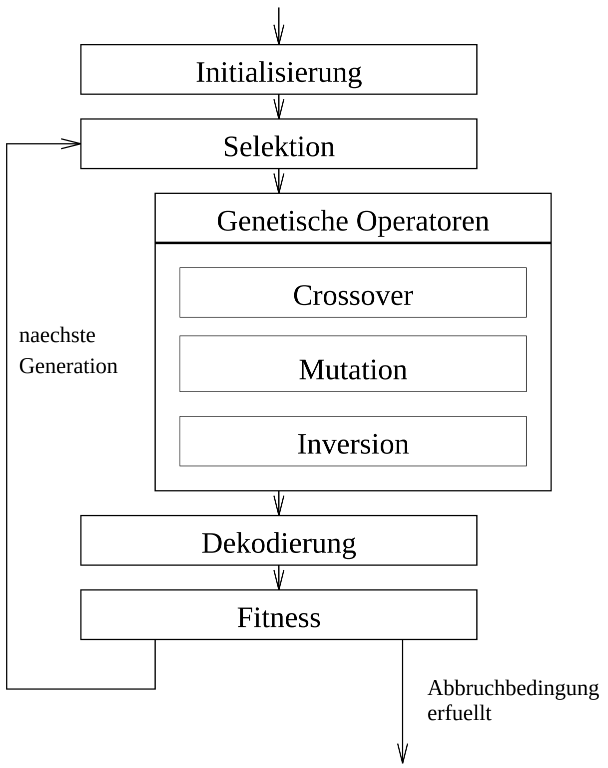


Abbildung 2.2: Hauptschleife des Genetischen Algorithmus

2.4 Genetische Operatoren

Neben problemspezifischen Operatoren, die Wissen über die Steuerdaten dazu benutzen, Strings nur dieser Kodierung sinnvoll zu manipulieren, verwendet jeder Genetische Algorithmus das Crossover und den Mutationsoperator. Sie haben sich als grundlegend für den Erfolg der Genetischen Algorithmen erwiesen.

2.4.1 Das Crossover

Das Crossover kreuzt zwei Strings, indem Teile der Strings herausgeschnitten und ausgetauscht werden. Die Schnittpunkte werden zufällig bestimmt. Der gebräuchlichste Crossover-Operator wählt für beide Strings einen einheitlichen Schnittpunkt und schneidet sie so in zwei Teile. Enthält beim einen String der vordere Teil, beim anderen der hintere Teil eine nützliche Eigenschaft, dann kombiniert ein Nachkomme diese Teile und verbessert seine Fitneß gegenüber der seiner Eltern.

Beispiel:

$$
\begin{array}{l}
Crossover\{101011, 111001\}: \\
101011 \longrightarrow 10|1011 \longrightarrow 101001 \\
111001 \longrightarrow 11|1001 \longrightarrow 111011
\end{array}
$$

Alle aus einer bestimmten Population durch Crossover zu bildenden Möglichkeiten zusammen decken u.U. nicht den gesamten Suchraum ab. Der Mutationsoperator bewirkt, daß im Prinzip jeder Punkt des Suchraums erreichbar ist.

2.4.2 Die Mutation

Die Mutation verändert zufällig den Wert eines Genes. Die Mutationsrate bestimmt, welche Strings und wie oft ein String mutiert wird. Die Mutation fügt neue Eigenschaften hinzu, denn das Crossover kann nur kombinieren, was schon vorhanden ist. Die Mutation stellt sicher, daß auch die fehlenden Regionen erreichbar sind. Ein Individuum mit niedriger Fitness wird möglicherweise ersetzt, obwohl es ein wichtiges Merkmal besaß, doch die Mutation verhindert, daß es unwiederbringlich verloren geht.

2.5 Die Fitneß

Die Anwendung erzeugt von den Steuerdaten abhängige Ergebnisdaten. Das können numerische Daten wie Funktionswerte, oder symbolische Daten sein. Die Bewertungsfunktion schließt von diesen Ergebnisdaten auf die Güte der Steuerdaten und ordnet ihnen einen Fitneßwert zu.
Beispiel:
Bei der Funktion $f(x) = x * x$ sind die Funktionswerte die Ergebnisdaten und können direkt als Fitneßwerte verwendet werden: $1011_2 \longrightarrow 11_{10} \longrightarrow 121_{10}$

Nach der Beschreibung der Teile und ihrer Aufgaben im Rahmen der Arbeitsweise eines Genetischen Algorithmus folgt nun der Vergleich dieser Arbeitsweise mit der herkömmlicher Optimierungsverfahren.

2.6 Herkömmliche Optimierungsverfahren

Bei herkömmlichen Optimierungsverfahren wird ausgehend von einem Startpunkt mit Hilfe von Zusatzinformation der nächste Punkt bestimmt, um so zu einer immer besseren Bewertung zu gelangen. Solche Zusatzinformation kann z.B. der Gradient der Bewertungsfunktion sein. In den Suchalgorithmus wird das Wissen eingebaut, wie man die zu optimierenden Größen verändern muß, um den Zielwert in der gewünschten Weise zu beeinflussen. Die zu optimierenden Größen werden direkt manipuliert.

Genetische Algorithmen dagegen starten mit einer Population zufällig gewählter Punkte im Suchraum. Die Suche findet an vielen Punkten gleichzeitig statt. Bei herkömmlichen Optimierungsverfahren kann es passieren, daß der einzelne Suchpunkt in einem lokalen Maximum steckenbleibt. Durch die verteilte Suche bei genetischen Algorithmen wird erreicht, daß so ein Punkt aus der Population verschwindet, wenn andere Punkte das lokale Maximum übertroffen haben.

Die Fitneß-Werte der Strings sind bei genetischen Algorithmen die einzige Information, die den Suchprozeß leiten. Man muß also nicht wissen, wie Änderungen in den Steuerdaten die Fitneß beeinflussen, sondern lediglich dem Genetischen Algorithmus eine Bewertungsfunktion für die erzeugten Steuerdaten zur Verfügung stellen. Da kein problemspezifisches

Wissen erforderlich ist, können Genetische Algorithmen immer dann angewendet werden, wenn man die erzielten Resultate bewerten kann.

Die beschriebene Arbeitsweise eines Genetischen Algorithmus gibt die Aktionen während der Optimierung an. Mit den Begriffen Schema und Building Block wird nun erklärt, wie durch diese Aktionen eine gerichtete Suche ausgeführt wird.

2.7 Schemata und Building Blocks

Die Beobachtung von Ähnlichkeiten zwischen Strings mit hoher Fitneß führt zum Begriff des Schemas. Schemata mit einer bestimmten Gestalt, sogenannte *Building-Blocks* bilden die Bausteine für gute Lösungen. Sie erklären, wie mit zufällig wirkenden Genetischen Operatoren eine gerichtete Suche ausgeführt wird. John H. Holland beschreibt in [3] die Wirkungsweise der Schemata.

2.7.1 Schemata

Mit dem Fortschreiten der Suche beobachtet man Ähnlichkeiten zwischen den besten Strings einer Population, bestimmte Gene stimmen bei ihnen überein. Also besteht zwischen diesen Genen und einer hohen Fitneß ein starker Zusammenhang. Die restlichen Gene liefern dann einen abgestuften Beitrag zur Fitneß.

Diese teilweise übereinstimmenden Strings sind Elemente der Menge aller Strings mit diesen übereinstimmenden Genen. Die übereinstimmenden Stellen definieren den Begriff eines Schemas. Ein Schema ist eine Teilmenge des Suchraums, die alle Elemente enthält, die an bestimmten Stellen übereinstimmen. In der Notation werden die nicht übereinstimmenden Stellen mit einem Don't-Care-Symbol markiert.

Beispiel:

Schema	Strings des Schemas
$10\#\#1:$	$\{10001, 10011, 10101, 10111\}$

Im folgenden sei eine binäre Codierung angenommen, da sie sehr häufig benutzt wird. Wenn l die Länge eines Strings ist, dann gibt es 2^l verschiedene Strings. Mit dem um das Don't-Care-Symbol erweiterten Alphabet $\{0,1,\#\}$ gibt es 3^l Schemata. Bei einem bestimmten String kann man an jeder Stelle zwischen dem Wert und dem Don't-Care-Symbol wählen. Daher ist jeder String Element von 2^l Schemata. Mit jedem Don't-Care-Symbol in einem Schema steigt die Anzahl seiner Elemente um eine Zweierpotenz.

Die Abbildung 2.3 verdeutlicht die Beziehungen zwischen den Schemata und den Strings des Suchraums, die in der Population enthalten sind. Die Kanten verbinden identische Strings. Sie zeigen, welche Strings Bestandteil der Population sind und welche Schemata diese Strings enthalten. Die Schemata bilden eine Menge von Teilmengen des Suchraums. Die Teilmengen haben je nach der Anzahl der Don't-Care-Symbole verschiedene Mächtigkeit. Ein bestimmter String kann in mehreren dieser Teilmengen enthalten sein (Siehe P1). Ein Schema ist in der Population vorhanden, wenn es durch mindestens einen seiner Strings vertreten ist. Jeder String ist Vertreter vieler Schemata und ein Schema kann mehrere Vertreter haben (Siehe S10).

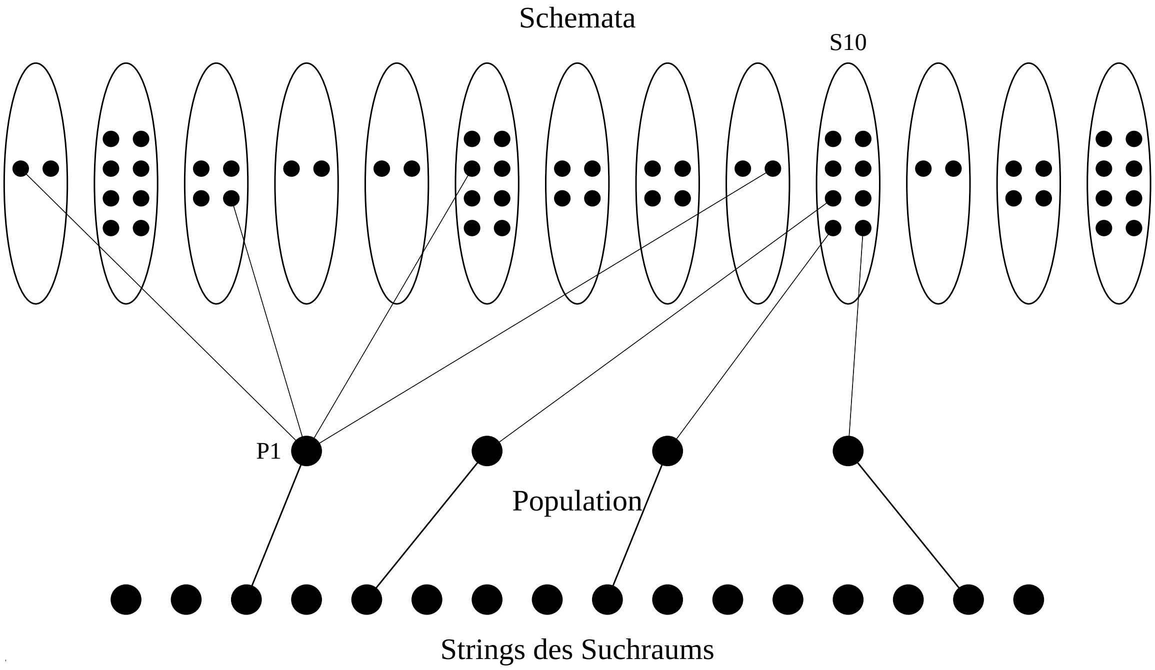


Abbildung 2.3: Schemata und ihre Strings, durch die sie in der Population vertreten sind.

2.7.2 Manipulation der Schemata

Die Schemata sind durch ihre Vertreter, die Strings, in der Population vorhanden. Bei jedem Erzeugen einer neuen Generation werden die Strings manipuliert. So werden die Schemata, deren Vertreter sie sind gleichzeitig manipuliert.

Wenn während der Selektion die Strings entsprechend ihrer Fitneß vermehrt oder verringert werden, dann verändert das auch die Präsenz von Schemata. Sie werden gleichzeitig indirekt manipuliert. Die Fitneß eines Schemas sei die durchschnittliche Fitneß aller seiner Elemente. Sie wird durch die durchschnittliche Fitneß der Elemente angenähert, die in der Population vorhanden sind. Ein Schema mit hoher Fitneß wird also durch Strings hoher Fitneß vertreten, die durch ihre höhere Selektionswahrscheinlichkeit öfters in der Population vorkommen. Wie beim einzelnen String die Zahl der Nachkommen, so wird beim Schema die Zahl der Repräsentanten von der Fitneß bestimmt.

Die Bewertung eines Strings verändert die Bewertung aller Schemata, deren Vertreter dieser String ist. Die Schemata werden so implizit parallel bewertet. Dieser implizite Parallelismus ermöglicht die effiziente Suche in sehr großen Suchräumen.

Die Population stellt die Bewertungen aller vorhandenen Schemata kompakt durch die Fitneß der Strings dar. Die Genetischen Operatoren Crossover und Mutation erzeugen neue Punkte, die die Bewertung vorhandener Schemata verfeinern und neue Schemata einführen. In [2] wird beschrieben, welchen Effekt die Genetischen Operatoren auf die Schemata ausüben. Ist ein neuer Punkt Repräsentant eines Schemas, das schon durch einen der alten Punkte vertreten ist, dann wird die Abschätzung der Fitneß des Schemas besser. Ist ein neuer Punkt Repräsentant eines Schemas, das bisher noch nicht vertreten war, dann wird ein neuer Teilraum des Suchraums bewertet. Es folgt die Beschreibung, welche Schemata durch einen neuen Punkt neu dazukommen und welche Schemata einen neuen

Repräsentanten erhalten.

2.7.3 Neue Punkte

Beim Crossover werden:

- neue Schemata eingeführt
- von alten Schemata neue Repräsentanten eingeführt.

Beispiel:

$$\begin{array}{l} Crossover\{1011, 1110\}: \\ 1011 \longrightarrow 10|11 \longrightarrow 1010 \\ 1110 \longrightarrow 11|10 \longrightarrow 1111 \end{array}$$

Der String 1011 war Repräsentant von 101#. Durch Crossover mit 1110 entsteht 1010. Das vierte Bit hat sich geändert, aber dieses Bit hat im Schema 101# ein Don't-Care-Symbol. Der neue String 1010 ist ein neuer Repräsentant von 101#.

Weder 1011 noch 1110 sind Repräsentanten von 10#0. Bei dem einen stimmen die festen Stellen des vorderen Teils, beim anderen die des hinteren Teils überein. Nach dem Crossover ist 1010 Repräsentant des neuen Schemas 10#0.
Beim Crossover können auch Schemata zerstört werden.

Beispiel:

$$\begin{array}{l} Crossover\{10011, 11000\}: \\ 10011 \longrightarrow 100|11 \longrightarrow 10000 \\ 11000 \longrightarrow 110|00 \longrightarrow 11011 \end{array}$$

Der String 10011 war Repräsentant von #0#|#1. Keiner der durch Crossover entstehenden Strings ist Repräsentant dieses Schemas. Im folgenden werden Schemata mit bestimmten Eigenschaften beschrieben, die eine größere Wahrscheinlichkeit haben, das Crossover zu überleben und daher am besten geeignet sind, die bisher beste Lösung zu erhalten.

2.7.4 Building-Blocks

Bei Schemata deren äußere feste Stellen nahe beieinander liegen, ist die Wahrscheinlichkeit am höchsten, daß sie vom Crossover nicht zerstört werden. Mit diesen *Building-Blocks* werden bessere Schemata zusammengesetzt. Waren die Eltern Repräsentanten von guten Schemata, deren feste Stellen beim einen vollständig links, beim andern rechts vom Crossover-Punkt lagen, entsteht ein Schema, das die Vorteile beider vereint.

Beispiel:

##110#1##|######
#########|#001##
$\Longrightarrow$
##110#1##|#001##

Wenn die Suche schon weit fortgeschritten ist, werden die Building-Blocks immer länger. Aufgrund ihrer hohen Fitneß sind sie in der Population stark vertreten, so daß die Wahrscheinlichkeit, durch Crossover zerstört zu werden, stark verringert wird. Durch die, infolge ihrer hohen Fitneß, große Zahl von Nachkommen ist es wahrscheinlich, daß nicht alle an einem Crossover teilnehmen. Werden aus einer Population, in der ein Bulding Block stark vertreten ist, zwei Individuen zum Crossover ausgewählt, dann ist die Wahrscheinlichkeit groß, daß die Partner Repräsentanten desselben Schemas sind. Es werden die nicht festen Stellen variiert, bis auch diese in einem noch besseren Schema zu festen Stellen werden.

Beispiel:

Schema: ##01#011###
01011|011000 $\longrightarrow$ 01011|011101
10010|011101 $\longrightarrow$ 10010|011000

In der Population entstehen, Generation für Generation, Repräsentanten immer besserer Schemata. Es folgt die Erklärung, warum durch diesen Prozeß ein Repräsentant entsteht, der den besten String darstellt, das Optimum.

2.8 Das Optimum

Die Strings erhalten entsprechend ihrer Fitneß Nachkommen. Implizit hat so jedes Schema seine eigene Fortpflanzungsrate, die durch die Fitneß der Vertreter des Schemas bestimmt wird. So werden immer bessere Schemata gefunden. Es folgt die Begründung, warum die Suche nach guten Schemata eine Suche nach dem optimalen String bewirkt.

Von den Schemata, die das Optimum enthalten, werden die am besten sein, die am wenigsten Elemente enthalten, denn bei ihnen wird die hohe Fitneß nicht durch die anderen Elemente weggemittelt. Bei der Suche nach guten Schemata werden sie die Population dominieren. Wenn die Zahl der festen Stellen in den Schemata steigt, wird das Optimum immer weiter eingegrenzt. Durch das Erzeugen neuer Punkte aus diesen Schemata wird dann das Optimum gefunden. (Vgl. Abbildung 2.4).

Die Strings bilden zusammen mit ihren Fitneßwerten die Datenbasis des Genetischen Algorithmus. In ihr ist aber mehr Information enthalten, denn zwischen den scheinbar unabhängigen Strings gibt es Zusammenhänge. Ihre Ähnlichkeiten stellen für die Suche eine wichtige zusätzliche Information dar.

Während der Genetische Algorithmus durch Stringmanipulationen aus alten Generationen neue erzeugt, findet implizit durch die Manipulation von Schemata eine Suche nach immer besseren Strings statt. Es folgt die Beschreibung von Eigenschaften der Kodierung und der Bewertungsfunktion, die die effektive Manipulation von Schemata unterstützen.

2.9 Eigenschaften der Kodierung und der Bewertung

Die Wahl der Bewertungsfunktion und der Kodierung für ein bestimmtes Problem liegt nicht fest. Oft gibt es mehrere u.U. sehr verschiedene Alternativen. Diese Wahl kann aber die Effektivität des Genetischen Algorithmus stark beeinflussen. Man möchte mit möglichst wenig Generationen auskommen und das Optimum soll auch gefunden werden. Genetische

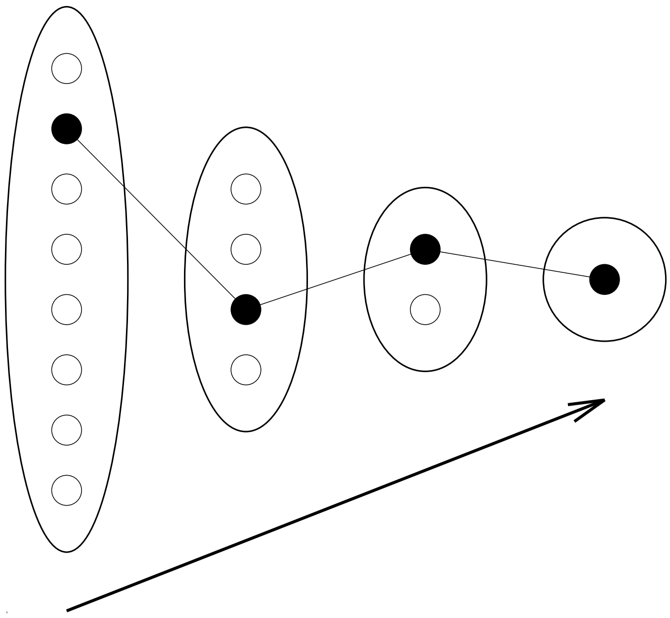


Abbildung 2.4: Schemata mit Optimum

Algorithmen werden bei sehr großen Suchräumen angewendet, bei denen eine aufzählende Suche, die jede mögliche Lösung testet, nicht durchgeführt werden kann. Ein Genetischer Algorithmus grenzt die Lösung, durch das Testen einiger Punkte des riesigen Suchraums und die daraus folgende Konzentration auf die erfolgversprechenden Teilräume, ein. Da nicht alle Punkte getestet werden, ist nicht sichergestellt , daß die Lösung gefunden wird. Daher müssen Kodierung und Bewertung so beschaffen sein, daß die Lösung in den Bereichen des Suchraums liegt, auf die sich die Suche konzentriert.

Wenn auch die Wahl der Bewertungsfunktion und der Kodierung nicht eindeutig ist, so gibt es dennoch Eigenschaften, die sie möglichst ausgeprägt besitzen sollten, damit die Suche sich auf die Teilräume konzentriert, die die Lösung enthalten. Diese Eigenschaften fördern die Effekte der Schemata und Building-Blocks, die die Effizienz der Suche verbessern.

2.9.1 Ähnlichkeit von Strings hoher Fitneß

Die Forderung, daß sich Strings mit einer hohen Fitneß ähneln, bedeutet, daß Strings, die in wenigen Genen vom Optimum abweichen, auch noch nahezu optimal sind. Die "Nähe" zum Optimum, die sich in der Ähnlichkeit ausdrückt, wird durch eine hohe Fitneß belohnt, so daß mit kleinen Änderungen der Strings in diesem Teil des Suchraums weitergesucht werden kann.

Die Kodierung muß sicherstellen, daß kleine Änderungen in dem String nicht große Änderungen in der zu optimierenden Struktur hervorrufen. Sind nicht ähnliche Strukturen durch ähnliche Strings kodiert, dann werden einer Menge ähnlicher Strukturen im ganzen Suchraum verstreute Punkte zugewiesen, und die "Nähe" zum Optimum zeigt sich nicht in der Ähnlichkeit der Strings.

Das folgende Beispiel nennt eine Kodierung, bei der eine kleine Änderung des Strings dazu führt, daß die dekodierte Struktur starke Unterschiede aufweist. Beim Problem des Handlungsreisenden werden Touren kodiert, die eine Menge von Städten verbinden. Kodiert man die Tour durch die Abfolge der Nummern der Städte, dann ergeben sich beim Crossover nicht immer legale Touren. Manche Städte können doppelt auftreten, während andere fehlen.

Dies wird bei einer Kodierung vermieden, die statt der Nummern der Städte die Nummern der Plätze auf einer Liste von Städten verwendet. Man beginnt mit der Liste aller Städte. Die gewählte Stadt wird jeweils aus der Liste entfernt, die damit immer kürzer wird. Die gleiche Nummer kann in der Kodierung mehrfach vorkommen, da sie immer eine andere Stadt bezeichnet. Andererseits isst sichergestellt, daß jede Stadt in der Tour enthalten ist.

Ändert man eine Zahl in der Kodierung einer Tour, dann bleibt der rechts von ihr stehende Teilstring sinnvoll, da die Länge der Liste der restlichen Städte gleich bleibt und somit die Platzangaben einen existierenden Platz beschreiben. Durch die Wahl eines anderen Listenplatzes wird eine andere Stadt aus der Liste entfernt. Damit ändert sich die Zuordnung von Listenplatz und Stadt. Die Platzangaben des Reststrings beziehen sich dadurch auf andere Städte, obwohl die Platzangaben nicht geändert wurden. Eine Änderung von einer Zahl im String bewirkt die Veränderung eines Großteils der dekodierten Tour. Die Tabelle 2.1 und die Abbildung 2.5 zeigen ein Beispiel.

Platz	Tour1								Tour2							
	1	5	2	5	3	3	1	1	1	5	6	5	3	3	1	1
	1	6	3	8	5	7	2	4	1	6	8	7	4	5	2	3
1	1	2	2	2	2	2	2	4	1	2	2	2	2	2	2	3
2	2	3	3	4	4	4	4		2	3	3	3	3	3	3	
3	3	4	4	5	5	7			3	4	4	4	4	5		
4	4	5	5	7	7				4	5	5	5	5			
5	5	6	7	8					5	6	7	7				
6	6	7	8						6	7	8					
7	7	8							7	8						
8	8								8							

Tabelle 2.1: Städtelisten

Die Bewertungsfunktion darf nicht so beschaffen sein, daß das Optimum des Problems bei der kleinsten Änderung seine Fitneß fast vollständig verliert, denn dann hat die Lösung den Charakter einer "Nadel im Heuhaufen" und der Genetische Algorithmus kann die "Nähe" zum Optimum wegen der fehlenden Fitneß nicht belohnen.

2.9.2 Eigenschaften der Building-Blocks

Building-Blocks existieren immer dann, wenn es für das Problem gute Teillösungen gibt. Die Teillösungen sollen kompakt weitergegeben und so zu besseren Teillösungen kombiniert werden können. Die Kodierung muß so gewählt werden, daß die zu den Teillösungen gehörenden Schemata möglichst kurz sind.

Die Forderung nach Additivität der Building-Blocks besagt, daß ein aus zwei Building-Blocks zusammengesetztes Schema eine höhere Fitneß besitzen muß als die Building-Blocks für sich. Es gibt Fälle, für die das nicht zutrifft und bei denen das kombinierte Schema trotz

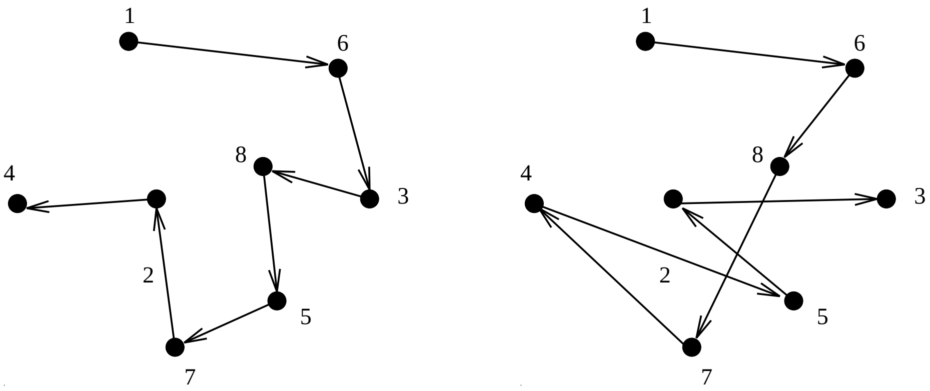


Abbildung 2.5: Touren

seiner guten Eltern eine schlechte Lösung darstellt. Solche Probleme werden GA-irreführend genannt, da bei ihnen gute Lösungen nicht aus guten Teillösungen zusammengesetzt werden können, und so die Suche fehlgeleitet wird.

Beispiel:

String	Fitneß
00	0.4
01	0.5
10	0.1
11	0.6

Schema	Fitneß
#0	0.25
#1	0.55
0#	0.45
1#	0.35

Das Schema 0# hat eine größere Fitneß als 1#. Also ist es am wahrscheinlichsten , daß 0# und #1 zu 01 kombiniert werden, obwohl 11 das Optimum ist.

Der komplementäre, Nichtlinearität genannte Fall tritt ein, wenn zwei schlechte Teillösungen durch das Crossover zu einer sehr guten Lösung zusammengesetzt werden. Die jeweiligen Gene für sich genommen liefern keinen Beitrag zur Fitneß, es sei denn, sie treten zusammen auf, d.h. sie sind in ihrer Bedeutung stark voneinander abhängig. (Vgl. Abbildung 2.6). In solchen Fällen stößt der Genetische Algorithmus nur mit geringer Wahrscheinlichkeit auf diese Kombination, da die Gene für sich allein keine Building-Blocks bilden. Solch eine nicht gerichtete Zufallssuche ist natürlich sehr ineffizient.

Die Kodierung von Teillösungen in Form von Building Blocks und ihre Bewertung entsprechend ihrer Ähnlichkeit zur Lösung unterstützt die Effekte der Schemata. Will man eine Teillösung eines Problems möglichst gut kodieren oder bewerten, so muß man eine Vorstellung davon haben, wie eine Teillösung aussieht. Dazu ist problemspezifisches Wissen nötig, das über die Identifizierung einer Lösung hinausgeht. Nur zu diesem Zweck ist bei Genetischen Algorithmen problemspezifisches Wissen nötig, und der Benutzer kann,

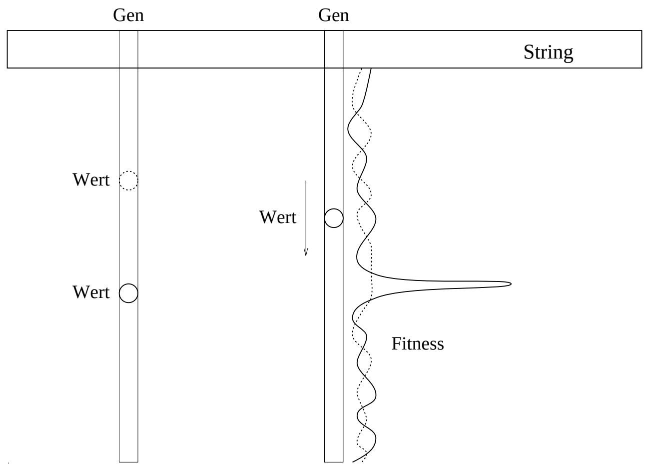


Abbildung 2.6: Gene mit starker Abhängigkeit

je nachdem wieviel Vorwissen ihm zur Verfügung steht, die Performanz des Genetischen Algorithmus steigern.

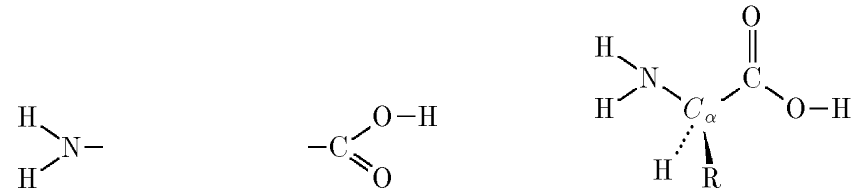
Kapitel 3

Proteine

Zuerst wird die chemische Struktur von Proteinen erklärt. Dann wird erläutert, wie diese die Herausbildung räumlicher Strukturen ermöglicht. Diese räumlichen Strukturen werden in drei Kategorien eingeteilt. Anschließend folgt eine Aufzählung von Verfahren mit denen die Strukturen aufgeklärt bzw. vorhergesagt werden. Abschliessend wird gezeigt, welche Bedeutung die freie Energie eines Proteins für die Ausbildung seiner räumlichen Struktur hat. Die Grundlagen des Aufbaus der Proteine sind in [8] und [9] dargestellt.

3.1 Chemische Struktur

Proteine sind Kettenmoleküle, die durch die Verknüpfung von kleinen Molekülen entstehen. Diese Bausteine sind Aminosäuren, von denen in Proteinen 20 verschiedene Arten vorkommen. Eine Aminosäure besteht aus einem zentralen Kohlenstoffatom C_α mit je einer Aminogruppe NH_2 und einer Carboxylgruppe $COOH$. Ein Wasserstoffatom ist mit der dritten Valenz des Kohlenstoffatoms verbunden. Die einzelnen Aminosäuren unterscheiden sich durch eine Seitenkette R, die die verbleibende Bindung besetzt. Die Abbildung 3.1 zeigt den allgemeinen Aufbau der Aminosäuren.



Aminogruppe   Carboxylgruppe   Aminosäure mit Seitenkette R

Abbildung 3.1: Grundstruktur der Aminosäuren

Das Kettenmolekül wird gebildet, indem jeweils die Carboxylgruppe einer Aminosäure mit der Aminogruppe der nächsten Aminosäure reagiert. Unter Abspaltung eines Wassermoleküls entsteht eine Peptidbindung. Die Abbildung 3.2 zeigt die Ausbildung der Bindung.

Der Anfang der entstehenden Kette wird durch die Aminosäure gebildet, deren Aminogruppe erhalten bleibt. Der Anfang wird wegen des Stickstoffatoms der Aminogruppe auch das N-terminale Ende genannt. Entsprechend heißt das Ende mit der Carboxylgruppe das

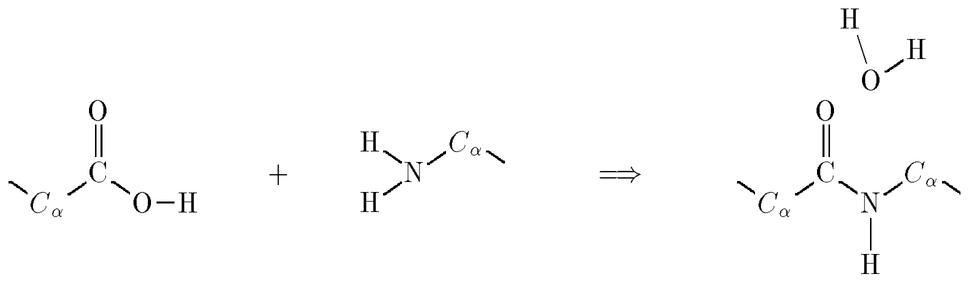


Abbildung 3.2: Entstehung der Peptidbindung

C-terminale Ende. Ein bestimmtes Protein ist durch die Abfolge der Aminosäuren vom N-terminalen Ende zum C-terminalen Ende definiert.

Durch die Peptidbindungen sind jeweils das Stickstoffatom der Aminogruppe, das zentrale Kohlenstoffatom und das Kohlenstoffatom der Carboxylgruppe Bestandteil der durchgehenden Kette von Bindungen in dem Molekül. Diese Atome bilden das Rückgrat des Proteins, bei dem jeweils an den zentralen Kohlenstoffatomen die Seitenketten oder Reste der Aminosäuren abzweigen.

3.2 Räumliche Struktur

Die räumliche Gestalt des Moleküls ist durch seine Bindungsstruktur noch nicht festgelegt. Die zahlreichen rotierbaren Bindungen ermöglichen eine große Zahl potentieller räumlicher Faltungen. Man spricht auch von der Konformation des Moleküls. Von den drei Bindungsarten, die im Rückgrat vorkommen, ist nur die Peptidbindung vergleichsweise starr. Das Rückgrat kann sich also an zwei Stellen pro Aminosäure drehen. Die Seitenketten enthalten je nach Aminosäure weitere drehbare Bindungen. Dadurch ergeben sich bei festliegender Faltung des Rückgrats, innerhalb des freibleibenden Raumes, für die Seitenketten zusätzliche mögliche Faltungen. Trotz der vielen Möglichkeiten, sich zu falten, nimmt ein Proteinmolekül eine eindeutig bestimmte räumliche Struktur an. Sie ist durch die Abfolge der Aminosäuren festgelegt. Die Moleküle des gleichen Proteins sind alle in derselben Weise gefaltet.

Die räumliche Struktur eines Proteins wird in drei Kategorien unterteilt. Die Primärstruktur ist durch die Abfolge der Aminosäuren, ihre Sequenz, bestimmt. Betrachtet man kurze Ausschnitte aus dem Molekül, dann treten 2 Arten von lokalen Faltungen immer wieder auf. Bei der ersten ist das Rückgrat in Form einer Helix gewunden. Sie wird α-Helix genannt. Bei der zweiten liegen die Atome des ausgestreckten Rückgrats in einer Ebene. Parallel dazu liegen ein oder mehrere ebenfalls so geformte andere Teile der Kette. Diese Struktur wird Faltblatt-Struktur genannt. Ist ein Teil der Kette nicht auf eine der beiden Weisen gefaltet, dann nennt man dies eine Coil-Struktur. Die räumliche Faltung des gesamten Proteins, die durch die Koordinaten jedes einzelnen Atoms festgelegt ist, nennt man die Tertiärstruktur. Für den Fall, daß sich mehrere Proteinmoleküle zu einem Verband zusammenlagern spricht man von der Quartärstruktur des Molekülverbunds.

3.3 Strukturaufklärung

Proteine spielen in der Biologie eine herausragende Rolle. Ihre vielfältigen Funktionen ergeben sich aus der großen Vielfalt ihrer Formen. Proteine sind an der chemischen Übertragung von Nervenimpulsen an den Synapsen beteiligt. Rezeptorproteine erkennen die im synaptischen Spalt freigesetzten Botenmoleküle. Das Immunsystem unterscheidet körpereigene und körperfremde Stoffe durch Oberflächenmerkmale der beteiligten Moleküle. Aus Proteinen aufgebaute Antikörper lagern sich an ihr Antigen an. Ihre räumliche Struktur enthält Teile, die nach dem Schlüssel-Schloß-Prinzip zu den Oberflächenstrukturen des Antigens passen. Proteine, die den Ablauf von Stoffwechselreaktionen in einer Zelle steuern, werden Enzyme genannt. [14] beschäftigt sich mit dieser Art von Proteinen. Sie sind Biokatalysatoren und beschleunigen jeweils eine ganz bestimmte Reaktion mit festgelegten Reaktionspartnern. Unterschiedliche Proteine katalysieren unterschiedliche Reaktionen. Für ihre Wirkungsweise ist ihre räumliche Struktur ausschlaggebend. Einbuchtungen in ihrer Oberfläche, sogenannte Domänen, passen spezifisch zu einem Teil des Moleküles eines Reaktionspartners und katalysieren so diese Reaktion. Um also den Wirkungsmechanismus verstehen zu können, muß die räumliche Struktur bekannt sein.

Die Bestimmung der Sequenz der Aminosäuren ist mit dem geringsten Aufwand verbunden. Sie wird auf chemischem Wege durch das schrittweise Abtrennen jeweils einer Aminosäure an einem Ende der Kette bestimmt. Diese Verfahren sind in [10] beschrieben. Die Sekundärstruktur wird rechnerisch aus den Atomkoordinaten der bekannten Tertiärstruktur bestimmt. Bei Proteinen, die in kristalliner Form vorliegen, läßt sich die Tertiärstruktur durch Röntgenbeugungsanalysen untersuchen. In [11, S. 135-145] ist das Verfahren der Röntgenbeugungsanalyse beschrieben. Ein anderes Verfahren benutzt die magnetische Kernresonanz der Atome. Diese Verfahren sind sehr aufwendig und deshalb sucht man nach einem Weg, die Tertiärstruktur mit Hilfe der Primärstruktur vorherzusagen. Von vielen Proteinen ist die Primärstruktur bereits bekannt, aber erst von ca. 350 Proteinen ist die Tertiärstruktur bestimmt.

3.4 Vorgang der Faltung

Die Tertiärstruktur entsteht durch den Prozeß der Faltung. Wie sich das Molekül faltet, wird von Kräften bestimmt, die zwischen den Atomen auftreten. Einige Seitenketten enthalten ganz oder teilweise geladene Atome, so daß anziehende und abstoßende Coulomb-Kräfte vorhanden sein können. Van-der-Waals-Kräfte lassen Atome sich anziehen, bis sich ihre Elektronenhüllen abstoßen. Aufgrund dieser Kräfte verändern sich die drehbaren Bindungen, wobei jede dieser Bindungen ein Rückstellmoment entwickelt, wenn sie aus ihrer Nullage gedreht wird. Im sich einstellenden Gleichgewicht hat das Molekül dann seine endgültige Faltung erreicht.

In der ersten Phase der Faltung bilden sich lokal jeweils die Sekundärstrukturen heraus. Sowohl α-Helix als auch Faltblatt werden durch Wasserstoffbrücken stabilisiert. Diese bilden sich zwischen dem verbliebenen Wasserstoff der Aminogruppe und dem Sauerstoff der ehemaligen Carboxylgruppe aus. Bei der α-Helix verbinden sie die Windungen der Helix, beim Faltblatt verbinden sie die parallel liegenden Stränge.

In der zweiten Phase der Faltung kommen die Kräfte über größere Distanz zum Tragen. Die bereits ausgebildeten Sekundärstrukturen ordnen sich zur Tertiärstruktur an.

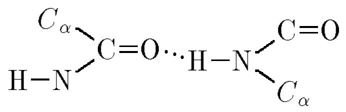


Abbildung 3.3: Wasserstoffbrücke

Im Verlauf der Faltung kann sich zwischen zwei Stellen der Kette, die sich nahe genug kommen eine kovalente Bindung bilden, wenn es sich um die Seitenketten der Aminosäure Cystein handelt. Die Thio-Gruppen SH spalten Wasserstoff ab und bilden eine Disulfid-Bindung zwischen den beiden Schwefelatomen. Solch eine Bindung stabilisiert die Faltung des Proteins.

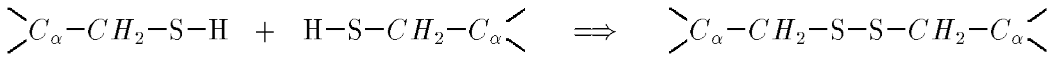


Abbildung 3.4: Disulfidbrücke

Die Kräfte, die die Faltung bewirken, verrichten Arbeit. Die Fähigkeit, Arbeit zu verrichten ist die Definition von Energie. Das ungefaltete Protein besitzt also eine potentielle Energie, die während der Faltung verringert wird, bis sie im Gleichgewichtszustand ihr Minimum erreicht hat. [17] enthält ein Kapitel über die potentielle Energie von Molekülen.

3.5 Strukturvorhersage

Zur Vorhersage werden verschiedene Verfahren benutzt. Durch den Vergleich übereinstimmender Teilsequenzen kann man auf ähnliche Strukturen schließen. Zur Vorhersage der Sekundärstruktur führt man an Proteinen bereits bekannter Struktur statistische Untersuchungen des Zusammenhangs zwischen Sequenz und Struktur durch. Dies führt zu Regeln, deren Trefferwahrscheinlichkeit über der eines bloßen Ratens liegt. Beim Ansatz der molekularen Dynamik versucht man den Faltungsprozeß durch die Simulation der dabei auftretenden Kräfte nachzuvollziehen. Schließlich wendet man verschiedene Minimierungsverfahren an, um das globale Minimum der potentiellen Energie eines Proteins zu bestimmen.

Kapitel 4

Anwendung zur Strukturvorhersage

In Kapitel 3 wurden verschiedene Verfahren zur Strukturvorhersage von Proteinen vorgestellt. Der hier gewählte Ansatz ist der der Energieminimierung. Genetische Algorithmen sind Optimierungsalgorithmen. Die Population des Genetischen Algorithmus enthält Punkte aus dem Raum der möglichen Konformationen des Proteins. Die Fitnesswerte der Elemente der Population ergeben sich aus den Energien der Konformationen. Durch die Anwendung der Genetischen Operatoren entstehen im Laufe einer Abfolge von Generationen Konformationen immer niedrigerer Energie.

Bei der Anwendung eines Genetischen Algorithmus auf ein spezielles Problem gibt es eine Schnittstelle zwischen dem eigentlichen Genetischen Algorithmus und den problemspezifischen Funktionen. Die Schnittstelle liegt bei der Berechnung der Fitneßwerte. Der Genetische Algorithmus übergibt die Binärstrings den Benutzerfunktionen. Diese berechnen aus ihnen die Fitneßwerte. Im folgenden werden für die Anwendung auf Proteine die Benutzerfunktionen und ihre Einbindung in den Genetischen Algorithmus beschrieben.

Zunächst muß eine Repräsentation für das Proteinmolekül gewählt werden. Diese Datenstruktur enthält auch Teile, die die Konformation des Moleküls beschreiben. Dann muß eine Kodierung gewählt werden, die die Konformation eines Proteinmoleküls innerhalb der Population des Genetischen Algorithmus als String darstellt. Aus dieser Darstellung müssen in jeder Generation die Konformationen erzeugt werden, so daß die Datenstruktur das Molekül vollständig beschreibt. Mit dieser Darstellung werden bei der Berechnung der Fitneßwerte die Energiewerte bestimmt.

4.1 Repräsentation eines Moleküls

Die Repräsentation des Moleküls wird während der Initialisierungsphase eines Optimierungslaufs erzeugt. Die Initialisierung ist von der Sequenz des Proteins, das optimiert wird abhängig. Der Zeitpunkt und die Stelle im Programm, wo die Initialisierung durchgeführt wird, sind in Kapitel 4.3 beschrieben.

Das Molekül wird durch die Sequenz seiner Atome repräsentiert. Die Atome jeweils einer Aminosäure sind zu einem Block zusammengefaßt. Die Abfolge der Blöcke entspricht der Sequenz der Aminosäuren. Für jede Aminosäure existiert ein Building Block. Er enthält die Atome einer Aminosäure in einer bestimmten Reihenfolge. Bei der Initialisierung wird aus der Sequenz und den Building Blocks die Sequenz der Atome zusammengestellt.

Die Bindungen zwischen den Atomen ergeben sich implizit durch die Benennung der Atome. Die Atome eines Building Blocks sind mit erweiterten Namen versehen. Dem chemischen Atomnamen folgt eine Kombination von Kürzeln für griechische Buchstaben und Zahlen. Durch diese Bezeichnung nach der *IUPAC*-Norm[1] lassen sich die Bindungen zwischen den Atomen einer Aminosäure rekonstruieren. Die Bindungen zwischen den Aminosäuren sind die Peptidbindungen. Die an ihnen beteiligten Atome sind mit N und C bezeichnet.

Durch die *IUPAC*-Bezeichnungen wird eine Verbindung zwischen dem Index eines Atoms innerhalb eines Building Blocks und einem bestimmten Atom in der Strukturformel einer Aminosäure hergestellt. Dieser Index wird in anderen Einträgen eines Building Blocks benutzt, um sich auf die Atome zu beziehen. Er dient der Definition von Drehwinkeln[2] und für die Festlegung von an der Energieberechnung beteiligten Atomen[3]. Dazu werden die Indizes des Building Blocks bei der Initialisierung in die Indizes für das gesamte Molekül umgerechnet.

4.1.1 *IUPAC*-Norm

In diesem Abschnitt werden an dem Beispiel der Aminosäure Tryptophan die *IUPAC*-Bezeichnungen der Atome vorgestellt. Die Abbildung 4.1 zeigt die Positionen der Atome in der Strukturformel. Die ausführliche Beschreibung der Regeln zur Namensgebung findet man in [18]. Hier folgt eine kurze Zusammenfassung.

Die Nichtwasserstoffatome der Seitenkette werden ausgehend von dem zentralen Kohlenstoffatom C_α mit den griechischen Buchstaben $\beta, \gamma, \delta, \ldots$ indiziert. An einer Verzweigung werden die Bindungen entsprechend einer Rangfolge der ersten Atome der Verzeigungen nummeriert. Die Nummerierung erfolgt nach abnehmenden Atommassen der Atome, bei gleichen Atomen entscheidet das nächste Atom einer Verzweigung. In jeder Seitenkette beschreibt der griechische Index den Abstand zu C_α. Die zusätzliche Zahl ist die Nummer der Abzweigung. Die Wasserstoffatome erhalten denselben Index wie ihr Bindungspartner. Sind mehrere Wasserstoffatome an denselben Partner gebunden, dann werden sie nummeriert.

Die Tabelle 4.1 nennt die entsprechenden Atomnamen, die im Building Block verwendet werden.

Atomindex	0	1	2	3	4	5	6		
Atomname	N	CA	C	O	CB	CG	CD1		
Atomindex	7	8	9	10	11	12	13		
Atomname	CD2	NE1	CE2	CE3	CZ2	CZ3	CH2		
Atomindex	15	16	17	18	19	20	21	22	23
Atomname	HA	HB1	HB2	HE1	HD1	HZ2	HH2	HZ3	HE3

Tabelle 4.1: Atomnamen

[1] Siehe Abschnitt 4.1.1
[2] Siehe Abschnitt 5.10
[3] Siehe Abschnitt 6.5

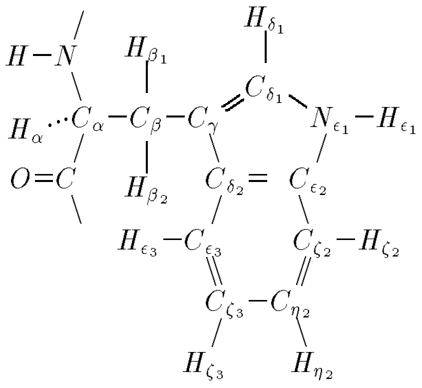


Abbildung 4.1: Tryptophan

4.1.2 Building Blocks

Die Building Blocks wurden mit Hilfe zweier Quellen zusammengestellt. Erstens das *GROMOS*-Programm und zweitens das *SYBYL*-Programm. *GROMOS* ist ein Programm zur Optimierung von Proteinmolekülen mit herkömmlichen Optimierungsverfahren. *SYBYL* ist ein Programm für das Molekulare Modellieren.

GROMOS bietet die Möglichkeit aus der Sequenz eines Proteins eine Datei zu erzeugen, die das Protein repräsentiert. Es benutzt dazu ebenfalls Building Blocks. Allerdings enthalten diese keine Atomkoordinaten einer Standardkonfiguration. Die Atomkoordinaten werden benötigt, um aus der Beschreibung der Konformation des Proteins[4] die Atomkoordinaten des gesamten Moleküls zu erzeugen[5].

Außerdem sind in der *GROMOS*-Building-Block Datei nicht alle Atome vorhanden. *GROMOS* benutzt das Konzept der *United Atoms*. Dabei werden die Wasserstoffatome an einem aliphatischen Kohlenstoffatom mit diesem zu einem vereinigten Atom zusammengefaßt. Ein aliphatisches Kohlenstoffatom ist Bestandteil einer Kohlenwasserstoffkette, im Gegensatz zu einem zyklischen, das in Ringstrukturen enthalten ist. Die Berechnung der Energie wird mit den vereinigten Atomen durchgeführt.

Die Berechnung der Energie in dieser Arbeit lehnt sich an das in *GROMOS* verwendete Verfahren an. Von den *GROMOS*-Building-Blocks werden für jedes Atom seine Ladung und die Bezeichnung seines Atomtypes verwendet. Zur Bedeutung dieser Werte für die Energieberechnung siehe Kapitel 6.

Es werden alle Atome repräsentiert. Die Berechnung der Atomkoordinaten wird für alle Atome durchgeführt. Die Berechnung der Energie verwendet die zusammengefaßten Wasserstoffatome nicht. So kann nach der Berechnung der Atomkoordinaten die Repräsentation eines kompletten Proteinmoleküls im *PDB*[6]-Format erzeugt werden.

Die Building Blocks mit den Koordinaten aller Atome wurden mit Hilfe von *SYBYL* erzeugt. In *SYBYL* kann eine Aminosäuresequenz angegeben werden, aus der eine Datei im *PDB*-Format erzeugt wird. Für jede Aminosäure wurde ein Protein erzeugt, daß aus drei gleichen Aminosäuren besteht. Auf diese Weise erhält man auch die Building Blocks für die

[4]Siehe Abschnitt 4.2

[5]Siehe Kapitel 5

[6]Brookhaven-Proteindatenbank

Aminosäuren am Anfang und am Ende eines Proteins. In *GROMOS* müssen am Anfang und am Ende die fehlenden Atome hinzugefügt werden. Am N-terminalen Ende kommt so ein Wasserstoffatom hinzu, am C-terminalen Ende eine Hydroxylgruppe.

Die bisher behandelten Teile der Building Blocks werden für die Repräsentation des Proteins verwendet. Sie enthalten weitere Teile, die bei der Initialisierung eingelesen werden. Sie werden bei der Erzeugung der Koordinaten und bei der Berechnung der Energie benötigt. In den Kapiteln 5 und 6 werden diese Teile beschrieben.

4.2 Kodierung

Die Konformationen sind in Form von Binärstrings zu kodieren. Bei der Dekodierung wird daraus eine Darstellung des Moleküls erzeugt, die für die Berechnung der Energie dieser Konformation geeignet ist. Die Energiefunktion benutzt kartesische Koordinaten der Atome des Moleküls. Kapitel 5 beschreibt die Erzeugung der kartesischen Koordinaten.
Diese werden aber nicht zur Kodierung der Konformationen verwendet. Nach der Building-Blocks Hypothese arbeitet ein Genetischer Algorithmus am besten, wenn ähnliche Lösungen auch ähnliche Kodierungen besitzen. Ähnliche Konformationen sind solche, bei denen Teile des Moleküls gleich gefaltet sind. Bei kartesischen Koordinaten führen die Unterschiede in einem Teil des Moleküls dazu, daß die Koordinaten des übereinstimmenden Teils gedreht und verschoben sind. Gleichartig gefaltete Teile sind so in der Kodierung nicht zu erkennen. Energetisch günstig gefaltete Teile eines Proteins stellen aber Teillösungen dar, die zu besseren Teillösungen kombiniert werden können. Die Kodierung mit kartesische Koordinaten unterstützt die Kombination von Teillösungen nicht und ist daher ungeeignet.

Gleich gefaltete Teile eines Proteins zeichnen sich dadurch aus, daß die Winkel der rotierbaren Bindungen übereinstimmen. Sie verändern sich bei Verschiebung oder Drehung des Teils nicht. Daher werden sie zur Kodierung der Konformation gewählt. Diese Drehwinkel werden im Unterschied zu den kartesischen die internen Koordinaten genannt.

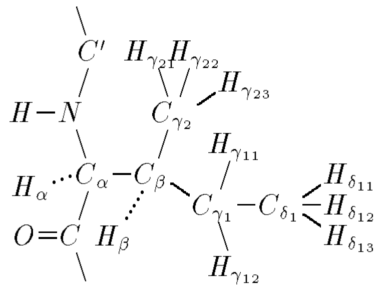
Ein zusätzlicher Vorteil ist, daß aus jeder Kombination von Drehwinkeln eine sinnvolle Konformation erzeugt werden kann, während die kartesischen Koordinaten der Atome nicht unabhängig voneinander gewählt werden können.

Bei kodierten Drehwinkeln stellt also jeder mögliche Binärstring eine mögliche Lösung dar. Der Fall, daß bei einer aus Drehwinkeln erzeugten Konformation Atome kollidieren oder sich durchdringen, drückt sich durch einen sehr hohen Energiewert aus. Es ist daher nicht nötig, diese Fälle extra auszuschließen.

In Abhängigkeit von dem zu optimierenden Protein ist zu seiner Darstellung eine unterschiedliche Anzahl von Drehwinkeln erforderlich. Die Aminosäuren besitzen durch ihre verschiedenen Seitenketten unterschiedlich viele Drehwinkel. Die Länge des Proteins und seine Aminosäuren bestimmen die Anzahl der Drehwinkel.

Nach der Dekodierung des Binärstrings erhält man eine Abfolge von Drehwinkeln. Während der Initialisierung wird anhand der Sequenz und der Building Blocks bestimmt, wie die Drehwinkel den Aminosäuren zugeordnet werden. Die Drehwinkel einer Aminosäure sind hintereinander angeordnet. Sie bestehen aus drei Drehwinkel des Rückgrats und den Drehwinkeln der Seitenketten. Die Abfolge von Drehwinkeln wird aufgeteilt, und die Drehwinkel des Rückgrats und der Seitenketten werden jeweils zusammengefaßt. Diese Aufteilung erklärt sich durch die Vorgehensweise bei der Berechnung der Koordinaten, siehe

Kapitel 5. Die Abbildung 4.2 zeigt für die Aminosäure Isoleucin den entsprechenden Ausschnitt aus der Abfolge von Drehwinkeln.



$\cdots$	$\angle(\overrightarrow{CC_\alpha})$	$\angle(\overrightarrow{C_\alpha N})$	$\angle(\overrightarrow{NC'})$	$\angle(\overrightarrow{C_\alpha C_\beta})$	$\angle(\overrightarrow{C_\beta C_{\gamma_1}})$	$\cdots$

Abbildung 4.2: Drehwinkel von Isoleucin

Jeder Drehwinkel ist durch einen Teil gleicher Länge des Binärstrings repräsentiert. Auf ihm sind die Werte des Intervalls $[-\pi, \pi]$ durch einen Gray-Code dargestellt. Dadurch ändert sich die Binärdarstellung eines Winkels der die Schranke von $-\pi$ nach $+\pi$ überschreitet kontinuierlich.

Aus den dekodierten Drehwinkeln werden die kartesischen Koordinaten der Atome erzeugt und in die Repräsentation des Moleküls eingetragen. Dann wird die Energie des Moleküls berechnet. Die Kapitel 5 und 6 beschreiben diese Programmteile im einzelnen. Im nächsten Abschnitt wird erklärt, wie die bisher besprochenen Teile in den Genetischen Algorithmus integriert werden.

4.3 *GENESIS*

Die Implementierung des Genetischen Algorithmus basiert auf dem Programmpaket *GENESIS*. Im folgenden wird zunächst *GENESIS* vorgestellt. Dann wird beschrieben, wie die Benutzerfunktionen eingebunden werden.

Verschiedene Anwendungen von Genetischen Algorithmen unterscheiden sich hauptsächlich durch die zu optimierenden Strukturen und ihre Bewertungsfunktionen. Die anderen Teile des Genetischen Algorithmus können von einer Anwendung auf eine andere übertragen werden.

4.3.1 Der Wrapper

GENESIS ist ein in C geschriebenes Programmpaket für Genetische Algorithmen, das es dem Benutzer ermöglicht, seine Bewertungsfunktion einzubinden. Es benutzt zur Kodierung der Individuen Binärstrings. Als Hilfe bei der Dekodierung kann der Benutzer eine Schablone angeben, wie der Binärstring interpretiert werden soll. Entsprechend der Schablone wird der Binärstring in Felder von C-Standardtypen übersetzt. Zeiger auf diese Felder werden der benutzerdefinierten Bewertungsfunktion übergeben. Ist die Bewertungsfunktion keine mathematische Funktion, dann müssen die Zahlen zur Kodierung der zu optimierenden Strukturen verwendet werden.

Die Schablone enthält Angaben über die Anzahl, die Länge in Bits, den Typ und den Wertebereich der Argumente. Der sogenannte Wrapper erzeugt aus einer Datei, die die in C geschriebene Bewertungsfunktion und die Schablone für deren Argumente enthält, eine Datei, die neben der Bewertungsfunktion eine Schnittstelle zum Hauptprogramm enthält. Sie besteht aus einer Funktion, die den Genotyp erhält, diesen entsprechend der Schablone für die Argumente der Bewertungsfunktion dekodiert und die Bewertungsfunktion mit den dekodierten Argumenten aufruft. Die vom Wrapper erzeugte Datei wird übersetzt und mit dem Hauptprogramm zusammengebunden.

Bei der Anwendung auf Proteine erhält die Bewertungsfunktion ein Feld von Drehwinkeln als Argument. Da verschiedene Proteine unterschiedliche Anzahlen von Drehwinkeln enthalten, sind die benötigten Schablonen verschieden. Ein Programm erzeugt aus einer Datei, die die Bewertungsfunktion enthält, und einer Datei, die die Sequenz des Proteins enthält, eine Datei, die die entsprechende Schablone enthält. Diese Datei wird dann als Eingabe für den Wrapper verwendet.

4.3.2 Die Bewertungsfunktion

Die Bewertungsfunktion, die von der vom Wrapper erzeugten Schnittstellenfunktion aufgerufen wird, besteht aus drei Teilen. Der erste Teil ist die Initialisierungsfunktion. Ihr Argument ist die Sequenz des Proteins. Sie wird nur einmal, beim ersten Aufruf der Bewertungsfunktion, ausgeführt. Sie erzeugt die Datenstrukturen zur Repräsentation des Moleküls und initialisiert weitere Datenstrukturen, die von den folgenden beiden Teilen benutzt werden. Der zweite Teil ist die Berechnung der kartesischen Koordinaten der Atome. Der dritte Teil ist die Berechnung der Energie des Proteins. Die Schnittstellenfunktion übergibt die berechnete Energie als Fitneßwert an den Genetischen Algorithmus.

4.3.3 Die Ein- und Ausgabedateien

GENESIS arbeitet dateiorientiert. Von einer Eingabedatei werden Parameter eingelesen, die jeweils einen Testlauf beschreiben. Das sind z.B. die Populationsgröße, die Crossover- und die Mutationsrate. Außerdem können verschiedene Abbruchkriterien angegeben werden. Ein Abbruchkriterium ist z.B. erfüllt, wenn die maximale Anzahl von Generationen erreicht ist oder wenn die Population zu einem gewissen Grad konvergiert ist. Schließlich wird die Art und der Umfang von Ergebnisdateien festgelegt.

Nach jeder Generation werden in die Ausgabedatei die beste und die durchschnittliche Fitneß sowie Werte über die Konvergenz ausgegeben. Als Ergebnis eines Testlaufs werden die besten gefundenen Lösungen in eine Datei geschrieben. Um einen Testlauf an der abgebrochenen Stelle weiterführen zu können, besteht die Möglichkeit, die gesamte Population in einer Datei zu speichern.

4.4 Programmstruktur

Die anwendungsspezifischen Teile für die Proteine und *GENESIS* wurden zu einer ersten seriellen Version des Programms zusammengefügt. Selbst bei Proteinen mit relativ wenigen Aminosäuren werden lange Binärstrings zu deren Repräsentation gebraucht. Wegen der sich daraus ergebenden Größe des Suchraums ist eine große Population angebracht. Zusammen mit der großen Komplexität der Energiefunktion ist für die Berechnung einer Generation ein großer Zeitbedarf zu erwarten. Deshalb wurde der Genetische Algorithmus parallelisiert. Das Kapitel 7 nennt die Änderungen, die hierzu nötig waren.

Mit der parallelen Version wurden die Testläufe durchgeführt. Sie werden im Kapitel 8 zusammen mit den verwendeten Eingangsdaten beschrieben. Das Kapitel 9 enthält die Auswertung der Ergebnisse. Die Auswertungsprogramme verwenden dazu die seriellen Funktionen zur Berechnung der Koordinaten und der Energie.

Die Abbildung 4.3 stellt die Programme und die verwendeten Daten in einer Übersicht dar.

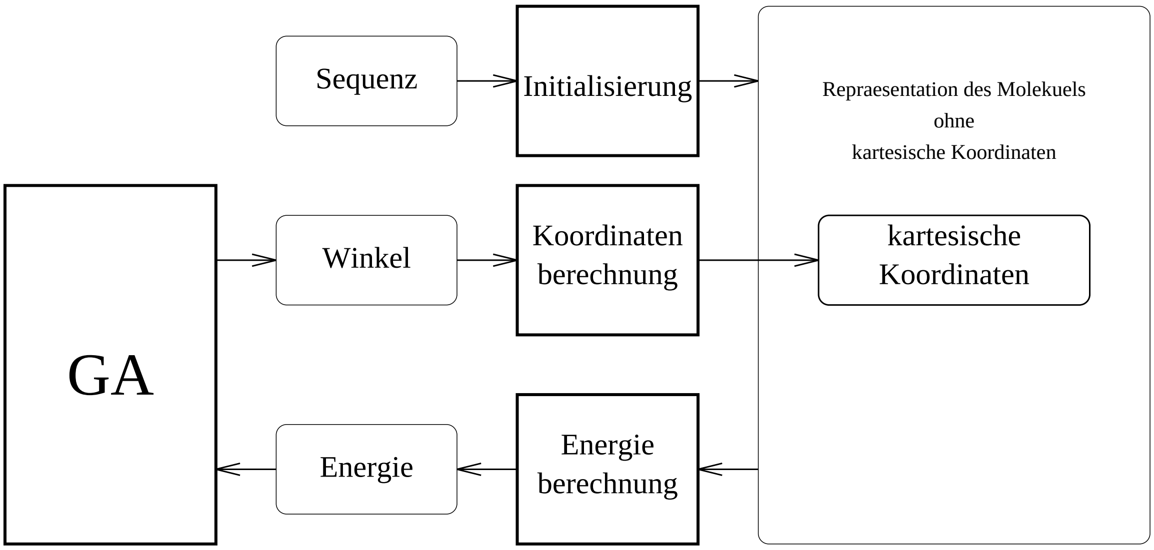


Abbildung 4.3: Programmstruktur

Kapitel 5

Molekulares Modellieren

Die Dekodierung des Binärstrings ergibt die Menge der Drehwinkel einer Konformation. Aus diesen werden die kartesischen Koordinaten erzeugt, mit denen die Energie berechnet wird.

Die kartesischen Koordinaten der Atome definieren die

- Längen der chemischen Bindungen
- Winkel zwischen zwei chemischen Bindungen desselben Atoms
- Drehwinkel entlang chemischer Bindungen

Ein Molekül ist dann eindeutig durch Drehwinkel bestimmt, wenn außer für die rotierbaren Bindungen, für alle diese Größen ihre Standardwerte verwendet werden.

Die Modellierung des Moleküls geschieht in vier Phasen. Zuerst wird das Rückgrat erzeugt. An das Rückgrat werden dann die Aminosäuren angepaßt. Für jede Aminosäure existiert eine Standardkonformation, ein sogenannter Building Block. Die in ihm enthaltenen kartesischen Koordinaten definieren die Lage der Aminosäureatome relativ zueinander. Die drei Atome, die Bestandteil des Rückgrats sind, werden mit den entsprechenden Atomen im Rückgrat zur Deckung gebracht. Dadurch werden die Koordinaten aller Aminosäureatome verschoben und rotiert, wobei ihre relative Lage zueinander erhalten bleibt.

In der dritten Phase werden das Wasserstoffatom der ehemaligen Amino-Gruppe H und das Sauerstoffatom O der ehemaligen Carboxyl-Gruppe in ihrer Lage korrigiert. Diese ist von den Bindungen im Rückgrat abhängig, die die Building Blocks verbinden.

In der vierten Phase werden alle drehbaren Bindungen in den Seitenketten von ihren Standardwerten in die Position gedreht, die den dekodierten Drehwinkeln entspricht. Dabei werden die Koordinaten aller Atome verändert, die Bestandteil des Teils der Seitenkette hinter der drehbaren Bindung sind.

Es folgt die Aufzählung der verwendeten Begriffe aus der Vektoralgebra und ihrer Anwendung auf Teile eines Moleküls. Dann werden Rotationen um Koordinatenachsen und um beliebige Achsen im Raum eingeführt, die in den Berechnungen der einzelnen Phasen benutzt werden. Anschließend werden die Phasen detailiert beschrieben.

5.1 Definitionen

Die kartesischen Koordinaten bilden einen Euklidschen Vektorraum.

Ortsvektor Der Ortsvektor $\vec{o}$ eines Punktes ist durch seine Koordinaten $\vec{o} = (x, y, z)$ gegeben.

Richtungsvektor Der Richtungsvektor $\vec{o_1 o_2}$ von einem Punkt $\vec{o_1}$ zu einem anderen $\vec{o_2}$ ist durch den Differenzvektor $\vec{o_1 o_2} = \vec{o_2} - \vec{o_1}$ der Ortsvektoren bestimmt.

Skalarprodukt Das Skalarprodukt zweier Vektoren $s = \vec{v_1} \cdot \vec{v_2}$ ist durch die Formel $\vec{v_1} \cdot \vec{v_2} = x_1 x_2 + y_1 y_2 + z_1 z_2$ definiert.

Betrag Den Betrag eines Vektors berechnet der Ausdruck $|\vec{v}| = \sqrt{\vec{v} \cdot \vec{v}}$.

Vektorprodukt Das Vektorprodukt $\vec{v_3} = \vec{v_1} \times \vec{v_2}$ zweier Vektoren ist ein Vektor, der auf beiden Vektoren senkrecht steht und mit diesen ein Rechtssystem bildet. Die Formel lautet:

$$\begin{pmatrix} x_3 \\ y_3 \\ z_3 \end{pmatrix} = \begin{pmatrix} y_1 z_2 - z_1 y_2 \\ z_1 x_2 - x_1 z_2 \\ x_1 y_2 - y_1 x_2 \end{pmatrix} = \begin{pmatrix} \begin{vmatrix} y_1 & y_2 \\ z_1 & z_2 \end{vmatrix} \\ \begin{vmatrix} z_1 & z_2 \\ x_1 & x_2 \end{vmatrix} \\ \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} \end{pmatrix}$$

5.2 Eine Bindung

Eine Bindung zwischen einem Atom a_1 und einem Atom a_2 wird durch den Richtungsvektor $\vec{a_1 a_2}$ dargestellt. Die Länge $l_{a_1 a_2}$ einer Bindung ist definiert durch den Betrag des Richtungsvektors.

5.3 Der Winkel zwischen zwei Bindungen

Der Winkel α zwischen zwei Bindungen ist definiert durch die jeweiligen Richtungsvektoren $\vec{a_2 a_1}, \vec{a_2 a_3}$ von dem Atom mit zwei Bindungen a_2 zu den Atomen mit einer Bindung a_1, a_3. Der eingeschlossene Winkel $\alpha = \angle(\vec{a_2 a_1}, \vec{a_2 a_3})$ wird berechnet durch:

$$\alpha = \arccos\left(\frac{\vec{a_2 a_1} \cdot \vec{a_2 a_3}}{|\vec{a_2 a_1}||\vec{a_2 a_3}|}\right)$$

5.4 Der Drehwinkel entlang einer Bindung

Der Drehwinkel entlang einer Bindung, auch genannt dihedraler Winkel, ist durch die Koordinaten von vier Atomen $a_1, \ldots, a_4$ definiert. Die drehbare Bindung liegt zwischen dem zweiten und dem dritten Atom. Sieht man in dieser Richtung die Bindung $\vec{a_2 a_3}$ entlang, dann bildet die Bindung zwischen dem dritten und dem vierten Atom $\vec{a_3 a_4}$ mit der zwischen dem ersten und dem zweiten Atom $\vec{a_1 a_2}$ einen Winkel. Um diesen Winkel ist die Bindung $\vec{a_2 a_3}$ aus ihrer Nullage gedreht. In dieser Blickrichtung ist der Wert des Winkels positiv, wenn sich $\vec{a_3 a_4}$ im Uhrzeigersinn dreht. Siehe Abbildung 5.1.

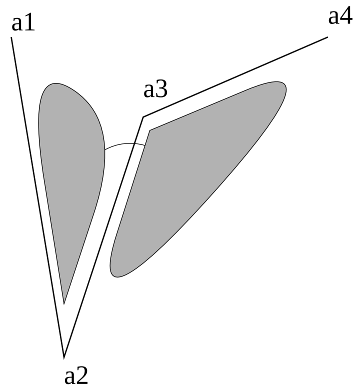


Abbildung 5.1: Drehwinkel

In der Nullage liegen alle vier Atome in einer Ebene. Die Richtungsvektoren $a_1\vec{a}_2$ und $a_3\vec{a}_4$ definieren mit dem Richtungsvektor $a_2\vec{a}_3$ jeweils eine Ebene. Der Drehwinkel $\angle(a_2\vec{a}_3)$ ist mit dem Schnittwinkel der Ebenen identisch. Dieser ist gleich dem Winkel zwischen zwei Orthogonalenvektoren $o_{\vec{123}}, o_{\vec{234}}$ der Ebenen. Das Vektorprodukt zweier Vektoren, die eine Ebene aufspannen, ist ein Orthogonalenvektor der Ebene.

Der einschliessende Winkel zwischen zwei Vektoren nimmt Werte zwischen 0 und π an. Der Drehwinkel für größere Werte entspricht einer Drehung entgegen dem Uhrzeigersinn, bei negativem Wert des Drehwinkels. Das Vorzeichen $\sigma(o_{\vec{123}}, o_{\vec{234}})$ des Winkels zwischen den Orthogonalenvektoren ist durch ihr Vektorprodukt bestimmt. Es ist kollinear mit $a_2\vec{a}_3$. Zeigt es in dieselbe Richtung wie $a_2\vec{a}_3$, dann ist der Drehwinkel positiv. In diesem Fall stimmen die Vorzeichen der Koordinaten der kollinearen Vektoren überein. Die Berechnungsformel lautet:

$$\angle(a_2\vec{a}_3) = \sigma(o_{\vec{123}}, o_{\vec{234}})\angle(a_1\vec{a}_2 \times a_2\vec{a}_3, a_2\vec{a}_3 \times a_3\vec{a}_4)$$

5.5 Rotation um eine Koordinatenachse

Die Grundlagen für die Berechnung von Rotationen mittels Matrizen finden sich z.B. in [22]. Rotiert man den Vektor $(1,0,0)$ mit dem Winkel α im mathematisch positiven Sinn um die z-Achse, dann ist seine x-Koordinate der Cosinus und seine y-Koordinate der Sinus von α. Der Vektor $(0,1,0)$ wird auf $(-\sin(\alpha), \cos(\alpha), 0)$ abgebildet. Für alle Punkte der xy-Ebene berechnet sich das Ergebnis durch Multipikation mit der Matrix:

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & - \\ \sin(\alpha) & \cos(\alpha) & - \\ 0 & 0 & - \end{pmatrix}$$

Bei einem beliebigen Punkt wird die Projektion auf die xy-Ebene rotiert, wobei die z-Koordinate unverändert bleibt. Die vollständige Matrix lautet daher:

$$\begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Die Matrizen für die Rotation um die x-Achse und die y-Achse lauten:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix} \quad \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix}$$

Zur Hintereinanderausführung zweier Rotationen kann man die Matrizen multiplizieren. Diese Operation ist assoziativ, aber nicht kommutativ.

Eine Rotation um eine Koordinatenachse kann als Koordinatentransformation interpretiert werden, wobei der Vektor fest bleibt und sich das Koordinatensystem dreht.

5.6 Erzeugung des Rückgrats

Es gibt drei Typen von Drehwinkeln im Rückgrat. Sie werden nach der Art der Bindung unterschieden:

ϕ Drehwinkel der Bindung zwischen Stickstoff und zentralem Kohlenstoff.

ψ Drehwinkel der Bindung zwischen zentralem Kohlenstoff und Kohlenstoff der ehemaligen Carboxylgruppe.

ω Drehwinkel der Bindung zwischen Kohlenstoff der ehemaligen Carboxylgruppe und Stickstoff.

Pro Aminosäure kommt jeder Winkel einmal vor, außer an den Enden. Dem N-terminalen Ende fehlt der ϕ-Winkel und dem C-terminalen Ende fehlt sowohl der ψ-Winkel als auch der ω-Winkel. Besitzt ein Protein n Aminosäuren, dann gibt es $3 * (n-1)$ Drehwinkel im Rückgrat.

Die Drehwinkel legen die Orientierung des Rückgrats im Raum nicht fest. Es besitzt drei Freiheitsgrade. Zuerst kann die Position des ersten Atoms frei gewählt werden. Das entspricht einer Verschiebung aus dem Ursprung. Als nächstes kann die erste Bindung in jede Richtung zeigen. Schliesslich kann die zweite Bindung um die Achse der ersten Bindung gedreht werden.

Hier wurden die Freiheitsgrade wie folgt gewählt. Das erste Atom wird in den Ursprung gelegt. Die erste Bindung verläuft entlang der x-Achse. Die zweite Bindung liegt in der xy-Ebene, mit positiver y-Koordinate. Das folgende Verfahren stammt aus [13, S. 430-432]. [16] enthält Molekülberechnungen in formalisierter Form.

Das Rückgrat wird schrittweise erzeugt, indem jeweils ein neues Atom a_n hinzugefügt wird. Seine Position ist bestimmt durch

- die Bindungslänge $l_{a_{n-1}a_n}$
- den Bindungswinkel $\beta = \angle(a_{n-1}\vec{a}_{n-2}, a_{n-1}\vec{a}_n)$
- und den Drehwinkel $\delta = \angle(a_{n-2}\vec{a}_{n-1})$

Es werden zwei Koordinatensysteme verwendet. Das fertig berechnete Rückgrat liegt im globalen Koordinatensystem vor. Das lokale Koordinatensystem befindet sich mit seinem

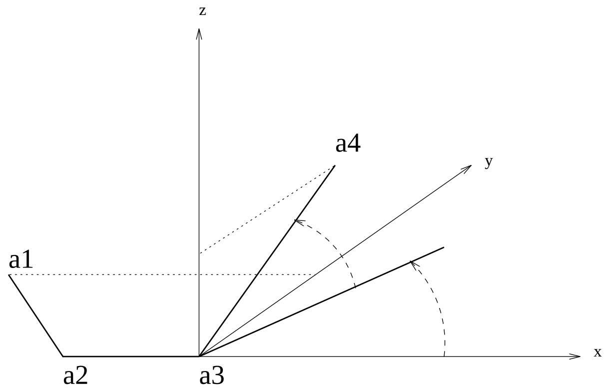


Abbildung 5.2: Lokales Koordinatensystem

Ursprung jeweils beim zuletzt berechneten Atom a_{n-1}. Das neue Atom a_n wird auf der x-Achse plaziert. Die Orientierung des lokalen Koordinatensystems relativ zu den letzten drei Atomen $a_{n-3}, a_{n-2}, a_{n-1}$ ist so gewählt, daß mit zwei Rotationen um seine Koordinatenachsen die richtigen Bindungs- und Drehwinkel eingestellt werden. Die lokalen Koordinaten des neuen Atoms werden dann in die globalen transformiert. Für das nächste Atom wird ein neues lokales Koordinatensystem definiert.

Das lokale Koordinatensystem ist so orientiert, daß die Atome $a_{n-3}, a_{n-2}, a_{n-1}$ in der xy-Ebene liegen. Das Atom a_{n-3} besitzt bezüglich des lokalen Koordinatensystems eine positive y-Koordinate, so daß ein in der xy-Ebene liegender Punkt die Nullage des Drehwinkels δ definiert. Siehe Abbildung 5.2.
Zuerst wird das Atom a_n mit der Bindungslänge $l_{a_{n-1}a_n}$ auf der x-Achse plaziert:

$$\vec{a_{n_0}} = (l_{a_{n-1}a_n}, 0, 0)$$

Dann wird um die z-Achse mit der Differenz des Bindungswinkels β zu π gedreht. Die Differenz wird gebildet, weil die x-Achse mit der Bindung $\vec{a_{n-2}a_{n-1}}$ einen Winkel von π bildet. Da $\sin(\pi - \beta) = \sin(\beta)$ und $\cos(\pi - \beta) = -\cos(\beta)$ gilt, lautet die Rotationsmatrix:

$$Rot_z = \begin{pmatrix} -\cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta) & -\cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Schließlich wird um die x-Achse mit dem Drehwinkel δ gedreht. Die Rotationsmatrix hierfür lautet:

$$Rot_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta) & -\sin(\delta) \\ 0 & \sin(\delta) & \cos(\delta) \end{pmatrix}$$

Die Rotationsmatrix für die gesamte Drehung lautet:

$$Rot_n = Rot_x * Rot_z = \begin{pmatrix} -\cos(\beta) & -\sin(\beta) & 0 \\ \sin(\beta)\cos(\delta) & -\cos(\beta)\cos(\delta) & -\sin(\delta) \\ \sin(\beta)\sin(\delta) & -\cos(\beta)\sin(\delta) & \cos(\delta) \end{pmatrix}$$

Die lokalen Koordinaten des Atoms a_n sind $\vec{a_{n_l}} = Rot_n * \vec{a_{n_0}}$. Diese müssen nun in die globalen Koordinaten transformiert werden.

Die Rotation des Vektors $\vec{a_{n_0}}$ kann man auch als Koordinatentransformation interpretieren. Dabei ist $\vec{a_{n_0}}$ der Koordinatenvektor des bereits gedrehten Vektors im neuen lokalen Koordinatensystem und $\vec{a_{n_l}}$ ist sein in das vorherige lokale Koordinatensystem transformierter Koordinatenvektor. Führt man diese Transformation für jedes lokale Koordinatensystem bis zum Beginn der Kette durch, dann erhält man die globalen Koordinaten des neuen Atoms.

Die Verkettung der Transformationen ergibt die Matrix $Trans_n$ für die Transformation in das globale Koordinatensystem. Beim Hinzufügen eines neuen Atoms a_{n+1} wird sie durch ihr Produkt mit der Drehmatrix Rot_{n+1} ersetzt.

$$\begin{aligned} \vec{a_{n_g}} &= Rot_1 * Rot_2 * \cdots * Rot_{n-1} * \overbrace{Rot_n * \vec{a_{n_0}}}^{\vec{a_{n_l}}} \\ \vec{a_{n_g}} &= \underbrace{Rot_1 * Rot_2 * \cdots * Rot_{n-1} * Rot_n}_{Trans_n} * \vec{a_{n_0}} \end{aligned}$$

Zuletzt werden noch die Koordinaten des Atoms a_{n-1} addiert, da es der Ursprung der lokalen Koordinatensysteme ist:

$$\vec{a_n} = \vec{a_{n-1}} + \vec{a_{n_g}}$$

Für die Atome a_1, a_2, a_3 wird die Transformation so gewählt, daß die Freiheitsgrade in der oben genannten Weise festgelegt werden. Für a_1 und a_2 ergeben sich keine Änderungen des Koordinatensystems. Für a_3 ist der Bindungswinkel $\angle(a_1, a_2, a_3)$ gegeben, während ein Drehwinkel noch nicht vorhanden ist. Deshalb liegt a_3 in der xy-Ebene. Ab a_4 sind Bindungs- und Drehwinkel definiert.

5.7 Anpassung der Building Blocks

Die Building Blocks enthalten die Atomkoordinaten einer Standardkonfiguration der jeweiligen Aminosäure. Die drei Atome b_1, b_2, b_3, die Bestandteil des Rückgrats sind werden mit den entsprechenden Atomen a_1, a_2, a_3 des berechneten Rückgrats zur Deckung gebracht. Hierzu wird durch Lösung eines linearen Gleichungssystems eine Drehmatrix Rot berechnet.

Die zentralen Kohlenstoffatome a_2, b_2 befinden sich im Ursprung des Koordinatensystems. Die beiden Bindungen, die Bestandteil des Rückgrats sind sollen aufeinander abgebildet werden. Damit diese Abbildung eindeutig ist werden außerdem die Vektorprodukte der beiden Richtungsvektoren aufeinander abgebildet.

$$\vec{nb} = \vec{b_2b_1} \times \vec{b_2b_3} \quad \vec{na} = \vec{a_2a_1} \times \vec{a_2a_3}$$

$$\vec{b_2b_1} \longrightarrow \vec{a_2a_1} \quad \vec{b_2b_3} \longrightarrow \vec{a_2a_3} \quad \vec{nb} \longrightarrow \vec{an}$$

$$Rot * \begin{pmatrix} \vec{b_2b_1}_x & \vec{b_2b_3}_x & \vec{nb}_x \\ \vec{b_2b_1}_y & \vec{b_2b_3}_y & \vec{nb}_y \\ \vec{b_2b_1}_z & \vec{b_2b_3}_z & \vec{nb}_z \end{pmatrix} = \begin{pmatrix} \vec{a_2a_1}_x & \vec{a_2a_3}_x & \vec{na}_x \\ \vec{a_2a_1}_y & \vec{a_2a_3}_y & \vec{na}_y \\ \vec{a_2a_1}_z & \vec{a_2a_3}_z & \vec{na}_z \end{pmatrix}$$

Das lineare Gleichungssystem wird mit einem Algorithmus aus [21] gelöst. Die Atome des Building Blocks werden abgebildet und die Koordinaten des zentralen Kohlenstoffatoms im Rückgrat werden addiert.

5.8 Rotation um eine beliebige Achse

Eine drehbare Bindung $\vec{ao}$ definiert eine Achse im Raum, um die alle Atome a_i gedreht werden, die Bestandteil des Teilmoleküls an $\vec{o}$ sind. Die Rotation geschieht in fünf Schritten. Zuerst wird der Ursprung des Koordinatensystems nach $\vec{o}$ verlegt. Dazu wird bei dem zu drehenden Atom der Vektor $\vec{o}$ von den Koordinaten subtrahiert. Durch Rotationen um die x-Achse und die y-Achse wird der Vektor $\vec{ao}$ mit der z-Achse zur Deckung gebracht. Dann kann die eigentliche Rotation mit dem Winkel ϱ um die z-Achse durchgeführt werden, bevor die Rotationen um die y-Achse und die x-Achse wieder rückgängig gemacht werden. Zum Schluß werden die Koordinaten von $\vec{o}$ wieder addiert. Die fünf Rotationsmatrizen werden zu der Matrix *Rot* aufmultipliziert, mit der die Atomkoordinaten abgebildet werden.

Die erste Rotation um die x-Achse dreht den Vektor $\vec{ao}$ in die xz-Ebene. Den Drehwinkel δ_x erhält man aus der Projektion $\vec{\pi_{yz}}(\vec{ao})$ von $\vec{ao}$ auf die yz-Ebene. Für die inverse Rotation wird dieser Winkel negativ. Die Matrizen für die Hin- und Rückdrehung lauten:

$$\vec{\pi_{yz}}(\vec{ao}) = (0, \vec{ao}_y, \vec{ao}_z)$$

$$\sin(\delta_x) = \frac{\vec{\pi_{yz}}(\vec{ao})_y}{|\vec{\pi_{yz}}(\vec{ao})|} \qquad \cos(\delta_x) = \frac{\vec{\pi_{yz}}(\vec{ao})_z}{|\vec{\pi_{yz}}(\vec{ao})|}$$

$$Rot_{x+} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta_x) & -\sin(\delta_x) \\ 0 & \sin(\delta_x) & \cos(\delta_x) \end{pmatrix} \quad Rot_{x-} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta_x) & \sin(\delta_x) \\ 0 & -\sin(\delta_x) & \cos(\delta_x) \end{pmatrix}$$

Die zweite Rotation dreht den Vektor in der xz-Ebene um die y-Achse, bis er mit der z-Achse zur Deckung kommt. Der Winkel $\epsilon_x = \angle(\vec{ao}, x - Achse)$ wird durch die Drehung um die x-Achse nicht verändert. Da $\delta_y = -(\pi/2 - \epsilon_x)$ gilt, ergibt sich für die Einträge der Rotationsmatrix und ihrer Inversen:

$$\sin(\delta_y) = -\cos(\epsilon_x) \qquad \cos(\delta_y) = \sin(\epsilon_x)$$

$$Rot_{y+} = \begin{pmatrix} \cos(\epsilon_x) & 0 & -\sin(\epsilon_x) \\ 0 & 1 & 0 \\ \sin(\epsilon_x) & 0 & \cos(\epsilon_x) \end{pmatrix} \quad Rot_{y-} = \begin{pmatrix} \cos(\epsilon_x) & 0 & \sin(\epsilon_x) \\ 0 & 1 & 0 \\ -\sin(\epsilon_x) & 0 & \cos(\epsilon_x) \end{pmatrix}$$

Die Matrix $Rot + z$ für die Rotation um die z-Achse ist die Standard-Rotationsmatrix mit dem Rotationswinkel ϱ. Die Berechnungsformel für die Roatation mit dem Winkel ϱ um die Achse $\vec{ao}$ lautet:

$$Rot = Rot_{x-} * Rot_{y-} * Rot_z * Rot_y * Rot_x \quad \vec{a_i'} = Rot * (\vec{a_i} - \vec{o}) + \vec{o}$$

5.9 Anpassung von *H* und *O*

Die Richtung der Bindung zwischen dem Stickstoff N und dem Wasserstoffatoms H ist relativ zu den Bindungen von N zum zentralen Kohlenstoff C_α und zum Kohlenstoff C_{-1}

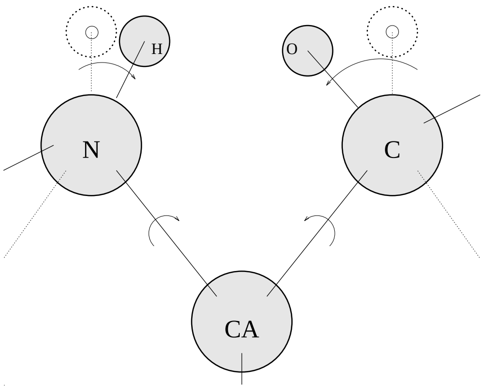


Abbildung 5.3: NH und CO

der vorherigen Aminosäure festgelegt. Der Building Block enthält C_{-1} nicht und der Drehwinkel $\angle(N, C_\alpha)$ im berechneten Rückgrat wird i.a. von dem Wert abweichen, den er in dem Molekül hatte, aus dem der Building Block stammt.

Die Position des Sauerstoff O hängt in gleicher Weise von der Bindung zwischen C und dem Stickstoff N_{+1} der folgenden Aminosäure ab. Die Drehwinkel $\angle(N, C_\alpha)$ und $\angle(C_\alpha, C)$ in dem Molekül, aus dem der Building Block stammt, sind während der Initialisierung eingelesen worden. Die Differenzen zu den in dem berechneten Rückgrat vorkommenden Werten bestimmen die Korrekturdrehwinkel. Das Atom H wird um die Achse $\vec{C_\alpha N}$ und das Atom O wird um die Achse $\vec{C_\alpha C}$ gedreht. Siehe Abbildung 5.3.

5.10 Die Drehwinkel der Seitenketten

Die Drehwinkel sind durch jeweils vier Atome definiert. Die Indizes dieser Atome sind Bestandteil der Building Blocks. Während der Initialisierung werden zusammen mit den Atomkoordinaten daraus die Werte berechnet, die die Drehwinkel in den Building Blocks besitzen. Das zweite und das dritte Atom ergeben das Paar von Atomen a_i, a_j, das die Drehachse bestimmt. Aus der Differenz der dekodierten Drehwinkel und ihrer Werte im Building Block ergeben sich die Korrekturdrehwinkel. Jedem Drehwinkel ist eine Menge von Atomen zugeordnet, deren Position von dem Drehwinkel abhängt. Diese Atome werden mit dem jeweiligen Korrekturdrehwinkel um die Achse $\vec{a_i a_j}$ gedreht. Ihre Indizes sind Bestandteil der Building Blocks.

5.11 Disulfide

Die Disulfidbrücken der Proteine bilden sich während der Faltung aus. Wenn sich die Thiolgruppen SH der Cysteine nahe genug kommen entsteht die kovalente Bindung.

Bei der Erzeugung des Moleküls aus den Drehwinkeln wird in einem letzten Schritt entschieden, ob bei dieser Konformation eine Disulfidbrücke vorhanden wäre. Dazu werden die Abstände der Schwefelatome der Thiolgruppen bestimmt. Eine Disulfid-Bindung besitzt eine Standardlänge. Wenn die Abweichung der Abstände von der Standardlänge eine Schranke unterschreitet, wird die Bindung ausgebildet.

Ein Protein kann mehrere Cysteine enthalten. Aus den möglichen Paaren wird eine Menge von Paaren zusammengestellt, bei denen die Abweichung von der Standardbindungslänge möglichst gering ist. Der benutzte Algorithmus lautet:

für i von 1 bis n
    wenn i noch frei
    erzeuge Bindung mit $j \in [i+1, \ldots, n]$
        so daß Abweichung $<$ Schranke und minimal
        wobei j noch frei
    markiere j als belegt

Dieses Kapitel beschrieb die mathematischen Hilfsmittel und ihre Anwendung zur Erzeugung der kartesischen Koordinaten. Im nächsten Kapitel wird erläutert, wie die Energie der erzeugten Konformation berechnet wird.

Kapitel 6

Energieberechnung

Aus den dekodierten Drehwinkeln wurde eine Repräsentation des Proteins erzeugt. Die kartesischen Koordinaten der Atome sind nun für die Berechnung der Energie vorhanden. Die Energie liefert einen Wert für die Wechselwirkung der Atome.

Die Wechselwirkungen unterteilen sich in solche zwischen Atomen, die durch Bindungen verbunden sind $\mathcal{B}$ und solche zwischen Atomen, die nicht direkt miteinander verbunden sind $\mathcal{N}B$.

Die bindungslosen Wechselwirkungen werden jeweils zwischen zwei Atomen betrachtet. Die Wechselwirkung für alle Atome ergibt sich durch die Summation der paarweisen Wechselwirkungen. Die bindungslosen Wechselwirkungen sind nur vom Abstand der Atome abhängig während die Bindungswechselwirkungen auch von Winkeln zwischen Bindungen bestimmt werden.

Die beiden Arten von Wechselwirkungen unterteilen sich jeweils in verschiedene Typen. Jeder Typ wird durch einen eigenen Term in der Energiefunktion repräsentiert.

6.1 Bindungswechselwirkungen

Die Bindungswechselwirkungen umfassen:

Bindungsdehnung E_b Der Abstand der Atome stimmt nicht mit der Standardbindungslänge überein. Eine Dehnung oder Stauchung wirkt sich quadratisch auf die Energie aus.

Bindungswinkel E_ω Der Winkel zwischen zwei Bindungen desselben Atoms weicht vom Standardwert ab. Die Abweichung wirkt sich quadratisch aus.

Bindungsdrehwinkel E_δ Der Drehwinkel entlang einer Bindung befindet sich nicht in seiner Nullage. Zur Definition des Drehwinkels siehe Abschnitt 5.4. Zwischen der Abweichung und der Energie besteht ein sinusförmiger Zusammenhang. Je nach Art der Bindung kann es während einer vollen Rotation mehrere Nullagen mit minimaler Energie geben.

Bindungsebenenwinkel E_τ Ein Atom ist mit drei anderen Atomen verbunden. Jeweils zwei Bindungen definieren eine Ebene. Wenn das Atom in der Mitte außerhalb der Ebene der drei anderen Atome liegt, ergibt sich zwischen dieser und einer durch zwei

Bindungen definierten Ebene ein Winkel. Sein Standardwert hängt von der normalen räumlichen Anordnung der Bindungen eines Atoms ab. Eine Abweichung wirkt sich quadratisch aus.

6.2 Bindungslose Wechselwirkungen

Die bindungslosen Wechselwirkungen unterteilen sich in:

Elektrostatik E_e Ladungen im Molekül führen zu Anziehung oder Abstoßung. Gleiche Vorzeichen bei den Ladungen bewirken eine Erhöhung der Energie. Diese ist dem Abstand der Atome umgekehrt proportional. Durch polare Bindungen entstehen partielle Ladungen im Molekül. Die Seitenketten bestimmter Aminosäuren haben sauren oder basischen Charakter. Bei neutralem pH-Wert enthalten sie nach einer Säure-Base-Reaktion ganze Ladungen.

Van-der-Waals E_v Diese Wechselwirkung setzt sich aus zwei Teilen zusammen. Der erste Teil repräsentiert die starke Abstoßung, die eintritt, wenn sich zwei Atome so nahe kommen, daß sich ihre Elektronenwolken berühren. Betrachtet man die Atome als harte Kugeln, dann setzt diese Wechselwirkung abrupt ein. Dies wird durch einen ab einem bestimmten Abstand stark wachsenden Term angenähert. Der zweite schwächere Anteil repräsentiert Ladungsverschiebungen in den Elektronenwolken, die zu einer Anziehung führen.

6.3 Berechnungsformeln

Die Tabelle 6.1 listet die Berechnungsformeln für die einzelnen Terme der Energiefunktion auf. Die Terme für die Bindungswechselwirkungen enthalten jeweils einen Standardwert $*_0$ für die betrachtete Größe. Alle Terme enthalten einen Proportionalitätsfaktor K_*. Er bestimmt den Zusammenhang zwischen der Abweichung vom Standardwert bzw. dem Abstandsausdruck und der Energie. Die Standardwerte und Proportionalitätsfaktoren hängen von der Art der beteiligten Atome ab. Der Index n bezeichnet jeweils den n-ten Summanden der betrachteten Größe. Die Indizes i und j bezeichnen Größen des i-ten und j-ten Atoms.

Der Wert des Parameters δ_{0_n} entspricht einem Energiemaximum des entsprechenden Bindungsdrehwinkels. Der Wert des Parameters m_n ist größer als Eins, wenn die Energie bei einer Drehung um 360 Grad mehrere Maxima durchläuft.

Der Van-der-Waals-Term besteht aus der Überlagerung eines anziehenden und eines anstoßenden Anteils. Die Anteile besitzen unterschiedliche Exponenten. Nähern sich zwei Atome aus großem Abstand, dann überwiegt zunächst der anziehende Anteil. Er führt zu einem Absinken der Energie bis zu einem Minimum, das den optimalen Abstand repräsentiert. Bei weiterer Annäherung wirkt sich der größere Exponent des abstoßenden Anteils aus. Er wächst nun schneller als der anziehende Anteil und erreicht große Werte, wenn sich die Atome berühren. Abbildung 6.1 zeigt schematisch diesen Verlauf der Van-der-Waals-Energie.
Die Energiefunktion setzt sich folgendermaßen aus den Termen zusammen:

Term	Formel	Bezeichner
Bindungsdehnung	$\frac{1}{2}K_{b_n}(b_n - r_{0_n})^2$	$b_n \ldots$ Bindungslänge n
Bindungswinkel	$\frac{1}{2}K_{\omega_n}(\omega_n - \omega_{0_n})^2$	$\omega_n \ldots$ Bindungswinkel n
Bindungsdrehwinkel	$K_{\delta_n}(1 + \cos(m_n\delta_n - \delta_{0_n}))$	$m_n \ldots$ Anzahl Maxima $\delta_n \ldots$ Bindungsdrehwinkel n
Bindungsebenenwinkel	$\frac{1}{2}K_{\tau_n}(\tau_n - \tau_{0_n})^2$	$\tau_n \ldots$ Bindungsebenenwinkel n
Elektrostatik	$\frac{Q_iQ_j}{4\pi\epsilon_0\epsilon_r r_{ij}}$	$Q_* \ldots$ Ladungen $\epsilon_* \ldots$ Dielektrizitätskonstanten $r_{ij} \ldots$ Abstand der Atome
Van-der-Waals	$\frac{K_{12_{ij}}}{r_{ij}^{12}} - \frac{K_{6_{ij}}}{r_{ij}^{6}}$	$K_{12_{ij}} \ldots$ Abstoßungsparameter $K_{6_{ij}} \ldots$ Anziehungsparameter

Tabelle 6.1: Berechnungsformeln der Energiefunktionsterme

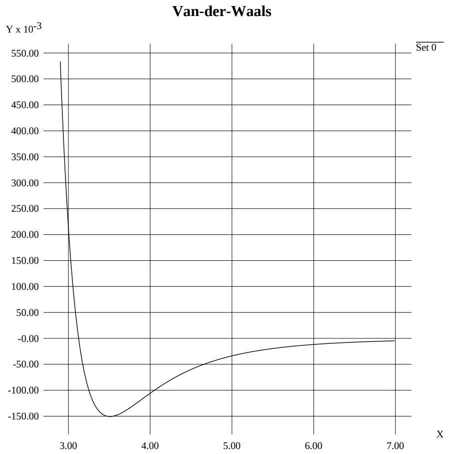


Abbildung 6.1: Schematischer Verlauf der Van-der-Waals-Energie

$$\mathcal{E} = \mathcal{B} + \mathcal{N}B =$$
$$\sum_n E_b(n) + \sum_n E_\omega(n) + \sum_n E_\delta(n) + \sum_n E_\tau(n) + \sum_{i<j} E_e(i,j) + \sum_{i<j} E_v(i,j)$$

Der nächste Abschnitt erklärt, wie die Standardwerte $*_0$ und die Proportionalitätsfaktoren K_* gewählt werden.

6.4 Parameter

Diese Form der Energiefunktion wird empirische Energiefunktion genannt. Sie verzichtet auf die Berechnung quantenmechanischer Effekte und rechnet statt dessen mit den Koordinaten der Atomkerne. Die Elektronen der Atomhüllen werden nicht explizit in der Berechnung berücksichtigt. Dadurch bleibt der Berechnungsaufwand auch für größere Moleküle, wie Proteine, in einem vertretbaren Rahmen.

Die Verschiedenheit der Wechselwirkungen zwischen den unterschiedlichen Atomtypen wird durch die Wahl der Standardwerte und der Proportionalitätskonstanten nachgebildet. Ihre Werte werden so eingestellt, daß die mit ihnen durchgeführten Berechnungen mit experimentellen Daten übereinstimmen. Eine weitere Möglichkeit ist die Bestimmung durch quantenmechanische Berechnungen an kleinen überschaubaren Molekülen. In dieser Arbeit wurden die Parameter des *GROMOS*-Programms [19] gewählt.

Von den oben genannten Termen wurden drei nicht berücksichtigt. Auf die Berechnung der Energie der Bindungslängen E_b, der Bindungswinkel E_ω und der Bindungsebenenwinkel E_τ konnte verzichtet werden. Sie besitzen durch die Vorgehensweise beim Aufbau des Moleküls ihre Standardwerte und leisten daher keinen Beitrag zur Energie. Von allen Bindungsdrehwinkeln wird für jene ein Energiebeitrag berechnet, die die verschiedenen Konformationen des Proteins definieren. Die restlichen Bindungsdrehwinkel besitzen ihre Standardwerte. Damit setzt sich die verwendete Energiefunktion aus folgenden Termen zusammen:

$$\begin{aligned}&\mathcal{E} = \mathcal{B} + \mathcal{N}B = \\ &\textstyle\sum_n E_\delta(n) + \sum_{i<j} E_e(i,j) + \sum_{i<j} E_v(i,j)\end{aligned}$$

Im nächsten Abschnitt wird gezeigt, wie die Atompaare erzeugt werden, für die die Energieterme berechnet werden.

6.5 Paarbildung

Die bindungslosen Wechselwirkungen werden für alle Atompaare berechnet. Damit jedes Paar nur einmal berücksichtigt wird, besitzt das zweite Atom immer einen größeren Index als das erste. Einige der Paare werden von der Berechnung ausgeschlossen. Atome, die durch eine Bindung, oder mit einem dazwischenliegenden Atom über zwei Bindungen verbunden sind, werden aus der Menge der Paare entfernt. Die Wechselwirkungen zwischen solchen Nachbarn ersten und zweiten Grades werden von den Bindungen bestimmt. Die Parameter der Energiefunktion gelten für Atome, die nicht Nachbarn ersten oder zweiten Grades sind. Z.B. ist die Länge einer chemischen Bindung kleiner als die Summe der Atomradien, so daß für Nachbarn ersten Grades sehr große Energiewerte berechnet würden.

Die auszuschließenden Paare von Atomen innerhalb einer Aminosäure sind in den Building Blocks enthalten. Dort sind für jedes Atom die Indizes der Atome vermerkt, mit denen kein Paar gebildet wird. Durch die Verbindung der Building Blocks durch das Rückgrat ergeben sich auszuschließende Paare von Atomen aus verschiedenen Building Blocks. Diese werden durch Indices dargestellt, deren Werte den größten Index innerhalb des Building Blocks übersteigt. Sie entsprechen einem Weiterzählen in den nächsten Building Block hinein.

Im folgenden wird am Beispiel der Aminosäure Asparagin gezeigt, wie mit Hilfe der Strukturformel und den durch die *IUPAC*-Bezeichnungen festgelegten Indices[1] die Liste der Ausschlüsse für die Building Blocks erstellt wurden. Die Wasserstoffatome ohne Index wurden mit den Kohlenstoffatomen zu *United Atoms* zusammengefaßt und werden bei der

[1] Siehe Abschnitt 4.1.1

Energieberechnung nicht berücksichtigt. Das Wasserstoffatom, das mit dem Stickstoff N^{11} verbunden ist, gehört zum nächsten Building Block. Da seine Position in den verschiedenen Building Blocks nicht die gleiche ist, kann hier kein Index für es festgelegt werden. Während der Initialisierung liegt die Sequenz fest und der Index des Wasserstoffatoms kann bestimmt werden.

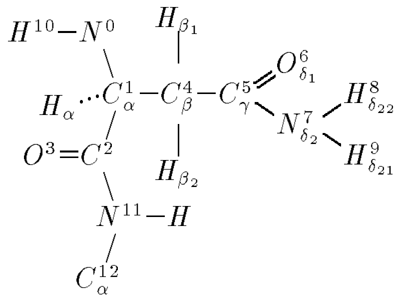


Abbildung 6.2: Atomindizes

Die Tabelle 6.2 nennt die sich aus der Abbildung 6.2 ergebende Liste von Ausschlüssen.

von	nach					
0	1	2	4	10		
1	2	3	4	5	10	11
2	3	4	11	12		
3	11					
4	5	6	7			
5	6	7	8	9		
6	7					
7	8	9				
8	9					

Tabelle 6.2: Ausschlüsse

Für die Erzeugung der Paare während der Berechnung wird eine Datenstruktur erzeugt, die für jedes Atom die Indices aller Partneratome enthält. Das Atom steht an der ersten Stelle der Paare. Die zweite Stelle enthält die Indizes der Partneratome. Diese sind in Form von mehreren Intervallen repräsentiert, die jeweils durch einen auszuschliessenden Partner unterbrochen werden. Ein Intervall besteht aus einem Start-und einem Ende-Wert, so daß alle Indizes dazwischen ein Partneratom bezeichnen.

6.6 Disulfide

Im letzten Schritt des Aufbaus des Moleküls wurde entschieden, ob und wo eine Disulfid-Bindung vorliegt. Bei der Faltung eines natürlichen Proteins entstehen die Disulfid-Bindungen spontan und stellen daher eine Verringerung der Energie dar. Die Änderung der Energie wird durch einen zusätzlichen Term in der Energiefunktion berücksichtigt.

Die Entscheidung über das Vorhandensein einer Disulfid-Brücke wurde anhand des Abstandes der Schwefelatome gefällt. Da die Positionen der Schwefelatome und der Kohlenstoffatome, mit denen sie verbunden sind, von der Faltung des Moleküls abhängen, befindet sich die Disulfid-Brücke i.a. nicht in ihrem Standardzustand. Deshalb werden bei ihnen die Terme der Bindungswechselwirkungen berechnet, die sonst nicht berechnet werden müssen. Es sind dies:

- Die Bindungslänge zwischen den Schwefelatomen $E_b(dis_i)$
- Die Bindungswinkel jeweils zwischen der Bindung der Schwefelatome und der Bindung zwischen Schwefel und Kohlenstoff $E_{\omega_1}(dis_i), E_{\omega_2}(dis_i)$
- Der Drehwinkel entlang der Bindung zwischen den Schwefelatomen $E_\delta(dis_i)$

Diese Beiträge stellen eine Erhöhung der Energie dar. Die Entscheidung zur Ausbildung der Bindung wirkt sich durch eine Verringerung der Energie um einen festen Betrag C aus. Dieser repräsentiert die Bildungsenergie der kovalenten Bindung. Je nachdem, wie nahe die Bindung dem Ideal kommt, bleibt von diesem Energiegewinn, nach der Berücksichtigung der obigen Beiträge, mehr oder weniger viel übrig.

Dieser feste Betrag wurde so gewählt, daß er bei der größten Abweichung von der Standardbindungslänge, bei der noch eine Bindung ausgebildet wird, gerade vom Term für die Bindungslänge kompensiert wird. Der Term für die Energie der Disulfidbindung i lautet:

$$\mathcal{D} = -C + E_b(dis_i) + E_{\omega_1}(dis_i) + E_{\omega_2}(dis_i) + E_\delta(dis_i)$$

Die Disulfidbindung wirkt sich auch auf die bindungslosen Wechselwirkungen aus. Durch die Abspaltung zweier Wasserstoffatome werden alle Paare, die diese enthalten nicht mehr berücksichtigt. Weitere Paare müssen ausgeschlossen werden, da durch die Disulfidbindung neue Nachbarn erster und zweiter Ordnung entstehen. Die partielle Ladung der an der Disulfidbindung beteiligten Schwefelatome wird zu Null.

In der seriellen Version wurden diese Änderungen dadurch berücksichtigt, daß Ladungen und Parameter auf Null gesetzt wurden. Diese Änderungen wurden bei der nächsten Konformation wieder rückgängig gemacht, da diese i.a. andere Disulfidbrücken besitzt. Die Intervalle wurden trotz der auszuschliessenden Paare nicht verändert. Die zuviel berechneten Werte wurden bei der Berechnung des Disulfidteils der Energie wieder abgezogen. In der parallelen Version mußte anders vorgegangen werden, siehe Kapitel 7.

Kapitel 7

Parallelisierung

In Kapitel 4 wurde erwähnt, daß nach der Implementierung einer seriellen Version die Parallelisierung vorgenommen wurde, um den Genetischen Algorithmus zu beschleunigen. Hierzu stand der Parallelrechner *MASPAR-MP1216* zur Verfügung. Das Programmiermodell des *MASPAR*-Systems erlaubt es, aus einem bestehenden seriellen Programm die Teile auszulagern, die parallelisiert werden sollen.

Zunächst werden allgemeine Überlegungen zur Parallelisierung Genetischer Algorithmen angeführt. Dann folgt die Beschreibung des Hardwareaufbaus der *MASPAR* und des Programmiermodells, mit dem man auf ihr Programme entwickelt. Anschließend wird die bei dieser Anwendung durchgeführte Parallelisierung beschrieben. Hierbei wird erklärt, welche Änderungen von *GENESIS* dazu nötig waren.

7.1 Parallele Genetische Algorithmen

Genetische Algorithmen bieten sich zur Parallelisierung an. Die Berechnung der Fitneßwerte der Individuen geschieht vollkommen unabhängig voneinander. Die Fitneßwerte der gesamten Population werden nacheinander berechnet, bevor sie in den Selektionsentscheidungen verwendet werden. Durch die Unabhängigkeit der Berechnung der Fitneßwerte können sie auch gleichzeitig berechnet werden.

Dies ist das synchrone Modell. Es gibt eine zentrale Verwaltung, die die Berechnung des Fitneßwertes durch untergeordnete Einheiten parallel durchführen läßt. Erst wenn alle Fitneßwerte berechnet sind fährt die Zentrale mit der Berechnung fort. Sie führt die Selektion und die Anwendung der Genetischen Operatoren durch. Beim asynchronen Modell gibt es diese zentrale Verwaltung nicht. Die einzelnen Prozessoren greifen auf einen gemeinsamen Speicher zu, in dem die Population repräsentiert ist.

Bei dem Ansatz mit Subpopulationen wird die Suche entsprechend der Anzahl der verfügbaren Prozessoren mehrfach ausgeführt. Jede Suche benutzt ihre eigene Population. Durch die Entkoppelung der Populationen können sich mehrere verschiedene Lösungen ausbilden. Die durch lokale Suche gefundenen Lösungen können zwischen den Subpopulationen ausgetauscht werden. Eine Anwendung, die diesen Ansatz verwirklicht, findet man in [6].

In dieser Arbeit wurde das synchrone Modell gewählt. Es eignet sich für Probleme, bei denen die Berechnung der Fitneßwerte relativ zu den übrigen Berechnungen des Genetischen Algorithmus den größten Teil der Zeit in Anspruch nimmt. Für das synchrone Modell spricht ferner, daß die Berechnungszeiten für verschiedene Individuen gleich sind. Zudem ist dieses Modell dem Typ des verwendeten Parallelrechners angepaßt.

7.2 Die *MASPAR MP-1*

Die *MASPAR* gehört zur Klasse der *SIMD*-Parallelrechner. Das heißt, daß in jedem Prozessor des Parallelrechners derselbe Befehl ausgeführt wird, aber mit unterschiedlichen Daten. Das Programm hat die gleiche Gestalt, wie man es von seriellen Programmen kennt, es gibt also nur einen Kontrollfluß. Die angesprochenen Variablen sind jedoch in mehreren Instantiierungen vorhanden. Jede Instanz residiert in einem anderen Speicher. Jeder dieser Speicher enthält von jeder Variablen eine Instanz. Er bildet mit dem Datenprozessor, der die Berechnungen jeweils für die in ihm vorhandenen Variablenwerte durchführt, ein *Processing Element* (*PE*). Die *MASPAR* besteht aus einem Feld von 128×128 *PE's*. Ein *PE* enthält 16K Speicher. Der Hardwareaufbau der *MASPAR* ist in [27] beschrieben.

Der Prozessor eines *PE's* erhält den auszuführenden Befehl von der *Array Control Unit* (*ACU*). Sie enthält 128K Speicher für Variablen, die in jeder der parallelen Berechnungen den gleichen Wert besitzen. Variablen im *ACU*-Speicher werden *singulär*, solche im *PE*-Speicher werden *plural* genannt und im Programmtext durch entsprechende Schlüsselwörter gekennzeichnet. Singuläre und plurale Variablen können in Berechnungen gemischt verwendet werden, nur die Zuweisung eines pluralen Rechenergebnisses an eine singuläre Variable macht keinen Sinn.

Zwischen den *PE's* können Daten ausgetauscht werden. Dafür stehen zwei Mechanismen zur Verfügung. Für die Kommunikation mit den direkten acht Nachbarn eines *PE's* gibt es eine schnelle Verbindung. Für die Kommunikation zwischen beliebigen *PE's* gibt es eine langsamere Verbindung. Zwischen der *ACU* und den *PE's* werden Daten durch Variablenzuweisungen ausgetauscht.

Die *MASPAR* ist mit einem seriellen *UNIX*-Rechner verbunden. Auf diesem sogenannten *Front End* werden die Programme entwickelt und übersetzt. Die Programme werden vom Benutzer auf dem Front End gestartet und starten ihrerseits die Ausführung auf der *MASPAR*.

Das übliche Vorgehen bei der Parallelisierung eines bereits existierenden Programms ist die Auslagerung der Teile , deren Parallelisierung eine Zeitersparnis bringt. Es laufen also zwei Programme auf zwei Rechnern, die entsprechend synchronisiert werden. Der serielle Teil läuft auf dem Front End, der parallele Teil auf der *MASPAR*. Beim synchronen Programmiermodell ist immer nur ein Rechner aktiv. Die Kontrolle wird z.B. vom Front End an die *MASPAR* übergeben und kehrt nach dem Ende des parallelen Abschnitts zum Front End zurück. Beim asynchronen Modell sind beide Rechner gleichzeitig aktiv und die Synchronisation wird von den Programmteilen übernommen.

Die seriellen und parallelen Programmteile werden vom Übersetzer anhand der Erweiterung des Dateinamens unterschieden. Aus den Objektdateien für *Front End* und *MASPAR* erzeugt der Binder ein ausführbares Programm, daß der Benutzer auf dem *Front End* startet.

Der Übergang der Kontrolle von einem Rechner auf den anderen wird durch den Aufruf einer Funktion des jeweils anderen Teiles bewirkt. Die Programme können immer nur auf Daten im Speicher des Rechners zugreifen, auf dem sie ablaufen. Benötigt eine Funktion

Daten, die aus dem Adreßraum des anderen Rechners stammen, dann müssen diese vorher übertragen werden. Werden die Ergebnisse des Funktionsaufrufs in dem aufrufenden Programmteil benötigt, dann müssen sie anschließend zurückübertragen werden.

Es ist möglich, sowohl zwischen *Front End*-Speicher und *ACU*-Speicher als auch zwischen *Front End*-Speicher und *PE*-Speicher Daten auszutauschen. Im letzteren Fall liegen die Daten auf dem *Front End* in einem zusammenhängenden Speicherbereich. Zur Übertragung auf bzw. von den *PE's* werden sie zu gleichen Teilen aufgeteilt bzw. zusammengefaßt.

Sowohl für den Funktionsaufruf als auch für die Datenübertragung müssen Funktionsnamen und Variablennamen des jeweils anderen Rechners angesprochen werden. Sie werden im Programmtext durch ein spezielles Schlüsselwort gekennzeichnet und dadurch in anderen Programmteilen bekannt.

Für die *MASPAR* stehen zwei Programmiersprachen zur Verfügung. Die *Maspar Parallel Application Language MPL* und *PARALLAXIS*. *MPL*[1] ist ein C-Derivat, daß Elemente enthält, mit denen die speziellen Eigenschaften der *MASPAR* genutzt werden können. *PARALLAXIS* ist eine Programmiersprache für das maschinenunabhängige parallele Programmieren. Sie erlaubt, von der tatsächlichen Anzahl von *PE's* abzusehen und eine dem Problem angepaßte Konfiguration von virtuellen *PE's* zu benutzen. Die Ausführung des Programms kann auf einem seriellen Rechner simuliert werden.

In dieser Arbeit wurde *MPL* gewählt. Sowohl *GENESIS* als auch die serielle Version der Energieberechnung sind in C implementiert. Nach dem Programmiermodell der Auslagerung des parallelen Teils konnte der serielle Teil unverändert auf dem *Front End* laufen, während der ausgelagerte Teil entsprechend den Sprachmitteln von *MPL* parallelisiert wurde.

7.3 Parallelisierung des Genetischen Algorithmus

Zunächst wird das allgemeine Vorgehen bei der Auslagerung der Fitneßberechnung erklärt. Anschließend wird erklärt, welche der in den Funktionen zur Initialisierung, zur Berechnung der Koordinaten und zur Energieberechnung benutzten Variablen singulär und welche plural gewählt wurden. Hierbei mußte auch der begrenzte Speicher beachtet werden. Schliesslich wird auf die Änderungen eingegangen, die durch die Tatsache nötig wurden, daß sich bei verschiedenen Konformationen unterschiedliche Disulfidbrücken ergeben.

7.3.1 Parallele Fitneßberechnung

Die Fitneßberechnung wird aus dem seriellen Programm ausgelagert. Die Fitneßfunktion besteht aus der Initialisierung, der Koordinatenberechnung und der Energieberechnung. Die Koordinaten und Energien der verschiedenen Proteinkonformationen werden parallel berechnet. Jedes *PE* berechnet die Energie eines Proteins. Als Populationsgröße wurde die Anzahl der *PE's* gewählt. Dadurch wird die parallele Energieberechnung einmal pro Generation aufgerufen.

In der seriellen Version werden in einer Schleife nacheinander die Genotypen der Individuen an die vom Wrapper erzeugte Schnittstellenfunktion übergeben. Diese Funktion

[1] Siehe [23, 24, 26]

dekodiert den Genotyp und ruft die Fitneßfunktion mit dem Phänotyp auf. Sie liefert den berechneten Fitneßwert zurück, der dann bei dem Individuum eingetragen wird.

Bei der parallelen Version wird diese Schleife durch den Aufruf der parallelen Version der Schnittstellenfunktion ersetzt, die dann auf der *MASPAR* abläuft. Der mehrfache Aufruf wird also durch einen Aufruf der parallelen Version ersetzt. Die parallele Version wird von der geänderten Version des Wrappers, dem Mpwrapper erzeugt.

Vor dem Aufruf der parallelen Schnittstellenfunktion werden die Genotypen der Population zur *MASPAR* übertragen. Nachdem die Schnittstellenfunktion auf der *MASPAR* terminiert ist, kehrt der Kontrollfluß zum Front End zurück. Jetzt werden die berechneten Fitneßwerte von der *MASPAR* zum *Front End* übertragen und in die Population eingetragen. Danach wird wie in der seriellen Version fortgefahren.

Die parallele Schnittstellenfunktion dekodiert die zuvor übertragenen Genotypen parallel. Im Unterschied zur seriellen Version wird der erhaltene Phänotyp nicht nur temporär gespeichert, sondern im *PE*-Speicher abgelegt. Bei der seriellen Version wird nach jedem Aufruf der Schnittstellenfunktion die Liste der besten Individuen aktualisiert. Ersetzt das gerade evaluierte Individuum ein Element der Liste, dann wird sein Phänotyp benötigt. Da der Phänotyp des zuletzt dekodierten Genotyps temporär gespeichert wird, ist er verfügbar.

Bei der parallelen Version wird die Bestenliste erst nach der parallelen Evaluierung der ganzen Population aktualisiert. Deshalb müssen von allen Individuen die Phänotypen noch vorhanden sein. Die Bestenliste ist Bestandteil des seriellen Programmteils. Die Phänotypen werden also von der *MASPAR* zum *Front End* übertragen.

Die Funktionen zur Übertragung der Genotypen und Phänotypen werden vom Mpwrapper entsprechend der Schablone für die Argumente der Fitneßfunktion erzeugt. Sie sind Bestandteil des seriellen Programmteils. Deshalb erzeugt der Mpwrapper zwei Dateien, eine für die parallele Schnittstellenfunktion und eine für die Übertragungsfunktionen.

7.3.2 Singuläre und plurale Daten

Bei der Berechnung der Energiewerte der Konformationen unterscheiden sich nur die Atomkoordinaten. Diese sind plural und befinden sich in den *PE*-Speichern. Die verwendeten Parameter der Energiefunktion und die Intervalle zur Erzeugung der Atompaare sind für alle Konformationen gleich. Für den Aufbau des für alle Konformationen gleichen Gerüstes des Proteins wird jeweils derselbe Building Block verwendet. Alle diese Daten werden singular gewählt und befinden sich im *ACU*-Speicher.

Die Speichergröße der *ACU* von 128K führte zu Platzproblemen. Daraufhin wurden nur noch die Building Blocks eingelesen, deren Aminosäure in der Sequenz des Proteins vorkam. Vor allem wird nur von jeweils einer Aminosäure die Anfangs- und die Endversion benötigt. Der dynamisch angeforderte Speicher für die Intervalle wurde nachträglich dem je nach den auszuschließenden Paaren tatsächlich nötigen Bedarf angepaßt.

Für die Koordinaten stehen 16K *PE*-Speicher zur Verfügung. Es wurden nur die Koordinaten von Atomen repräsentiert, die an der Berechnung der Energie teilnehmen. Die Repräsentation aller Atome wird durch die serielle Version bei der Auswertung der Ausgabedateien erzeugt. Für die Proteine mit denen die Testläufe durchgeführt wurden, reichte der *PE*-Speicher aus. Die Koordinaten eines Atoms bestehen aus drei Gleitkommazahlen. Ihr Speicherbedarf beträgt 12 Bytes. Die folgende Tabelle listet den Platzbedarf der Koordinaten. Ein Teil des *PE*-Speichers muß für lokale plurale Variablen verfügbar bleiben.

Sequenz	Aminosäuren	Atome	Bytes	Kbytes
pdb1cbh	36	323	3876	3.78
pdb1crn	46	402	4824	4.71
pdb1fdx	54	450	5400	5.27
pdb2ovo	56	519	6228	6.08

Die Funktion zur Lösung des linearen Gleichungssystems bei der Anpassung der Building Blocks wurde parallelisiert. Zweidimensionale Felder sind als Felder von Zeilenzeigern repräsentiert. Die Zeilenzeiger wurden plural gewählt. Der verwendete Algorithmus bedingt, daß möglicherweise auf den *PE's* auf verschiedene Feldelemente zugegriffen wird. Das bedeutet, daß durch die pluralen Zeilenindizes auf verschiedene Zeilenzeiger zugegriffen wird. Wären die Zeilenzeiger singulär, dann lägen sie im *ACU*-Speicher und es könnte nur nacheinander auf sie zugegriffen werden. Deshalb wurden sie plural gewählt.

7.3.3 Parallele Disulfidberechnung

Die Energie von vielen Konformationen wird parallel berechnet. In den unterschiedlichen Konformationen bilden sich verschiedene Disulfidbindungen aus. Die Parameter der Energiefunktion und die Ladungen der Atome sind singulär. Sie können nicht wie bei der seriellen Version verändert werden, da sie für alle Konformationen verwendet werden.

Außerdem hat sich gezeigt, daß das Abziehen von zuviel berechneten Beiträgen zu falschen Ergebnissen führen kann. Die Schwefelatome der Disulfidbindung kommen sich sehr nahe. Der Beitrag zur Van-der-Waals-Energie kann so groß werden, daß die berechnete Energie der anderen Paare bei der Addition weggerundet wird. Wird der Wert wieder abgezogen, dann bleibt null übrig. Deshalb werden diese Paare von vornherein nicht berechnet.

Der geschilderte Effekt war ein Hinweis, daß bei den unterschiedlichen Größenordnungen der Beiträge die Reihenfolge der Summation das Ergebnis beeinflußt. Deshalb werden für jedes Atom zuerst die Beiträge mit den Partneratomen mit den größten Indizes berechnet. Das sind die Atompaare mit den größten Abständen. Wenn die Abstände kleiner werden, haben sich die kleinen Beiträge schon aufsummiert, so daß der Rundungsfehler kleiner ist. Die Beiträge der Paare jeweils eines Atoms werden zur Gesamtsumme addiert, so daß beim nächsten Atom wieder neu mit der Summation begonnen wird.

Statt der Änderung von Parametern und Ladungen werden neue Intervalle gebildet, so daß zusammen mit Flags entschieden wird, ob der Beitrag eines Paares berücksichtigt wird. Aus den Intervallen der seriellen Version werden drei Mengen von Intervallen berechnet. Die erste Intervallmenge erzeugt nur Paare, die keine Schwefel- und Wasserstoffatome aus Cysteinen enthalten. Diese Paare ändern sich nicht durch Disulfidbindungen.

Die zweite Intervallmenge erzeugt alle Paare, die ein Schwefelatom aus einem Cystein enthalten. Die Partneratome sind keine Schwefel- oder Wasserstoffatome. Das Schwefelatom steht an der ersten Stelle der Paare. Die dritte Intervallmenge erzeugt alle Paare, die ein Wasserstoffatom aus einem Cystein enthalten. Die Partneratome sind keine Schwefel- oder Wasserstoffatome. Das Wasserstoffatom steht an der ersten Stelle der Paare.

Für jede Konformation geben plurale Flags für die Schwefel- und Wasserstoffatome an, ob sie von einer Disulfidbindung betroffen sind. Diese Flags werden entsprechend der Entscheidung, eine Disulfidbindung auszubilden, gesetzt. Die Flags bestimmen, ob der Energiebeitrag der entsprechenden Paare berücksichtigt wird.

Für die verbleibenden Kombinationen der Schwefel- und Wasserstoffatome untereinander wird anhand der vorhandenen Disulfidbrücken entschieden, ob das betreffende Paar einen Energiebeitrag liefert.

Kapitel 8

Testläufe

Dieses Kapitel beschreibt die auf der *MASPAR* durchgeführten Testläufe. Zunächst wird erläutert, warum die in *GENESIS* implementierten Selektionsverfahren für die Anwendung auf Proteine ungeeignet sind. Dann werden die verschiedenen Versionen des Genetischen Algorithmus vorgestellt, mit denen die Testläufe durchgeführt wurden. Anschließend werden die Proteine genannt, deren Energie minimiert wurde. Schließlich werden für jeden Testlauf die gewählten Werte aller Parameter genannt.

8.1 Exponentielle Energiewerte

Während der ersten Testläufe hat sich eine Eigenheit der Energiefunktion als problematisch herausgestellt. Es zeigte sich keinerlei Konvergenz der Bitstrings[1]. Das sprach für einen unzureichenden Selektionsdruck. Die Ursache waren die exponentiell steigenden Energiewerte schlechter Faltungen.

Der Van-der-Waals-Term steigt in der zwölften Potenz, wenn sich Atome sehr nahe kommen. Durch das Crossover werden Moleküle aus Teilen anderer Moleküle zusammengesetzt. Dadurch entstehen in jeder Generation mehrere Moleküle, bei denen sich Atome nahe kommen bzw. miteinander kollidieren. Dies drückt sich durch eine sehr große Energie aus.

Um auch in einer Population mit solchen Ausrutschern einen ausreichenden Selektionsdruck zu erhalten werden Skalierungsverfahren[2] angewendet. Diese basieren alle auf einer Mittelwertbildung der Fitneßwerte. Es hat sich herausgestellt, daß diese Verfahren bei Ausrutschern, die in dieser exponentiellen Größenordnung liegen, ungeeignet sind. Der Mittelwert wird schon durch einen Ausrutscher so stark erhöht, daß die Unterschiede zwischen den restlichen Individuen verschwinden. Im Extremfall führt das dazu, daß alle Individuen identische Erwartungswerte für die Anzahl ihrer Nachkommen erhalten.

Abbildung 8.1 zeigt eine typische Verteilung der Energiewerte nach der Initialisierung in der ersten Generation. Im linken Schaubild sind die Energiewerte mit logarithmischer Skalierung über den Individuen aufgetragen. Im rechten Schaubild sind die Individuen vorher nach ihren Energiewerten sortiert worden. Der kleinste Wert ist $1.82142 * 10^8$,der größte Wert ist $2.05644 * 10^{30}$ und der Mittelwert beträgt $1.95896 * 10^{26}$.

[1] Siehe Abschnitt 9.1.1

[2] siehe Abschnitt 8.1

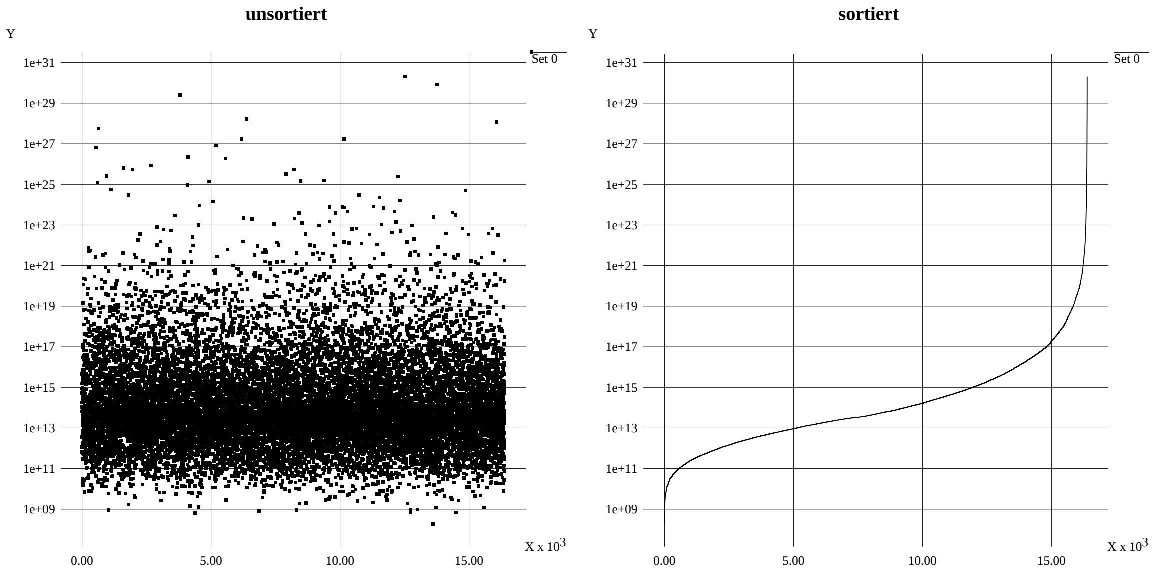


Abbildung 8.1: Verteilung

Genetische Algorithmen bestimmen die Erwartungswerte mittels der Fitneßwerte. Das Ziel ist diese Fitneß zu maximieren. Setzt man Genetische Algorithmen zur Minimierung ein, dann müßen die Werte der Bewertungsfunktion, genannt Zielwerte, in Fitneßwerte umgerechnet werden. *GENESIS* minimiert die Zielwerte. Deshalb wird ein Schrankenwert bestimmt, von dem die Zielwerte subtrahiert werden, um Fitneßwerte zu erhalten. Der Beste, mit dem kleinsten Zielwert, hat dann den größten Fitneßwert. Mit diesen Fitneßwerten werden dann die Erwartungswerte entsprechend der relativen Fitneß berechnet.

Von der Wahl des Schrankenwertes hängen die Fitneßwerte und damit ihr Durchschnitt ab. Somit hängen die relativen Fitneßwerte von ihm ab. Mit ihm kann der Selektionsdruck eingestellt werden. Je größer er gewählt wird, desto kleiner werden die Unterschiede der relativen Fitness. Um den Selektionsdruck zu erhöhen senkt man den Schrankenwert. Senkt man ihn unter den größten Zielwert ab, dann müssen die entstehenden negativen Fitneßwerte auf null gesetzt werden.

Die folgende Aufzählung nennt zwei Verfahren mit denen wahlweise bei *GENESIS* der Schrankenwert bestimmt wird. Das dritte Verfahren wurde zusätzlich implementiert. Anschließend werden mit den Werten der obigen Verteilung die Erwartungswerte berechnet, die die Individuen bei den verschiedenen Verfahren erhalten.

Window Scaling Als Schranke wird der größte Wert der n letzten Generationen verwendet. n wird in der Eingabedatei angegeben. Im Fall von $n = 1$ ist der größte Wert der aktuellen Generation die Schranke. So entstehen zwar keine negativen Werte, aber ein Ausreißer kann den Selektionsdruck beeinträchtigen.

Sigma Scaling Neben dem Mittelwert μ wird auch die Standardabweichung σ bestimmt. Die Schranke S ergibt sich nach der Formel $S = \mu + f * \sigma$. Der Faktor f wird von der Eingabedatei gelesen. Ist $f = 1$ dann ist der aus dem Fitneßwert $F_i = S - Z_i$ berechnete Erwartungswert $E_i = F_i/\mu$, der einem Zielwert $Z_i = \mu - \sigma$

zugeordnet wird, doppelt so groß wie der eines durchschnittlichen Zielwertes $Z_j = \mu$. Mit wachsendem f nimmt der Selektionsdruck ab.

Linear Scaling Die Zielwerte werden mit der linearen Skalierfunktion $S(x) = ax + b$ in Fitneßwerte umgerechnet. Die Parameter a und b hängen vom Mittelwert μ und vom kleinsten Zielwert z_{min} ab. Sie werden nach folgenden Formeln berechnet:

$$a = -\frac{\mu}{\mu - z_{min}} \quad , \quad b = (1-a)\mu$$

Durch $S(x)$ wird der Mittelwert auf sich selbst und der minimale Wert auf das doppelte des Mittelwerts abgebildet:

$$S(\mu) = a\mu + (1-a)\mu = \mu$$
$$S(z_{min}) = az_{min} + (1-a)\mu = \mu + a(z_{min} - \mu) = \mu + \mu\frac{z_{min}-\mu}{z_{min}-\mu} = 2\mu$$

Das beste Individuum hat also den doppelten Fitneßwert eines durchschnittlichen. Negative Werte werden auf null gesetzt. Die Erwartungswerte ergeben sich aus den relativen Fitneßwerten.

Beim Window Scaling erhält das beste Individuum einen Erwartungswert von 1.0001, und ein durchschnittliches Individuum einen Erwartungswert von 1. Wie man aus der Abbildung 8.1 entnimmt, liegen die Zielwerte fast der ganzen Population zwischen dem minimalen und dem durchschnittlichen Wert. Ihre Erwartungswerte liegen damit in dem Intervall $[1, 1.0001]$.

Beim Sigma Scaling ist die Standardabweichung mit $1.7513 * 10^{28}$ sogar größer als der Mittelwert $\mu = 1.95896 * 10^{26}$. Das heißt, daß ein Individuum, um einen Erwartungswert 2 zu erhalten, einen Zielwert kleiner als null haben müßte. Das beste Individuum erhält den Erwartungswert 1.00032, ein durchschnittliches 0.989254.

Beim Linearen Scaling entspricht der Parameter a der Steigung der Skalierungsgerade. Sie wird als -1 berechnet. Der Parameter b hat den Wert $3.91792 * 10^{26}$, das ist das Doppelte von μ. Diese Skalierungsfunktion entspricht einer Subtraktion von der Schranke $2 * \mu$. Der größte Erwartungswert ist 1.00074. Der kleinste Erwartungswert beträgt zwar 0.415808, aber trotzdem besitzen die meisten Individuen fast den gleichen Erwartungswert. 10 Zielwerte sind größer als die Schranke und ergeben einen Erwartungswert von null. Außer ihnen besitzen lediglich 16 Individuen einen Erwartungswert, der nicht in dem Intervall $[0.99, 1.00074]$ liegt. Das heißt, daß fast die ganze Population einen Erwartungswert von ca. 1 besitzt.

Die mit diesen drei Skalierungsverfahren erhaltenen Erwartungswerte erzeugten keinen ausreichenden Selektionsdruck, so daß keine Konvergenz eintrat.

Ein Versuch zur Lösung des Problems war die Logarithmierung der Werte. Dies sollte die Abstände zu den exponentiell großen Werten stärker verringern, als die zwischen den kleineren Werten. Das brachte nicht den gewünschten Erfolg, siehe 9. Ein weiterer Versuch war die Veränderung der Energiefunktion. Bei dieser Version steigt die Energie linear statt exponentiell, wenn der Atomabstand kleiner als $1\mathring{A}$ wird. Dadurch sollten die exponentiellen Ausrutscher verhindert werden. Auch mit diesem Ansatz stellte sich keine Konvergenz ein, siehe Kapitel 9.

Das Problem wurde dadurch gelöst, daß ein Rankingverfahren verwendet wurde. Der Erwartungswert eines Individuums richtet sich nicht nach einem Verhältnis von Fitneßwerten, sondern nach seinem Rang i in der Population. Die Rangfolge wird durch Sortieren der Individuen nach ihren Fitneßwerten bestimmt. Das Verfahren heißt $(\mu, \lambda)-LinearRanking$ siehe [5]. Die Formeln lauten:

$$p(a_i) = \begin{cases} \frac{1}{\mu}\left(\eta_{max} - 2(\eta_{max} - 1)\frac{i-1}{\mu-1}\right) & ,1 \leq i \leq \mu \\ 0 & ,\mu < i \leq \lambda \end{cases} \quad \eta_i = p(a_i) * \lambda \quad \eta_{best} = \frac{\lambda}{\mu} * \eta_{max}$$

Die Selektionswahrscheinlichkeiten $p(a_i)$ multipliziert mit der Größe der Population λ ergeben die Erwartungswerte η_i. Eine Anzahl der schlechtesten Individuen $\lambda - \mu$ erhält keine Nachkommen $p(a_i) = 0$. Der Rest der Population μ erhält beginnend beim Besten $\eta_{best} = \eta_1$ absteigende Erwartungswerte. Der Unterschied von einem Rang zum folgenden ist gleichbleibend und kann durch die Wahl von η_{max} von null ($\eta_{max} = 1$) bis zu einem Maximalwert ($\eta_{max} = 2$) verändert werden. Dadurch kann der Selektionsdruck eingestellt werden, wobei die Summe aller Erwartungswerte immer gleich der Populationsgröße λ ist.

8.2 Die Versionen des Genetischen Algorithmus

Um verschiedene Versionen des Genetischen Algorithmus testen zu können, wurden an *GENESIS* Änderungen vorgenommen. Im *GENESIS*-Manual wird für Erweiterungen vorgeschlagen, die geänderten Teile zu ersetzen und ein neues Programm zu erzeugen. Das würde bedeuten, daß für jede Version ein eigenes Programm erzeugt würde. Stattdessen wird durch zusätzliche Einträge in der Eingabedatei zwischen den Versionen gewählt. Dazu enthält das Programm alle Teile der verschiedenen Versionen. Beim Einlesen der Eingabedatei werden Funktionsvariablen mit der der Version entsprechenden Funktion besetzt. Außerdem werden zusätzlich nötige Parameter eingelesen. Das hat den Vorteil, daß für jeden Testlauf die Version und die Wahl aller Parameter in der Eingabedatei festgehalten ist. Außerdem gibt es so nur für unterschiedliche Proteine unterschiedliche Programme.

Die Versionen unterscheiden sich erstens durch die Wahl des Crossover-Operators. Zweitens wird die Funktion *measure* ausgetauscht, die Werte für die jeweilige Selektion berechnet. Schliesslich wird die Funktion *selection* entsprechend dem Selektionsverfahren ersetzt.

Die verschiedenen Versionen sind:

default Dies ist die ursprüngliche Version von GENESIS. Ihre Verwendung bei Proteinen ist aus den in Abschnitt 8.1 genannten Gründen nicht sinnvoll.

log Die Fitneßwerte werden logarithmiert. Das Selektionsverfahren bleibt das von *default*.

loglin Die logarithmierten Werte werden bei der Selektion linear skaliert, siehe Abschnitt 8.1.

top Zur topologischen Version siehe Abschnitt 8.3. Als Selektionsverfahren wird (μ, λ)-Linear Ranking verwendet. Die Parameter μ und η_{max} sowie die Anzahl der Nachbarschaftsringe wird eingelesen.

sch Die Kachel-Version ist eine Variante der topologischen Version, siehe Abschnitt 8.3. Die Parameter μ und η_{max} sowie die Größe der Kacheln wird eingelesen.

rank Bei dieser Version des Ranking ist der Selektionsdruck maximal und hängt von der Anzahl der Ränge ab, siehe Abschnitt 8.1.

ml Beim (μ, λ)-Linear Ranking kann der Selektionsdruck durch den Parameter η_{max} eingestellt werden, siehe 8.1.

8.3 Topologische Versionen

Beim herkömmlichen Genetischen Algorithmus besitzt die Population keine topologische Struktur. Sie ist eine Menge von Individuen, in der die Anordnung keine Rolle spielt. Für das Crossover wird beliebig zufällig aus ihr ausgewählt. Bei der Selektion wird eine neue Population gebildet, in der die Anordnung der Individuen nicht von der in der Vorgängerpopulation abhängt.

In der Natur spielt der Ort eine wichtige Rolle. Vererbungsvorgänge finden hauptsächlich lokal statt. Die Nachkommen verteilen sich ausgehend von ihrem Entstehungsort. Die Fitneß ist relativ zur Fitneß der Individuen der unmittelbaren Umgebung. Das führt dazu, daß sich gebietsweise verschiedene Lösungen ausbilden können, die in keinem direkten Wettbewerb stehen. Es werden vor allem zwischen Mitgliedern der gleichen Lösung Erbinformationen ausgetauscht.

Man kann bei Genetischen Algorithmen eine topologische Struktur der Population nachbilden. Es ist zu erwarten, daß so auch stark unterschiedliche Lösungen lange nebeneinander existieren können. Der Nachteil der schlechteren Lösungen wirkt sich nicht so schnell aus. In Suchräumen mit sehr vielen fast gleichwertigen Lösungen muß sich der Genetische Algorithmus also nicht auf eine festlegen. Die Vielfalt in der Population bleibt länger erhalten und man erhält mehrere Lösungsvorschläge.

Die topologischen Versionen *top* und *sch* ordnen die Population auf einem Torus an. Die Fitneß wird relativ zu den Nachbarindividuen berechnet. Die Nachkommen besetzen Nachbarfelder ihrer Eltern. Der nötige Platz wird geschaffen, indem die Individuen in der Richtung der nächstgelegenen Lücke verschoben werden, die ein Individuum ohne Nachkommen hinterlassen hat. Der Abstand wird auf dem Torus, also in vier Richtungen bestimmt. Durch den Schiebevorgang werden andere Lücken und Eltern mitverschoben.

Das Crossover wird immer zwischen direkten Nachbarn ausgeführt. Bei der herkömlichen, nicht angeordneten Population, wird ein zusammenhängender Teil der Population gekreuzt, dessen Größe durch die Crossoverrate bestimmt wird. Da die Population bei der topologischen Selektion nicht durchmischt wird, werden die Crossover-Partner über die Population verteilt gewählt. Der erste Partner wird unter den noch nicht benutzten Individuen zufällig gewählt. Der zweite Partner wird unter den direkten Nachbarn gewählt, wobei jene ausgelassen werden, die schon einmal benutzt wurden oder ein Nachkomme des ersten Partners sind. Gibt es keinen solchen Nachbarn, dann wird der zuletzt überprüfte benutzt.

Die Versionen unterscheiden sich in der Art der Berechnung der Fitneß relativ zu den Nachbarn. Es wird eine Menge von Nachbarn ausgewählt. Die Fitneß des Individuums wird durch den Quotienten seiner Fitneß mit der Summe der Fitneßwerte aller Nachbarn ersetzt. Bei der Version *top* gibt der Parameter *Nachbarschaftsringe* an, bis zu welchem Abstand andere Individuen zur Nachbarschaft gehören. Bei der Version *sch* ist der Torus in Kacheln gleicher Kantenlänge aufgeteilt. Die Individuen einer Kachel bilden für jedes Individuum auf ihr die Nachbarschaft.

Protein	Aminosäuren	Atome	Drehwinkel	Disulfidbrücken	Helix	Faltblatt
pdb1cbh	36	323	149	2		*
pdb1crn	46	402	190	3	*	*
pdb1fdx	54	450	223	0		
pdb2ovo	56	519	251	3	*	*

Tabelle 8.1: Testproteine

8.4 Testproteine und Parameter

Da die Berechnungszeit der Energie mit dem Quadrat der Anzahl der Atome steigt, wurden kurze Proteine verwendet, die Abschnitte mit Sekundärstrukturen enthalten. Bilden sich Sekundärstrukturen in den Lösungen aus, dann sind sie bei der optischen Beurteilung zu erkennen. Die Tabelle 8.1 nennt die Proteine, ihre Anzahl von Aminosäuren, Drehwinkeln und Disulfidbrücken. Vorhandene Sekundärstrukturen sind mit einem Stern markiert.

Die Crossoverwahrscheinlichkeit wurde einheitlich mit 0.6 gewählt. Das Programm von *GENESIS*, mit dem man die Parameter eines Testlaufs eingibt, berechnet nach der Formel:

$$p_m = \frac{\sqrt{\frac{2.72}{Stringl\ddot{a}nge}}}{2 * Populationsgr\ddot{o}\ss e}$$

einen Vorschlag für die Mutationswahrscheinlichkeit. Für die Längen der Testproteine ergibt sich immer der gerundete Wert von 0.000001 pro Bit. Für den Wert, ab dem ein Bit als konvergiert betrachtet wird, wurde einheitlich 0.95 gewählt.

In *GENESIS* wird ein Crossoveroperator mit zwei Crossoverpunkten verwendet. Beim Crossover der Proteinfaltungen wird der Drehwinkel verändert, in dem die Crossoverstelle liegt. Deshalb steigt mit der Anzahl der Crossoverpunkte die Wahrscheinlichkeit für Kollisionen. Für die Testläufe wurde der *GENESIS*-Crossoveroperator durch einen mit einem Crossoverpunkt ersetzt.

Die Tabelle 8.2 listet für jedes Testprotein die Testläufe, die Version des Genetischen Algorithmus und die gewählten Parameter auf. Die Namen der Testläufe identifizieren die Ein- und Ausgabedateien. "Ringe" steht für die Anzahl von Nachbarschaftsringen bei der Version *top*. "Kachel" steht für die Kantenlänge der Kacheln bei der Version *sch*.

Wegen der langen Laufzeiten war es nötig, die Restart-Option von *GENESIS* in Anspruch zu nehmen. Dazu wird in der Eingabedatei angegeben, daß die letzte Generation in einer Datei gesichert werden soll. GENESIS schreibt die Population in der entpackten Form, das heißt ein Zeichen pro Bit. Dies war bei der Größe der Population unmöglich, da z.B. der gepackte Genotyp von pdb1cbh 187 Bytes pro Individuum benötigt. Das wären $187 * 8 * 16384 = 24510464 = 24.5 Mbyte$ für eine Population. Die Funktion wurde abgeändert, so daß die Genotypen gepackt abgespeichert werden. Dann sind für pdb1cbh noch $3.1 Mbyte$ nötig.

Sequenz	Testname	Version	η_{max}	μ	η_{best}	Ränge	Ringe	Kachel
pdb1crn	ml1	ml	2.0	13000	2.52			
	ml2	ml	2.0	13000	2.52			
	ml3	ml	2.0	13000	2.52			
	ml4	ml	2.0	13000	2.52			
	ranksc3	rank				13000		
	ranksc4	rank				13000		
	ranksc5	rank				13000		
	ranksc6	rank				13000		
	sch1	sch	2.0	13000	2.52			8
	sch2	sch	2.0	13000	2.52			32
	top2	top	2.0	13000	2.52		1	
	top3	top	2.0	15000	2.18		3	
pdb1fdx	fdx1	ml	2.0	13000	2.52			
	fdx2	ml	1.6	13000	2.02			
	fdx3	ml	1.8	13000	2.27			
	fdx4	ml	2.0	15000	2.18			
	sch3	sch	1.8	15000	1.97			32
pdb2ovo	ovo1	ml	2.0	13000	2.52			
	ovo2	ml	2.0	15000	2.18			
	ovo3	ml	1.8	15000	1.97			

η_{max} ... Selektionsdruck
μ ... Individuen mit Erwartungswert > 0
η_{best} ... Erwartungswert des Besten

Tabelle 8.2: Testläufe

Kapitel 9

Ergebnisse

Die Ergebnisse der Testläufe liegen in Form verschiedener Dateien vor. Diese Dateien werden durch ein Auswertungsprogramm nach bestimmten Kriterien interpretiert, um die berechneten Lösungen zu beurteilen. Zunächst wird der Inhalt der Ergebnisdateien beschrieben. Dann werden die Berechnungen des Auswertungsprogramms erläutert. Es folgen die Ergebnisse für die einzelnen Testläufe.

9.1 Ergebnisdateien

9.1.1 Ausgabedatei

GENESIS gibt nach jeder Generation in eine Ausgabedatei statistische Werte über die Konvergenz der Bitstrings und die Werte der Zielfunktion aus. Die Werte sind:

Generation Dieser Wert ist die Nummer der Generation.

Evaluierungen Dieser Wert gibt die Anzahl der bisherigen Berechnungen der Zielfunktion an. Da die Populationsgröße einheitlich ist, beträgt der Wert das 16384-fache der Generationsnummer.

Verlorene Bits Dieser Wert entspricht der Anzahl der Bits, die in der ganzen Population denselben Wert besitzen. Sie können nur noch durch Mutationen und nicht mehr durch das Crossover einen anderen Wert erhalten.

konvergierte Bits Für jedes Bit wird berechnet, in wieviel Prozent der Population der jeweils häufigere Wert vorkommt. In der Eingabedatei wird ein Prozentsatz angegeben, ab welchem Wert ein Bit als konvergiert angesehen wird. Der Wert in der Ergebnisdatei gibt die Anzahl der konvergierten Bits an.

Konvergenz Der für jedes Bit berechnete Wert der Konvergenz wird über alle Bits gemittelt. Die Konvergenz gibt die durchschnittliche prozentuale Konvergenz aller Bits an.

Online Performance Die Online Performance ist ein Maß für die Verbesserung der Zielfunktionswerte. Sie ist der Durchschnitt aller bisher berechneter Zielfunktionswerte.

Der Name leitet sich von der Bedeutung dieses Wertes ab, die dieser bei Online-Anwendungen von Genetischen Algorithmen besitzt. Dabei bedeutet jede Berechnung der Zielfunktion eine Interaktion mit einer Umgebung des Genetischen Algorithmus, an die dieser sich optimal anpassen soll. In diesem Fall ist es wichtiger schlechte Lösungen zu vermeiden als möglichst schnell die Lösungen zu verbessern.

Offline Performance Die Offline Performance ist ebenfalls ein Maß für die Verbesserung der Zielfunktionswerte. Ihr Name kommt von der Bedeutung des Wertes bei Offline-Anwendungen von Genetischen Algorithmen. Hier ist der Prozeß der Optimierung von der Anwendung der gefundenen Lösungen getrennt. Zwischenzeitlich erzeugte schlechte Lösungen kommen nicht in einer Interaktion mit einer Umgebung zur Anwendung. Das Hauptaugenmerk liegt auf der Verbesserung der bisher besten Lösung. Die Offline Performance ist der Durchschnitt der jeweils besten Zielfunktionswerte aller Generationen.

Bester Dieser Wert ist der beste, also kleinste Zielfunktionswert in dieser Generation.

Durchschnitt Dieser Wert ist der Durchschnitt aller Zielfunktionswerte dieser Generation.

9.1.2 Bestendatei

In der Datei der Besten werden die Phänotypen der besten Lösungen ausgegeben, die gefunden wurden. In der Eingabedatei wird angegeben, wieviele Beste gesichert werden sollen. Nach jeder Generation wird die Liste der Besten auf den neuesten Stand gebracht. Selbst wenn eine Lösung durch die Selektion oder die Genetischen Operatoren verloren geht, bleibt sie in der Liste erhalten, falls sie nicht durch bessere Lösungen ersetzt wird.

Als Erweiterung zu *GENESIS* kann in der Eingabedatei eine Liste von Generationen angegeben werden, in denen der Beste der Generation als Phänotyp in eine Datei ausgegeben wird. So kann nicht nur die Entwicklung der Zielfunktionswerte, sondern auch die der Lösungen verfolgt werden. Bei den topologischen Versionen kann eine Liste von Generationen angegeben werden, in denen die Zielfunktionswerte der ganzen Population in eine Datei ausgegeben werden. Hier ist die örtliche Verteilung der Zielfunktionswerte von Interesse.

9.2 Auswertungsprogramm

Das Auswertungsprogramm liest die Phänotypen der besten gefundenen Lösungen. Aus den Drehwinkeln werden die kartesischen Koordinaten erzeugt. Die Drehwinkel und die Loordinaten sind die Basis für die Berechnung von Ergebnisdaten, die in verschiedene Dateien ausgegeben werden.

9.2.1 Statistische Werte

Die Entwicklung der statistischen Werte der Konvergenz und der Werte der Zielfunktion werden graphisch dargestellt. Für das Schaubild der Konvergenz werden die Zähler der verlorenen und konvergierten Bits zur Länge des Binärstrings in Relation gesetzt. Die Werte der y-Achse sind somit Prozentzahlen. Die Werte der Zielfunktion werden mit logarithmischer y-Achse dargestellt, da sie während des Testlaufs ihre Größenordnung ändern.

Wenn gegen Ende eines Testlaufs die Werte des Besten über lange Zeit konstant bleiben ist die Population konvergiert. Trotz hoher durchschnittlicher Konvergenz können immer noch einige Bits nicht konvergiert sein. Wenn mehr als 90 Prozent aller Bits konvergiert sind, werden die Nummern der nicht konvergierten Bits in eine Datei ausgegeben. Das Auswertungsprogramm bestimmt anhand dieser Nummern, welche Drehwinkel nicht konvergiert sind.

9.2.2 *PDB*-Dateien

Die besten Lösungen eines Testlaufs werden im *PDB*-Format in Dateien ausgegeben. So können sie mit Programmen zum Molekularen Modellieren wie zum Beispiel *SYBYL* inspiziert werden. Ebenso wird mit den Dateien verfahren, die die Besten bestimmter Generationen enthalten. So kann bei der optischen Bewertung die Entwicklung des Besten verfolgt werden.

9.2.3 Ähnlichkeit von Konformationen

Zur Beurteilung der Güte der gefundenen Lösungen werden Ähnlichkeitsvergleiche mit den originalen Konformationen der Proteine durchgeführt. Verglichen werden die *RMS*-Werte und die Drehwinkel. Der erreichte minimale Energiewert der Lösung wird mit der Energie des Originalproteins verglichen. Die Energie des Originals wird mit der Energiefunktion berechnet, die in dem Genetischen Algorithmus verwendet wird.

Die Winkel und *RMS*-Werte der Originalkonformationen wurden aus der *PDB*-Datei der Proteindatenbank berechnet. Dazu wurden die Koordinaten der *PDB*-Datei in die Datenstruktur zur Darstellung eines Proteins eingelesen. Die *RMS*-Werte werden direkt aus den Koordinaten bestimmt. Für die Berechnung der Drehwinkel werden die Angaben in den Building Blocks verwendet. Sie geben die Indizes der Atome an, die den Drehwinkel definieren. Für die Energieberechnung werden die in der *PDB*-Datei geladen vorliegenden Enden des Proteins in die ungeladene Form umgewandelt.

Root Mean Square

Der *RMS*-Wert ist ein Mittelwert der Abweichungen der Abstände der zentralen Kohlenstoffatome C_α. Er bildet also ein Maß für die Ähnlichkeit der Faltung des Rückgrats. Für die Konformation der Lösung K_l und die Konformation des Originals K_o werden die Abstände r_i^l und r_i^o zwischen den C_α Atomen berechnet. Diese Werte sind von der Lage im Raum unabhängig. Bei n C_α Atomen gibt es $m = n(n-1)/2$ Abstände r_i. Aus den Abständen r_i^l und r_i^o ergeben sich die Abweichungen $d_i = |r_i^l - r_i^o|$. Der *RMS*-Wert berechnet sich nach der Formel:

$$RMS(K_l, K_o) = \sqrt{\sum_{i=1}^{m} d_i^2}$$

RMS-Werte werden berechnet für den Vergleich der Lösungen mit den Originalen. Außerdem werden sie für den Vergleich von Lösungen untereinander berechnet. Diese geben Auskunft über die Ähnlichkeit der besten Lösungen. Schließlich werden für Lösungen aus verschiedenen Testläufen *RMS*-Werte bverechnet, um zu beurteilen, ob verschiedene Testläufe ähnliche Ergebnisse erzeugen.

Drehwinkel

Wenn nur Teile der Faltung übereinstimmen, dann kann das am *RMS*-Wert nicht direkt erkannt werden. Deshalb werden die Drehwinkel des Originals mit denen der Lösungen verglichen. Die Differenzen der Drehwinkel werden nach Rückgrat und Seitenketten getrennt in eine Datei ausgegeben. Wenn Teile der Lösung und des Originals ähnlich gefaltet sind, zeigt sich das durch kleine Differenzen.

Disulfidbrücken

Ein weitere Aussage über die Ähnlichkeit von Konformationen machen die Disulfidbrücken. Deshalb werden die Paare von Aminosäureindizes der Disulfidbrücken in den Lösungen denen des Originals gegenübergestellt. So ist zu erkennen, ob die korrekten Disulfidbrücken gefunden wurden, oder ob sich statt dessen andere, oder zusätzliche ausgebildet haben.

9.3 Ergebnisse der Testläufe

In den folgenden Abschnitten werden die Ergebnisse der Testläufe jeweils unter verschiedenen Aspekten betrachtet. Zunächst wird gezeigt, wie sich das Versagen der Standardselektionsmechanismen äußerte. Dann werden für die Testproteine für jeden Testlauf der beste erreichte Energiewert und der *RMS*-Wert der zugehörigen Konformation aufgelistet. Als weiteres Ähnlichkeitsmaß werden dann für einige der besten Lösungen die Abweichungen der Drehwinkel von denen im Original angegeben. Anschliessend werden zur optischen Beurteilung der Lösungen ihre Molekülgraphiken denen der Originale gegenübergestellt. Im folgenden Abschnitt zeigen Schaubilder den Verlauf der statistischen Daten. Dann wird der RMS-Wert zum Vergleich der erzeugten Lösungen untereinander herangezogen. Im nächsten Abschnitt wird anhand von Molekülgraphiken des Besten in bestimmten Generationen des Testlaufs die Entwicklung der Besten beurteilt. Schliesslich werden im letzten Abschnitt für die topologischen Versionen die Fitneßwerte aller Individuen einer Generation über ihrer Position in der quadratisch angeordneten Population angezeigt.

9.3.1 *GENESIS*-Selektion

In *GENESIS* hat man die Wahl zwischen Window-Skaling und Sigma-Skaling[1]. Bei Testläufen mit diesen Verfahren änderte sich bis zur 60. Generation die Konvergenz nicht. Sie blieb konstant bei den 50.3 Prozent der Initialisierungspopulation. Das zeigt, daß die unterschiedlichen Fitneßwerte zu keinen unterschiedlichen Selektionswahrscheinlichkeiten geführt haben. Da die Selektionsentscheidungen gleichverteilt zufällig sind, bleiben die Bits der Genotypen gleichverteilt.

Bei diesen Testläufen wurde eine Option von *GENESIS* verwendet, die sicherstellt, daß das bisher beste Individuum immer in der nächsten Generation vertreten ist. Die über Generationen unveränderte Kurve in Abbildung 9.1 zeigt, daß die Selektionsentscheidungen

[1]Siehe Abschnitt 8.1

während dieser Generationen zu keiner Verbesserung geführt haben. Das Crossover erzeugte zufällig verteilte neue Punkte im Suchraum. Die Verbesserungen der Fitness entstanden durch diese Zufallssuche.

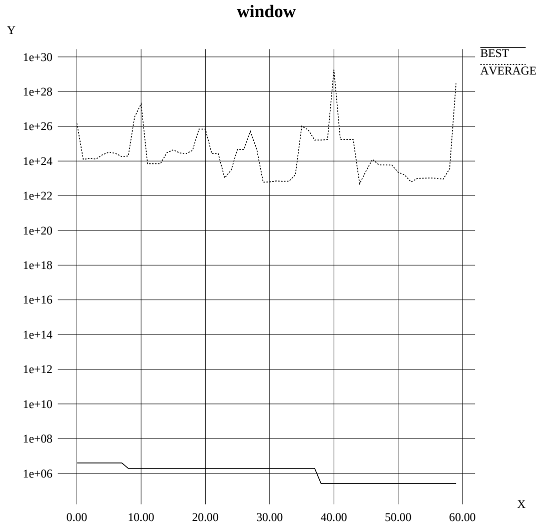


Abbildung 9.1: Window

Im Testlauf log4 wurden die Fitneßwerte logarithmiert bevor das Sigma-Skaling angewandt wurde[2]. Im Testlauf loglin1 wurden die logarithmierten Fitneßwerte linear skaliert. In Abbildung 9.2 ist zu erkennen, daß bis zur Generation 137 kein Bit konvergierte. Die Konvergenz stagniert, bevor die 60 Prozent überschritten sind. Abbildung 9.2 zeigt, daß das lineare Skaling gegenüber dem Sigma-Skaling zu einer höheren Konvergenz führte. Das erste konvergierte Bit tritt jedoch erst bei Generation 133 auf.
Bei den Testläufen cut4 und cut6 wurde der Anstieg der Energie begrenzt, indem ein Atomabstand kleiner als $1\AA$ zu einem linearen Anstieg führte. Der durchschnittliche Energiewert der ersten Generation von cut4 hat dadurch den Wert $1.24 * 10^8$, der von cut6 $3.87 * 10^9$, während dieser Wert ohne Beschränkung oberhalb $1 * 10^{20}$ liegt. Abbildung 9.3 zeigt, daß die Beschränkung der Energiewerte nicht ausreicht, um einen ausreichenden Selektionsdruck zu erzeugen.

Bei den folgenden Testläufen wurde als Selektionsverfahren das Ranking verwendet. Dadurch trat bei der Optimierung der Energiewerte eine ausreichende Konvergenz ein. Die Schaubilder des Abschnitts 9.3.5 zeigen den Verlauf der statistischen Werte.

9.3.2 Energie und *RMS* der Besten

Die Tabellen 9.1, 9.2, 9.3 und 9.4 listen für jedes Testprotein die Energiewerte und die *RMS*-Werte des Besten eines Testlaufes auf. In der ersten Zeile der Tabellen sind die Energiewerte der Originalkonformationen aufgeführt.

Die Energiewerte unterschreiten die Energie des Originals. Das heißt, daß die Originalkonformation bei dieser Energiefunktion nicht die Konformation mit der minimalen

[2]Siehe Abschnitt 8.1

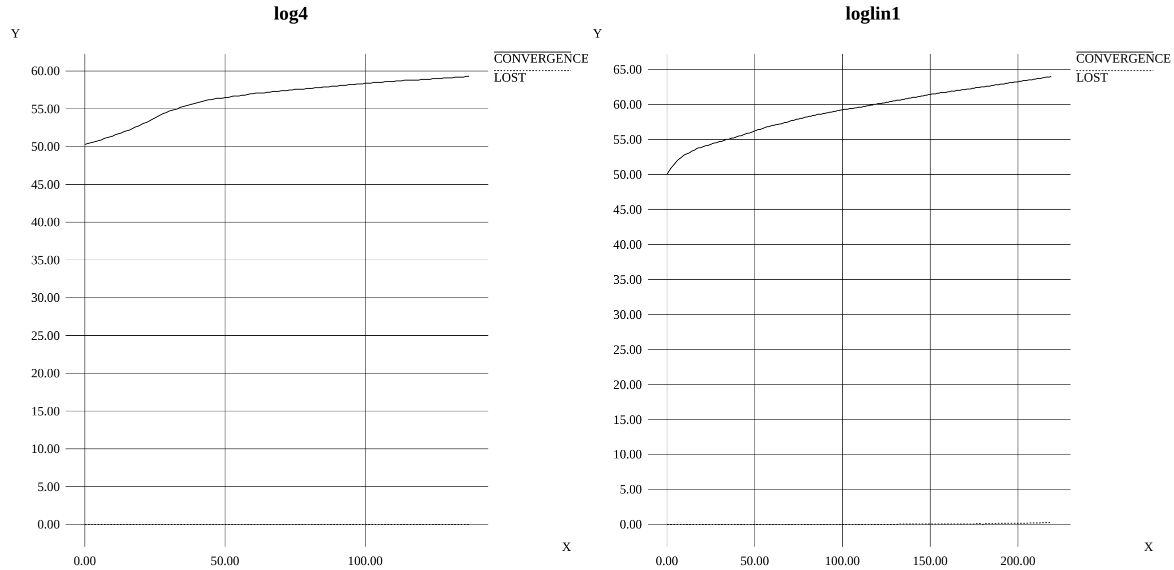


Abbildung 9.2: Logarithmierte Fitneß

Protein pdb1cbh			Energie 1347.18	
Test	Version	Generationen	Energie	RMS
cbh1	ml	499	453.18	8.28
cbh2	ml	299	463.33	8.93
cbh3	ml	288	488.62	12.24

Tabelle 9.1: pdb1cbh

Energie ist. Daraus folgt, daß die empirische Energiefunktion nicht die korrekten Energiewerte berechnet. Die hohen *RMS-Werte* zeigen große Unterschiede der Konformationen der Lösungen zu den Originalkonformationen an.

9.3.3 Winkel

Die Tabelle 9.5 listet für vier Testläufe, die jeweils mit verschiedenen Proteinen durchgeführt wurden, die Differenzen der Drehwinkel im Rückgrat zu denen im Original auf. Die Drehwinkel Φ, Ψ und ω sind im Abschnitt 5.6 definiert. Die Winkel des Originals wurden mit denen der besten Lösung verglichen. Die Werte sind im Bogenmaß angegeben. Die maximale Abweichung ist $\pm\pi$.

Es sind keine Bereiche zu erkennen, in denen bei mehreren aufeinander folgenden Aminosäuren die Drehwinkel kleine Abweichungen besitzen. Eine Abweichung von 10 Grad entspricht einem Wert von 0.17. Die ω-Winkel der starren Peptidbindung zeigen im Vergleich zu den Φ und Ψ-Winkeln keine kleineren Abweichungen. Die kleinsten Abweichungen besitzt der Ψ-Winkel. Die Tabellen 9.6 und 9.7 zeigen für die gleichen Testläufe die Differenzen der Drehwinkel in den Seitenketten.

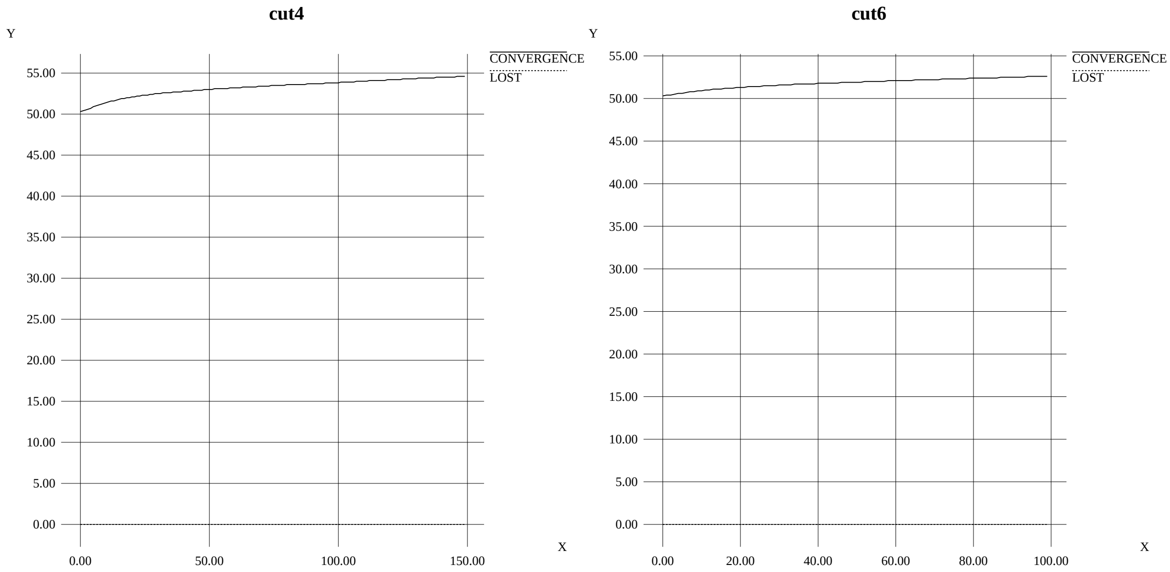


Abbildung 9.3: Begrenzte Energiewerte

9.3.4 Molekülgraphiken

Die Abbildungen 9.4, 9.5 und 9.6 zeigen die graphische Darstellung der Konformationen. Die Konformationen der besten Lösungen einiger Testläufe werden denen des Originals gegenübergestellt. Die Graphiken sind von *SYBYL* erzeugt. Zur Verdeutlichung ist die Faltung des Rückgrats hervorgehoben. Bei den Lösungen sind keine Helizes zu beobachten. Es fällt auf, daß die Lösungen keine dichte Packung aufweisen.

9.3.5 Konvergenzschaubilder

Die Abbildung 9.7 zeigt die Bitkonvergenzen, wie sie bei einem Testlauf mit Ranking auftreten. Nach der zufälligen Anfangsinitialisierung hat jedes Bit ca. 50 Prozent Konvergenz. Dieser Wert steigt, bis nach mehreren Generationen die ersten Bits zu 100 Prozent konvergiert sind. Dann erhält der Eintrag “lost” Werte ungleich null. Die Abbildung 9.8 zeigt die typische Konvergenzkurve einer topologischen Version. Durch die Selektion mit Fitneßwerten relativ zu den Nachbarn können verschiedene Lösungen länger nebeneinander bestehen. Durch das lokale Crossover bleibt die Vielfalt länger erhalten.

Die Schaubilder von Online- und Offline-Performance, siehe Abbildung 9.9, unterscheiden sich dadurch, daß Ausrutscher in die Online-Performance miteingehen, während bei der Offline-Performance nur die Besten der Generationen verwendet werden. Deshalb sind bei der Online-Performance Sprünge nach oben zu beobachten.

Die Abbildung 9.10 zeigt die starken Schwankungen der durchschnittlichen Fitneß. Durch die unterschiedliche Größenordnung der beiden Werte ist der Verlauf der besten Fitneß im einzelnen Schaubild besser zu erkennen.

Im Schaubild 9.11 ist zu erkennen, daß sich durch die Konvergenz die durchschnittliche Fitneß der besten Fitneß angenähert hat. Die Erhöhungen der durchschnittlichen Fitneß entstehen durch die Mutation.

Protein pdb1crn			Energie 2066.73	
Test	Version	Generationen	Energie	RMS
ml1	ml	201	1358.2	9.49
ml2	ml	229	1408.8	8.39
ml3	ml	200	1425.2	7.86
ml4	ml	185	1470.5	12.17
ranksc3	rank	89	1903.2	11.11
ranksc4	rank	301	1426.1	-
ranksc5	rank	219	1678.6	12.71
ranksc6	rank	199	1406.0	6.49
sch1	sch	99	64335.8	24.64
sch2	sch	89	58361.1	8.84
top2	top	153	5586.2	17.91
top3	top	149	43338.9	16.03

Tabelle 9.2: pdb1crn

Protein pdb1fdx			Energie 281977.88	
Test	Version	Generationen	Energie	RMS
fdx1	ml	229	1978.5	26.62
fdx2	ml	199	1787.9	35.37
fdx3	ml	199	1675.3	16.89
fdx4	ml	219	1536.9	33.93
sch3	sch	238	641893.2	13.29

Tabelle 9.3: pdb1fdx

9.3.6 Ähnlichkeit der Ergebnisse

Die Tabelle 9.8 zeigt die Ähnlichkeit zwischen den besten Lösungen eines Testlaufs. Die auftretenden großen *RMS*-Werte zeigen, daß die besten Lösungen nicht alle Abwandlungen einer Lösung sind. Kleine *RMS*-Werte treten bei ähnlichen Lösungen auf. Ist der *RMS*-Wert zweier Lösungen sehr klein, dann sind ihre *RMS*-Werte zu den anderen Lösungen ähnlich.

Bei der Sicherung der Besten wird ein Genotyp nur gespeichert, wenn er noch nicht vorhanden ist. Die Individuen 0 und 1 in der Tabelle 9.8 besitzen einen *RMS*-Wert von null. Die Koordinaten ihrer C_α-Atome stimmen überein. Der letzte Winkel ihrer Phänotypen ist verschieden, aber das hat auf die Position des letzten C_α-Atoms keinen Einfluß.

In der Tabelle 9.9 sind *RMS*-Werte von Lösungen aus verschiedenen Testläufen desselben Proteins aufgelistet. Die erzeugten besten Lösungen sind verschieden.

9.3.7 Disulfidbrücken

Die Tabelle 9.10 zeigt, daß die originalen Disulfidbrücken größtenteils nicht ausgebildet werden. Da sich die Konformationen von der des Originals unterscheiden, bilden sich statt dessen andere Disulfidbrücken aus. Das Protein pdb1fdx besitzt im Original keine Disul-

Protein pdb2ovo			Energie 2338.53	
Test	Version	Generationen	Energie	RMS
ovo1	ml	179	1905.0	12.74
ovo2	ml	259	1286.3	11.41
ovo3	ml	239	1257.3	10.42
sch4	sch	120	1054787.0	10.83

Tabelle 9.4: pdb2ovo

fidbrücken. Im Testlauf fdx2 sind bis zu zwei Stück entstanden.

9.3.8 Nicht konvergierte Bits

Bei dem Testlauf ml1 stagnierte die Konvergenz ab einer gewissen Generation. Die Anzahl der nichtkonvergierten Bits lag über der durch die Mutation veränderten Anzahl von Bits. Die verschiedenen Werte der zugehörigen nicht konvergierten Drehwinkel haben somit keinen Einfluß auf die Energiewerte. Sie liegen an den Enden der Kette und in Seitenketten. Tabelle 9.11 listet für Testläufe, bei denen mehr als 90 Prozent der Bits konvergiert sind, die Winkel, in denen die nichtkonvergierten Bits liegen.

9.3.9 Die Besten der Generationen

Die Abbildung 9.12 zeigt für den Testlauf fdx3 die Molekülgraphiken des jeweils Besten einer Generation.

Sequenz pdb1crn		
Testlauf ml3		
$\triangle\Phi$	$\triangle\Psi$	$\triangle\omega$
	2.196	0.122
0.402	-0.119	-0.385
-2.981	0.054	-1.056
0.639	-1.135	0.447
1.780	-0.579	-0.188
0.830	-0.607	-0.198
-2.674	0.080	0.013
2.720	0.028	0.987
-0.242	0.236	0.991
-2.719	0.507	0.934
-3.029	-0.451	1.098
2.833	-0.103	0.284
-2.941	-1.454	1.754
-1.327	0.088	0.646
-2.444	0.648	-2.258
-2.880	-1.357	0.011
0.676	0.058	0.256
-2.070	-1.137	0.618
0.555	-0.685	0.027
2.999	-0.880	0.713
0.993	-1.111	1.161
1.203	0.849	0.171
-0.969	0.558	1.660
-1.443	-0.723	0.059
0.075	0.580	0.241
1.852	-1.001	1.345
-0.704	-0.010	0.488
-2.770	-0.317	1.334
2.631	-0.691	-0.216
-2.188	0.183	-0.638
-1.029	-0.449	-2.386
0.255	-0.905	0.505
3.041	-0.833	-0.528
0.599	-0.065	-0.634
1.060	0.968	-0.166
2.348	-0.040	-0.032
-0.380	-0.195	0.612
0.163	0.418	-0.464
-1.066	0.424	0.343
0.604	1.201	0.265
0.411	0.066	0.371
-1.986	-0.834	2.651
1.407	0.157	0.183
-0.339	0.449	0.529
0.453	0.032	1.568
-0.393		

Sequenz pdb1crn		
Testlauf ranksc6		
$\triangle\Phi$	$\triangle\Psi$	$\triangle\omega$
	1.755	-0.431
-0.015	-0.506	-0.618
0.780	-0.602	-0.160
1.136	1.086	-0.167
-0.208	-0.634	0.260
0.370	0.332	-0.547
-2.882	-0.048	1.161
-0.459	0.641	1.239
-3.010	0.623	0.764
2.772	-0.371	-0.385
0.928	0.003	1.300
2.833	0.720	1.561
2.410	-0.098	1.398
1.263	-0.574	0.971
3.017	0.120	0.350
2.286	-0.351	-2.155
-0.963	0.493	2.085
-2.984	-1.217	0.587
3.132	-0.335	-0.273
-2.106	-0.665	-0.655
0.686	0.552	-1.588
0.767	-0.679	0.540
-2.012	0.098	0.599
-1.872	1.535	1.292
2.456	-0.224	1.782
-2.879	-0.835	-0.446
-1.317	0.696	0.954
-1.801	0.579	-0.354
-2.977	0.935	-0.253
3.126	0.147	0.313
-1.894	-0.339	-0.527
0.752	-0.641	0.652
1.471	-0.673	-0.074
0.433	0.181	-1.205
0.177	-1.143	-0.804
0.219	-0.224	-0.352
1.724	0.075	1.649
-2.647	0.240	0.947
-1.704	-0.196	0.104
1.469	-1.444	1.437
0.895	0.256	0.083
-1.305	-0.147	2.043
-2.729	0.181	0.570
-0.959	0.615	-0.268
0.404	-0.110	0.739
0.301		

Sequenz pdb1fdx		
Testlauf fdx3		
$\triangle\Phi$	$\triangle\Psi$	$\triangle\omega$
	1.063	-0.432
1.309	0.755	-0.097
-0.460	-0.034	-0.978
-0.537	0.007	0.693
1.694	0.323	0.157
0.614	0.639	1.366
-2.242	0.228	0.448
1.891	0.083	1.632
-2.199	-0.248	-0.008
-2.215	-0.407	2.417
2.815	-0.718	0.289
-2.269	0.031	2.861
-2.464	0.267	-0.316
-3.034	0.389	-0.104
-2.865	0.164	1.423
2.113	0.967	0.347
-2.620	-0.558	2.503
2.179	-1.325	2.012
-2.418	0.783	0.076
-1.763	-0.365	-1.502
-0.768	0.385	3.008
-2.006	0.227	-0.269
1.031	0.202	-0.256
0.803	-0.042	-1.002
-0.200	-0.406	-0.288
2.373	-0.724	1.440
0.566	-1.325	1.241
-2.663	0.652	1.589
0.483	-0.072	-0.255
-0.521	-0.401	-0.328
0.042	-1.280	1.104
1.985	-0.118	0.492
0.370	-0.060	0.155
-2.629	0.378	-0.418
-0.763	-0.912	2.109
-2.638	-0.380	1.232
-1.665	-0.372	-3.095
-1.858	0.844	-0.213
0.319	-0.869	-0.334
0.238	-0.092	0.488
-0.673	-0.257	-2.039
-1.306	-0.101	0.919
-2.600	0.206	0.358
-0.919	-0.084	-2.606
-0.131	-0.885	-0.099
1.135	0.460	-0.111
-2.608	0.207	-0.294
2.476	0.127	-2.686
-2.628	0.830	0.460
-2.461	-0.235	0.159
-0.037	1.510	-0.802
3.089	0.518	0.736
0.567	0.080	-0.554
0.893		

Sequenz pdb2ovo		
Testlauf ovo3		
$\triangle\Phi$	$\triangle\Psi$	$\triangle\omega$
	-0.527	0.225
-0.862	-0.306	1.092
-0.699	-0.862	1.107
-2.795	-0.666	-1.181
0.241	-0.006	-1.035
0.887	-0.381	0.282
-0.879	0.515	-0.476
-0.647	-0.146	0.290
-2.512	0.519	0.357
2.795	0.125	-0.088
-0.328	-1.911	0.412
0.286	-0.399	0.402
-0.409	0.851	-0.518
0.193	-0.532	0.650
0.848	0.470	-1.326
0.226	-0.541	0.090
0.389	0.036	0.876
-0.629	0.140	0.270
-0.929	0.702	1.298
-0.670	-0.335	0.653
-1.055	-0.796	-0.578
1.267	0.318	0.218
-0.081	0.035	-1.099
-3.036	0.193	0.210
-0.324	-1.119	1.397
-2.087	0.095	-0.635
0.102	-0.506	0.039
-1.127	0.114	1.206
1.405	-0.139	0.950
-0.664	-0.577	-0.763
-3.121	-0.323	-0.299
-1.997	1.052	0.648
0.801	-0.315	-2.490
2.574	0.074	2.158
3.126	0.663	0.890
-1.999	0.831	-1.746
-0.576	0.241	-3.026
-2.303	0.226	0.715
-2.168	1.097	1.599
1.381	-0.456	-0.410
-0.872	-1.211	1.327
-0.006	-0.845	-0.210
1.788	0.546	-1.914
-2.525	0.676	0.812
-2.545	-0.578	-0.101
2.649	0.202	-2.517
0.172	-0.379	0.962
-1.735	-0.079	1.138
-2.237	0.297	-0.401
-0.669	0.278	-0.109
2.798	-0.674	-0.325
3.019	-0.216	0.070
-0.602	-0.525	1.986
0.054	-0.027	3.119
1.521	0.334	1.299
-0.079		

Tabelle 9.5: Drehwinkel des Rückgrats

Sequenz pdb1crn				
Testlauf ml3				
$\triangle\varrho_1$	$\triangle\varrho_2$	$\triangle\varrho_3$	$\triangle\varrho_4$	$\triangle\varrho_5$
-0.097				
-2.120				
0.354				
1.713				
—				
-2.720				
1.745	0.893			
0.755				
—				
-1.298	-0.699	-1.957	0.494	-3.108
1.673				
1.896	-1.399			
-0.880	2.725			
-1.980	-1.606			
-0.040				
0.440				
0.185	0.154	2.742	1.097	-1.228
1.728	1.733			
—				
—				
-1.975				
—				
0.183	1.343	-0.616		
—				
1.630	2.590			
-1.084				
—				
1.658				
2.416	-0.956			
-1.798				
—				
0.409				
-2.116	1.394			
0.200	-0.056			
2.853	0.574			
—				
—				
—				
0.183				
2.972				
—				
—				
0.653	1.851			
1.274	2.502			
—				
0.535	2.980			

Sequenz pdb1crn				
Testlauf ranksc6				
$\triangle\varrho_1$	$\triangle\varrho_2$	$\triangle\varrho_3$	$\triangle\varrho_4$	$\triangle\varrho_5$
1.989				
0.034				
0.011				
1.327				
—				
-0.406				
-0.004	0.163			
-2.472				
—				
2.746	-2.528	3.118	-1.224	2.758
2.041				
0.374	0.871			
1.899	0.209			
-0.348	-2.097			
2.157				
-0.462				
1.252	0.663	1.147	-2.474	-0.375
-1.052	0.996			
—				
—				
0.135				
—				
-1.719	0.287	-3.076		
—				
1.366	-3.110			
0.597				
—				
1.461				
-1.462	2.358			
-2.049				
—				
1.980				
-2.011	0.240			
-0.076	1.405			
-0.233	1.310			
—				
—				
—				
-1.688				
0.665				
—				
—				
-0.003	-2.235			
1.838	2.975			
—				
1.290	-2.972			

Tabelle 9.6: Rotamere

Sequenz pdb1fdx			
Testlauf fdx3			
$\triangle \varrho_1$	$\triangle \varrho_2$	$\triangle \varrho_3$	$\triangle \varrho_4$
—			
1.196	-0.115		
-2.198			
1.133	-0.566		
0.420	2.005		
0.651	-0.296		
-0.448			
-0.275			
-2.076	-1.150		
—			
2.670			
—			
—			
2.764			
0.158	-2.192	-0.661	-1.924
—			
0.098	-2.755	2.769	
-1.659			
—			
2.148			
-0.154	0.595		
-2.941	0.836		
0.648	0.969		
3.031	0.627	-1.168	
—			
-1.850			
-3.077	1.545		
-2.747	-0.707		
—			
0.185	1.611		
-0.481	2.295		
—			
-0.461	2.189		
0.272			
2.335			
2.996	0.338		
0.819	0.501		
1.986			
—			
-2.451			
3.022			
—			
1.235			
2.466			
-0.262			
—			
-2.390			
—			
—			
—			
-2.969	2.419		
—			
-0.933	2.129	-0.145	
-1.101	2.536		

Sequenz pdb2ovo				
Testlauf ovo3				
$\triangle \varrho_1$	$\triangle \varrho_2$	$\triangle \varrho_3$	$\triangle \varrho_4$	$\triangle \varrho_5$
1.820	2.001			
—				
—				
1.740				
-0.885				
1.750				
-1.853	1.624			
-0.339				
0.230				
1.582	-0.419	-1.535		
-0.337	-2.786			
—				
-1.532	-0.948	-1.174	1.195	
—				
—				
-1.212				
-2.086				
-0.069	-1.499	0.835		
1.154	0.707	2.736		
-1.416	2.820			
0.758	0.404	0.135	-1.478	1.519
—				
-2.195	1.469			
0.083				
—				
2.096				
0.001	-2.966			
-1.026	-0.538			
-2.490	-0.151	-1.257	0.484	
2.245				
1.762	-0.614			
—				
-2.752	0.078			
-1.172	0.763	-0.862	0.408	
-1.630				
0.095	-0.896			
-1.230	-2.058			
-2.743				
1.127	-0.431			
—				
-0.271				
-0.642				
0.482	0.682	2.626		
-2.148				
0.467	-1.803			
—				
2.337				
-0.861	-0.548			
0.093				
-2.107	-1.048			
-0.979				
-2.155	0.176			
1.295	-3.013			
—				
-2.767	1.617	1.788	-2.156	
-0.222				

Tabelle 9.7: Rotamere

von\nach	1	2	3	4	5	6	7	8	9
0	0.00	7.60	7.46	7.40	7.41	7.13	7.15	5.00	6.91
1		7.60	7.46	7.40	7.41	7.13	7.15	5.00	6.91
2			0.70	1.03	2.10	2.81	1.07	8.64	1.66
3				0.54	1.88	2.87	0.70	8.44	1.64
4					1.93	2.80	0.44	8.26	1.58
5						2.74	1.89	8.12	2.54
6							2.56	8.03	2.89
7								8.03	1.45
8									7.73

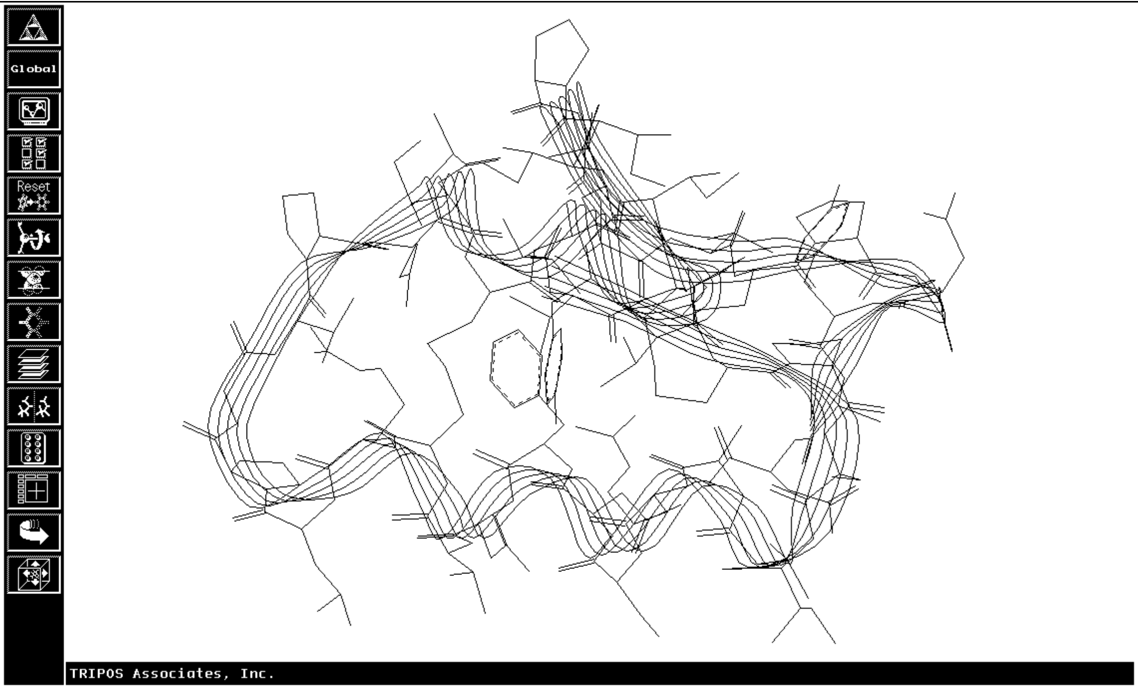
Tabelle 9.8: *RMS* der Lösungen von ml4

von\nach	0	1	2	3	4	5	6	7	8	9
0	7.99	8.01	8.01	8.01	8.01	8.81	8.05	8.07	7.99	8.81
1	8.16	8.17	8.19	8.19	8.19	9.06	8.22	8.24	8.16	9.06
2	7.68	7.70	7.71	7.71	7.71	8.50	7.75	7.77	7.68	8.51
3	8.26	8.26	8.28	8.28	8.28	9.15	8.31	8.34	8.26	9.15
4	8.33	8.33	8.36	8.36	8.36	9.22	8.38	8.41	8.33	9.22
5	8.34	8.34	8.36	8.36	8.36	9.23	8.39	8.42	8.34	9.23
6	8.29	8.29	8.32	8.32	8.32	9.20	8.34	8.37	8.29	9.20
7	8.55	8.55	8.58	8.58	8.58	9.57	8.60	8.64	8.55	9.57
8	8.27	8.26	8.29	8.29	8.29	9.22	8.31	8.35	8.27	9.23
9	8.28	8.29	8.31	8.31	8.31	9.17	8.34	8.36	8.28	9.17

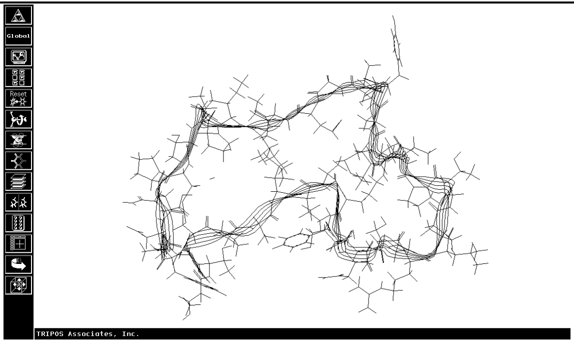
Tabelle 9.9: *RMS* der Lösungen von ml1 nach ml2

Sequenz pdb1crn					Sequenz pdb1fdx			Sequenz pdb2ovo				
Testlauf ml1					Testlauf fdx2			Testlauf ovo3				
Lösung	Cysteinpaare				Lösung	Cysteinpaare		Lösung	Cysteinpaare			
	15 25	3 31	2 39						15 34	7 37	23 55	
1	0	0	0	3 39	1	10 13		1	0	0	0	7 34
2	0	0	1		2	10 13		2	0	0	1	
3	0	0	1		3	17 37		3	0	0	1	
4	0	0	1		4	10 13		4	0	0	1	
5	0	0	1		5	10 13		5	0	0	1	
6	0	0	1		6	10 13		6	0	0	1	
7	0	0	1		7	10 13		7	0	0	1	
8	0	0	1		8	10 13		8	0	0	1	
9	0	0	1		9	17 37		9	0	0	1	
10	0	0	1		10	10 13		10	0	0	1	
11	0	0	1		11	17 37		11	0	0	1	
12	0	0	1		12	17 37		12	0	0	0	7 34
13	0	0	1		13	10 13		13	0	0	1	
14	0	0	1		14	10 13	17 40	14	0	0	1	
15	0	0	1		15	10 13	17 37	15	0	0	1	
16	0	0	1		16	17 37		16	0	0	1	
17	0	0	1		17	17 37		17	0	0	1	
18	0	0	1		18	10 13		18	0	0	1	
19	0	0	1		19	17 37		19	0	0	1	
20	0	0	1		20	17 37		20	0	0	1	
21	0	0	1		21	10 13		21	0	0	1	
22	0	0	1		22	10 13		22	0	0	1	
23	0	0	1		23	17 40		23	0	0	1	
24	0	0	1		24	17 37		24	0	0	1	
25	0	0	1		25	17 37		25	0	0	1	
26	0	0	1		26	10 13	17 40	26	0	0	0	34 37
27	0	0	1		27	17 37		27	0	0	1	
28	0	0	1		28	10 13	17 40	28	0	0	1	
29	0	0	1		29	10 13		29	0	0	1	
30	0	0	1		30	17 37		30	0	0	1	

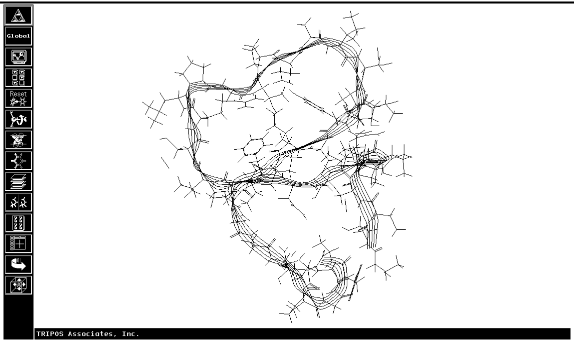
Tabelle 9.10: Disulfidbrücken



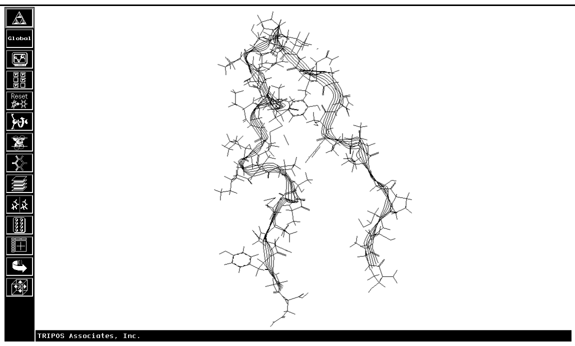
Originalkonformation



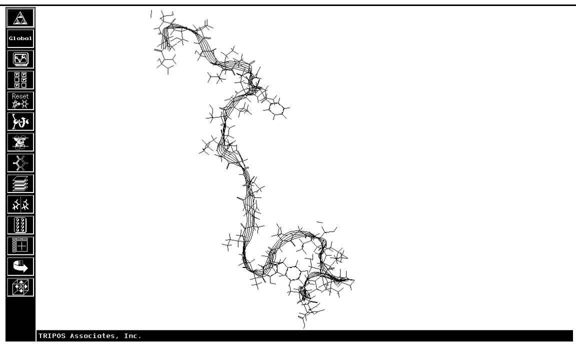
Testlauf ml3



Testlauf ranksc6

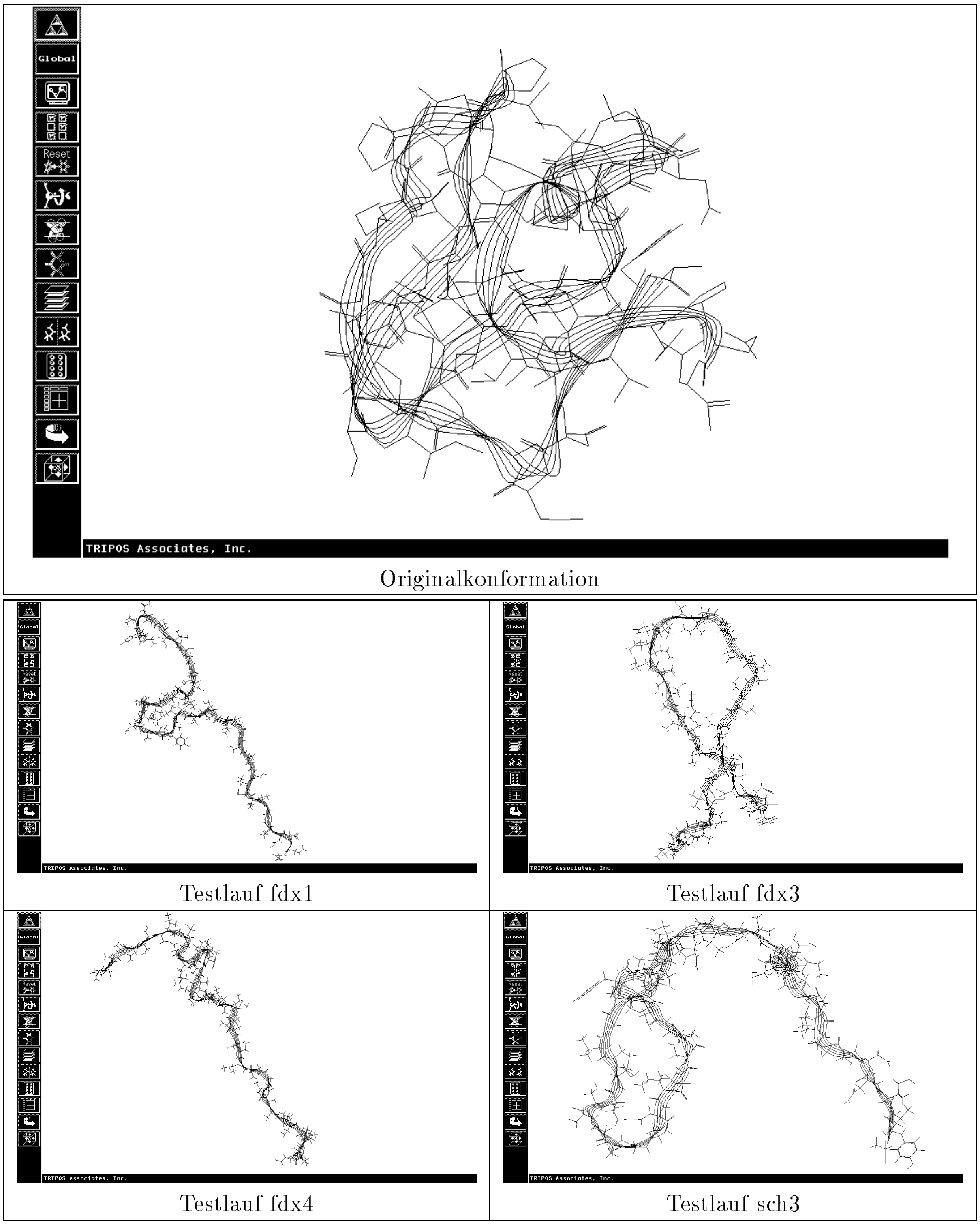


Testlauf sch2



Testlauf top2

Abbildung 9.4: Sequenz pdb1crn



Originalkonformation

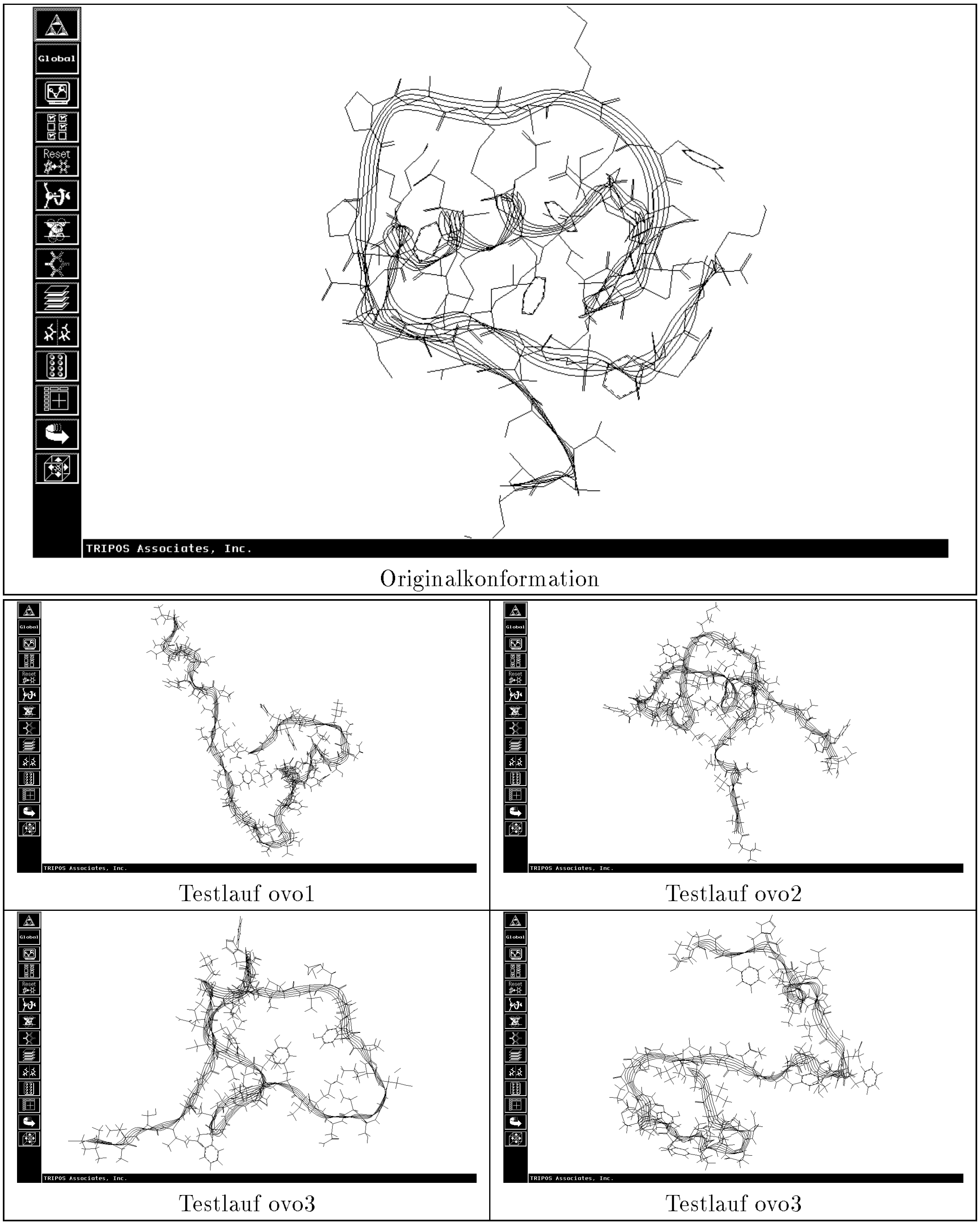
Testlauf fdx1

Testlauf fdx3

Testlauf fdx4

Testlauf sch3

Abbildung 9.5: Sequenz pdb1fdx



Originalkonformation

Testlauf ovo1

Testlauf ovo2

Testlauf ovo3

Testlauf ovo3

Abbildung 9.6: Sequenz pdb2ovo

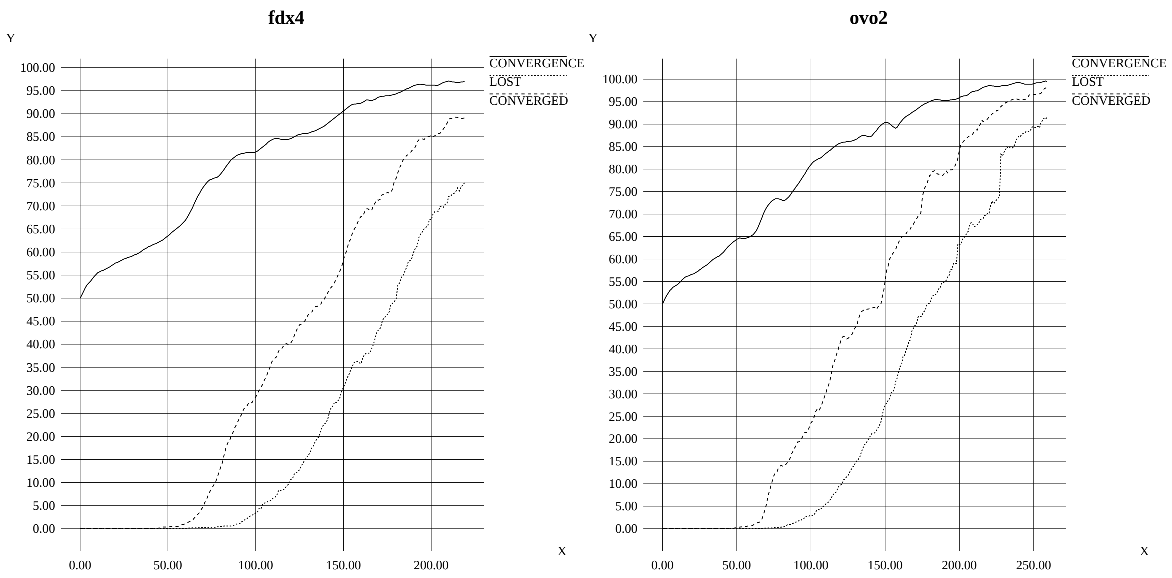


Abbildung 9.7: Version mit Ranking

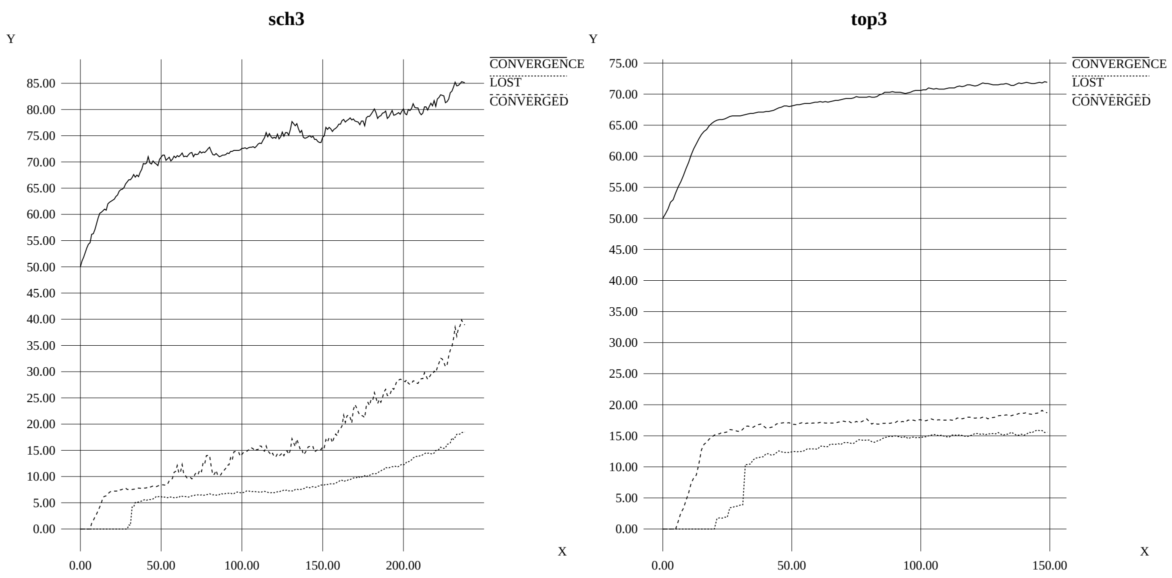


Abbildung 9.8: topologische Version

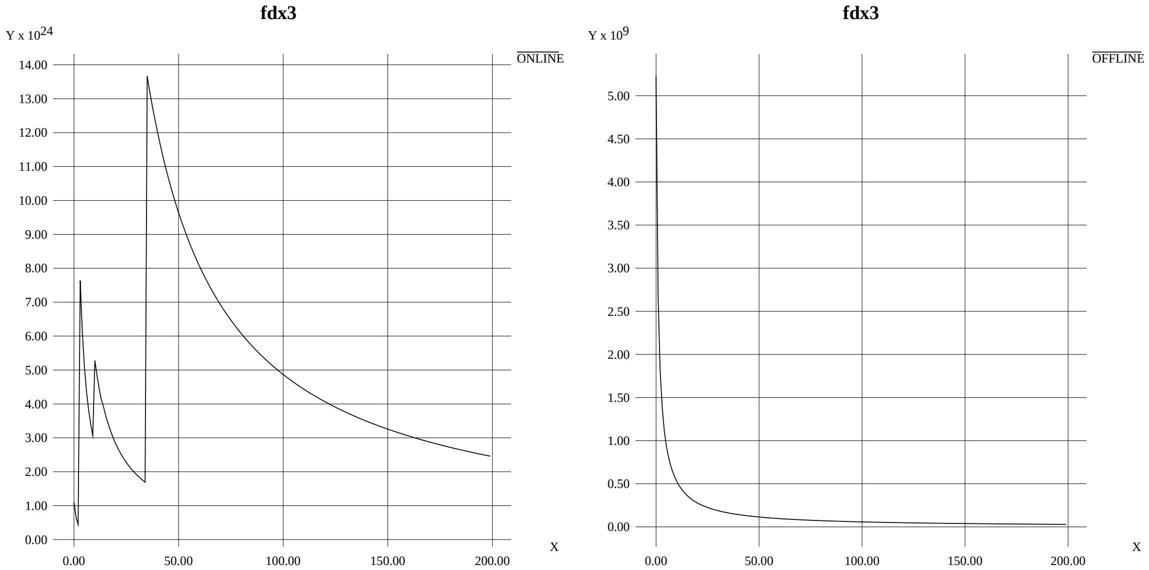


Abbildung 9.9: On- und Offline-Performance

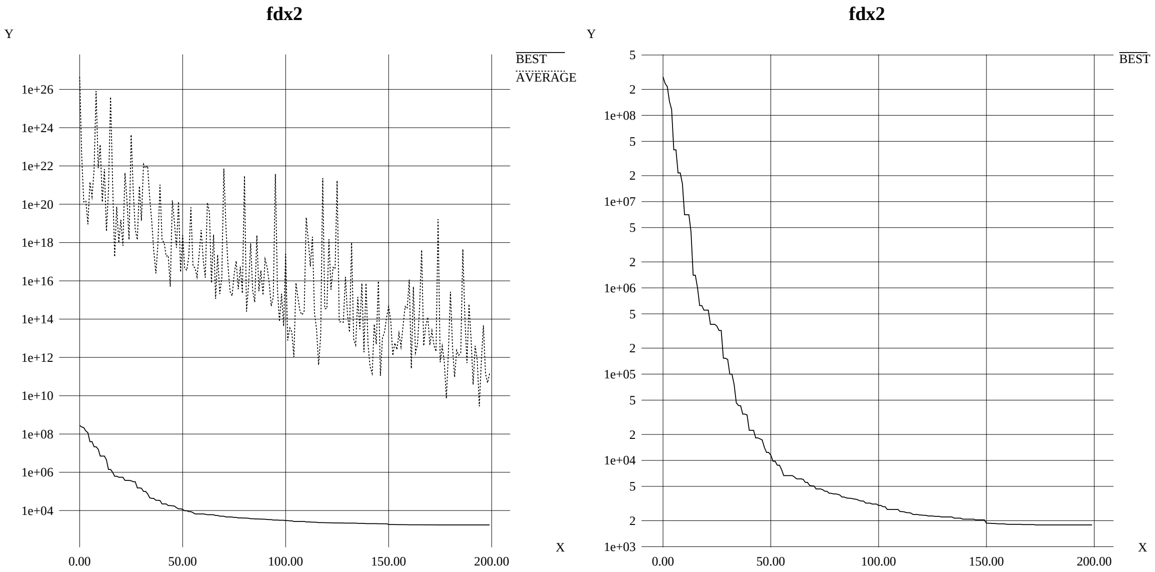


Abbildung 9.10: Fitneßwerte

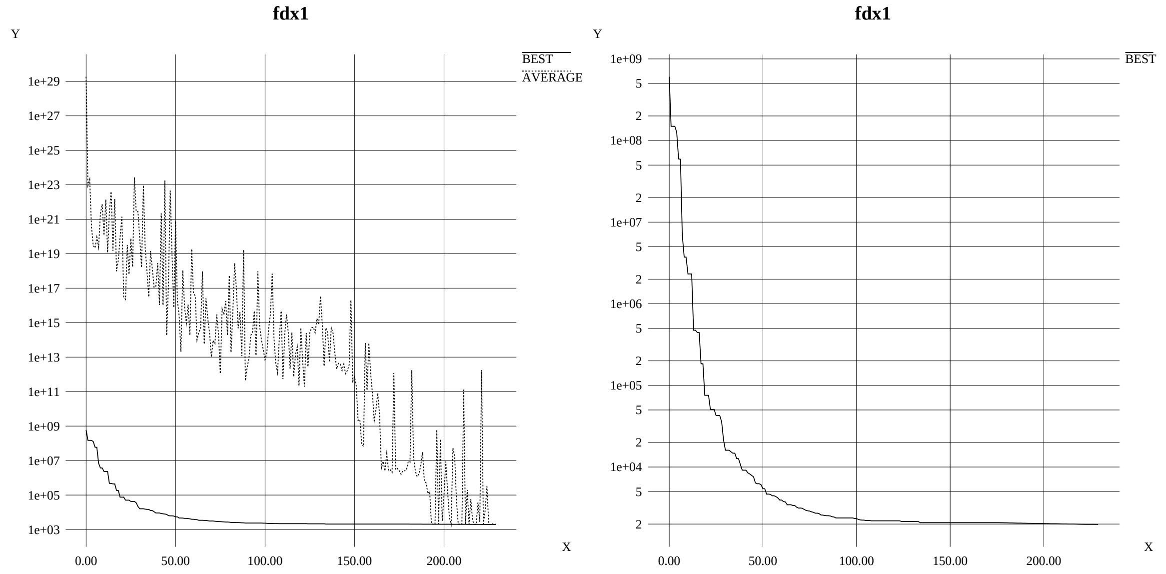


Abbildung 9.11: Konvergenz der durchschnittlichen Fitneß

Sequenz pdb1crn							
Testlauf ml1							
Φ	Ψ	ω	ϱ_1	ϱ_2	ϱ_3	ϱ_4	ϱ_5
	5	6	2				
1	0	4	1				
0	0	0	0				
0	2	0	0				
0	0	1					
2	0	0	0				
0	0	1	1	1			
0	0	2	2				
0	0	0					
0	1	0	0	0	0	1	0
0	0	0	1				
0	0	0	0	1			
0	0	1	0	1			
1	0	1	2	0			
0	1	0	0				
0	0	1	0				
0	0	0	0	0	0	1	1
0	0	1	0	0			
0	0	0					
0	0	0					
0	1	0	0				
0	1	0					
0	0	1	1	0	0		
0	0	0					
0	0	1	1	1			
0	0	0	1				
0	0	0					
0	0	2	0				
0	0	1	0	1			
1	0	0	2				
0	0	1					
1	3	1	0				
0	0	0	1	1			
0	2	0	2	4			
0	1	0	1	0			
0	0	1					
0	0	1					
0	0	0					
2	0	0	0				
1	1	2	0				
1	0	0					
0	2	3					
0	0	1	0	7			
4	6	3	7	8			
4	3	1					
6			7	1			

Sequenz pdb2ovo							
Testlauf ovo2							
Φ	Ψ	ω	ϱ_1	ϱ_2	ϱ_3	ϱ_4	ϱ_5
	8	6	4	6			
5	4	4					
5	1	0					
0	0	1	1				
0	1	0	0				
1	1	1	4				
1	0	0	0	1			
0	0	0	3				
1	1	0	1				
2	0	0	0	2	2		
0	1	0	0	0			
0	0	0					
2	0	1	3	1	0	1	
2	0	0					
0	0	0					
0	1	0	0				
0	3	0	2				
0	0	1	5	4	0		
0	0	1	4	2	3		
0	1	1	0	1			
2	0	1	1	0	0	1	0
0	0	0					
0	1	0	1	0			
0	0	4	0				
0	0	0					
0	0	0	0				
0	0	0	3	0			
0	0	0	0	0			
0	1	0	0	0	0	0	
0	1	1	1				
0	1	0	2	0			
1	0	0					
0	0	0	1	0			
1	0	0	0	1	1	1	
0	0	0	2				
0	0	0	0	0			
0	0	0	1	0			
0	0	0	0				
0	0	1	0	1			
0	2	0					
2	0	0	0				
1	1	0	3				
0	1	0	0	0	1		
3	0	1	0				
1	1	2	3	7			
0	0	1					
0	0	0	0				
1	1	0	4	3			
0	0	1	0				
0	2	0	0	1			
0	0	0	0				
2	1	3	0	0			
0	0	1	2	3			
0	0	0					
1	0	1	3	2	1	2	
1			9				

Tabelle 9.11: Nichtkonvergierte Bits

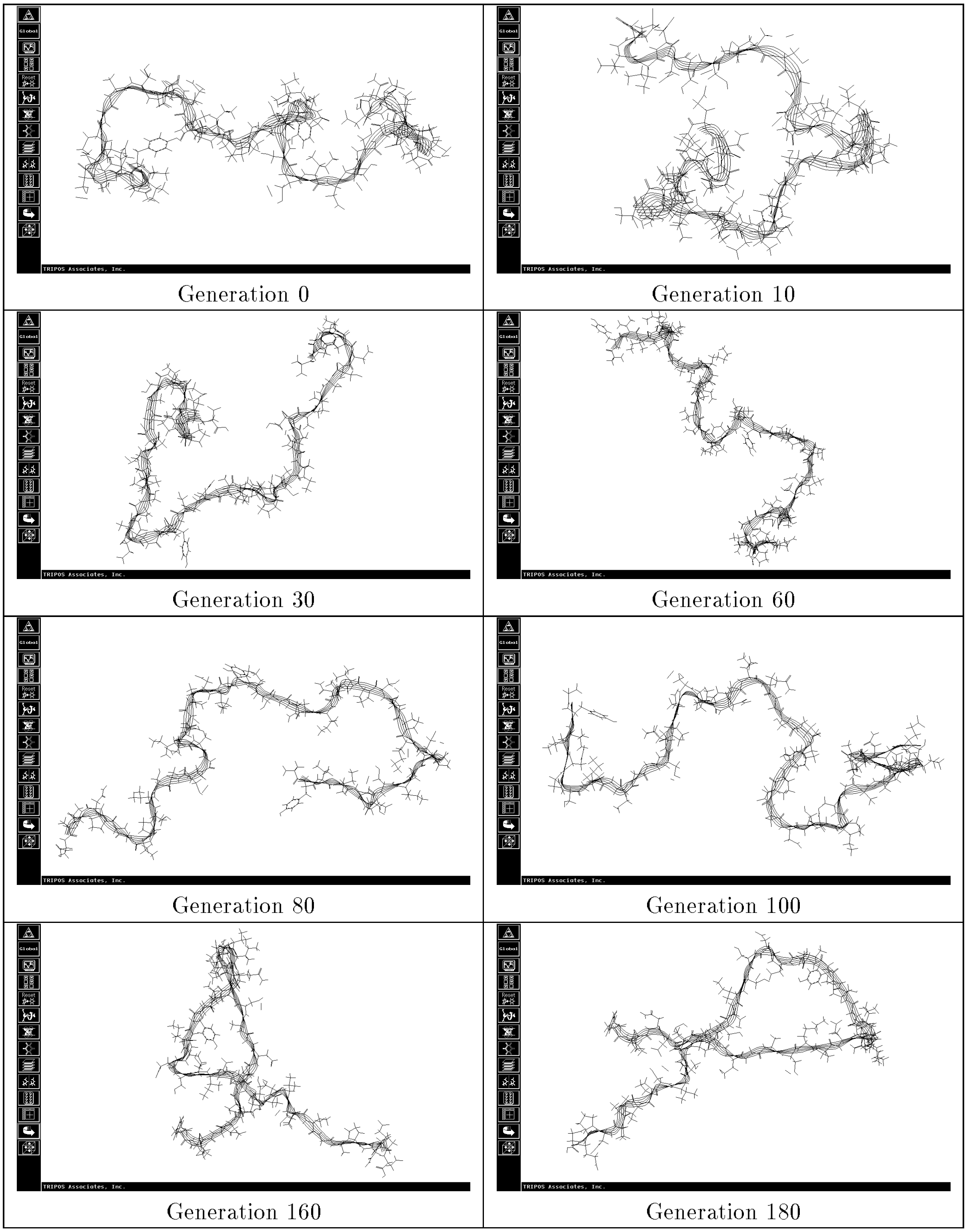


Abbildung 9.12: Generationsbeste des Testlaufs fdx3

Kapitel 10

Diskussion und Ausblick

In diesem Kapitel werden die Testergebnisse beurteilt. Beurteilt werden die erzeugten Konformationen, die Eigenschaften des Genetischen Algorithmus und der Einsatz der *MASPAR*. Aus der Beurteilung ergeben sich Verbesserungsvorschläge. Es wird die Anregung für einen anderen Ansatz vorgestellt.

Die berechneten *RMS*-Werte haben gezeigt, daß die erzeugten Konformationen die natürlichen Proteinfaltungen nicht korrekt vorhersagen. Der Vergleich der Drehwinkel zeigte, daß keine Abschnitte mit gleicher Faltung auftreten.

10.1 Voraussetzungen

Der Ansatz beruht auf drei Thesen:

1. Ein Protein faltet sich in die energetisch günstigste Konformation.
2. Die verwendete empirische Energiefunktion gibt die Energieunterschiede der verschiedenen Konformationen korrekt wieder.
3. Der Genetische Algorithmus findet das globale Minimum.

Die erste These ist zwar plausibel aber nicht unumstritten[1]. Es ist z.B. möglich, daß sich ein Protein bedingt durch den Faltungspfad in ein lokales Minimum der Energiefunktion faltet. Wenn die Energiebarriere zu einem günstigeren Zustand durch thermische Fluktuationen nicht überwunden werden kann, ist das Protein in dieser Konformation stabil.

Während der biologischen Synthese von Proteinen in den Ribosomen einer Zelle wird das Kettenmolekül schrittweise aufgebaut. Das heißt, daß sich der bereits existierende Teil bereits faltet, ohne daß die noch fehlenden Teile Einfluß auf die Faltung nehmen können. Die hinzukommenden Teile falten sich "auf" einen bereits vorgefalteten Teil. Der Faltungsvorgang ist ein anderer, wenn das Kettenmolekül bereits vollständig vorliegt und sich mit der ausgestreckten Form beginnend faltet. Dennoch wurde für viele Proteine gezeigt, daß sie sich nach einer Denaturierung spontan wieder in ihre natürliche Konformation falten und damit auch ihre katalytischen Eigenschaften wiedererlangen. Die Denaturierung erreicht man durch die Änderung des ph-Wertes der Umgebung oder durch Auflösung der

[1] Siehe [12, S. 63]

Disulfidbrücken mit reduzierenden Chemikalien. Zur Renaturierung macht man diese Maßnahmen rückgängig.

Die zweite These verlangt, daß empirische Energiefunktionen die quantenmechanischen Energiefunktionen genau genug annähern. Bei den Testergebnissen unterschritten die erzeugten Konformationen die für die natürliche Konformation berechneten Energiewerte. Geht man von der Richtigkeit der ersten These aus, dann folgt daraus, daß die empirische Energiefunktion nicht die korrekten Verhältnisse wiedergibt.

Für die dritte These gibt es bei Genetischen Algorithmen keine Garantie[2]. Diese Garantie können andere Optimierungsverfahren, außer vollständig aufzählende Verfahren, nur für Funktionen mit bestimmten bekannten Eigenschaften geben. Solche Eigenschaften sind z.B. Differenzierbarkeit und Stetigkeit. Bei Anwendungen finden sich aber meistens komplexe Funktionen die diese Eigenschaften nicht besitzen. Genetische Algorithmen finden in großen komplexen Suchräumen mit vertretbarem Aufwand annähernd optimale Lösungen. Falls es im Raum der Konformationen eines Proteins viele sich unterscheidende Konformationen mit annähernd gleichen Energien gibt, dann reicht der vergleichsweise kleine Vorteil des globalen Minimums nicht aus, damit es vom Genetischen Algorithmus sicher gefunden wird. Bei den Testläufen mit gleichen Parametern unterschieden sich die besten erzeugten Konformationen. Die ähnlichen Energiewerte sprechen dafür, daß es viele verschiedene Faltungen mit vergleichbaren Energiewerten gibt.

In solchen Suchräumen tritt besonders leicht vorzeitige Konvergenz ein. Ein zu groß gewählter Selektionsdruck vermindert zu schnell die Vielfalt in der Population. Die beim Ranking verwendeten Erwartungswerte für das beste Individuum entsprechen den beim linearen Skaling verwendeten (ca. 2). Tests mit kleinerem η_{max} und damit geringerem Selektionsdruck erfordern entsprechend längere Laufzeiten.

10.2 Verbesserungen

Ansätze für Verbesserungen sind also die Energiefunktion und die Verhinderung von vorzeitiger Konvergenz. Die empirische Energiefunktion ist durch die verwendeten Parameter gegeben. Verwendet wurden die Parameter von *GROMOS*. Es können Parameter aus anderen Quellen verwendet werden. Die verwendeten Parameter von *GROMOS* gelten für kristalline Proteine und solche in Lösung[3]. Als Erweiterung können die Moleküle des Lösungsmittels mit berücksichtigt werden. Dadurch ergeben sich Wechselwirkungen mit hydrophilen und hydrophoben Seitenketten. Für die Faltung und besonders für die Ausbildung der Sekundärstruktur sind die Wasserstoffbrücken wichtig. Die Energiefunktion von *GROMOS* berücksichtigt ihre Effekte durch das Gleichgewicht zwischen Van-der-Waals und Elektrostatischer Energie[4]. Andere Energiefunktionen wie die von *CHARMM*[5] berechnen die Energie der Wasserstoffbrücken explizit.

Die Peptidbindung im Rückgrat ist relativ starr. Der Drehwinkel ω wurde zur Beschreibung der Faltung des Rückgrats verwendet. Der Parameter für den Energiebeitrag ist entsprechend der Starrheit größer als der der Drehwinkel ϕ und ψ. Dadurch, daß die

[2]Siehe [4, S. 74]
[3]Siehe [19, S. II-2]
[4]Siehe [19, S. II-31]
[5]Siehe [20]

Beiträge der Van-der-Waals Energie sehr viel größer sind, fällt das nicht ins Gewicht. Infolgedessen weichen die ω Winkel stark von ihrem Standardwert ab. Eine Repräsentation ohne variable ω-Winkel wird realistischere Faltungen des Rückgrats erzeugen.

GENESIS bietet die Möglichkeit, die Anfangspopulation mit Bitstrings zu initialisieren. Die verbleibenden Individuen werden zufällig initialisiert. Man kann für die Anfangspopulation Faltungen benutzen, die man aus Sekundärstrukturvorhersagen erzeugt. Der Genetische Algorithmus kann dann die stückweise energetisch günstigen Faltungen zu Tertiärstrukturvorhersagen zusammensetzen.

Bei den Schaubildern der statistischen Werte fallen die starken Schwankungen der durchschnittlichen Fitness auf. Durch das Crossover werden Teilfaltungen so zusammengefügt, daß Kollisionen auftreten. Diese bewirken die extremen Energiewerte, die die durchschnittliche Fitness ansteigen lassen. Als Erweiterung zu einem reinen Genetischen Algorithmus kann man diese Kollisionen verhindern, indem man den Drehwinkel, an dem die Teile zusammengefügt werden, lokal optimiert.

Man kann aus einem Genetischen Algorithmus und einem herkömmlichen Optimierungsverfahren einen hybriden Ansatz erstellen. Der Genetische Algorithmus lokalisiert zunächst Bereiche des Suchraums, in denen gute Lösungen liegen. In dieser Phase wird die Fähigkeit, große Suchräume schnell einzugrenzen, ausgenutzt. Das deterministische Verfahren findet dann die lokalen Minima der verschiedenen Lösungsvorschläge. Dabei wird die Konvergenzgarantie für das lokale Minimum ausgenutzt.

10.3 Einsatz der *MASPAR*

Durch die Parallelisierung auf der *MASPAR* ergibt sich eine Beschleunigung um etwa den Faktor 430 gegenüber der seriellen Version auf einer *SUN SPARCstation ELC*. Die hohe Rechenleistung der *SUN* kompensiert einen Teil der 16384-fachen Prozessorenanzahl der *MASPAR*. Die Tabelle 10.1 stellt die Rechenzeiten für die Testproteine gegenüber und nennt die Rechenzeit, die nötig ist, um eine Population der Größe 16384 auf der *SUN* zu berechnen. Die Zeiten umfassen die Berechnung der Koordinaten und der Energie. Der Zeitbedarf für die Übertragung der Daten vom *Front End* auf die *MASPAR* sind hier nicht berücksichtigt. In Tabelle 10.2 sind die Zeiten aufgeführt, die zum Berechnen der Konvergenzstatistiken und für die Übertragung der Binärstrings benötigt werden. Die Binärstrings werden in einen zusammenhängenden Speicherbereich kopiert, damit sie mit einem Übertragungsvorgang auf die *MASPAR* übertragen werden können. Die Zeit für das Kopieren ist in den Zeiten der Tabelle 10.2 enthalten. Die letzte Spalte gibt an, welche Beschleunigung sich mit Berücksichtigung der Übertragungszeiten ergibt.

Sequenz	*MASPAR* 16K Individuen	*SPARC*station 1 Individuum	*SPARC*station 16K-Population	Beschleunigung
pdb1cbh	2 min 1 sek	3.15 sek	14 h 20 min	426
pdb1crn	3 min 6 sek	4.9 sek	22 h 18 min	431
pdb1fdx	3 min 53 sek	6.1 sek	27 h 45 min	429
pdb2ovo	5 min 9 sek	8.05 sek	36 h 38 min	427

Tabelle 10.1: Rechenzeiten ohne Übertragungszeiten

Sequenz	Konvergenz	Übertragung	Beschleunigung
pdb1cbh	57 sek	7 sek	403
pdb1crn	1 min 12 sek	8.5 sek	413
pdb1fdx	1 min 30 sek	10.5 sek	410
pdb2ovo	1 min 41 sek	12 sek	424

Tabelle 10.2: Übertragungszeiten

Eine Möglichkeit zu weiterer Beschleunigung ergibt sich bei der Berechnung der Bitkonvergenzstatistiken. Diese wird nach der Berechnung der Energie auf dem *Front End* durchgeführt. Die neue Population liegt nach der Anwendung der genetischen Operatoren und vor der Berechnung der Energie fest. Unter Ausnutzung des asynchronen Programmiermodells der *MASPAR* kann man gleichzeitig die Energie auf der *MASPAR* und die Bitkonvergenzstatistiken auf dem *Front End* berechnen.

Bei den topologischen Versionen erwies sich das Feststellen der nächstgelegenen Lücke und das Verschieben von Individuen, um Platz für die Nachkommen zu schaffen, als zeitaufwendig. Um diese Operationen parallel ausführen zu können muß die Population vom *Front End* auf die *MASPAR* verlagert werden.

10.4 Ausblick und Fazit

Der in dieser Arbeit gewählte Ansatz minimierte die Energie der Konformationen, um die natürliche Faltung vorherzusagen. Die Ähnlichkeit der Vorhersage zur natürlichen Faltung diente zur Beurteilung des Ergebnisses der Minimierung. Ein anderer Ansatz kann die Ähnlichkeit schon während der Optimierung als Maß für die Güte der Vorhersage benutzen. Die Individuen repräsentieren nicht die Konformation eines bestimmten Proteins, sondern eine Menge von Regeln zur Tertiärstrukturvorhersage aus Sequenzen. Die Fitneß der Individuen ergibt sich aus der Ähnlichkeit ihrer Vorhersagen zu den Konformationen bekannter Proteine. Der beste Regelsatz eines Optimierungsvorganges kann dann für unbekannte Proteine Vorhersagen treffen, oder mit einer Kontrollmenge von bekannten Proteinen getestet werden.

Eine weitere Möglichkeit ist die Verwendung eines Classifier-Systems[6]. Hierbei besteht ein Individuum aus einer Regel und alle Regeln zusammen führen die Vorhersage aus. Die erreichte Fitneß wird entsprechend dem Anteil am Erfolg auf die beteiligten Regeln verteilt. Neue Regeln entstehen wie beim Genetischen Algorithmus durch Selektion entsprechend der Fitneß und Crossover.

Abschließend ist zu sagen, daß bei einer Weiterentwicklung des in dieser Arbeit gewählten Ansatzes der Schwerpunkt auf der Auswahl bzw. Entwicklung einer aufwendigeren Energiefunktion liegen muß. Ein Fortschritt wäre die Entwicklung einer Kodierung von Konformationen, bei der durch Crossover erzeugte Konformationen wenige Kollisionen aufweisen.

[6]Siehe [4, Kapitel 6]

Literaturverzeichnis

[1] Linder, Hermann. 1971: *Biologie*. Genetik 238-273. Carl Ernst Poeschel Verlag GmbH, Stuttgart.

[2] John H. Holland. 1975: *Adaptation in Natural and Artificial Systems* The University of Michigan Press.

[3] Holland, John H. 1976: *Adaptation*, in: R. Rosen and F.M. Snell (Eds.), Progress in Theoretical Biology 4. Academic Press, New York. 263-293.

[4] Goldberg, David E. 1953,1989: *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Publishing Company, Inc.

[5] Thomas Bäck, Frank Hoffmeister. *Extended Selection Mechanisms in Genetic Algorithms* in: Proceedings of the 4th International Conference on Genetic Algorithms and their Applications, 92-99. Morgan Kaufmann Publishers.

[6] H. Mühlenbein, M. Schomisch and J. Born. 1991: *The parallel genetic algorithm as function optimizer* in: Parallel Computing 17, 619-632, 1991. North-Holland.

[7] John J. Grefenstette, modified by Nicol N. Schraudolph. 1990: *A User's Guide to GENESIS 1.2ucsd.*

[8] Lubert Stryer. 1987: *Biochemie* Vieweg & Sohn VerlagsGmbh.

[9] Richard E. Dickerson and Irving Geis. 1971: *Struktur und Funktion der Proteine* Verlag Chemie GmbH, Weinheim/Bergstr.

[10] Hugo Fasold. 1973:*Die Struktur der Proteine* Verlag Chemie GmbH, Weinheim/Bergstr.

[11] P.v. Sengbusch. 1979: *Molekular- und Zellbiologie* Springer Verlag, Berlin, Heidelberg.

[12] Walton, Alan G. 1981: *Polypeptides and Protein Structure*. Elsevier, New York.

[13] Barry Robson and Jean Garnier. 1988: *Introduction to Proteins and Protein Engineering.* Elsevier Science Publishers B.V.

[14] Alan Fersht. 1985: *Enzyme Structure and Mechanism* W. H. Freeman and Company, New York.

[15] J.Ellis Bell, Evelyn T. Bell. 1988: *Proteins and Enzymes* Prentice-Hall, Inc.

[16] G.M. Crippen, T.F. Havel. 1988: *Distance Geometry and Molecular Conformation.* Research Studies Press Ltd.

[17] John W. Moore, Ralph G. Pearson. 1981: *Kinetics and Mechanism* John Wiley & Sons, Inc.

[18] IUPAC-IUB Commission on Biochemical Nomenclature. 1969: *Abbreviations and Symbols for the Description of the Conformation of Polypeptide Chains. Tentative Rules.* in: Biochemistry, Volume 9, Number 18, September 1, 1970.

[19] Prof. Dr. W.F. van Gunsteren and Prof. Dr. H.J.C. Berendsen. *GROMOS Manual.* BIOMOS b.v., Biomolecular Software.

[20] Bernard R. Brooks, Robert E.Bruccoleri, Barry D.Olafson, David J. States, S.Swaminathan, and Martin Karplus. 1982: *CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations* in: Journal of Computational Chemistry, Vol. 4, No. 2, 187-217 (1983). John Wiley & Sons, Inc.

[21] William H. Press, Brian P. Slannery, Faol A. Teukolskiy, William P. Vetterling. 1988: *Numerical Recipes in C.* Cambridge University Press.

[22] W.D. Fellner. 1988: *Computer Grafik* BI Wissenschaftsverlag.

[23] MasPar Parallel Application Language (MPL). Februar 1990: *User Guide, Beta 2.* MasPar Computer Corporation

[24] MasPar Parallel Application Language (MPL). September 1990: *Reference Manual.* MasPar Computer Corporation

[25] MasPar Programming Environment (MPPE). *User Guide, Beta 2.* MasPar Computer Corporation

[26] MasPar Commands. Februar 1990: *Reference Manual, Beta 2.* MasPar Computer Corporation

[27] W. Butscher. 1991: *The Dataparallel Computer MasPar MP1* in: Supercomputer '91 Proceedings. Springer Verlag.