Fragmentbasierte Entwicklung neuer Leitstrukturen *in silico* durch Integration von 3D-Pharmakophorhypothesen

Dissertation

zur Erlangung des Doktorgrades
der Fakultät für Mathematik, Informatik
und Naturwissenschaften
der Universität Hamburg

vorgelegt von
Boris Kröplien
aus Hamburg

Hamburg, 2006

Universität Hamburg

1. Gutachter: Prof. Dr. Bernd Meyer
2. Gutachter: Prof. Dr. Volkmar Vill

Tag der Disputation: 26.01.2007

Die vorliegende Arbeit wurde in der Zeit von März 2001 bis Juli 2006 am Institut für Organische Chemie des Departments Chemie der Universität Hamburg, Direktor: Prof. Dr. Joachim Thiem, durchgeführt.

Herrn Prof. Dr. B. Meyer danke ich für die Überlassung des Themas sowie für die stets wertvolle und freundliche Unterstützung bei der Durchführung dieser Arbeit.

"*It would appear that we have reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in 5 years.*"

John von Neumann, 1949

Inhalt

Abkürzungsverzeichnis

1D, 2D, 3D	ein-, zwei-, dreidimensional
AAV	Allgemeine Arbeitsvorschrift
ADME/Tox	*adsorption, distribution, metabolism and excretion / toxicity*
AIDS	*acquired immunodeficiency syndrome*
C	Programmiersprache[1]
CD4	*cluster of differentiation* 4
csv	*comma separated values*
d_{max}	maximaler Abstand, den ein Ankerpaar überspannen kann
d_{min}	minimaler Abstand, den ein Ankerpaar überspannen kann
DNA	*deoxy ribonucleic acid*
E	Energie
E_{max}	Energie des energieungünstigsten Rotamers
E_{min}	Energie des energiegünstigsten Rotamers
GP120	Glykoprotein 120
HAART	*highly active antiretroviral therapy*
HIV	*human immunodeficiency virus*
HSQC	*heteronuclear single quantum coherence*
HTS	*high throughput screening*
LCAO	*linear combination of atomic orbitals*
log P	Octanol-Wasser Verteilungskoeffizient
MD	Moleküldynamiksimulationen
mol2	Dateiformat für Moleküle
$newE_{max}$	Neuer Grenzwert für die maximale Energie der Rotamere eines RBR
NMR	*nuclear magnetic resonance*
NOE	*nuclear Overhauser enhancement*
NOESY	*nuclear Overhauser enhancement and exchange spectroscopy*
PDB	*Brookhaven Protein Data Bank*
QSAR	*quantitative structure activity relationships*
RNA	*ribo nucleic acid*
SPL	SYBYL *programming language*
SPR	*surface plasmon resonance*
STD	*saturation transfer difference*
vdW	van-der-Waals

Ein Glossar der in dieser Arbeit verwendeten englischen Begriffe sowie eine Liste der getroffenen Definitionen befindet sich in den Anhängen 7.1 und 7.2. Ein schematischer Überblick über die wichtigsten Schritte der vorliegenden Arbeit befindet sich als Referenz in Abschnitt 3.8.

Aminosäuren

Aminosäure	Abkürzung	Code	Aminosäure	Abkürzung	Code
Alanin	Ala	A	Leucin	Leu	L
Arginin	Arg	R	Lysin	Lys	K
Asparagin	Asn	N	Methionin	Met	M
Asparaginsäure	Asp	D	Phenylalanin	Phe	F
Cystein	Cys	C	Prolin	Pro	P
Glutamin	Gln	Q	Serin	Ser	S
Glutaminsäure	Glu	E	Threonin	Thr	T
Histidin	His	H	Tryptophan	Trp	W
Glycin	Gly	G	Tyrosin	Tyr	Y
Isoleucin	Ile	I	Valin	Val	V

Abweichung von den SI-Einheiten

Bei der Beschreibung chemischer Bindungen werden nicht immer SI-Einheiten verwendet. Bindungslängen und interatomare Abstände werden in dieser Arbeit in Ångström [Å] angegeben, wobei ein Ångström 10^{-10} m (0.1 nm) entspricht.
Energien werden im Rahmen dieser Arbeit in Kilokalorien [kcal] angegeben, wobei 1 kcal 4.1868 Kilojoule entspricht.

Farbgebung in den Bildern

Sofern nicht anders angegeben wurde für die in dieser Arbeit gezeigten Strukturbilder die folgende Farbgebung verwendet: Proteinoberflächen sind gelb dargestellt, Liganden sind entweder einfarbig oder koloriert gezeigt. Bei der Kolorierung nach Atomtypen wurden Sauerstoffatome rot, Stickstoffatome dunkelblau, Wasserstoffatome cyan, Schwefelatome gelb und Kohlenstoffatome weiß dargestellt. Bei der Kolorierung der im Rahmen dieser Arbeit neu generierten Liganden wurden die Atome und Bindungen entsprechend der

Beschreibung in der Bildunterschrift zu Abbildung 35 auf Seite 51 eingefärbt. *Docking-spheres* wurden als farbige Kugeln dargestellt.

Programmcode

Die im Rahmen dieser Dissertation programmierten Skripte und Programme sind im Anhang 7.3 abgedruckt. Aus Platzgründen ist von Gruppen sehr ähnlicher Skripte jeweils nur eines exemplarisch präsentiert. Um die Vielfältigkeit der Anwendbarkeit der Programme zu erhöhen, wurden sie modular aufgebaut und variierbar miteinander verknüpft. Hierdurch war und ist es möglich, bestimmte Arbeitsschritte einzeln abzuändern, zu wiederholen oder auch mit Hilfe eines externen Programms zu ersetzen.

1 Einleitung

1.1 Wirkstoffdesign historisch

Bis in das 20. Jahrhundert hinein war die Basis der Wirkstoffentwicklung das tradierte Wissen aus der Volksmedizin, gelegentlich ergänzt durch Zufallsentdeckungen. Im 19. Jahrhundert wurde mit der systematischen Testung neuer Arzneimittel im Tiermodell begonnen.[2] Des Weiteren gewannen aus Naturprodukten isolierte und nachsynthetisierte Naturstoffe an Bedeutung, mit denen es erstmals möglich war, Wirkstoffe genau zu dosieren. Tierversuche sind bis heute ein wichtiges Mittel bei der Wirkstoffentwicklung, konnten aber auf vielen Gebieten durch ethisch weniger bedenkliche und oft auch aussagekräftigere Systeme wie Zellkultursysteme oder *in vitro*-Studien ersetzt werden.

Die Suche nach neuen Wirkstoffen orientiert sich an den so genannten Leitstrukturen (*leads*). Dies sind Moleküle, die zwar einige der gewünschten Eigenschaften tragen, zum Beispiel Bindung an das Zielprotein, denen aber andere noch fehlen, zum Beispiel orale Verfügbarkeit. Basierten Leitstrukturen früher meist auf Naturstoffen, ist heute der Ursprung vieler Leitstrukturen das Hochdurchsatz-*screening* (*high throughput screening* - HTS).[3-5] Hierbei werden pro Tag 10.000 bis 100.000 synthetische Verbindungen in vollautomatischen HTS-Anlagen auf ihre Bindung an ein Zielprotein (*target*) untersucht. Eine Grundvoraussetzung für das HTS ist das Vorhandensein einer großen Anzahl von Verbindungen. Hierfür bedienen sich die Pharmafirmen der Methoden der kombinatorischen Chemie (siehe Abschnitt 1.2.1) und halten riesige Substanzbibliotheken in so genannten *compound repositories* (roboterisierten Chemikalienlagern) vor.[6;7] Die verschiedenen *screening*-Verfahren (z.B. HTS) liefern eine große Anzahl von *hits* (Verbindungen, die Bindung oder Wirkung im Assay gezeigt haben). Aus diesen werden in einem "*hit to lead*" genannten Prozess mehrere Leitstrukturen ausgewählt oder entwickelt, auf denen dann die eigentliche Wirkstoffentwicklung basiert.[8]

1.2 Kombinatorische Chemie

1.2.1 Kombinatorische Synthese

Die kombinatorische Synthese ist eine der Grundlagen der Wirkstoffforschung. Methoden, die es erlauben, viele Verbindungen definierter Struktur parallel oder in einer Mischung

gleichzeitig herzustellen, wurden aufbauend auf den Arbeiten zur Synthese von Peptidbibliotheken entwickelt. Einen ausführlichen Überblick über die bislang beschriebenen kombinatorischen Synthesen niedermolekularer organischer Verbindungen, in Lösung oder an fester Phase, geben Balkenhohl *et al.* und Terrett *et al.*[9;10]

1.2.2 *In silico*-Kombinatorik

Die kombinatorische Chemie ermöglicht den Zugang zu Millionen von Verbindungen, die für das *high throughput screening* benötigt werden. Für den entsprechenden Prozess *in silico*, das so genannte *virtual screening*,[11] werden die digitalen Repräsentationen sehr vieler Moleküle benötigt. Analog zur kombinatorischen Chemie, haben sich Verfahren der *in silico*-Kombinatorik entwickelt. Einen guten Überblick geben hier Leach *et al.*[12] Grundsätzlich wird zwischen fokussierten und exploratorischen Bibliotheken unterschieden. Fokussierte Substanzbibliotheken haben eine deutliche Ausrichtung auf ein bestimmtes *target*. Sie enthalten Moleküle, die sich in vielen Eigenschaften ähneln und einen ausgewählten Parameter variieren. Exploratorische Bibliotheken hingegen sind angelegt um einen weiten Bereich physikochemischer und struktureller Eigenschaften abzudecken.

In einer viel beachteten Veröffentlichung von 1999 wiesen Teague *et al.* auf die Probleme hin, die sich daraus ergeben, wenn Bibliotheken allzu wirkstoffartig angelegt sind.[13] Da anwendbare Wirkstoffmoleküle meist bereits eine auf orale Verfügbarkeit hin optimierte Polarität und Molmasse haben, lassen sie wenig Spielraum für den Prozess der Leitstrukturoptimierung. Teague *et al.* schlagen vor, die Bibliotheken mit einer Molmassenverteilung von 100-350 g/mol und einem logP-Wert von 1-3, also mit Leitstrukturen, deren Hydrophobizität und Molekulargewicht im Laufe der Optimierung noch steigen kann, anzulegen.

1.3 Eignung von Substanzen als Wirkstoffe

Beim Erstellen von Substanzbibliotheken, *in vitro* und *in silico*, bei der Auswahl von Substanzen für *screenings* sowie beim Design neuer Substanzen, die als Wirkstoffe eingesetzt werden sollen, stellt sich immer wieder die Frage, wie zwischen wirkstoffartig und nicht-wirkstoffartig unterschieden werden kann.

Bei dieser Fragestellung geht es nicht darum vorherzusagen, welche Moleküle gute Binder für das entsprechende *target*-Protein sein werden (*primary constraints*), sondern um eine Auswahl von Molekülen, die möglichst günstige *secondary constraints*, also günstige

ADME/Tox-Eigenschaften,[14;15] eine gute Verfügbarkeit an ihrem Wirkort und geringe bis keine negativen Nebenwirkungen haben. Die Verfügbarkeit am Wirkort hängt unter anderem stark von den physikochemischen Eigenschaften der Verbindung ab, so können zum Beispiel sehr große Moleküle im allgemeinen schlecht biologische Membranen durchdringen.[16] Verschiedene Ansätze zur systematischen Untersuchung von heutigen Wirkstoffen wurden veröffentlicht, der wohl bekannteste stammt von Lipinski *et al.*[16]
In dieser Studie wurden die Eigenschaften von 2245 Wirkstoffen, die im Jahr 2001 in den USA bereits zugelassen waren, oder sich in klinischen Studien befanden, analysiert. Die Autoren stellten fest, dass beim Vorhandensein von mehr als einer der folgenden fünf Eigenschaften eine schlechte Absorption oder Permeation wahrscheinlich ist (*rule of five*):

- Das Molekül weist mehr als fünf H-Bindungsdonoren auf (Summe von OH und NH).
- Das Molekulargewicht ist größer als 500 Dalton.
- Der log P Wert ist größer als 5.
- Das Molekül weist mehr als zehn H-Bindungsakzeptoren auf (Summe von O- und N-Atomen).
- Das Molekül hat mehr als fünf rotierbare Bindungen.

Verbindungsklassen, die biologische Transporter nutzen, und daher den Regeln nicht gehorchen müssen, stellen Ausnahmen dar.

Zahlreiche neuere Methoden, die ebenfalls auf einer Analyse der physikochemischen Eigenschaften der Verbindungen beruhen, wurden veröffentlicht.[17-24]
Einen anderen Ansatz wählten Bemis und Murcko 1996 mit dem *shapes*-Konzept,[25] das in Abschnitt 1.3.1 näher beschrieben wird, da es für die vorliegende Arbeit von großer Bedeutung ist.

1.3.1 Das *shapes*-Konzept

Guy W. Bemis und Mark A. Murcko führten 1996 eine Analyse der etwa 6700 Substanzen durch, die zum damaligen Zeitpunkt in der *Comprehensive Medicinal Chemistry database*[26-28] aufgelistet waren.[25] Sie entfernten zunächst alle Verbindungen, die keine humanmedizinischen Wirkstoffe waren, beispielsweise Kontrastmittel, Geschmackstoffe oder Konservierungsmittel.[29]

Die verbliebenen 5120 Verbindungen betrachteten sie als Graphen[30;31] und analysierten sie hinsichtlich ihrer Struktur.

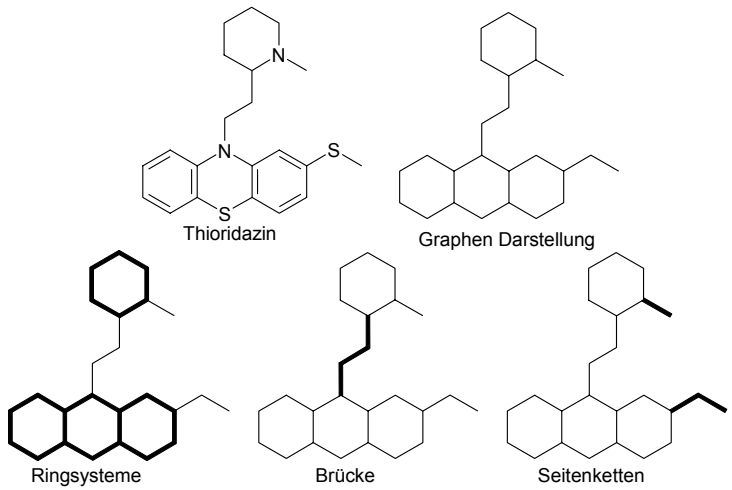


Abbildung 1: Betrachtungen von Molekülen als Graphen. Aufteilung in Ringsysteme, Brücken und Seitenketten am Beispiel des Neuroleptikums Thioridazin. In fett sind die entsprechenden Strukturelemente gezeigt.[25]

Wie in Abbildung 1 am Beispiel des Thioridazins gezeigt ist, teilten sie die Struktur der Moleküle in Gerüste (die Gesamtheit von Ringsystemen und Brücken), Ringsysteme, Brücken (*linker*) und Seitenketten auf. Sie führten Häufigkeitsanalysen der in den Wirkstoffmolekülen vorhandenen Ringsysteme, Brücken[25] und Seitenketten[32] durch.

Die Analyse der Brücken ergab, dass in den 5120 Wirkstoffmolekülen Brücken mit 0 bis 7 Atomen existierten.

Zur Häufigkeitsanalyse der Ringsysteme verwendeten sie zwei unterschiedliche Ansätze:

In einer graphentheoretischen Betrachtung[31] ignorierten sie sowohl den Atomtyp der einzelnen Atome als auch ihre Hybridisierung. Dadurch kamen sie zu einer rein 2-dimensionalen Betrachtungsweise. Als Ergebnis dieser Analyse stellten sie fest, dass sich die Datenbank aus 1179 verschiedenen Gerüsten zusammensetzte, von denen 783 (66 %) einmalig sind, da sie nur in einem Wirkstoffmolekül dieser Datenbank auftauchen. In Abbildung 2 ist die Häufigkeitsverteilung der 32 häufigsten Grundgerüste (*frameworks*) dargestellt.

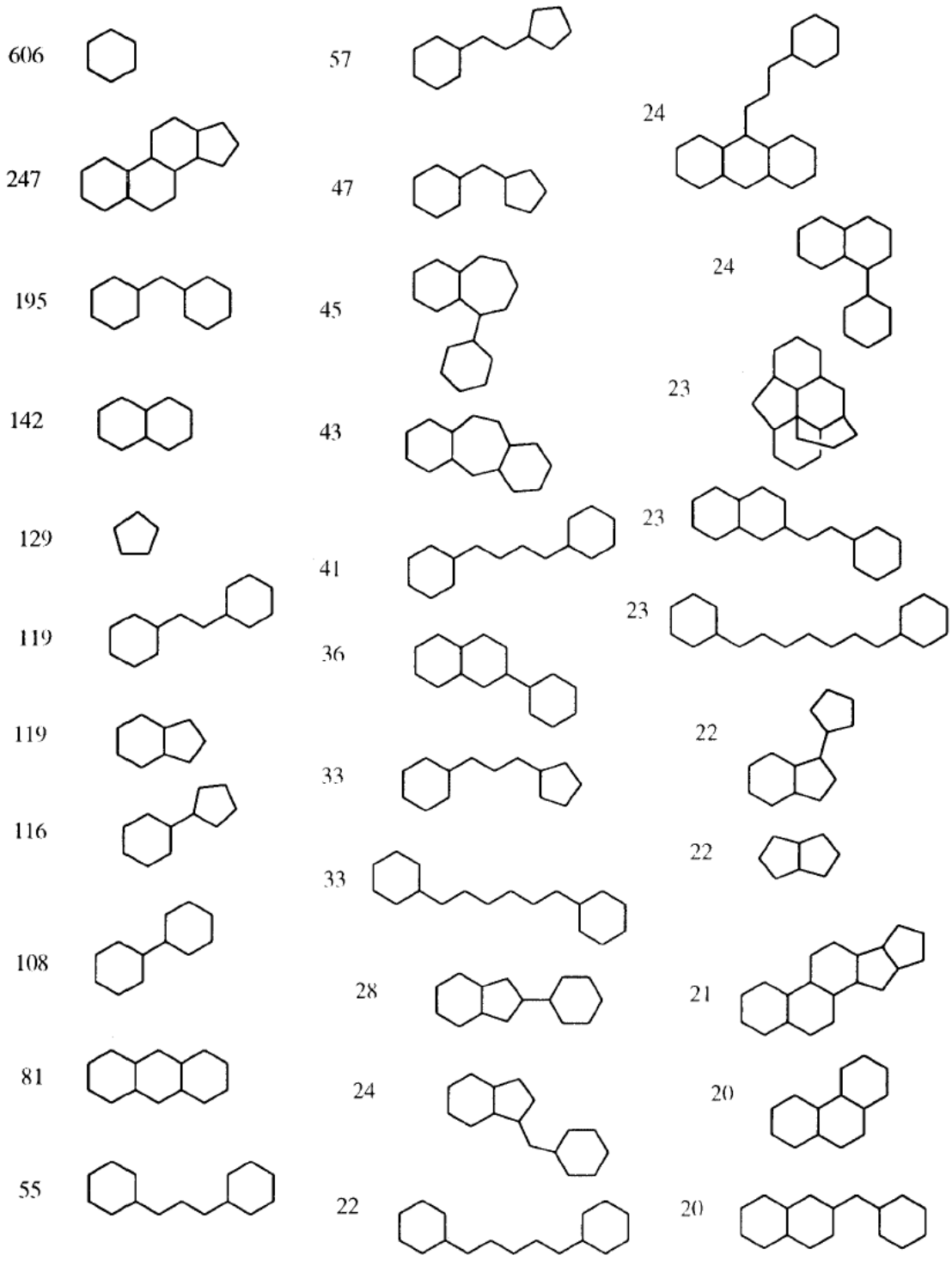


Abbildung 2: Die häufigsten Gerüste (*frameworks*) der 2D-Analyse von Bemis und Murcko. Zu jedem Gerüst ist sein Vorkommen in der Datenbank angegeben.[25]

Aus diesen 32 häufigsten Gerüsten lassen sich bereits über 50 % der in der Datenbank enthaltenen Moleküle aufbauen.

In ihrer erweiterten Analyse verwendeten sie einen Ansatz, der die Moleküle über ihre topologischen Torsionen beschreibt.[33] Hierbei werden die Hybridisierung, der Bindungstyp

und der Atomtyp berücksichtigt, wodurch die Information über die 3D-Struktur des Gesamtmoleküls erhalten bleibt. Durch die größere Anzahl an möglichen Bausteinen war zu erwarten, dass die Datenbank, aus diesem Blickwinkel betrachtet, aus mehr unterschiedliche Gerüste bestehen würde. Dies bestätigte sich mit einer Gesamtzahl von 2506 verschiedenen Gerüsten. In Abbildung 3 sind die 41 Gerüste, die in der Datenbank am häufigsten vertreten waren, aufgeführt. Wieder war eine große Anzahl (1908, entsprechend 76 %) der Gerüste einmalig. Die in Abbildung 3 gezeigten 41 häufigsten Gerüste repräsentieren bereits 1235 (24 %) der 5120 Wirkstoffmoleküle in der von Bemis und Murcko analysierten Datenbank.

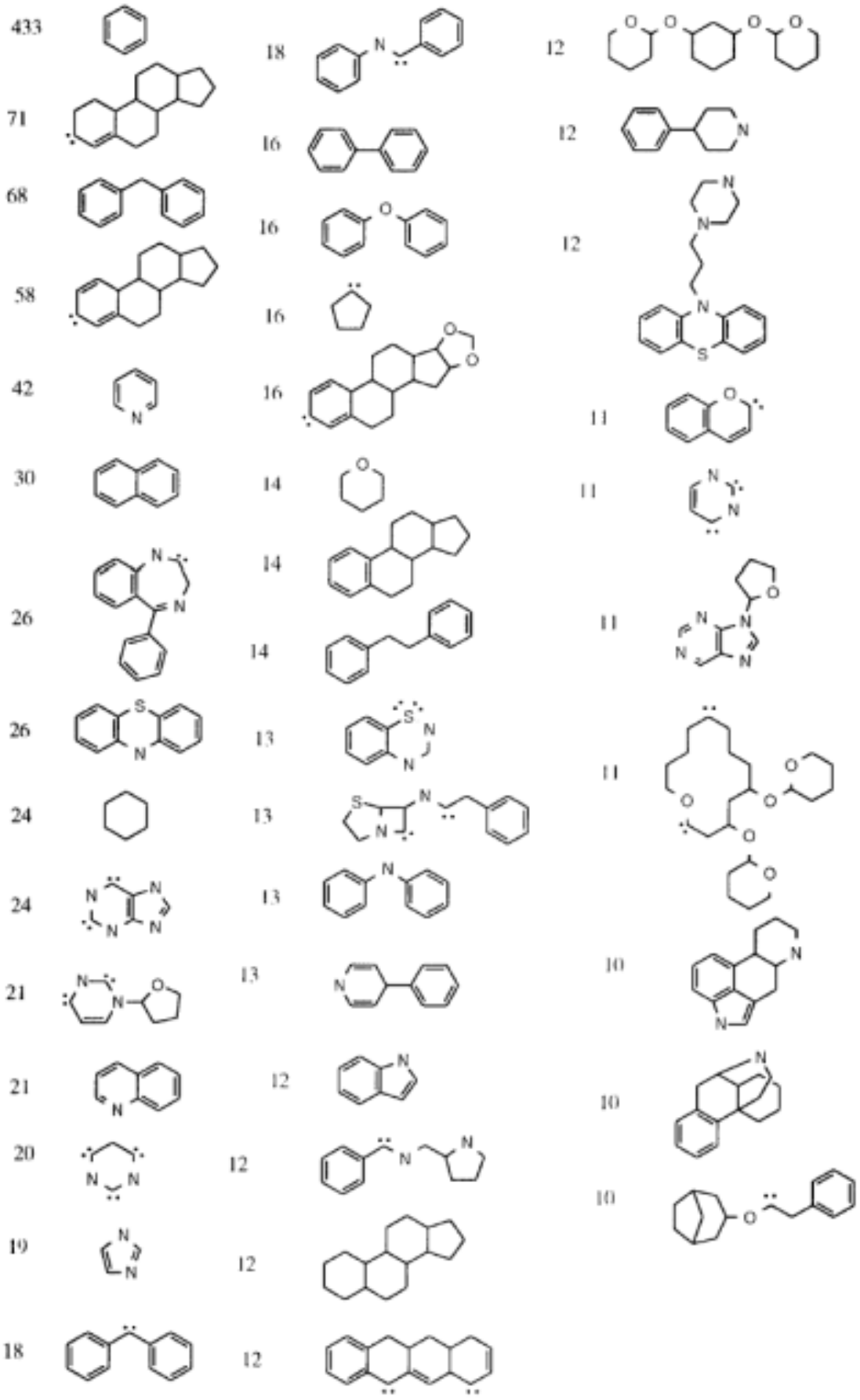


Abbildung 3: Die häufigsten Gerüste der 3D-Analyse von Bemis und Murcko.[25]

Die von den Autoren durchgeführte Analyse der Wirkstoffdatenbank ist sehr aufschlussreich. Bestimmte Grundgerüste sind in zugelassenen Wirkstoffen überdurchschnittlich häufig enthalten. Aus nur wenigen Bausteinen lässt sich wiederum ein Großteil der Wirkstoffe der Datenbank aufbauen. Dies könnte bedeuten, dass bestimmte Gerüste wirkstoffartiger sind als andere, sich also besser zum Aufbau neuer Wirkstoffe eignen. Eine wichtige Rolle für die Häufigkeit des Auftretens bestimmter Gerüste spielen selbstverständlich auch Patentrecht, Kosten, Synthetisierbarkeit und nicht zuletzt der Konservativismus ("*a tendency to make new compounds which are structurally similar to known compounds*"[25]). Trotzdem ist anzunehmen, dass die hier vorgestellte Analyse deutliche Hinweise darauf gibt, welche Gerüste besonders günstig für den Aufbau von neuen Wirkstoffen sein werden. Dies kann, wie in der vorliegenden Arbeit (siehe Abschnitt 3.1.1), genutzt werden, um im Vorfeld einer kombinatorischen Aufgabe eine Beschränkung der Bausteine auf solche, die sich in den Untersuchungen von Bemis und Murcko als besonders wirkstoffartig gezeigt hatten, vorzunehmen.

1.3.2 Peptide als Wirkstoffe

Da in dieser Arbeit der von Wülfken[34] entwickelte peptidische Ligand NMWQKVGTPL als Modellverbindung verwendet wurde, sind im Folgenden die wesentlichen Gesichtspunkte, bezüglich der Verwendung von Peptiden als Wirkstoffe aufgeführt.

Peptide haben oft starke Aktivität. Zum Beispiel als Hormone, Neurotransmitter oder Antibiotika spielen sie in biologischen Systemen oft eine große Rolle.
Aufgrund der leichten Variierbarkeit, der schnellen und gut automatisierbaren Synthese und der leichten Ableitbarkeit aus Protein-Protein-Wechselwirkungen (vgl. Abschnitt 3.2 HIV GP120) sind Peptide oft ein erster Ansatzpunkt in der Wirkstoffentwicklung. Sie eignen sich ausgezeichnet als Modellverbindungen.
Peptide als Medikamente einzusetzen ist hingegen schwierig. Insbesondere sprechen vier Gründe dagegen:

- Die schnelle Hydrolyse der Peptide durch Proteasen im Serum
- Die große konformationelle Flexibilität, die aufgrund von Entropieverlusten meist zu einer schlechten Bindungskinetik führt

- Die mangelnde orale Verfügbarkeit, die aus dem schnellen Abbau im Magen, Darm oder Serum resultiert
- Die vergleichsweise hohen Synthesekosten

Für einige spezielle Anwendungen werden Peptide als Wirkstoffe eingesetzt (z.B. Insulin, Ciclosporin, Leptin, Oxytocin, Erythropoetin), aber die negativen Eigenschaften der Peptide, besonders die schlechte Bioverfügbarkeit, verhindern in vielen Fällen den Einsatz als Therapeutika.[35-37]

Wo Peptide als Wirkstoffe eingesetzt werden, geschieht dies meist durch intravenöses oder subkutanes Spritzen. Der HIV-Fusions-Hemmer Fuzeon (T20, Enfuvirtide) beispielsweise muss zweimal täglich subkutan gespritzt werden, damit eine wirksame Konzentration aufrechterhalten wird. Bei oraler Gabe ist er wirkungslos.[38]

Für viele dieser Fälle, in denen ein guter peptidischer Binder vorlag, der aber als Wirkstoff nicht einsetzbar war, sind inzwischen Peptidomimetika entwickelt worden. Hierbei handelt es sich um Verbindungen, die von ihrer Wirkung her den Peptiden ähneln, aber nicht mehr Träger der unerwünschten Eigenschaften sind.[39]

1.4 Der Pharmakophor

Ein wichtiger Begriff bei der Beschreibung biologischer Wechselwirkungen ist der des Pharmakophors. Der Begriff wurde von Paul Ehrlich 1909 für das molekulare Muster eines Medikamentes (*pharmacon*) eingeführt, das die biologische Aktivität trägt (*phoros*).[40] Verwendet wird der Begriff heute meist für die 3D-Anordnung der die Wirkung vermittelnden funktionellen Gruppen eines Liganden innerhalb der Bindungstasche eines Proteins. Meist werden nicht spezielle funktionelle Gruppen definiert, sondern Eigenschaften, die eine an diesem Punkt befindliche Atomgruppe aufweisen muss, zum Beispiel positive Ladung, Wasserstoffbrücken-Akzeptoreigenschaften oder hohe Lipophilie. Ein Pharmakophor kann als Negativpassform des aktiven Zentrums des Rezeptors angesehen werden.

1.5 Methoden zur Bestimmung des Pharmakophors

Soll aufbauend auf einem gebundenen Liganden, ein Pharmakophor-Modell konstruiert werden, werden zwei Informationen benötigt:

- Welche Gruppen des Liganden vermitteln die Bindung zum Rezeptor?
- Welche dreidimensionale Anordnung nehmen diese Gruppen dabei ein?

Zur Ermittlung dieser beiden Informationen existieren verschiedene Verfahren, die im Folgenden beschrieben werden.

1.5.1 Alanin-Scan

Welche funktionellen Gruppen eines peptidischen Liganden an den Rezeptor binden, lässt sich zum Beispiel durch Substitutions-Experimente wie den Alanin-Scan analysieren. Hierbei werden die Aminosäuren eines bindenden peptidischen Liganden einzeln gegen Alanin ausgetauscht. Ein starker Abfall der Aktivität ist Indiz dafür, dass die entfernte Seitenkette wichtig ist. Die Durchführung eines Alanin-Scans ist einfach, die Ergebnisse sind aber nicht immer eindeutig. Führt der Austausch einer Aminosäure zum Aktivitätsverlust, kann sie entweder Teil des Pharmakophors sein oder eine strukturgebende Funktion gehabt haben. Um dies unterscheiden zu können, müssen weitere Methoden hinzugezogen werden. Eine vergleichbare Analyse des Bindungsepitops durch sukzessiven Austausch einzelner Gruppen ist auch an nicht-peptidischen Liganden möglich, jedoch meist synthetisch mit sehr viel größerem Aufwand verbunden.[41]

Zur Identifikation der bindungsaktiven Gruppen des Liganden anhand ihrer Nähe zum Protein bieten sich die STD(*saturation transfer difference*)-NMR-Spektroskopie und die Röntgenstrukturanalyse an.

1.5.2 STD-NMR-Spektroskopie

Die STD-NMR-Spektroskopie[42] kann genutzt werden, um aus Substanzbibliotheken die Liganden zu identifizieren, die an ein Zielprotein binden (*screening*).[43;44] Auch Informationen über Dissoziationskonstanten des Ligand-Rezeptor-Komplexes und damit über Bindungsstärken sowie über das Bindungsepitop[45] sind über diese Methode zugänglich.[42;43]

In Abbildung 4 ist das Prinzip der STD-NMR-Spektroskopie schematisch gezeigt.

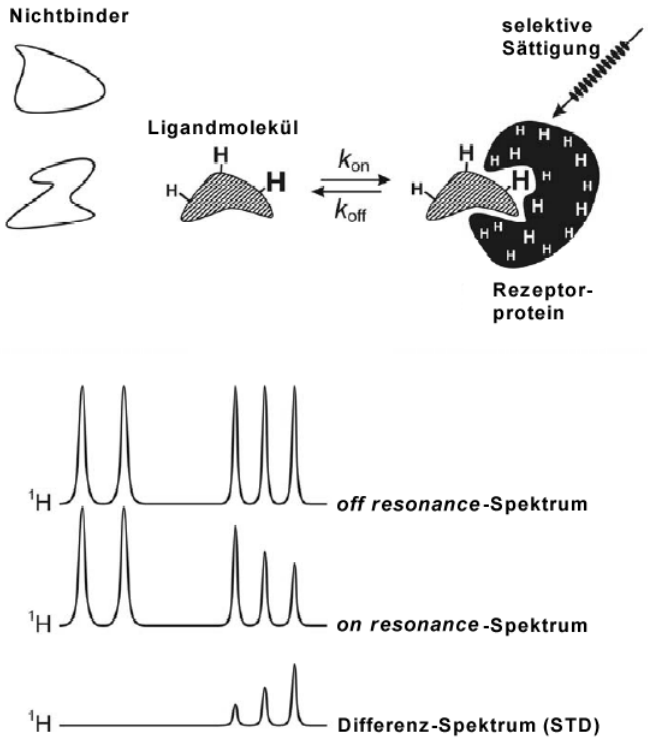


Abbildung 4: Schematische Darstellung des Prinzips der STD (*saturation transfer difference*)-NMR-Spektroskopie. Ein Teil der selektiven Sättigung des Rezeptorproteins wird auf den bindenden Liganden übertragen und von diesem in Lösung nach der Dissoziation beibehalten. Die Sättigung des Liganden ist umso intensiver, je enger der Kontakt zum Rezeptor war. Auf nicht bindende Moleküle wird keine Sättigung übertragen, somit zeigen sie im STD-Spektrum keine Signale.

Das STD-NMR-Spektrum ergibt sich aus der Differenz zweier Teilspektren, dem *on resonance*-Spektrum und dem *off resonance*-Spektrum.

Bei der Aufnahme des *on resonance*-Spektrums werden einige Resonanzen des Proteins selektiv durch Einstrahlen einer passenden Frequenz gesättigt, ohne dass die Signale der Liganden hierdurch beeinflusst werden. Durch Spindiffusion überträgt sich die Sättigung sehr schnell auf alle anderen Protonen des Proteins, so dass all diese Signale im NMR-Spektrum ausgelöscht werden. Auch auf den gebundenen Liganden überträgt sich die Sättigung. Bei seiner Dissoziation vom Protein trägt er diese immer noch, so dass die entsprechenden Signale des dann in Lösung vermessenen Liganden abgeschwächt sind.

Für die Aufnahme des *off resonance*-Spektrums liegt der Einstrahlpunkt für die Sättigung so, dass keine Proteinsignale getroffen werden, und erfolgt nur, um bei der Differenzbildung der on und *off resonance*-Spektren gleiche Bedingungen zu schaffen.

Moleküle, die nicht binden, zeigen im *on* und *off resonance*-Spektrum gleiche Signalintensitäten, so dass ihre Signale nach der Differenzbildung im Idealfall komplett verschwunden sind.

Bindende Moleküle hingegen zeigen im Differenz-Spektrum Signale. Dabei sind deren Intensitäten umso stärker, je mehr Sättigung sie während der Aufnahme des *on resonance*-Spektrums erhalten haben. Der Sättigungstransfer hängt dabei ab vom Abstand zwischen den bereits gesättigten Protonen des Proteins und den Protonen des Liganden, auf die Sättigung übertragen wird.

Die Intensitätsunterschiede der einzelnen Protonen eines Liganden innerhalb eines STD-NMR-Spektrums können somit zur Bestimmung des Bindungsepitops genutzt werden. Hierbei werden zum Teil noch weitere Informationen benötigt, da die Intensität der STD-Signale nicht ausschließlich vom Abstand der Protonen abhängt, sondern unter anderem auch vom Ligandenüberschuss, der T1-Relaxationszeit und der Kinetik der Bindung.[43;46]

Die STD-NMR-Methode konnte inzwischen erfolgreich auf Festphasensysteme,[47] auf in Liposomen eingebettete Transmembranproteine[48] und sogar auf Proteine in lebenden Zellen[49] übertragen werden.

1.5.3 Röntgenstrukturanalyse

Bei der Röntgenstrukturanalyse wird ein Einkristall der zu analysierenden Substanz mit einem gebündelten Röntgenstrahl beschossen. Die Wellenlängen der Röntgenstrahlen liegen im Ångströmbereich, daher werden sie an den Elektronen des Proteins gebeugt und gestreut. Aus dem hierbei entstehenden charakteristisches Beugungsmuster wird dann über Fourier-Transformationen die Elektronendichte bestimmt. Durch den Vergleich der Orte hoher Elektronendichte mit der Primärstruktur des Proteins kann auf die Lage der Atome im Raum und damit auf die dreidimensionale Struktur des Proteins zurück geschlossen werden.[50]

Die Elektronendichte an Wasserstoffatomen jedoch reicht für eine Beugung nicht aus. Daher ist die Position der Wasserstoffatome des analysierten Proteins auf diesem Wege nicht bestimmbar. Ist die genaue Position der Wasserstoffatome jedoch von Interesse, kann sie durch Neutronenstreuung an den Atomkernen des Kristalls ermittelt werden, denn die Streuung von Neutronen am Wasserstoffkern ist auch im Vergleich mit schwereren Kernen noch ausreichend groß.

Nur kristalline Strukturen können per Röntgenstrukturanalyse aufgeklärt werden, daher muss das zu untersuchende Protein kristallisieren. Dies hat dazu geführt, dass vielfältige Techniken

und Methoden entwickelt wurden um schnell die optimalen Bedingungen (Temperatur, Salzkonzentration, pH-Wert, Additive) für die Kristallisation eines Analyten ermitteln zu können.[51-54] Bei manchen Proteinen gelingt die Kristallisation nur nach Deglykosylierung oder Expression mit verkürzter Sequenz. Andere Proteine, vor allem die pharmakologisch besonders interessanten integralen Transmembranproteine, entziehen sich weitestgehend der Kristallisation, da es aufgrund der hydrophoben Transmembransegmente leicht zu Aggregation kommt.[55]

Zur Erzeugung von Einkristallen von Protein-Liganden-Komplexen gibt es verschiedene Möglichkeiten. Vor allem ist hier die Erzeugung gemischter Kristalle und das so genannte *soaking* zu erwähnen, das nachträgliche Einlegen eines Protein-Einkristalls in eine Lösung des Liganden.

In der 1971 angelegten *Brookhaven Protein Data Bank* (PDB) werden die so erzeugten Daten, insbesondere die Koordinaten, ihre Genauigkeit und Herkunft, abgespeichert, und sind somit öffentlich zugänglich.[56;57]

Einen guten Überblick über die Fragen, die bei der Verwendung einer Röntgenstruktur zu berücksichtigen sind, geben Davis *et al.* und weisen darauf hin, dass eine röntgenkristallographisch bestimmte Struktur immer die subjektive Interpretation einer Elektronendichtekarte durch einen Kristallographen ist.[58] Röntgenstrukturen mit guter Auflösung (kleiner als 2 Ångström) sind oft gut verwendbar.[59] Bei schlechterer Auflösung standen bei der Zuordnung der Elektronendichte zur Proteinstruktur sehr viele Interpretationsmöglichkeiten offen, was häufig zu schwerwiegenden Fehlern führte.[60]

Trotz aller Bedenken ist die Röntgenstrukturanalyse ein nützliches und weit verbreitetes Verfahren. Beim Vorliegen von Protein-Ligand-Mischkristallen kann es sowohl zur Identifikation der bindenden Gruppen des Liganden, als auch zur Analyse der 3D-Struktur des Liganden im gebundenen Zustand genutzt werden. Kuntz *et al.* stellten 2003 fest, dass die Übereinstimmung zwischen NMR- und Röntgenstrukturen, besonders im Bereich der *active sites* von Enzymen, sehr hoch ist.[61]

1.5.4 *Transferred* NOE-NMR-Spektroskopie

Eine weitere Methode zur Bestimmung der 3D-Struktur des gebundenen Liganden ist die Ausnutzung des *Nuclear Overhauser Effects*[62;63] in der *transferred* NOESY-NMR-Spektroskopie.

Hierbei wird beeinflussen sich zwei benachbarte Kernspins (Entfernung von 2 bis 5 Ångström) über die dipolare Kopplung durch den Raum. Eine Abweichung der Magnetisierung des einen Spins vom Gleichgewicht überträgt sich durch den Raum auf den anderen Kernspin, so dass dieser ebenfalls vom Gleichgewicht abweicht und sein beobachtbares Signal verstärkt oder abgeschwächt wird.

Dieser Effekt wird als *Nuclear Overhauser Effect* (NOE) bezeichnet. Er wird zur Identifikation räumlich benachbarter Kerne in mittels NOE-NMR-Spektroskopie untersuchten Molekülen genutzt. Da die Stärke des NOE vom Abstand der koppelnden Spins abhängt, können mit dieser Methode Atomabstände bestimmt werden.[64-66]

Ähnlich wie bei der oben beschriebenen STD-NMR-Spektroskopie enthält die Magnetisierung des Liganden bei der Dissoziation vom Protein noch Informationen über den gebundenen Zustand. In Lösung kann diese Magnetisierung ausgelesen, der übertragene NOE bestimmt und dadurch auf die interatomaren Abstände im gebundenen Liganden zurückgeschlossen werden.

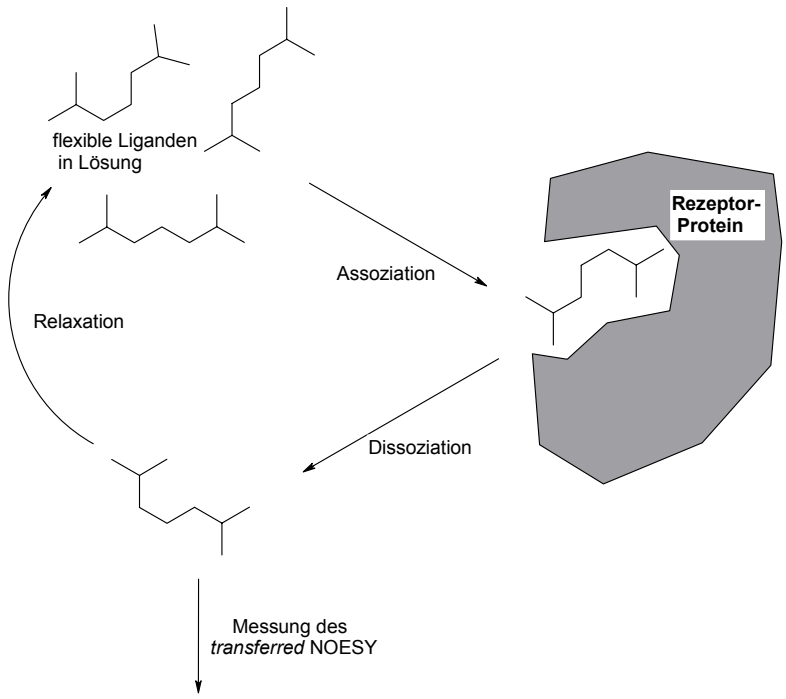


Abbildung 5: Ablauf der *transferred* NOESY-NMR-Spektroskopie. Der NOE, den der Ligand im gebundenen Zustand aufgebaut hat, baut sich nach der Dissoziation nur so langsam ab, dass er am ungebundenen Liganden noch gemessen werden kann.

Mit diesen interatomaren Abständen als Randbedingungen (*constraints*) kann dann eine Distanzgeometrierechnung durchgeführt werden, um zu den Raumkoordinaten der Atome und

damit zur 3D-Struktur des gebundenen Liganden zu kommen. Häufig wird noch eine Molekulardynamik-Rechnung (vgl. Abschnitt 1.6.3) angeschlossen, um zu identifizieren, welches der meist mehreren Strukturmodelle aus den NMR- und Distanzgeometrie-Methoden zutreffend ist.

1.6 Grundlagen des molecular modeling

Unter *molecular modeling* werden verschiedene, überwiegend computergestützte Methoden der Darstellung, Bearbeitung und Charakterisierung chemischer Strukturen und ihrer Wechselwirkungen verstanden.

Theoretisch müssten diese Systeme streng quantenmechanisch betrachtet werden.[67] Da die geschlossene Lösung der Schrödinger-Gleichung aber für größere Moleküle unmöglich ist, werden mehrere Näherungen und Vereinfachungen vorgenommen (Born-Oppenheimer, Hartree-Fock, LCAO), was zu den so genannten *ab initio*-Methoden führt.

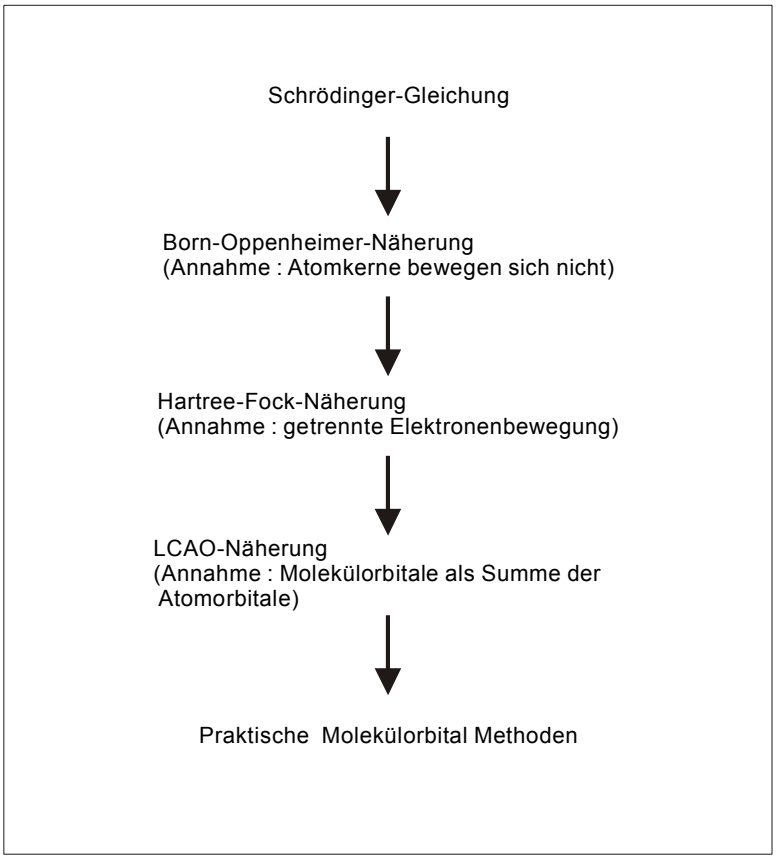


Abbildung 6: Übersicht über übliche Näherungen zur Lösung quantenmechanischer Probleme.[68]

Diese sind für Moleküle von bis zu 20 Atomen gut anwendbar und werden genutzt, um sehr grundlegende physikalische Eigenschaften der Verbindungen, wie ihre Bildungswärme, das Ionisationspotential, die Molekülgeometrie oder ihr Dipolmoment vorherzusagen.
Um auch noch größere Systeme genähert quantenmechanisch betrachten zu können, sind weitere Vereinfachungen möglich. Bei einer Größe von etwa 200 Atomen ist allerdings zur Zeit die Grenze erreicht, bis zu der ein Problem noch quantenmechanisch beschreibbar ist.
Für größere Systeme sind quantenmechanische Ansätze nicht geeignet. Hier werden klassisch-mechanische Modelle, wie die Molekülmechanik, verwendet. In dieser werden die Atome des Moleküls als Massekugeln, die Bindungen als Federkräfte betrachtet.

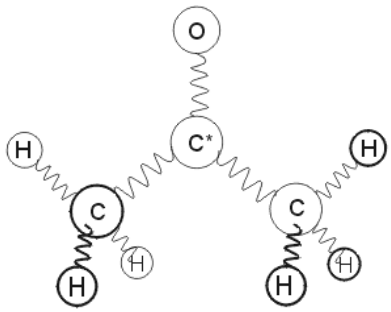


Abbildung 7: Federmodell eines Moleküls (hier Aceton)[68]

Die Modellierung dieser Kräfte zur Ermittlung einer energetisch günstigen 3D-Struktur ist die Aufgabe einer so genannten Kraftfeldberechnung.

1.6.1 Kraftfelder

Betrachtet nach der Kraftfeldmethode setzt sich die Gesamtenergie des zu beschreibenden Systems additiv aus vier Einzelbeitragen zusammen:

$$E = E_{Bindungslänge} + E_{Bindungswinkel} + E_{Torsion} + E_{nichtkovalent}$$

Formel 1: Energiebeiträge zur Gesamtenergie eines Moleküls (Kraftfeldmethode).[35]

In die Gleichungen, die die Änderung der Energie durch die Bindungsdehnung bzw. -stauchung, die Erweiterung oder Verengung des Bindungswinkels, die Verdrehung des Torsionswinkels und die van-der-Waals- und ionischen-Wechselwirkungen zwischen nichtgebundenen Atompaaren beschreibt, gehen einige Konstanten ein, die in jedem Kraftfeld

für jeden Atomtyp individuell definiert sind. Diese Parameter und damit die gesamten Kraftfelder sind immer für eine kleine Auswahl an Stoffklassen optimiert.
Wichtige Kraftfelder im Bereich der Biopolymere sind CHARMM,[69;70] AMBER[71;72] und GROMOS.[73;74] Relativ allgemein in der organischen Chemie anwendbar mit einem großen Parametersatz sind Tripos[75] und OPLS(AA).[76]
Mit den klassischen Potentialen der Molekülmechanik allein lassen sich dynamische Vorgänge, wie Bindungsbildung oder Bindungsbruch nicht beschreiben (siehe Abschnitt 1.6.3 Moleküldynamiksimulationen). Auch beruhen diese Methoden nur auf einer Beschreibung der Position des Atomkerns und lassen die Elektronen unberücksichtigt. Insofern sind sie, im Gegensatz zu den oben genannten quantenmechanischen Ansätzen, nicht in der Lage, Phänomene wie Delokalisierung, Aromatizität und Ionisierung zu erklären.

1.6.2 Energieminimierung

Energieminimierung ist eine Methode, eine energetisch günstige Konformation eines Moleküls zu ermitteln, dessen Energie aufbauend auf den oben genannten klassischen Potentialen der Molekülmechanik beschrieben wird.
Hierbei wird eine Startgeometrie des Moleküls iterativ verändert, so dass die Molekülenergie von einem Schritt zum nächsten verringert wird. Das Verfahren endet in einem lokalen Energieminimum. Welche Struktur als energetisch günstigste gefunden wird, hängt stark davon ab, von welcher Startstruktur ausgegangen und welches Kraftfeld verwendet wird.

1.6.3 Moleküldynamiksimulationen

Große Moleküle sind nicht starr. Die Funktion mancher Makromoleküle wird erst deutlich, wenn mehrere Konformere betrachtet werden, zum Beispiel beim *induced fit* von Enzymen. In der Moleküldynamik werden die Newtonschen Bewegungsgleichungen für die Atome eines Moleküls basierend auf einem vorher gewählten Kraftfeld berechnet. Dies erlaubt Aussagen über die thermodynamischen Eigenschaften, die Bewegung und über die verschiedenen Konformere von Molekülen. Moleküldynamiksimulationen bis zu einer Dauer von 10 ns mit über 100.000 Atomen sind heute auf modernen Rechnersystemen umsetzbar.[77]

1.7 Molecular modeling beim Wirkstoffdesign

1.7.1 Auswahl der Modellierungsmethode

Aufbauend auf den oben beschriebenen Grundlagen gibt es heute ein großes Repertoire von Methoden und Techniken, die beim Auffinden und Entwickeln neuer Wirkstoffe unterstützen können.[78] Eine herausragende Bedeutung hierbei hat die Modellierung von Protein-Ligand-Wechselwirkungen. Welche Methoden und Techniken hierfür eingesetzt werden können, hängt stark von den zur Verfügung stehenden Informationen über das zu modellierende System ab.

In Abbildung 8 ist ein Entscheidungsbaum für die Auswahl einer geeigneten Methode gezeigt.

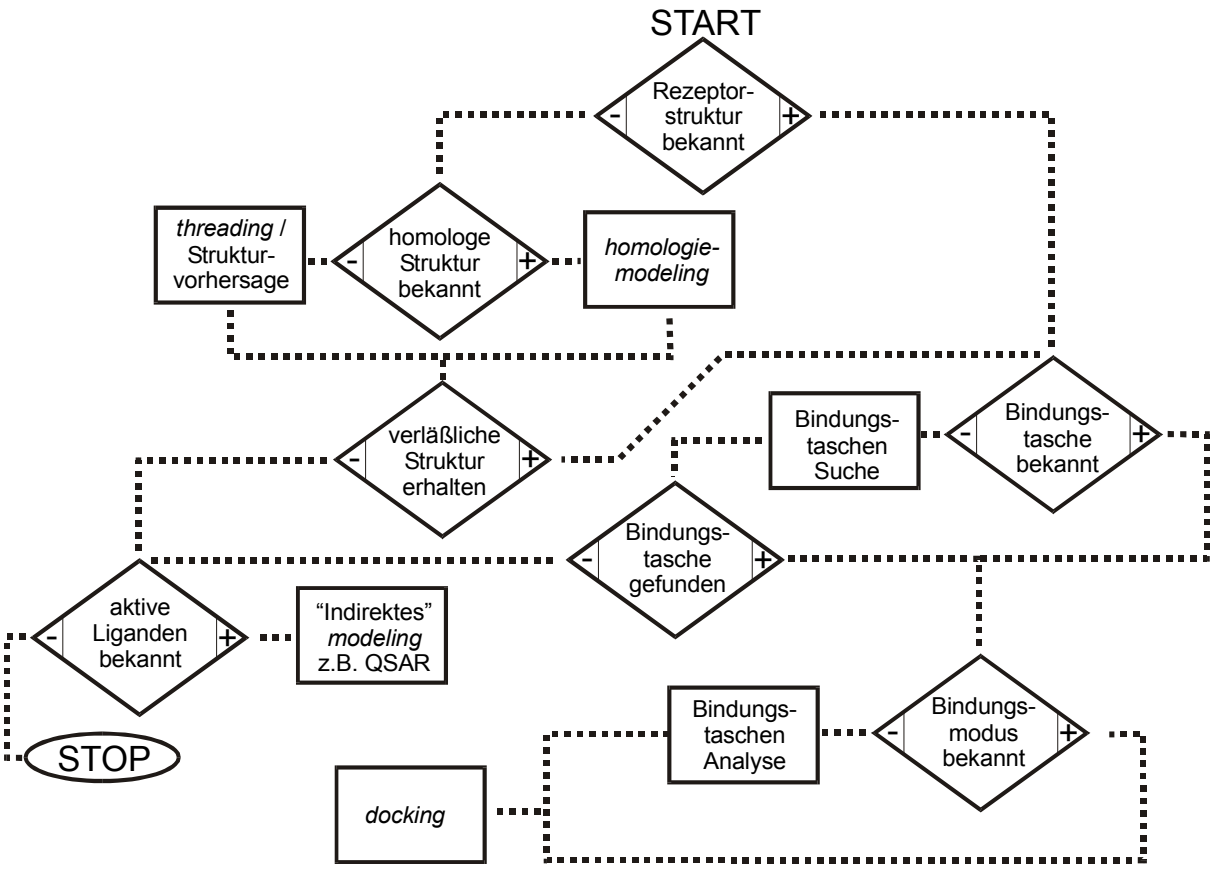


Abbildung 8: Entscheidungsbaum zur Auswahl einer geeigneten Methode des *molecular modeling*.[79]

Eine 3D-Struktur des Proteins liegt meist in Form einer Röntgenkristallstruktur vor. Ist dies nicht der Fall, kann sie oft über *threading* oder *homology modeling* vorhergesagt werden.[80-82] Sobald eine gemessene oder vorhergesagte 3D-Struktur vorliegt, muss die Bindungstasche identifiziert werden. Dies ist häufig anhand eines kokristallisierten Liganden möglich,

ansonsten existieren Methoden um eine Proteinoberfläche nach Bindungstaschen abzusuchen.[83]

Liegen sowohl die dreidimensionale Struktur des Proteins, als auch Informationen über die Bindungstasche vor, können die Methoden des *docking* oder des *de novo design* (siehe Abschnitte 1.7.2 und 1.7.3) angewendet werden.

Verfügt man nicht über eine geeignete 3D-Struktur des makromolekularen Rezeptors und gelingt es auch nicht, eine solche über *threading* oder *homology modeling* zu erzeugen, kann man sich bei der Suche nach neuen Wirkstoffen an bereits bekannten aktiven Molekülen orientieren. Hierfür existieren verschiedene unter dem Oberbegriff QSAR (quantitative Struktur-Wirkungsbeziehungen) zusammengefasste Techniken.[84-91]

1.7.2 *De novo design*

De novo design Programme identifizieren Positionen innerhalb der Bindungstasche eines Rezeptors, an denen dieser günstig mit einem Liganden wechselwirken könnte. Die Programm konstruieren neue Moleküle durch Kombination von Atomen oder Molekül-Fragmenten. Für den Aufbau der Liganden existieren zwei prinzipiell unterschiedliche Ansätze, der des sequentiellen Wachstums und der des "Platzierens und Verbrückens".

In Abbildung 9 sind diese beiden Vorgehensweisen einander schematisch gegenübergestellt. Einen guten Überblick über die zur Zeit gängigen *de novo design* Programme erschien 2005 in einem Übersichtsartikel von Schneider *et al.*[92]

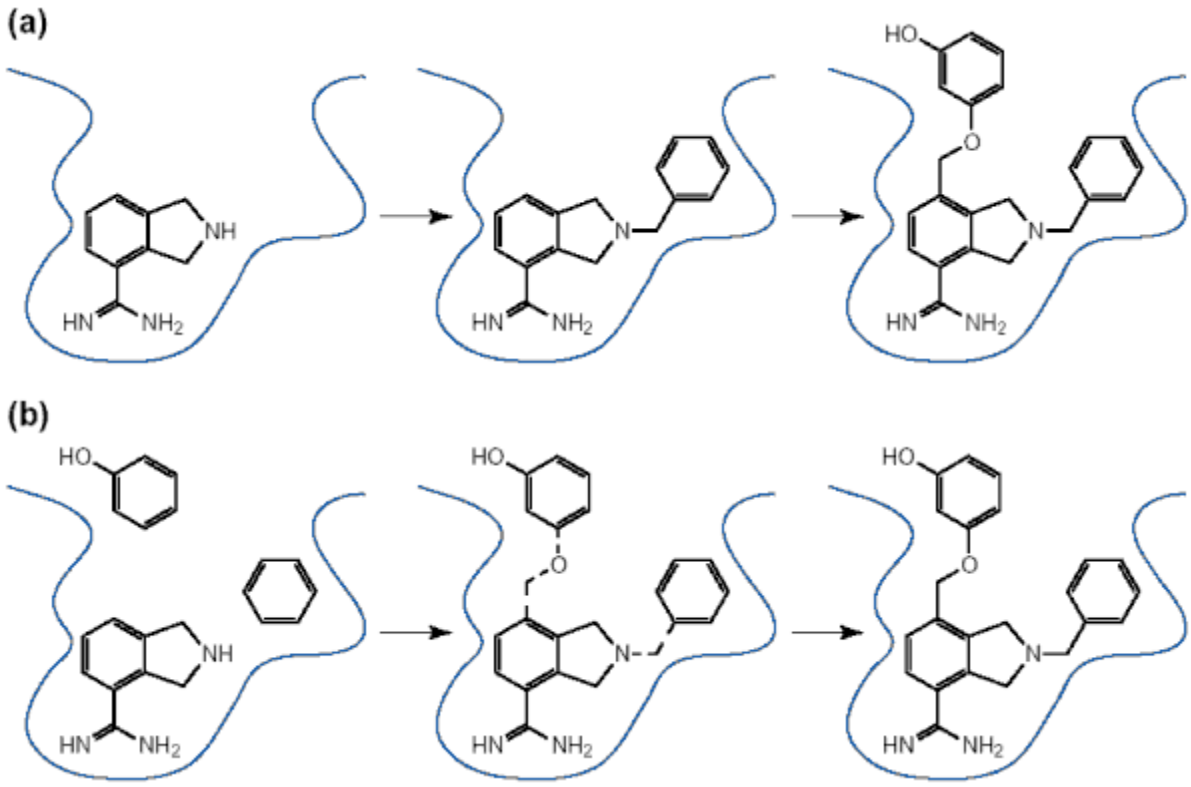


Abbildung 9: Zwei Strategien im *de novo design*. Die durchgezogene Linie repräsentiert die Bindungstasche an der Oberfläche eines Proteins. (a) sequenzielles Wachstum und (b) Platzieren und Verbrücken von Fragmenten.[93]

1.7.3 *Docking*

Docking ist ein integraler Bestandteil des Wirkstoffdesigns geworden.[94;95] Die Kosten im Entwicklungsprozess eines Wirkstoffs steigen stetig. Es ist zwar technisch möglich, aber sehr teuer, riesige Substanzbibliotheken in HTS-Assays (vgl. Abschnitt 1.1) zu untersuchen. Daher lohnt es sich durch *virtual screening*[96;97] aus diesen großen chemischen Bibliotheken nur eine überschaubare Anzahl viel versprechender Verbindungen herauszufiltern. Nur die Verbindungen, die sich *in silico* als interessant erweisen, werden dann *in vitro* auf ihre Wirkstofftauglichkeit getestet.

Docking-Programme bestehen grundsätzlich aus zwei Komponenten, einem Algorithmus, der viele verschiedene Orientierungen und Konformationen des zu dockenden Liganden in der Bindungstasche des Rezeptors erzeugt und einer so genannten *scoring-function*, die in der Lage ist, diese Anordnung von Rezeptor und Ligand energetisch zu bewerten. Damit gibt diese Funktion einen Anhaltspunkt für die zu erwartende Bindungsenergie.[98-100] Einen Überblick über die wichtigsten *scoring functions* geben Ajay *et al.*[101]

Große Bedeutung kommt der Behandlung der Flexibilität zu. Den Rezeptor flexibel zu betrachten und seine Anpassung an den bindenden Liganden zu erlauben, würde dem Konzept des *induced fit*[102] Rechnung tragen und sicherlich in vielen Fällen zu Ergebnissen führen, die dem natürlichen Zustand näher kämen.[103-108]

Komplette Rezeptorflexibilität zu berücksichtigen bedeutet einen immensen Rechenaufwand und ist im Rahmen eines *docking*-Programms bei der Leistungsfähigkeit heutiger Computersysteme nicht praktikabel.[109] Einige Programme behandeln Teile des Rezeptors als flexibel oder arbeiten parallel mit mehreren verschiedenen Versionen des Rezeptors, um seine Flexibilität zu berücksichtigen.[79;110;111]

Im Gegensatz zur Rezeptorflexibilität ist die Ligandenflexibilität vergleichsweise einfach zu implementieren, da hier eine geringere Anzahl an Freiheitsgraden zu berücksichtigen ist, was zu geringerem Rechenzeitaufwand führt. Da Liganden in den seltensten Fällen in ihrer energiegünstigsten Konformation an einen Rezeptor binden ist es nicht ausreichend, nur die energiegünstigste Konformation zu docken.[112;113] Daher wurden verschiedene Ansätze entwickelt um Liganden während des *docking* flexibel zu behandeln. Diese unterscheiden sich besonders darin, ob die Konformere des Liganden vor oder während des *docking* erzeugt werden. Werden die Konformere im Voraus erzeugt, können sie später immer wieder für verschiedene *docking*-Probleme verwendet werden, ohne dass dann zusätzliche Ressourcen für die Berücksichtigung der Flexibilität aufgewendet werden müssen.

Außerdem kann bei dieser Methode jedes *docking*-Programm verwendet werden, da die Konformere im späteren *docking* als starr betrachtet werden. Ansätze, die die Konformationen im Voraus erzeugen, gehen entweder nach der Methode der systematischen Konformationssuche vor (vgl. Abschnitt 1.8), sie erzeugen die Ligandenkonformationen basierend auf experimentell bestimmten Konformationen ähnlicher Liganden,[114] oder sie verwenden möglichst unterschiedliche Konformere des Liganden, um die Flexibilität möglichst vollständig zu repräsentieren.[113]

Werden die Konformere während der Orientierungssuche des Liganden innerhalb der Bindungsregion des Rezeptors generiert, können die Informationen über die sterischen und elektrostatischen Eigenschaften des Rezeptors zur Steuerung der Erzeugung der Konformere genutzt werden. Hierdurch lässt sich ansatzweise der *induced fit* des Liganden simulieren. Das so erzeugte Ensemble von Ligandenstrukturen ist jedoch auf diese eine Bindungstasche hin optimiert, so dass der Rechenaufwand für ein späteres *docking* des gleichen Liganden an einen anderen Rezeptor unverändert groß bleibt.

Zur Erzeugung der Konformere während des *docking* existieren verschiedene Methoden: Der inkrementelle Aufbau des Liganden mit Energieminimierungsschritten nach dem Knüpfen der einzelnen Bindungen,[115;116] *simulated annealing*-Ansätze,[117;118] Programme, die Monte Carlo Simulationen beinhalten[117;119-121] oder sich genetischer Algorithmen bedienen.[122-124]
Es existiert eine Vielzahl von *docking*-Programmen. Van Leuwen[125] hat eine Sammlung von *links* und Informationen zum Thema *docking* zusammengestellt, die als weitere Referenz dienen kann. Zu den am meisten verwendeten *docking*-Programmen gehören DOCK,[126] FLEXX,[115] GOLD,[122] AUTODOCK,[117] GLIDE,[127;128] QXP[129] und ICM.[130] Im Folgenden werden die wichtigsten *docking*-Programme kurz vorgestellt.
Das Programm DOCK, das in dieser Arbeit verwendet wurde, wird eingehend in Abschnitt 1.7.4 beschrieben.

FLEXX

FLEXX ist eines der am weitesten verbreiteten Tools für das fragmentbasierende *docking*. Diese Methode beruht darauf, dass der Ligand zunächst in Bestandteile zerschnitten, dann ein Basisfragment optimal gedockt und anschließend die restlichen Fragmente flexibel wieder angeknüpft werden. Die Ergebnisse fragmentbasierender Methoden sind sehr stark abhängig von der Wahl und der Platzierung des Basisfragmentes. In FLEXX wird die Bindungstasche durch *interaction sites* beschrieben. Das als starr angenommene Basisfragment wird so platziert, dass es mit einem Dreieck solcher Punkte überlagert. Die energiegünstigste Anordnung wird dann als Anker für die inkrementelle und flexible Neukonstruktion des ursprünglichen Liganden gewählt. Die Platzierung des Basisfragmentes durch FLEXX, im Vergleich zum Vorgehen von DOCK, ist in Abbildung 10 gezeigt. Als *scoring function* verwendet FLEXX eine Variante der Funktion von Böhm[131] und berücksichtigt Wasserstoffbrücken, Salzbücken, aromatische Wechselwirkungen, hydrophobe Wechselwirkungen sowie entropische Beiträge.

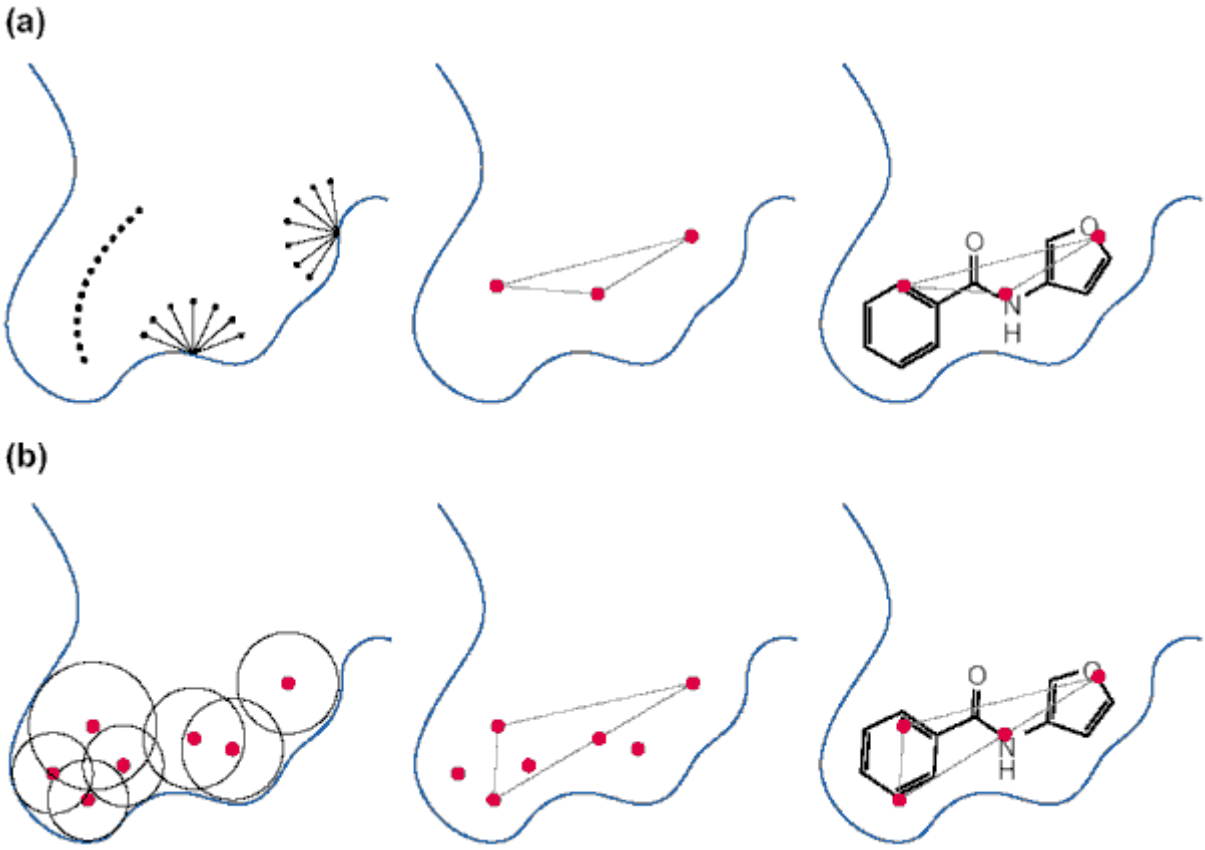


Abbildung 10: Schematische Darstellung des FLEXX-Algorithmus (a) und des DOCK-Algorithmus (b). Die geschwungene durchgezogene Linie repräsentiert die Bindungstasche an der Oberfläche eines Proteins. Die gepunktete Linie stellt eine Region günstiger lipophiler Kontakte dar, die fächerartigen Strukturen sind mögliche Wasserstoff-Brücken-Bindungen. FLEXX platziert das Basisfragment so, dass ein Dreieck seiner Atome mit einem Dreieck von solchen *interaction sites* in der Bindungstasche übereinstimmt. DOCK füllt die Bindungstasche mit *spheres* und bringt dann diese *spheres* mit Atomen des Liganden zur Deckung.[93]

Im Jahre 2001 erschien FLEXE, eine Variante von FLEXX, die in begrenztem Maße Seitenkettenflexibilität in der Bindungstasche berücksichtigt, indem sie verschiedene Konformere in diesem Bereich berücksichtigt.[111]

GOLD

GOLD (*Genetic Optimization for Ligand Docking*) arbeitet mit einem genetischen Algorithmus nach dem Inselmodell,[132] in dem die Orientierungen des Liganden und seine Wasserstoffbrücken zum Rezeptor in Form eines "Bitstring-Chromosoms" repräsentiert sind. Hierbei kodiert jedes Byte einen Torsionswinkel, während die Wasserstoffbrücken über zwei Integerstrings codiert sind. In einer Nachahmung der Evolution "entwickeln" sich diese

Orientierungen über Generationen weiter, da sie einem Selektionsdruck (Optimierung der Gesamtenergie) ausgesetzt sind. GOLD geht von der Röntgenstruktur des Rezeptors aus, in der (vgl. Abschnitt 1.5.3) Wasserstoffbrücken zwischen Ligand und Rezeptor nicht definiert sind. Um diese trotzdem berücksichtigen zu können, behandelt GOLD einen Teil des Rezeptors um die Bindungstasche herum, als flexibel. Hierdurch können sich die entsprechenden Hydroxylgruppen jeweils optimal zu ihren Bindungspartnern ausrichten. Auch den Liganden behandelt GOLD flexibel.

AUTODOCK

AUTODOCK[117;133] arbeitete in seinen ersten Versionen mit einem *simulated annealing*-Ansatz,[117] verwendet inzwischen aber ebenfalls einen genetischen Algorithmus, in dem die "Gene" die Zustandsvariablen "Ort", "Ausrichtung" und "Konformation" des Liganden beschreiben. Durch diese Umstellung des Algorithmus ist das Programm zum einen schneller, zum anderen in der Lage, Probleme höherer Dimensionalität, z.B. mehr als acht frei drehbare Bindungen im Ligand, zu berechnen.[133] AUTODOCK verwendet eine energiebasierte *scoring function*, die van-der-Waals- und elektrostatische Wechselwirkungen, Solvatations- und Entropieeffekte berücksichtigt.

In verschiedenen vergleichenden Studien[134-137] haben sich in der letzten Zeit GLIDE und ICM als die über einen weiten Bereich zuverlässigsten *docking*-Werkzeuge erwiesen.

GLIDE

GLIDE[128] arbeitet mit hierarchischen Filtern zur Einengung des Suchraumes. Die im ersten Schritt erzeugten Orientierungen des Liganden werden zur Clustern zusammengefasst, die dann im weiteren Vorgehen nur noch durch eine Orientierung repräsentiert werden. Diese Orientierungen der repräsentativen Liganden werden dann neu gedockt und minimiert, wobei nur die energetisch günstigsten erhalten bleiben. Diese wenigen Orientierungen werden dann nach dem Monte Carlo Verfahren optimiert und neu bewertet. Nur die endgültigen Orientierungen des Liganden werden so rechenintensiv bewertet. Am Anfang des *docking* hingegen filtert GLIDE sehr schnell und grob. Dadurch liefert GLIDE vergleichsweise genau gerechnete Ergebnisse in kurzer Zeit.[136]

ICM

ICM (*Internal Coordinate Mechanics*) behandelt den Liganden als vollkommen flexibel, den Rezeptor als starr. Für den Rezeptor konstruiert ICM fünf Potential-"Landschaften", die die elektrostatischen und hydrophoben Eigenschaften, die möglichen Wasserstoff-brückenbindungen und die van-der-Waals-Eigenschaften des Rezeptors beschreiben. Die Beiträge der Rezeptoratome zu diesen Potential-"Landschaften" werden bei ICM vor dem eigentlichen *docking* berechnet und in einem "*grid*" genannten Gitter zur späteren Verwendung im *scoring* abgespeichert. Als *scoring function* verwendet ICM die VLS (*virtual library screening*).[138] Das eigentliche *docking* verläuft dann über einen modifizierten Metropolis-Monte Carlo Ansatz, bei dem zwischen der Erzeugung einer neuen Orientierung und der Anwendung des Metropolis Kriteriums eine Energieminimierung (vgl. Abschnitt 1.6.2) durchgeführt wird, was die Effizienz der Methode deutlich erhöht.

Consensus docking und *-scoring*

Verschiedene veröffentlichte Artikel befassen sich damit, das "Beste *docking*-Programm" zu ermitteln. In Artikeln, die von den Autoren oder Koautoren einzelner *docking*-Programme geschrieben wurden, schneiden diese Programme oft überdurchschnittlich gut ab.[130;139] Glaubwürdiger, auf jeden Fall übertragbarer, sind Untersuchungen, in denen Anwender, meist aus der industriellen Forschung, die gängigen *docking*-Programme auf ihre Anwendbarkeit für reale Probleme testen.[97;98;140]

Die Strategie des *consensus docking* versucht durch die Kombination mehrerer Algorithmen, die Stärken der verschiedenen *docking*-Programme zu vereinen. So kann beispielsweise für ein *virtual screening* einer großen Datenbank von Liganden mit einem der schnellen Algorithmen eine Vorauswahl getroffen werden, die dann mit einem langsameren aber genaueren Algorithmus weiter reduziert wird. Ähnliches Vorgehen ist auch bei den *scoring functions* sinnvoll. So können die Ergebnisse durch die Verwendung von zwei oder drei *scoring functions* deutlich verbessert werden.[141-144]

1.7.4 DOCK

DOCK war das erste Programm, das das *docking*-Problem nicht in Form einer Simulation, sondern durch einen *clique-search* Ansatz anging.[145]

Die erste Version, wurde von Irwin D. Kuntz an der *University of California at San Francisco* entwickelt und 1982 veröffentlicht.[146] Die zur Zeit aktuelle Version ist DOCK 5.4.0.[147]
Das DOCK-Programmpaket beinhaltet unter anderem die Programme MS, SPHGEN, GRID und DOCK. Zunächst erzeugt das Programm MS eine Connolly-Oberfläche[148] des Rezeptors. Dann erzeugt das Programm SPHGEN durch einander überlappende Kugeln (in der DOCK-Terminologie *spheres* genannt) ein Negativbild dieser Connolly-Oberfläche der Bindungsregion des Rezeptors.

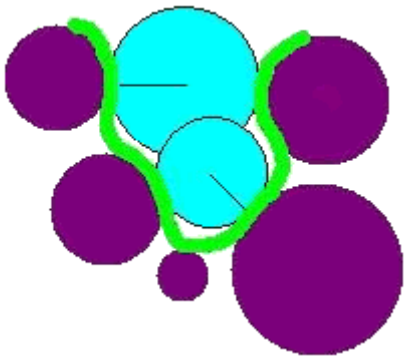


Abbildung 11: Die durchgezogene grüne Linie repräsentiert eine Connolly-Oberfläche, die über den fünf in violett dargestellten Atomen des Rezeptors erzeugt wurde. Die hellblau dargestellten *spheres*, die SPHGEN in dieser Region erzeugt hat, füllen die Bindungstasche aus. Ihre Mittelpunkte liegen auf den Oberflächennormalen (als dünne schwarze Linien dargestellt).[147]

In diese Kugeln passt DOCK dann Atome des Liganden ein, so, dass es zu einer möglichst großen Überlappung kommt (vgl. Abbildung 10b). Diese Methode des *docking* ist vergleichsweise sehr schnell[97;149;150] und hat bereits zu etlichen erfolgreichen Vorhersagen geführt,[151-157] auch beim *docking* von Proteinen an Proteine.[158] Spätere Versionen von DOCK sind in der Lage den Liganden während der Einpassung in die *spheres* des Rezeptors flexibel zu handhaben.[159] In der Ankersuche wird der Ligand zunächst virtuell zerschnitten und das resultierende größte starre Fragment als Ankerfragment (vergleichbar dem FLEXX Basisfragment, Seite 23) gedockt. Dann werden alle anderen Teile des Liganden im Rahmen einer Konformationssuche (Abschnitt 1.8) wieder angeknüpft.[160] Der resultierende Ligand kann abschließend in der Bindungstasche minimiert werden. Auch andere Ansätze zur Repräsentation der Flexibilität in DOCK wurden veröffentlicht, zum Beispiel das *docking* mehrerer als starr angenommener Konformere eines Liganden[113] oder die Verwendung von *distance geometry*-Methoden zum Aufbau einer optimierten Ligandenkonformation innerhalb der Bindungstasche.[161]

DOCK ist eines der schnellsten *docking*-Programme.[93;97] Daher wird es, neben FLEXX, besonders häufig für das *virtual screening* großer Datenbanken eingesetzt.[157]

Für das *scoring* bietet DOCK drei verschiedene Optionen: Das *energy scoring*, *contact scoring* und das *chemical scoring*.

Für das *energy scoring* verwendet DOCK einen auf dem AMBER Kraftfeld[71;72] beruhenden Kraftfeld-Ansatz, bei dem van-der-Waals- und elektrostatische Anteile berücksichtigt werden.[143;162]

$$E = \sum_{i=1}^{lig} \sum_{j=1}^{rec} \left(\frac{A_{ij}}{r_{ij}^{a}} - \frac{B_{ij}}{r_{ij}^{b}} + 332 \frac{q_i q_j}{D r_{ij}} \right)$$

Formel 2: DOCK Energiefunktion. E = intermolekulare Wechselwirkungsenergie, r_{ij} = Abstand der Atome i und j, A_{ij} und B_{ij} = van-der-Waals-Abstoßungs- und Anziehungs-Parameter, a und b = van-der-Waals-Abstoßungs- und Anziehungs-Exponenten, q_i und q_j = Punktladungen auf den Atomen i und j, D = Dieelektrizitätskonstante, 332 = Faktor zur Umrechnung der elektrostatischen Energie in kcal/mol.[163] Zusätzlich berücksichtigt DOCK die intramolekulare Energie des Liganden (vgl. Formel 1).

Das *contact scoring* verwendet eine Energiefunktion mit einem harten Gefälle. Alle Rezeptor-Ligand-Atomkontakte, die zwischen 4.5 und 2.8 Ångström liegen, werden als Kontakt gewertet und tragen positiv zur *score* bei. Alle engeren Kontakte werden mit einer vom Benutzer definierbaren Strafe belegt. Der entfernungsabhängige Verlauf des Potentials der *contact score* ist in Abbildung 12 gezeigt.

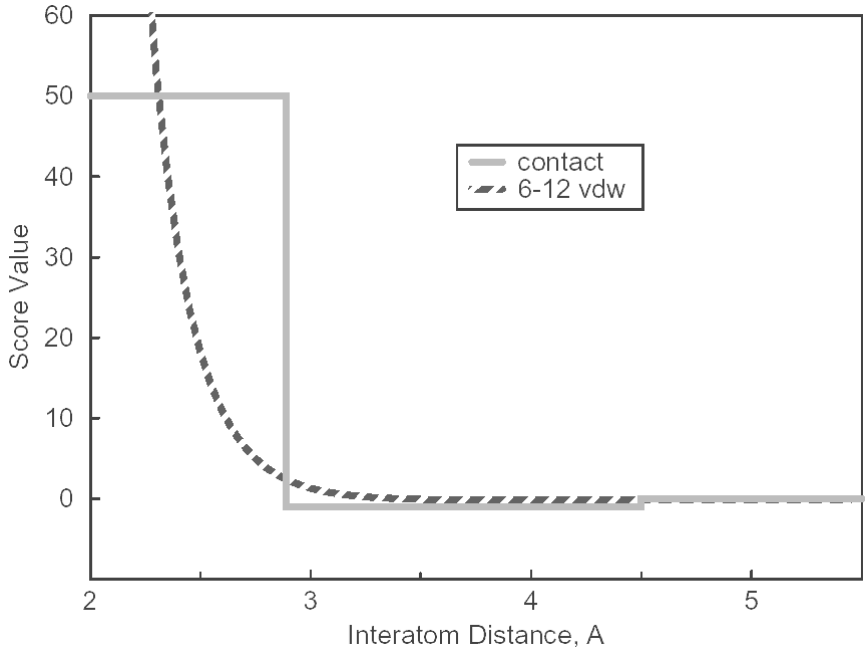


Abbildung 12: Verlauf des Potentials der DOCK-*contact score* (durchgezogene Linie) im Vergleich mit dem Verlauf eines 6-12-van-der-Waals-Potentials (gestrichelte Linie).[163]

Das *chemical scoring* erfordert eine aufwendige Parametrisierung. Es basiert auf der normalen *energy score*, erlaubt es aber dem Benutzer, mehr Kontrolle über das *scoring* auszuüben, ohne eine eigene *scoring function* programmieren zu müssen. Durch Hinzufügen von so genannten "*chemical labels*" kann der Benutzer von DOCK Bindung zwischen bestimmten Gruppen bestrafen oder begünstigen.[164]

Für alle drei Arten des *scoring* bietet das DOCK-Programmpaket[165] die Möglichkeit, die Beiträge des Rezeptors zur *score* im Voraus zu berechnen.[162] Bei diesem Vorgehen, das auch als "*force field on a grid*"[166;167] bezeichnet wird, werden, für das Beispiel der *energy score*, die van-der-Waals- und Coulomb-Einflüsse aller Rezeptoratome an Punkten eines virtuellen Gitters berechnet, das durch die Bindungstasche gelegt wird. Dieses *grid* wird zusammen mit dem Rezeptor abgespeichert.

Bei der Berechnung der intermolekularen Wechselwirkungsenergie (Formel 2) werden die am nächstliegenden Gitterpunkt summierten Einflüsse aller Rezeptoratome in die Berechnung einbezogen. Die Verwendung eines *grid* bietet sich dann an, wenn der Rezeptor als starr betrachtet wird. Es ist rechenzeiteffizient, wenn beabsichtigt wird viele Liganden oder viele Orientierungen zu bewerten.[163] In einer Untersuchung von Wu *et al.*[168] stellte sich heraus, dass die Verwendung eines *grid* (im Vergleich zu einem "*full force field*"-Ansatz, bei dem die

Wechselwirkung aller Atome einzeln berechnet wird) in einigen Fällen zu einer deutlichen Verschlechterung der Ergebnisse führte. Um in dieser Arbeit in kurzer Rechenzeit möglichst genaue Ergebnisse zu erhalten, wurde als Kompromiss ein *grid* mit sehr geringem Gitterabstand verwendet.

Für spezielle Anwendungen bietet DOCK eine große Anzahl von Möglichkeiten, die von dem bisher beschriebenen Vorgehen abweichen. So ist es möglich, bestimmte *spheres* des Liganden und des Rezeptors als so genannte *critical points* zu definieren.[169] Dies erlaubt dem Benutzer, bestimmte Wechselwirkungen in der Bindungsregion zu erzwingen, zum Beispiel um nur Orientierungen erzeugen zu lassen, in denen eine bekannte Wechselwirkung zwischen zwei funktionellen Gruppen reproduziert ist.

Noch einen Schritt weiter geht die Definition von "*ligand centers*". Bei Verwendung dieser Funktion nutzt DOCK bei der Einpassung des Liganden in den *sphere cluster* der Rezeptor-Bindungsregion nicht mehr die Schweratome des Liganden, sondern die neu definierten *ligand centers*. Durch die Kombination dieser beiden Methoden der *critical points* und *ligand centers* wurde im Rahmen dieser Arbeit das *constrained docking* (siehe Abschnitt 3.3.4) realisiert.

Zur Berücksichtigung von Rezeptorflexibilität, in Fällen wo sie eine entscheidende Rolle spielt, haben Knegtel *et al.* einen Ansatz programmiert, der mit Ensembles von Rezeptor-Strukturen arbeitet.[170]

Die im Rahmen dieser Arbeit verwendete Version 4.0.1 von DOCK wurde 1997 veröffentlicht.[165] Im Jahre 2001 veröffentlichten Ewing *et al.*[126] einen Artikel, in dem sie auf mögliche Strategien beim Arbeiten mit dieser Version von DOCK eingingen. Unter anderem validierten sie hier die Methode der Variation des random_seed, die in Abschnitt 3.5.2 verwendet wird. Seit Version 4 verwendet DOCK einen neuen und wesentlich schnelleren Algorithmus zum Einpassen der *spheres* des Liganden in die *spheres* des Rezeptors. Hierdurch konnte die *docking*-Geschwindigkeit nochmals verdoppelt werden.[165] Eine detaillierte Beschreibung des alten und des neuen Algorithmus geben Ferro *et al.*[171] und Kuhl *et al.*[172]

Für weitere Details zu DOCK sei hier auf das DOCK4-Handbuch[163] und auf die Abschnitte 3.3 bis 3.5 der vorliegenden Arbeit verwiesen. Einen Überblick über die wichtigsten Schritte bei der Arbeit mit DOCK gibt Abbildung 51.

1.8 Molekulare Konformation und Konformationsanalyse

Als Konformation wird die Orientierung von Molekülteilen zueinander bezeichnet, die sich durch Drehung um Einfachbindungen ergibt. Moleküle, die sich nur in der Verdrehung von Molekülteilen unterscheiden, werden als Konformationsisomere oder Konformere bezeichnet.[173]

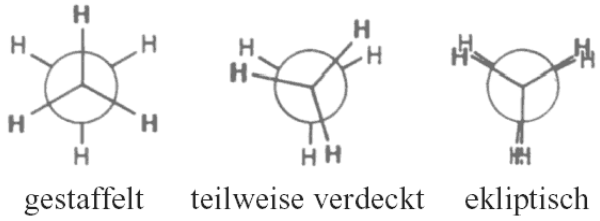


Abbildung 13: Drei Newman-Projektionen des Ethans. Das Molekül wird jeweils entlang seiner C-C-Einfachbindung betrachtet. Es sind die drei wichtigsten Extreme einer theoretisch unendlichen Anzahl möglicher Konformationen dargestellt.

Der Konformationsraum eines Moleküls ist die Menge aller seiner möglichen Konformere. Eine energetische Beurteilung des Konformationsraumes einer chemischen Struktur liefert ihre Energiehyperfläche. Jeder Punkt auf dieser Fläche (vgl. Abbildung 14) entspricht einer diskreten Konformation und dem dazugehörigen Energieinhalt.[174]

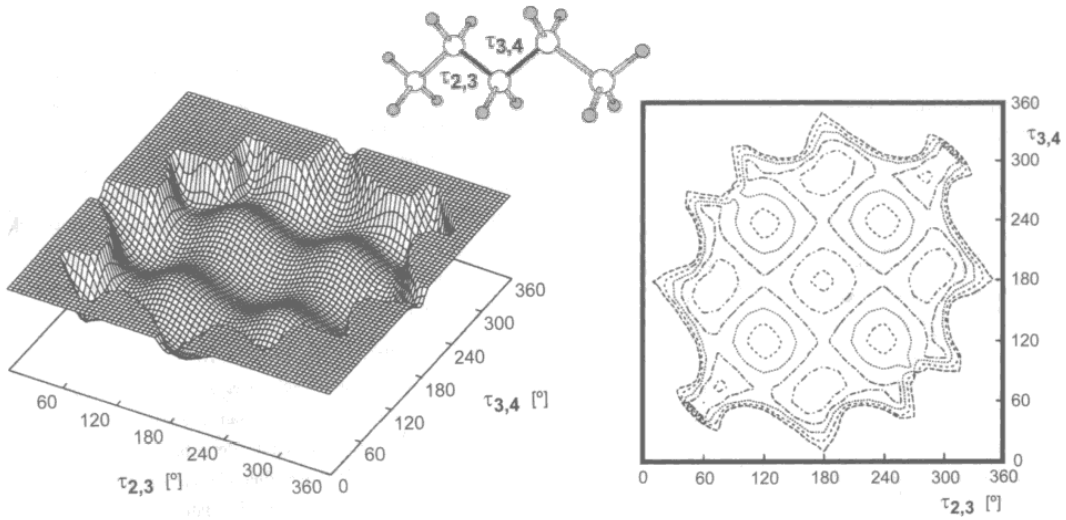


Abbildung 14: Energiehyperfläche des Konformationsraumes des Pentans. Links: 3D-Darstellung. Rechts: 2D-Darstellung mit Potentiallinien. Als Winkelinkrement wurden 5 Grad gewählt, so dass 8184 Rotamere berücksichtigt wurden.[174]

Die Analyse des Konformationsraumes eines Liganden, mit dem Ziel, die Konformation, die der Ligand bei der Bindung einnimmt, vorherzusagen oder zu beschreiben, ist von großer Bedeutung im Wirkstoffdesign. Die experimentelle Bestimmung dieser Konformation über 2D-NMR oder Röntgenkristallographie ist sehr aufwendig und kostenintensiv, zeigt aber, dass nicht immer die energetisch günstigste Konformation auch die biologisch aktive ist.
Um für das Wirkstoffdesign über eine repräsentative Auswahl von Konformeren günstiger Energie zu verfügen, kann der Konformationsraum eines Moleküls auf verschiedene Arten analysiert werden.
Die systematische Methode (auch angewendet im Rahmen dieser Arbeit, Abschnitt 3.1.4) besteht darin, jeden Torsionswinkel im Molekül schrittweise um einen konstanten Winkel zu variieren und die Energie jedes Rotamers auf der Energiehyperfläche aufzutragen. Bei einem Winkelinkrement von 60 Grad ergibt das bei Pentan (mit zwei relevanten drehbaren Bindungen) 36 diskrete Rotamere, eher wenig, um den Verlauf der entsprechenden Energiehyperfläche in Abbildung 14 abzuschätzen. Die Anzahl der zu berücksichtigenden Rotamere ergibt sich nach folgender Formel:

$$n = \left(\frac{360}{inc} \right)^{k}$$

Formel 3: Formel zur Berechnung der Anzahl der möglichen Rotamere n aus der Anzahl der rotierbaren Bindungen k und dem Winkelinkrement inc, um das die Bindungen gedreht werden.

Für ein Molekül mit fünf rotierbaren Bindungen sind bei einem Winkelinkrement von 10 Grad demnach bereits 60.5 Millionen Rotamere zu berücksichtigen, was zeigt, dass dieses Verfahren nur für Moleküle mit wenigen rotierbaren Bindungen und einem vergleichsweise grobem Winkelinkrement anwendbar ist.
Ab einer Anzahl von drei Bindungen, also sobald mindestens vier Atome betrachtet werden, kann es zu hochenergetischen Konformeren kommen, die physikalisch unsinnig sind und nicht berücksichtigt werden müssen. Diese treten immer dann auf, wenn sich zwei Atome näher kommen, als 3/4 der Summe ihrer van-der-Waals-Radien (vgl. Formel 4 und Abbildung 15).

$$d_{\min} = \frac{3}{4}\left(r(A)_{vdW} + r(B)_{vdW}\right)$$

Formel 4: Formel zur Berechnung der minimal erlaubten Distanz zwischen zwei nicht miteinander verbundenen Atomen A und B aus ihren van-der-Waals-Radien $r(A)_{vdW}$ und $r(B)_{vdW}$.

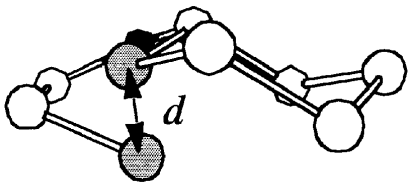


Abbildung 15: Van-der-Waals-Verletzung der schraffiert dargestellten Atome.[174]

Diese "van-der-Waals-verbotenen" Rotamere spielen physikalisch keine Rolle und können bei der systematischen Untersuchung des Konformationsraumes außer Acht gelassen werden.
Für Systeme mit mehr rotierbaren Bindungen, oder wenn eine feinere Abtastung des Konformationsraumes erforderlich ist, ist die Methode der systematischen Suche nicht mehr praktikabel.
In diesen Fällen können Monte Carlo- oder Moleküldynamik-Ansätze eingesetzt werden.[175-180]

1.9 AIDS

Gegen die Krankheit AIDS (*acquired immunodeficiency syndrome*), die durch das HIV (*human immunodeficiency virus*) ausgelöst wird, existiert bis heute kein wirksames Heilmittel und kein Impfstoff. Weltweit waren im Jahre 2005 über 40 Millionen Menschen mit dem HI-Virus infiziert. Fünf Millionen Menschen steckten sich 2005 neu mit dem Virus an, drei Millionen Menschen starben an AIDS beziehungsweise an den Folgeerkrankungen. Insgesamt hat AIDS bis heute 28 Millionen Opfer gefordert. Südostasien, das südliche Afrika und Lateinamerika sind, wie in Abbildung 16 zu sehen ist, besonders stark betroffen. Durch die hohen Infektions- und Sterberaten besonders unter jungen Menschen, droht in einigen Regionen in wenigen Jahren ein Zusammenbruch der ökonomischen und sozialen Strukturen.[181;182]

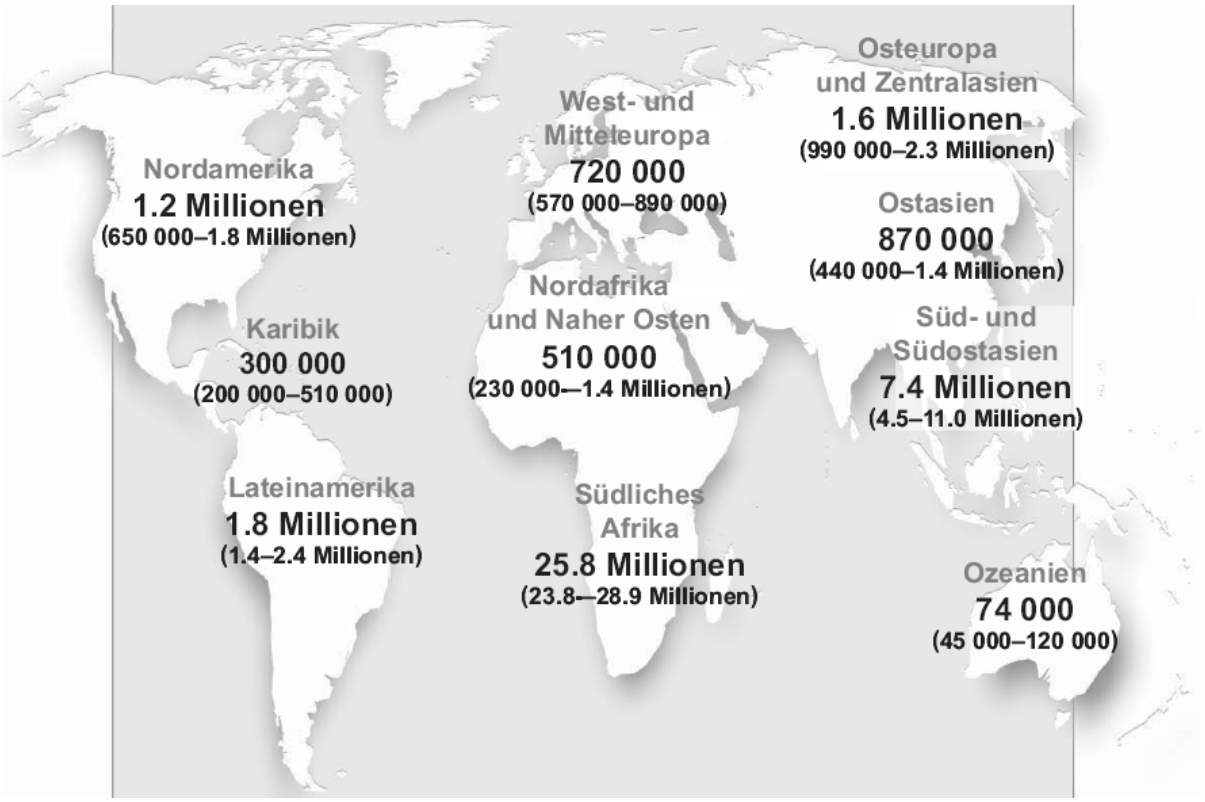


Abbildung 16: Globalen Ausbreitung von HIV. Die Bereiche in Klammern geben die Genauigkeit der jeweiligen Schätzung der Zahl der HIV-Infizierten an.[182]

Patienten werden heute mit der so genannten HAART (*highly active antiretroviral therapy*) behandelt, einer Kombination mehrerer Medikamente, die gegen unterschiedliche Stadien im Lebenszyklus von HIV gerichtet sind (vgl. Abbildung 17). Durch diese Behandlung kann die Viruslast gesenkt, das Fortschreiten der Krankheit verzögert und das Leben der Patienten verlängert werden.[183]

Aufgrund der komplizierten Dosierung der HAART, der zum Teil schweren Nebenwirkungen einiger Komponenten und des Umstandes, dass nach einiger Zeit resistente HIV-Stämme auftreten und die Patienten durch die HAART nicht geheilt werden können, sondern trotzdem sterben, besteht weiterhin ein großer Bedarf an neuen Wirkstoffen im Kampf gegen HIV.[184]

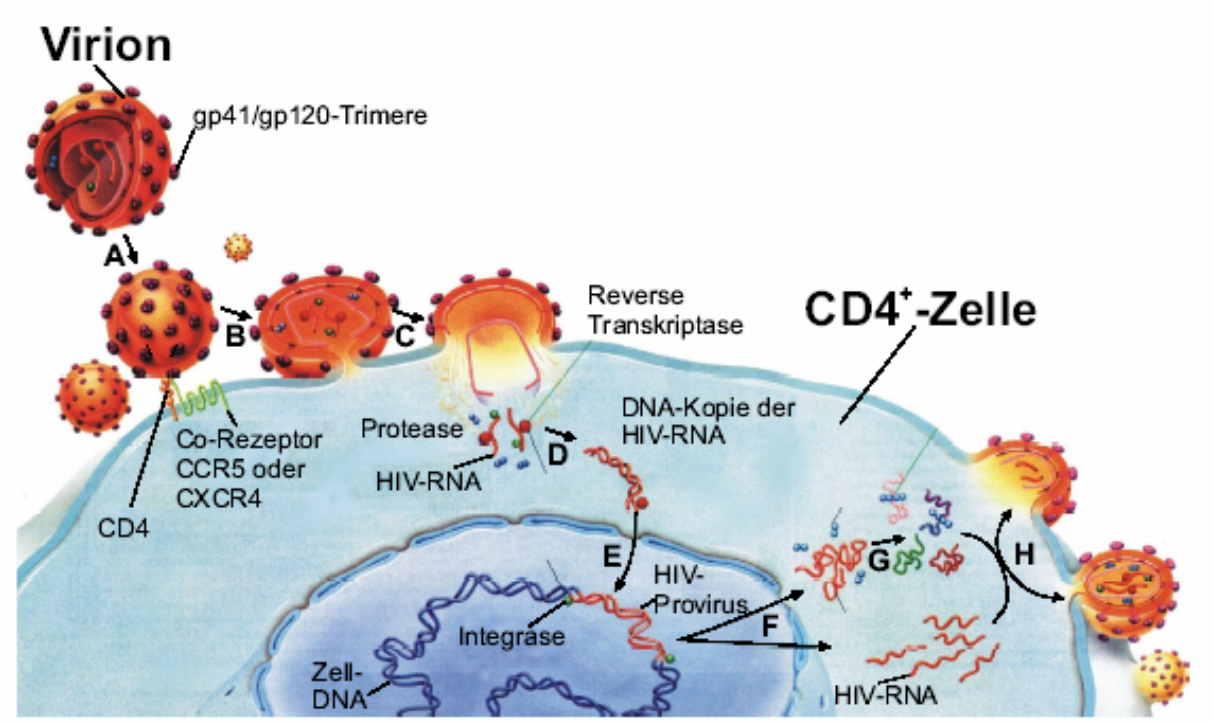


Abbildung 17: Schematische Abbildung des HIV-Lebenszyklus. Nach dem Binden des Virus an die Wirtszelle erfolgt eine Zellfusion (A, B und C). Das in die Zelle freigesetzte Genom wird anschließend durch die viruseigene Reverse Transkriptase in DNA umgeschrieben (D), in den Kern transportiert und dort durch die virale Integrase in das Genom der humanen Zelle integriert (E). Durch die normale Transkription (F) und Translation (G) der Wirtszelle entstehen wieder virale Proteine und virale RNA. Diese werden zur Oberfläche der Wirtszelle transportiert, wo sie als neue Viren ausknospen (H). Die Wechselwirkung zwischen CD4 und GP120 ist in Schritt A gezeigt. Die Komponenten der HAART richten sich gegen die Schritte B, D und G.[185]

1.9.1 Die CD4-GP120 Wechselwirkung

Der erste Kontakt zwischen der Zelle des humanen Immunsystems und dem HI-Virion ist die Wechselwirkung zwischen dem viralen Hüllprotein GP120 und dem Rezeptor CD4 auf der Zelloberfläche (Schritt A in Abbildung 17). Wenn es gelingt, diese Bindung zu verhindern, kann der Virus nicht in die Zelle eindringen und wird vom Immunsystem neutralisiert oder ist bereits nach wenigen Stunden nicht mehr infektiös.[186] Verschiedene Wirkstoffe, z.B. TNX-355, die auf eine Störung dieser Wechselwirkung abzielen, befinden sich derzeit in der klinischen Testung.[187]

Die CD4-GP120 Wechselwirkung ist gut beschrieben. Seit 1998 existiert eine Röntgenkristallstruktur (Abbildung 18) von Kwong *et al.*, in der ein gekürztes und partiell

deglykosyliertes GP120 mit der D1 und der D2-Domäne des CD4 und einem gegen GP120 gerichteten Antikörper kokristallisiert werden konnte.[188]

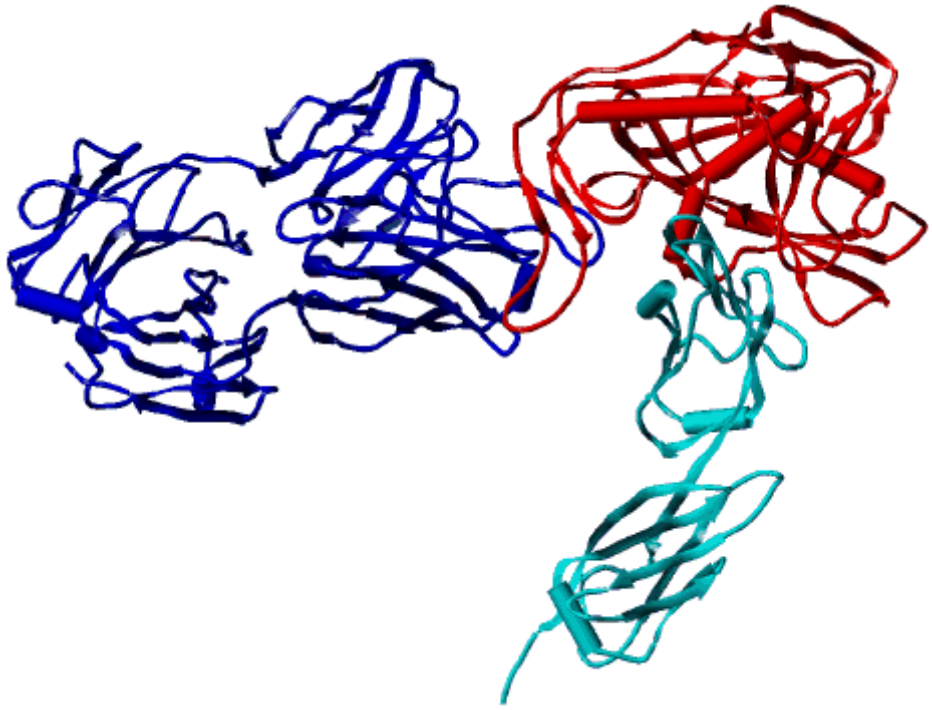


Abbildung 18: Röntgenstrukturanalyse eines ternären Komplexes aus CD4 (cyan, nur Domänen D1 und D2), GP120-core (rot) und dem Antikörpers 17b (blau, Fab-Fragment).[188]

Aufbauend auf dieser Röntgenstruktur untersuchte Wülfken 2001 die Grenzfläche CD4-GP120 und identifizierte diejenigen viralen Aminosäuren, die wahrscheinlich für die Bindung verantwortlich sind, da sie in engem Kontakt zum CD4 stehen.[34]

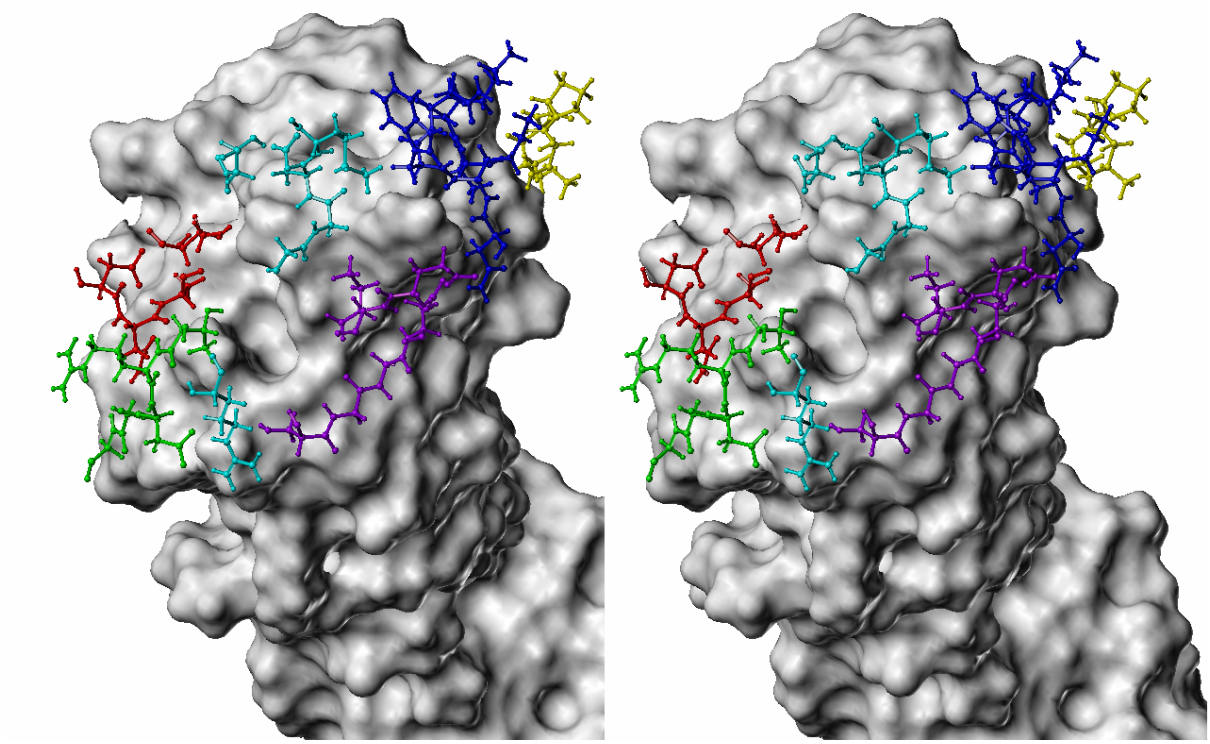


Abbildung 19: *Crossed* Stereodarstellung. Die sieben von Wülfken[34] identifizierten Peptidsequenzen des GP120, die sich in der Röntgenstruktur von Kwong *et al.*[188] in engem Kontakt zur Oberfläche des CD4 (hier weiß dargestellt) befinden. Die in der vorliegenden Arbeit näher untersuchten Sequenzen sind das dunkelblau dargestellte ^{425}NMWQKV^{430} und das gelb dargestellte ^{123}TPL^{125}.

Wülfken stellte fest, dass die beiden bindenden Peptide ^{425}NMWQKV^{430} und ^{123}TPL^{125} (in Abbildung 19 dunkelblau und gelb) ohne Lageänderung über ein Glycin verbrückt werden können. Das Peptidkonstrukt NMWQKVGTPL zeigte eine Dissoziationskonstante zum CD4 von 6 mM und inhibierte in einem Virusinhibitions-Assay[189] *in vitro* die Infektion von Zellen mit HIV. Mit Hilfe der STD-NMR-Spektroskopie ermittelte Wülfken 2001 das Bindungsepitop des Dekapeptides NMWQKVGTPL an das CD4, worauf in Abschnitt 3.2 näher eingegangen wird.

2 Problemstellung

Ziel dieser Arbeit ist es, Methoden zu entwickeln, die den Prozess der Leitstruktursuche unterstützen. Es stehen verschiedene Methoden zur Verfügung, um Pharmakophore zu charakterisieren. So liefert die STD-NMR-Spektroskopie Informationen über die an einer Bindung beteiligten Atome des Liganden und die *transferred* NOESY-NMR-Spektroskopie Informationen über die dreidimensionale Anordnung dieser Atome im Raum zum Zeitpunkt der Bindung. Ausgehend von diesen Daten soll die neue Methode wirkstoffähnliche Liganden vorschlagen. Die Methode soll an einem aktuellen Forschungsthema, der Wechselwirkung des GP120 des HIV mit dem humanen CD4, angewendet werden.
Diese Aufgabe lässt sich in folgende Teilprobleme untergliedern:

- Es soll eine Datenbank von Molekülen aufgebaut werden, die geeignet sind, typische Distanzen in einem gegebenen Pharmakophor (4 bis 20 Å) zu überspannen. Die in der Datenbank enthaltenen Moleküle sollen so ausgewählt werden, dass sie möglichst wirkstoffähnlich sind. Um mit geringem Aufwand eine möglichst große Zahl an Molekülen zu erzeugen, bietet es sich an, ein Verfahren zur *in silico*-Kombinatorik zum Aufbau der Datenbank zu entwickeln und zu verwenden. Anschließend sollen Skripte für die einfache Suche in der Datenbank bereitgestellt werden.
- Der Pharmakophor der CD4-GP120-Bindung soll *in silico* so vorbereitet werden, dass zum einen seine Charakteristika, die zur Suche in der Datenbank verwendet werden sollen, identifiziert werden können, und dass zum anderen ein *in silico*-Testsystem für die vorgeschlagenen neuen Liganden zur Verfügung steht. Als Datenbasis hierfür liegen eine Röntgenkristallstruktur von Kwong *et al.*[188] und STD-NMR-Daten von Wülfken[34] vor.
- Anhand des Pharmakophors soll dann eine Suche in der Datenbank durchgeführt werden. Die gefundenen *hits* sollen bewertet werden. Einige der besten *hits* sollen ausgewählt und *in silico* chemisch funktionalisiert werden.
- Abschließend soll die Bindung der funktionalisierten *hits* an das CD4 kritisch betrachtet werden. Bei einer vollständigen Wirkstoffentwicklung würde dieses zum Beispiel durch Synthese und *in vitro* Tests geschehen. Im Rahmen dieser Arbeit soll *unconstrained docking* zur Bewertung der entwickelten Liganden verwendet werden.

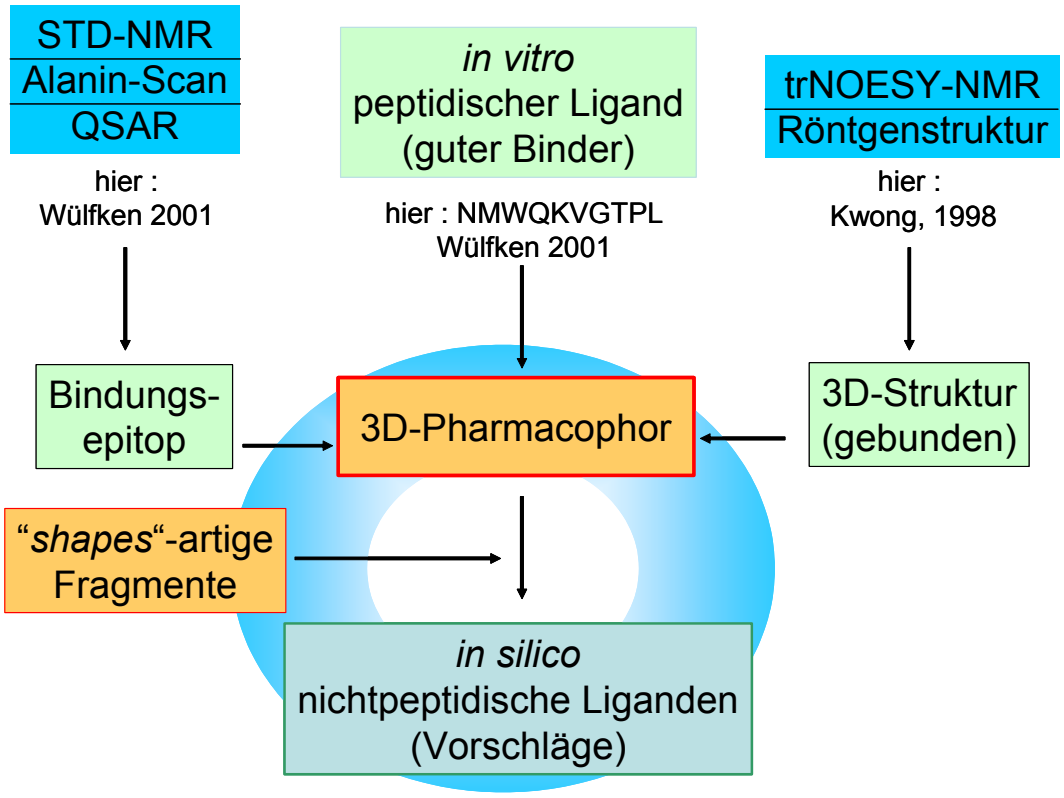


Abbildung 20: Die Problemstellung der vorliegenden Arbeit in graphischer Form. Ausgangspunkt ist ein peptidischer Ligand mit guten Bindungseigenschaften sowie Informationen über sein Bindungsepitop (ermittelt mittels STD-NMR-Spektroskopie) und seine 3D-Struktur im gebundenen Zustand (hier ermittelt aus einer Röntgenstruktur, beim Fehlen einer Röntgenstruktur auch gut zugänglich über die *transferred* NOESY-NMR-Spektroskopie). Nach Charakterisierung des 3D-Pharmakophors sollen hierfür neue Binder gefunden werden. Diese sollen basierend auf der "*shapes*"-Analyse von Bemis und Murcko[25] konstruiert werden.

3 Ergebnisse und Diskussion

Es ist üblich, in Pharmakophor-Modellen nur Schweratome zu berücksichtigen und Wasserstoffatome zu vernachlässigen.[35] Im Folgenden sollte jedoch ein Pharmakophor anhand ^{1}H-NMR-spektroskopischer Daten charakterisiert werden. Daher wurden die *shapes*[25] erweitert, indem alle chemisch sinnvollen Protonen hinzugefügt wurden.

Es wurde zunächst eine fragmentbasierte Molekül-Datenbank aufgebaut. Aus dieser können anhand einer gegebenen Distanz zwischen zwei Protonen eines Pharmakophors Moleküle ausgewählt werden, die in einer energiegünstigen Konformation zwei Protonen in diesen Abstand bringen können.

Angewendet werden sollte diese Datenbank auf die Wechselwirkung zwischen dem HIV-GP120 und dem humanen CD4. Zur Beschreibung dieser Wechselwirkung lag eine Röntgenkristallstruktur[188] und ein Bindungsepitop aus STD-NMR-Messungen[34] vor. Mit Hilfe von *modeling*-Methoden wurden diese Daten genutzt, um einen 3D-Pharmakophor zu konstruieren.

Anschließend wurde eine der Entfernungen in diesem Pharmakophor ausgewählt, ausgemessen und als Input für die Datenbank gewählt. Die besten *hits* der Antwort wurden *in silico* chemisch funktionalisiert, um ihre Bindungseigenschaften zu verbessern. Die jeweils energiegünstigsten Einzelmodifikationen wurden dann in jeweils einem Molekül zusammengefasst, dessen verbesserte Bindung an das CD4 abschließend durch *unconstrained docking* bestätigt wurde.

Ein zusammenfassender Überblick über die gesamte Struktur der Arbeit ist in Abschnitt 3.8, Abbildung 97 und Abbildung 98 gegeben (ab Seite 125).

3.1 Aufbau einer Strukturdatenbank

Viele pharmakologisch interessante Moleküle bestehen aus zwei oder mehreren Ringsystemen, die über Brücken (auch als *linker* bezeichnet) miteinander verknüpft sind.[25] Daher wurde in dieser Arbeit eine Datenbank von Molekülen, die dem Motiv "Ring-Brücke-Ring" entsprechen, konstruiert und zur Auswahl von Modellverbindungen für zukünftige Wirkstoffe verwendet.

Der Aufbau der Datenbank verlief in vier Schritten, die in den folgenden Abschnitten dargelegt werden.

Zunächst wurde die von Bemis und Murcko[25] durchgeführte Analyse der Struktur heutiger Wirkstoffmoleküle dafür genutzt, ausgewählte Ringsysteme und Brücken als Fragmente für

die Datenbank auszuwählen. Für jedes Ringsystem wurden chemisch die interessanten Positionen markiert.

Dann wurden die Fragmente *in silico* kombiniert und zu "Ring-Brücke-Ring-Systemen" (im Folgenden als RBR bezeichnet) verknüpft. Die RBR wurden einheitlich benannt, energieminimiert und in einer Datenbank abgelegt.

Anschließend wurde für jedes RBR eine systematische Konformationssuche (vgl. Abschnitt 1.8) durchgeführt, bei der die Torsionswinkel der Bindungen der Brücke systematisch variiert wurden.[190] In einer Abstandsmatrix (im verwendeten *modeling*-Programm SYBYL als "*distance map*" bezeichnet) wurden für jede Kombination von Torsionswinkeln die Entfernungen zwischen allen chemisch interessanten Wasserstoffatomen gespeichert.

Da die Konformationssuche auch die Rotamere mit hoher Energie gespeichert hatte, wurde abschließend ein "Energieschnitt" durch die Datenbank durchgeführt. Hierbei wurden nur Rotamere mit günstiger innerer Energie erhalten.

3.1.1 Auswahl der Ring-Bausteine

Um die Anzahl an Ringen, aus denen im Folgenden RBR aufgebaut werden sollten, möglichst gering zu halten und gleichzeitig Grundgerüste mit pharmakologisch günstigen Eigenschaften zu erhalten, wurden als Startpunkt die Arbeiten von Bemis und Murcko aus dem Jahre 1996 verwendet.[25]

Bemis und Murcko hatten analysiert, welche Grundgerüste in den heute verfügbaren Wirkstoffen mit welcher Häufigkeit vorkommen. Ihre Untersuchungen sind in der Einleitung (Abschnitt 1.3.1) zusammengefasst. Ein Ergebnis ihrer Studie war, dass sich aus den 41 häufigsten der 1235 Grundgerüste bereits 24 % der Wirkstoffe rekonstruieren lassen, dass also unter Verwendung nur weniger Bausteine viele der derzeit bekannten Wirkstoffe repräsentiert werden können.

Diese 41 Grundgerüste (Abbildung 21) wurden als Ausgangspunkt für die Auswahl von Ringsystemen für diese Arbeit gewählt.

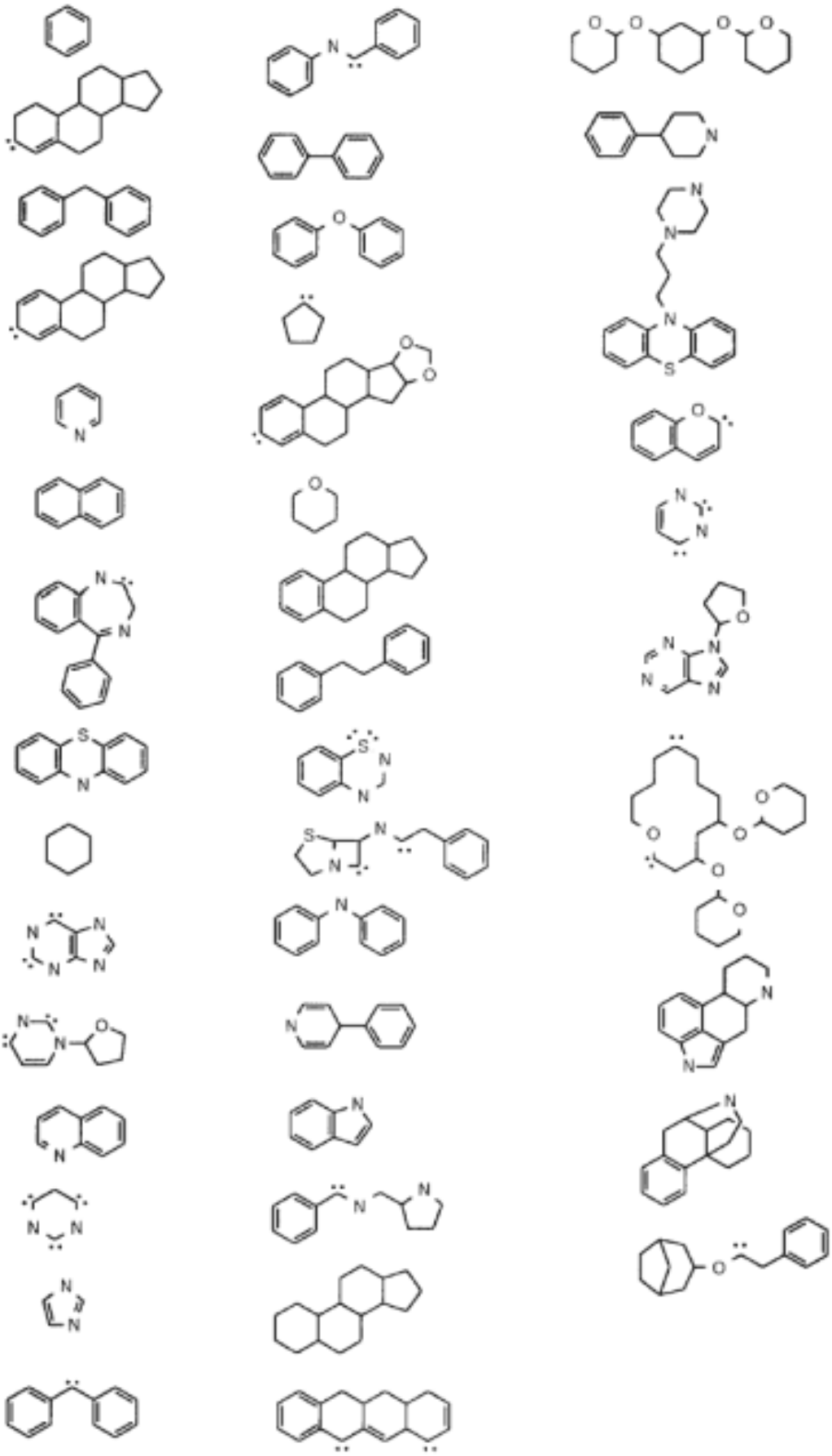


Abbildung 21: Die 41 häufigsten Gerüste der 3D-Analyse von Bemis und Murcko.[25]

Da in dieser Arbeit RBR kombinatorisch erzeugt werden sollten, wurden die 18 Grundgerüste, die bereits RBR waren, entfernt (Abbildung 22). Diese Grundgerüste waren insofern redundant, da sie als Ergebnis der kombinatorischen Verknüpfung in Abschnitt 3.1.3 erneut entstehen können.

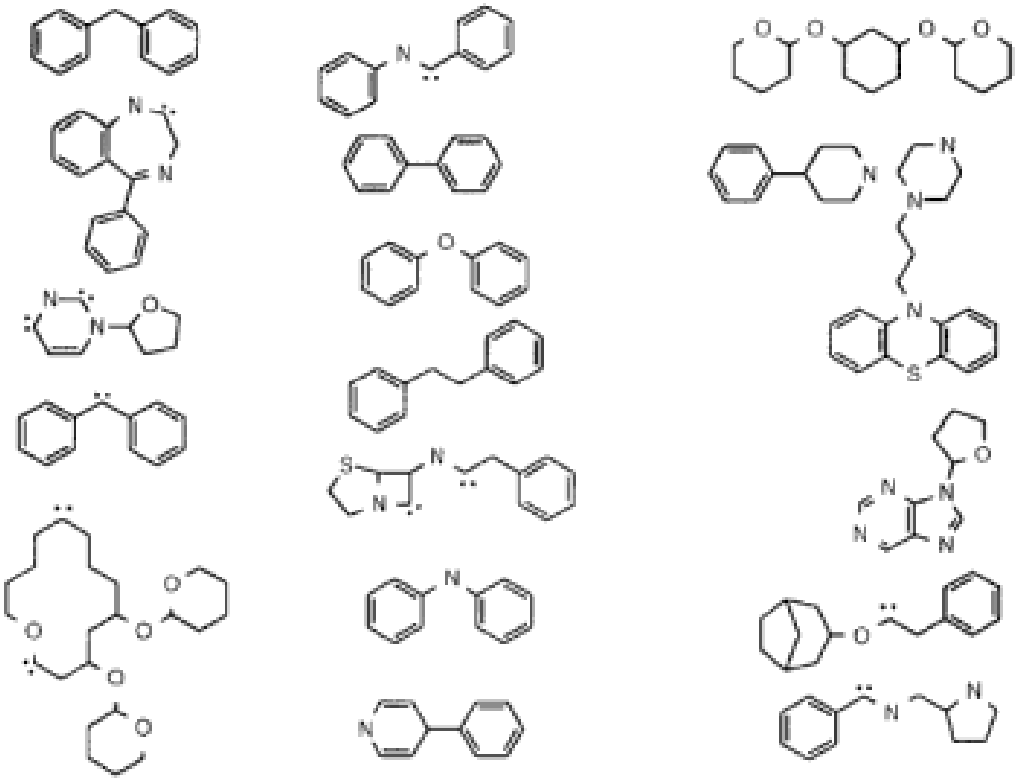


Abbildung 22: Diese Gerüste wurden aus der Auswahl entfernt, da sie bereits Ring-Brücke-Ring-Systeme (RBR) sind.

Unter den 23 verbliebenen Grundgerüsten konnten fünf identifiziert werden, die ausschließlich in Wirkstoffen vorkommen, die ein sehr eng eingeschränktes, bei vielen Leitstruktursuchen nicht erwünschtes Anwendungsgebiet haben. Die in Abbildung 23 gezeigten Grundgerüste der Phenothiazine (Neuroleptika, Antihistaminika, Antiemetika und Sedativa), Sulfonamidantibiotika, Tetrazykline (Breitbandantibiotika) und morphinartigen Alkaloide wurden aus der Auswahl von *shapes* für Schritt 3.1.3 entfernt.[191]

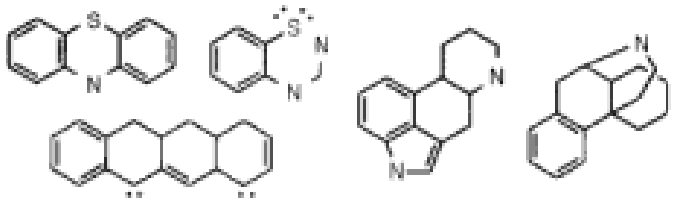


Abbildung 23: Fünf Gerüste, die aufgrund ihres eng eingeschränkten biologischen Wirkungsspektrums aus der Auswahl entfernt wurden.

Die spätere Suche in der Datenbank sollte anhand der sterischen Anforderungen des Pharmakophors erfolgen. Daher wurden sämtliche Atomtypen in den verbliebenen 18 Grundgerüsten vereinheitlicht, wodurch die Größe der Datenbank deutlich verringert werden konnte. Nur das Stickstoffatom im Indol-Gerüst behielt einen individuellen Atomtyp (N) um die planare Geometrie des Indols zu erhalten.
Allein die Information über die Hybridisierung an den jeweiligen Positionen des Gerüstes wurde erhalten. Chemische Funktionalisierung und Einführung von Heteroatomen in die sterisch bereits gut angepassten RBR erfolgte erst später und ist in Abschnitt 3.4 beschrieben.
Durch Vergleich der Gerüste untereinander fiel auf, dass weitere sieben Gerüste (Abbildung 24) entfernt werden konnten, da sie sterische Doubles anderer Gerüste (des Benzols, des Cyclohexans, des Indols oder des Naphtalins) sind.

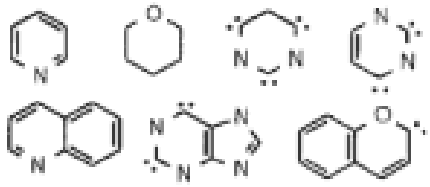


Abbildung 24: Sieben Gerüste, die aus der Auswahl entfernt wurden, da sie bei Betrachtung als 3D-Graphen sterische Doubles anderer Gerüste waren.

Von den fünf Steroid-Gerüsten, die noch in der Auswahl enthalten waren, wurden exemplarisch zwei ausgewählt, die sich in der Sättigung des A-Ringes unterschieden. Die anderen drei Steroid-Gerüste wurden entfernt.
Die verbleibende Auswahl enthielt noch zwei Fünfringe. Beide wurden entfernt und durch einen gesättigten Fünfring ersetzt, da an diesem durch das gleichzeitige Vorhandensein äquatorialer und axialer Positionen ein ausreichend breiter Raum von möglichen Substitutionen am Fünfring abgedeckt wird.

Von den 41 *shapes*, die Bemis und Murcko als die häufigsten identifiziert hatten, wurden in dieser Arbeit nach den hier beschriebenen Selektionsschritten die sieben *shapes* in Abbildung 25 verwendet.

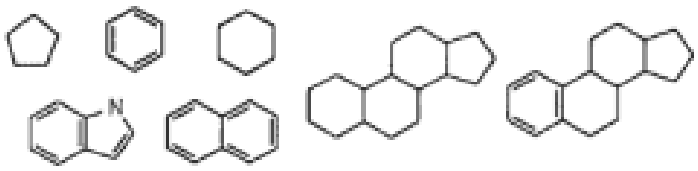


Abbildung 25: Die sieben Gerüste (*shapes*), die aus den 41, die Bemis und Murcko als häufigste beschrieben hatten (Abbildung 21), ausgewählt wurden.

Definition der Verknüpfungspunkte

Für jedes Ringsystem wurde festgelegt, welches Atom zur Verknüpfung mit den Brücken genutzt werden sollte. Bei den monozyklischen Ringsystemen (Abbildung 26) war diese Entscheidung trivial.

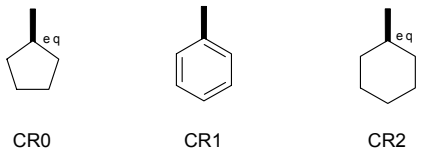


Abbildung 26: Die drei monozyklischen Ringsysteme. Die Verknüpfungspunkte sind jeweils als fett gedruckte Bindung dargestellt.

Bei den bizyklischen Ringsystemen konnte auf Vorarbeiten von Bemis und Murcko zurückgegriffen werden. Als Teil ihrer 2D-Analyse hatten sie untersucht, an welchen Atomen besonders häufig Brücken gebunden sind. Sie hatten dabei festgestellt, dass am Indol sowohl an C-2 als auch an C-3 (Abbildung 27) Brücken verknüpft sind.

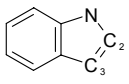


Abbildung 27: Strukturformel des Indols. Bemis und Murcko stellten fest, dass Brücken sowohl an C-2 als auch an C-3 verknüpft sind.

Um dieser Besonderheit des Bausteins gerecht zu werden, wurden am Indol-Gerüst beide Verknüpfungspunkte beim Aufbau der Datenbank berücksichtigt.
Da Bemis und Murcko in ihrer Untersuchung entsprechend für das Naphtalin-Gerüst festgestellt hatten, dass Brücken sowohl an der 1- als auch an der 2-Position verknüpft sind, wurden beide Verknüpfungsmöglichkeiten in der Datenbank berücksichtigt.

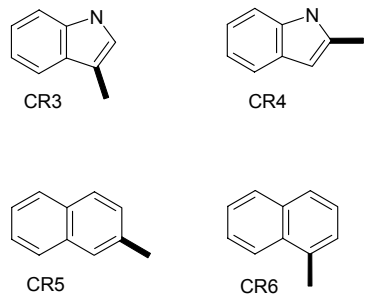


Abbildung 28: Die in die Datenbank eingebundenen bizyklischen Ringsysteme. Sowohl für das Indol-System als auch für das Naphtalin-System wurden zwei verschiedene Verknüpfungspunkte (mit fett gedruckten Bindungen markiert) berücksichtigt.

Die Verknüpfungspunkte für die Steroid-Gerüste wurden jeweils an C-17 festgelegt, da die meisten Steroide dort eine Seitenkette tragen.[192]

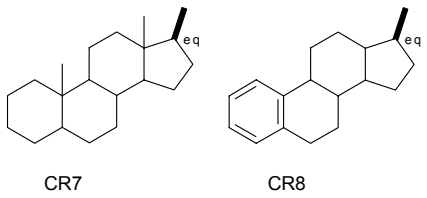


Abbildung 29: Die beiden in die Datenbank eingebundenen Steroid-Gerüste. Die Verknüpfungspunkte sind mit fett gedruckten Bindungen gekennzeichnet.

Definition der Ankerpunkte

Ankerpunkte sind die leicht chemisch modifizierbaren und repräsentativen Atome eines Ringsystems. In der systematischen Konformationssuche (Abschnitt 3.1.4) wird zwischen allen Ankerpunkten des RBR die Distanzmatrix berechnet. Zur Verringerung der Rechenzeit wurden verschiedene Kriterien zur Reduzierung der Ankerpunkte verwendet.

Bei den monozyklischen Ringsystemen (CR0, CR1 und CR2 in Abbildung 26) wurden lediglich Symmetriebetrachtungen zur Auswahl der Anker genutzt. Dies soll am Beispiel des Benzols erläutert werden.

Der Phenylring ist um die Bindung zur Brücke frei drehbar. Insofern besteht, bei einer Verknüpfung der Brücke an C-1 des Benzols, kein sterischer Unterschied zwischen den beiden ortho-ständigen Protonen H-2 und H-6, beziehungsweise zwischen den beiden meta-ständigen Protonen H-3 und H-5. Daher wurden als Anker für den Benzolring H-2, H-3 und H-4 definiert (Abbildung 30).

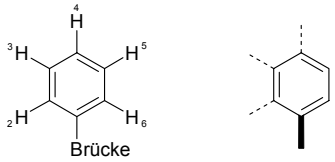


Abbildung 30: Strukturformel des Benzols, das in die Ring Datenbank als CR1 eingebunden wurde. Bei Verknüpfung der Brücke an C-1 (markiert mit einer fett gedruckten Bindung) besteht kein sterischer Unterschied zwischen den beiden ortho-ständigen Protonen H-2 und H-6. Insofern wurde von diesen nur H-2 als Ankerpunkt (durch eine gestrichelte Bindung dargestellt) definiert. Das gleiche gilt für die meta-ständigen Protonen H-3 und H-5 von denen nur H-3 als Ankerpunkt berücksichtigt wurde.

Bei den bizyklischen Ringsystemen (CR3, CR4, CR5 und CR6 in Abbildung 28) wurden nur Positionen als Anker definiert, die neue sterische Anordnungen ermöglichen. So wurde zum Beispiel am CR5 das H-2 nicht als Anker definiert. Die sterische Anordnung dieser zwei Atome ist identisch mit der, die am CR1 bereits als Anker an der ortho-Position beschrieben wurde (Abbildung 31).

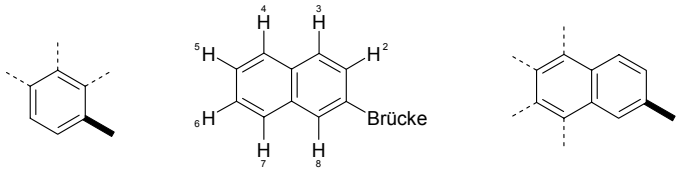


Abbildung 31: Das Naphtalin-Gerüst CR5 (Mitte), bei dem der Verknüpfungspunkt an C-1 mit einer fett gedruckten Bindung gekennzeichnet ist, im Vergleich mit dem als CR1 benannten Benzol-Gerüst (links). Die Protonen H-2 und H-3 des Naphtalins sind sterisch der ortho- und der meta-Position im Benzol äquivalent. Insofern wurden sie als Ankerpunkte für das Naphtalin nicht berücksichtigt. Die als Ankerpunkte definierten Positionen sind mit gestrichelten Bindungen markiert (rechts).

Des Weiteren wurden am Indol Symmetrie-Überlegungen in die Definition der Ankerpunkte einbezogen, so dass an einem der Indol-Ringsysteme nur zwei statt vier Anker definiert wurden (siehe Abbildung 32).

Für die Steroid-Gerüste (CR7 und CR8 in Abbildung 29) brachten weder Symmetrie- noch Redundanz-Betrachtungen eine Einschränkung der möglichen Ankerpunkte. Daher wurde hier die Häufigkeit der Substitution der einzelnen Wasserstoffatome des Steroides analysiert. Hierfür wurde das Gerüst im Programm SCIFINDER[193;194] als Ausgangspunkt für eine Suche nach chemischen Strukturen genutzt, die das jeweilige Grundgerüst als Teilstruktur enthielten. Die Ergebnisse dieser Suchen wurden visuell inspiziert, und Positionen, an denen besonders häufig Substitutionen auftraten, wurden als Anker definiert.

Die neun ausgewählten Ringsysteme wurden mit CR0 bis CR8 benannt und sind in Abbildung 32 gezeigt. Die Position am Ring, an der die Brücke (der *linker*) verknüpft werden soll, ist jeweils mit einer fett gedruckten Bindung markiert. Die Positionen, die als Ankerpunkte definiert wurden, sind in der Abbildung als gestrichelte Bindungen dargestellt. Auf eine Abbildung der Stereochemie der Ankeratome an CR0, CR2 und CR7 wurde zur Übersichtlichkeit der Abbildungen verzichtet. Der Verknüpfungspunkt der Brücken befand sich bei CR0, CR2, CR7 und CR8 in der äquatorialen Position, so dass die dort zu verknüpfende Brücke durch das Ringsystem eine möglichst geringe sterische Hinderung erfährt. Für die Ankeratome von CR0, CR2 und CR7 wurden jeweils sowohl die äquatoriale als auch die axiale Position berücksichtigt.

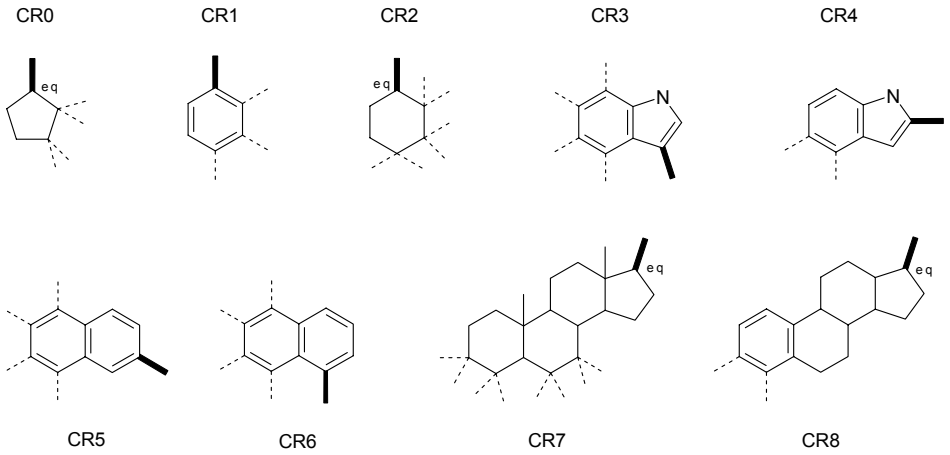


Abbildung 32: Die neun Ringsyteme CR0 bis CR8, aus denen die Ring-Datenbank aufgebaut wurde. Die Verknüpfungspunkte sind jeweils mit fett gedruckten Bindungen markiert, die Ankerpunkte als gestrichelte Bindungen. An CR0, CR2 und CR7 wurden für die Ankeratome sowohl die äquatorialen als auch die axialen Positionen berücksichtigt, auf eine Darstellung der Stereochemie wurde in dieser Abbildung zur Erhöhung der Übersichtlichkeit verzichtet.

Die neun in Abbildung 32 gezeigten Ringsysteme wurden in SYBYL[195] erstellt. Die Definition der Wasserstoffatome als Anker erfolgte durch Hinzufügen dieser Atome zu einem Tripos-Set. Es wurde eine Energieminimierung gemäß AAV 1a durchgeführt, um die Ringsysteme in einer möglichst energiegünstigen Konformation in der Datenbank abzulegen. Hierzu wurde als Platzhalter für den sterischen Anspruch der Brücke an der Verknüpfungsposition ein C-Atom eingeführt, bevor das Ringsystem minimiert wurde. Um die automatische Verknüpfung (Abschnitt 3.1.3) dieses Platzhalter-Atoms mit einer Brücke zu erleichtern, wurde diesem C-Atom die *atom ID* "1" zugewiesen.

3.1.2 Auswahl der Brücken

Auch für die Auswahl der Brücken konnten die Analysen von Bemis und Murcko herangezogen werden, in denen sich gezeigt hatte, dass in den untersuchten Wirkstoffen vor allem Brücken (*linker*) von 0 bis 7 Atomen Länge vorkamen.
Um die mittlere Rigidität der RBR zu erhöhen wurde auf die Verwendung von Brücken mit mehr als fünf frei drehbaren Bindungen verzichtet (vgl. Lipinskis *rule of five* auf Seite 3). Es wurden Brücken mit null bis vier Brücken-Atomen verwendet (Abbildung 33).

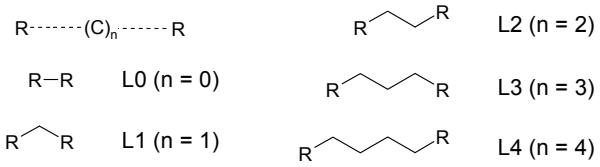


Abbildung 33: Brücken, bezeichnet mit L0 bis L4, aus denen die Brücken-Datenbank aufgebaut wurde. Mit n ist jeweils die Anzahl der durch die Brücke neu eingefügten Atome angegeben.

Die Brücken wurden in SYBYL erstellt. Hierbei wurden die Atome, die später durch die Platzhalter-C-Atome der Ringe (vgl. Abschnitt 3.1.1) ersetzt werden sollten, als Kohlenstoffatome ohne Wasserstoff-Substituenten gezeichnet. Um die automatische Verknüpfung (Abschnitt 3.1.3) zu erleichtern, wurde diesen beiden endständigen Atomen jeweils die *atom ID* "1" bzw. "2" zugewiesen. Die Brücken wurden mit L0 bis L4 benannt und vor ihrer Ablage in der Datenbank in eine gestreckte und gestaffelte Konformation (vgl. Abbildung 13) gebracht.

3.1.3 *In silico*-Kombinatorik: Aufbau der Datenbank DB1

Die Benennung der RBR erfolgte durch Verknüpfen der Namensteile. So wurde das RBR aus einem gesättigten Fünfring (CR0) und einem aromatischen Sechsring (CR1) verknüpft mit einer Brücke aus drei Atomen (L3) als CR0XL3XCR1 benannt (vgl. Abbildung 32 und Abbildung 33).

Bei der Kombination der RBR wurde darauf geachtet, kein RBR doppelt zu erzeugen, da dies die Datenbank nur redundant vergrößert hätte.

Die Verknüpfung zu RBR geschah automatisiert mit Hilfe des Skriptes 7.3.1 combine_all.spl und ist in Abbildung 34 am Beispiel des RBR CR3XL1XCR5 dargestellt. Zuerst wurden die drei zu verknüpfenden Molekülteile in drei Molekül-Ebenen von SYBYL geladen. Anschließend wurde die Brücke L1 mit dem ersten Ringsystem CR3 verknüpft. Hierfür wurde der SYBYL-Befehl "JOIN" auf die Atome mit den *atom IDs* "2" (Brücke L1) und "1" (Ring CR3) angewendet, was unter Löschung eines der Atome zum Entstehen des Ring-Brücke-Konstrukt CR3XL1 führte. Anschließend wurde dieses mit dem zweiten Ringsystem CR5 verknüpft.

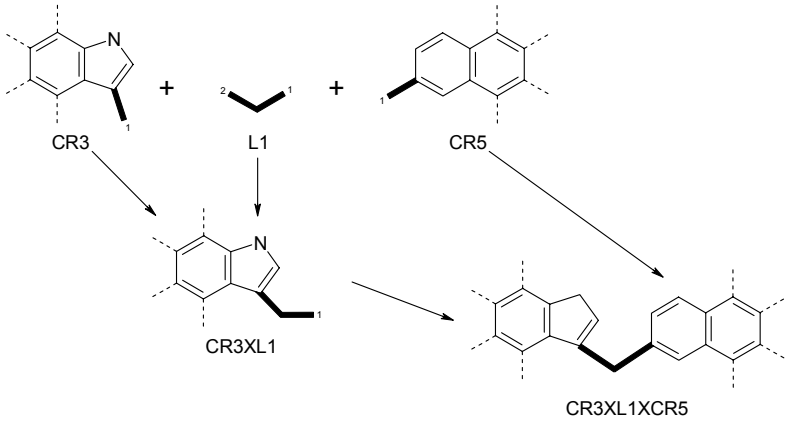


Abbildung 34: Schematische Darstellung der im Text näher beschriebenen Verknüpfung von zwei Ringsystemen und einer Brücke zu einem RBR am Beispiel von CR3XL1XCR5.

Die Information über die in Ring1 als Anker definierten Atome wurde während der Verknüpfung im Set ANCHORS_A bewahrt. Die Set-Definitionen innerhalb der anderen Molekülteile gehen während der Verknüpfung verloren. Um die Information über die in Ring2 als Anker definierten Atome zu bewahren, wurden diese Atome für die Verknüpfung auf den Atomtyp "Du" (*dummy*) gesetzt, so dass sie auch nach der Verknüpfung zum RBR noch eindeutig identifiziert werden konnten. Nach der Verknüpfung wurden alle Atome vom "Du"-Typ im Set ANCHORS_B zusammengefasst und dann auf den Atomtyp "H" zurückgesetzt. Dieses Vorgehen bedingt, dass erstens nicht von vorneherein "Du"-Atome in den Ring-Bausteinen enthalten sein dürfen und zweitens ausschließlich Wasserstoffatome als Anker definiert werden können. Auch die Information, welche Bindungen des fertigen RBR ehemals zur Brücke gehörten, welche Torsionswinkel also im der folgenden Konformationssuche variiert werden sollten, ging durch das Verknüpfen verloren. Es wurde daher ein Algorithmus entwickelt, der das Molekül nach rotierbaren Bindungen innerhalb der Brücke durchsucht (Skript combine_all.spl Zeile 372-381). Diese Bindungen wurden dann im Set BKL_ROTATABLE abgespeichert (Abbildung 35).

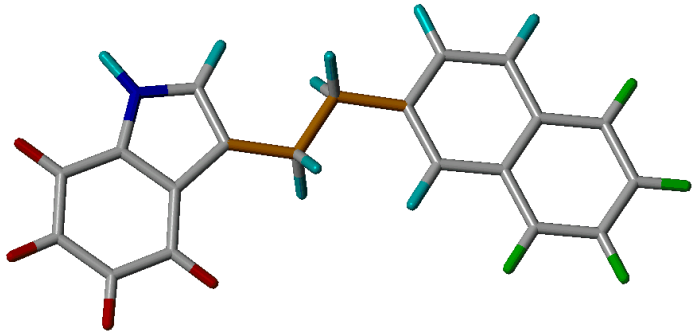


Abbildung 35: Der Farbcode für die Tripos-Sets, dargestellt am Beispiel des RBR CR3XL2XCR5. Die Anker am ersten Ringsystem CR3 sind rot dargestellt (Set ANCHORS_A). Die Anker am zweiten Ringsystem CR5 sind grün dargestellt (Set ANCHORS_B). Die im Set BKL_ROTATABLE zusammengefassten Bindungen der Brücke sind orange dargestellt. Der Rest des Moleküls ist "*by atom color*" eingefärbt, so dass Kohlenstoffatome weiß sind, Wasserstoffatome cyan und Stickstoffatome dunkelblau.

Die fertiggestellten RBR wurden in einer SYBYL-Datenbank abgelegt, die im Folgenden als Datenbank DB1 bezeichnet wird. Durch die Kombination der neun Ringsysteme CR0 bis CR8 mit den fünf Brücken L0 bis L4 ergaben sich (unter Vermeidung von doppelten Strukturen) 225 RBR.

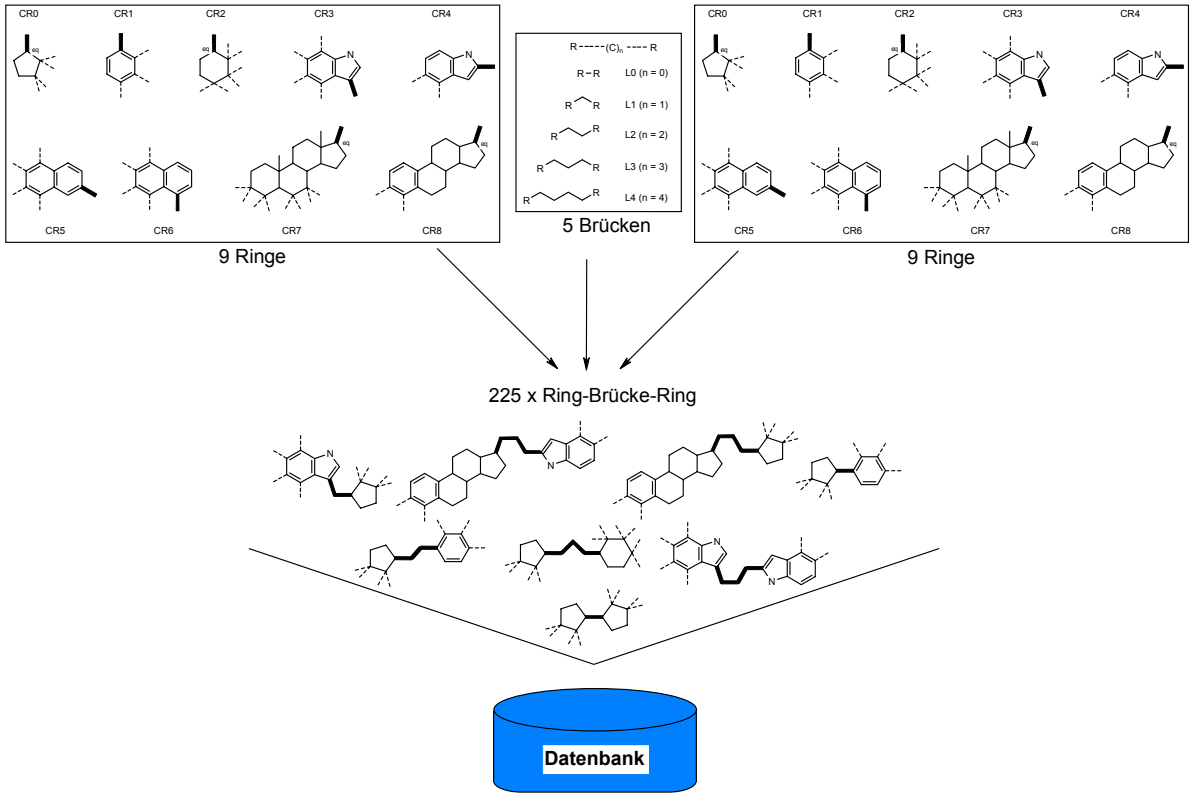


Abbildung 36: Aufbau der RBR-Datenbank durch *in silico*-Kombinatorik. Da vermieden wurde, doppelte RBR zu erzeugen (z.B. CR0XL1XCR2 und CR2XL1XCR0), wurden 225 statt 405 RBR in der Datenbank DB1 abgelegt.

Mit Hilfe des Skriptes 7.3.2 minimize_all.spl wurden anschließend alle RBR gemäß AAV 1a energieminimiert.

3.1.4 Konformationsanalyse: Aufbau der Datenbank DB2

Die in Abschnitt 3.1.3 erzeugten Moleküle befanden sich nach der Minimierung in einer energiegünstigen Konformation. Nur diese eine Konformation zu berücksichtigen würde jedoch der mitunter hohen Flexibilität der Moleküle nicht gerecht.

Um die systematischen Konformationssuchen (SYBYL-Terminus: *Systematic Conformational Search*) vorzubereiten, wurde das SPL-Skript 7.3.3 setupsearch_all.spl verwendet. Diese Suchen waren rechenaufwendig und es war bei vorherigen Versuchen immer wieder zum Ausfall des Systems gekommen. Daher wurden zuerst für alle Suchen die Einstellungen gespeichert. Der Start der Suchen erfolgte später mit einem zweiten Skript.

Für jedes RBR wurden alle die Bindungen als rotierbar definiert, die im Set BKL_ROTATABLE enthalten waren. Der Literatur ist zu entnehmen, dass 60 Grad als

Winkel-Inkrement für die meisten Moleküle ausreichend fein ist.[196] Da die Rechenzeit zur Erzeugung der Rotamere in der vorliegenden Arbeit aber nur einmalig aufgewendet werden musste und die betrachteten Moleküle nur über wenige (maximal fünf) Rotationsfreiheitsgrade verfügten, wurde eine feinere Abtastung in 10 Grad-Schritten verwendet. Diese ist als Mittelweg zwischen einer vollständigen Abtastung in 1 Grad-Schritten, mit immens hoher Rechenzeit und unter Produktion riesiger Datenmengen, und einer zu groben Abtastung (z.B. im 120 Grad-Schritten), bei der eventuell ganze Bereiche energiegünstiger Konformationen übersehen würden, zu sehen.

Die Einstellungen für die Bewertung von van-der-Waals-Wechselwirkungen wurden auf den vorgegebenen Werten ("General": 0.94, "1-4": 0.87, "H-Bond": 0.65) belassen. Durch die Verwendung dieser rechnerisch verringerten van-der-Waals-Radien wird ein relativ weiches Potential simuliert und der Flexibilität des Rezeptors Rechnung getragen.[190]

Es wurden alle Rotamere verworfen, die die mit diesen Faktoren skalierten van-der-Waals-Radien verletzten. Für jedes Rotamer wurde die Energie berechnet und in der Ausgabematrix abgespeichert. Der Wert für *Max Energy Difference* wurde mit 9999 kcal/mol so groß gewählt, dass hierdurch kein Rotamer aussortiert wurde. Rotamere mit einer Energie deutlich größer als 10 kcal/mol waren trotzdem nicht in den Ausgabematrizen zu finden, da diese bereits vorher aufgrund der Verletzung der van-der-Waals-Radien aussortiert wurden (vgl. Formel 4 und Abbildung 15). Die maximale Anzahl an Rotameren pro RBR ergibt sich nach Formel 3 aus der Anzahl der zu rotierenden Bindungen und dem Winkelinkrement. Nur die Rotamere, bei denen es nicht zu einer Verletzung der van-der-Waals-Radien kam wurden in der csv-Datei des RBR abgelegt. Zusätzlich beschränkt war die Anzahl der gespeicherten Rotamere durch eine interne Beschränkung von SYBYL, so dass maximal die jeweils 500000 energiegünstigsten Rotamere in der csv-Datei gespeichert wurden.

In die Abstandsmatrix (*distance map*), also die Liste der für jedes Rotamer zu messenden Entfernungen, wurden alle Entfernungen zwischen Atomen des Sets ANCHORS_A zu Atomen des Sets ANCHORS_B aufgenommen. Nur bei RBR, die auf beiden Seiten das gleiche Ringsystem trugen (z.B. CR1XL3XCR1), wurden redundante Entfernungen aussortiert (z.B. "Ring1-H-5(para) nach Ring2-H-1(ortho)", da ja schon "Ring1-H-1(ortho) nach Ring2-H-5(para)" bestimmt wurde).

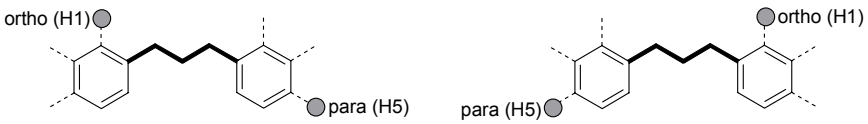
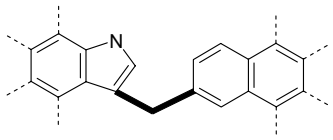


Abbildung 37: Bei gleichen Ringsystemen auf beiden Seiten der Brücke gab es redundante Entfernungen, die nicht mit in die Abstandsmatrix aufgenommen wurden. Dies ist hier gezeigt an den beiden ortho-para-Paarungen am RBR CR1XL3XCR1.

Die 225 vorbereiteten Suchen wurden mit der Option "HOLD_FOR_LATER" abgespeichert und standen so für das Skript 7.3.4 startsearches_all.spl zur Verfügung. In diesem Skript wurden dann die vorbereiteten Suchen mit einem Verzögerungsintervall von je einer Minute nacheinander gestartet, denn sofortiges und unverzögertes Starten hatten in Vorversuchen immer wieder aufgrund von Speicherproblemen zum Komplettausfall des Systems geführt.
Die Gesamtheit der Operationen "Verknüpfen von Bausteinen zu RBR", "Minimierung der RBR", "Vorbereitung der systematischen Konformationssuche" und die Suchen selbst, wurden mit einem Skript (7.3.5 go_all.spl) gestartet.
Die Skriptlaufzeit auf einer Silicon Graphics Octane2 betrug 22 Minuten. Nach Ende des Skriptes dauerte es noch knapp vier Stunden, bis das System die letzten Suchen bearbeitet hatte.
Die so erzeugten Ergebnisse lagen in einer SYBYL-internen Form vor, die sich nur über den SYBYL-Befehl "*analyze systematic search*" öffnen und betrachten lies. Um die Daten in ein Format zu bringen, das eine Bearbeitung und Suche auch ohne Verwendung von SYBYL erlaubt, wurde das SPL-Skript 7.3.6 export_dirs_csv.spl verwendet. Das Öffnen der Ergebnisse des systematischen Suchen in SYBYL sowie der Export der Tabellen in das csv (*comma separated values*)-Format war überaus zeitaufwendig. Um die Rechnung zu beschleunigen, wurden die Ergebnisse der systematischen Suchen auf mehrere unterschiedlich leistungsstarke (Octane1, Octane2) sgi-Workstations verteilt, die dann parallel am Öffnen und Exportieren der Daten arbeiteten. Zur Erzeugung der 225 csv-Dateien wurden insgesamt 245922 Minuten Rechenzeit auf den Rechnern benötigt (entsprechend 171 Tagen).
Es wurden 225 Tabellen (entsprechend 225 RBR) mit insgesamt 28.2 Millionen Zeilen (entsprechen 28.2 Millionen Rotameren) erzeugt. In den Feldern der Abstandsmatrix wurden dabei insgesamt 417.7 Millionen Einzelentfernungen vermessen. Die Gesamtheit dieser 225 Tabellen stellt die Datenbank DB2 dar. Die im Rahmen der Konformationssuche erzeugten Rotamere wurden zwar von SYBYL generiert und vermessen, jedoch im Rahmen der Konformationssuche nicht als *in silico*-Moleküle (mol2-Dateien) abgespeichert. Die Informationen über die erzeugten Rotamere, ihre Energien und die Abstandsmatrizen der

Ankerpaare wurde in den csv-Dateien gespeichert. Im späteren Verlauf der Arbeit wurden dann gezielt die Rotamere erzeugt und gespeichert, die bestimmten Anforderungen entsprachen.



CR3XL1XCR5

Abbildung 38: Das RBR CR3XL1XCR5, an dessen Beispiel im Folgenden die Durchführung der systematischen Konformationssuchen erklärt wird.

Tabelle 1: Auszug aus der csv-Datei der systematischen Konformationssuche des RBR CR3XL1XCR5. Jede Zeile beschreibt ein Rotamer, das sich von dem vorhergehenden Rotamer in einem der Rotationswinkel um 10 Grad unterscheidet. In der dritten Spalte ist die intramolekulare Energie dieses Rotamers aufgeführt, in den folgenden Spalten findet sich die Distanzen zwischen allen als Anker definerten Atomen der beiden Ringsysteme des RBR

Torsions-winkel 1	Torsions-winkel 2	Energie	"(12)CR3.H4--> (31)CR5.H8"	"(12)CR3.H4--> (32)CR5.H7"	…	"(15)CR3.H7--> (34)CR5.H5"
330	310	19.24	4.985	6.095	…	6.775
330	300	19.35	4.895	6.175	…	7.195
…	…	…	…	…	…	…
30	60	19.74	4.785	6.045	…	7.175
30	50	19.6	4.875	5.975	…	6.755

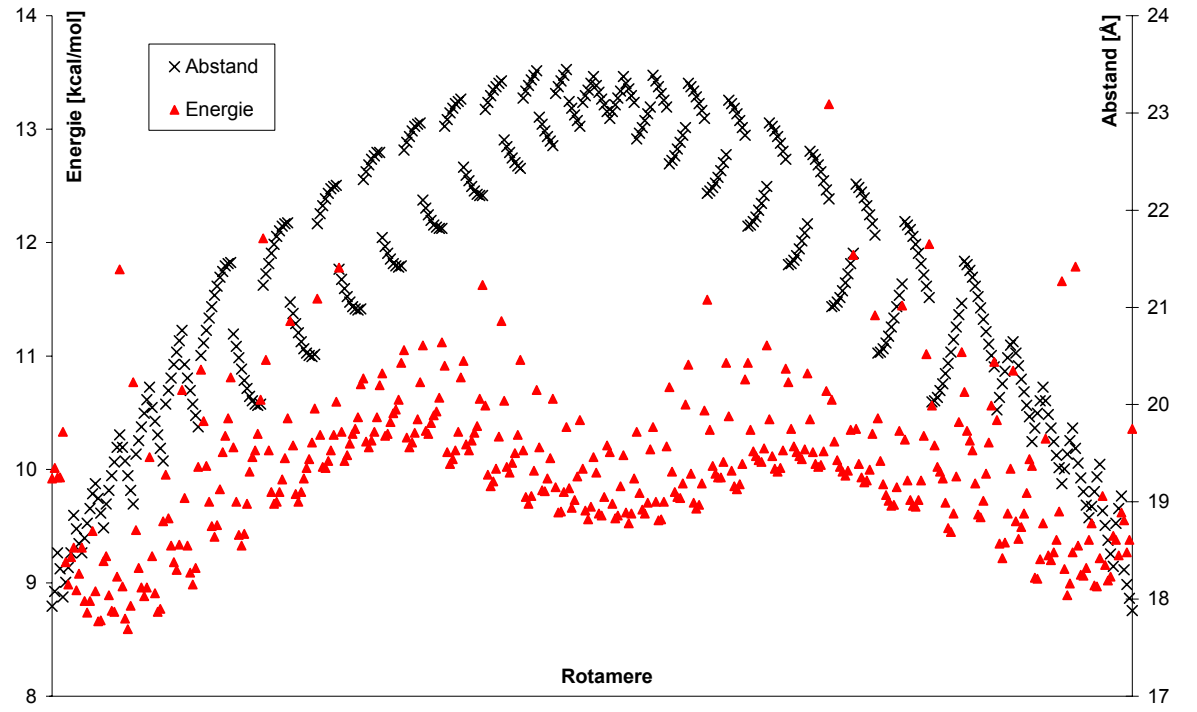


Abbildung 39: Zur Veranschaulichung der primären Ergebnisse der systematischen Konformationssuchen ist in hier für das Ankerpaar 10 des RBR CR3XL1XCR5 für jedes Rotamer aufgetragen, welche Distanz die beiden Atome mit den *atom IDs* "14" und "32" aufspannen (schwarzes x) und welche Energie das Rotamer hat (rotes Dreieck).

3.1.5 Aufbau der Datenbank DB3

Um die Daten aus den Konformationssuchen in eine Form zu bringen, in der eine Suchanfrage (zum Beispiel "Welche Anker an welchen Bausteinen sind geeignet, energiegünstig eine Entfernung von elf Ångström zu überbrücken?") effizient beantwortet werden kann, wurden zunächst verschiedene organisierende Dateien erzeugt:

Mit Hilfe von 7.3.7 csv_2_energy_all.sh wurde für jedes RBR eine energy-Datei erzeugt. Durch das Skript wurde hierzu die entsprechende csv-Datei nach dem Minimum und dem Maximum der Energiewerte durchsucht. Diese beiden Werte wurden dann in die neu erzeugte energy-Datei geschrieben.

Tabelle 2: Abbildung der energy-Datei der systematischen Konformationssuche des RBR CR3XL1XCR5. Die erste Zahl ist die Energie des günstigsten, die zweite die des ungünstigsten Rotamers dieses RBR.

17.69
23.09

Des Weiteren wurde mit dem SPL-Skript 7.3.8 writelogs_all.spl zu jedem RBR eine map-Datei erzeugt, in der für jede Spalte der Abstandsmatrix vermerkt war, welchen Abstand innerhalb des RBR sie enthielt. Das Skript gleicht in vielen Schritten dem Skript setupsearch_all.spl. Es bereitet aber keine systematische Suche vor, sondern speichert lediglich die Liste der zu vermessenden Entfernungen in einer map-Datei ab.

Tabelle 3: Abbildung der map-Datei des RBR CR3XL1XCR5. In der ersten Zeile ist die Anzahl der Ankerpaare abgelegt (16). In den 16 folgenden Zeilen steht jeweils zuerst die laufende Nummer eines Ankerpaars und in der zweiten Spalte eine Beschreibung, von welchem Anker-Atom (*atom ID*, Ringsystem.Atom-Name) des ersten Ringsystems zu welchem Anker-Atom des zweiten Ringsystems die Entfernung gemessen werden sollte.

16	
001	12 CR3.H4 --> 31 CR5.H8
002	12 CR3.H4 --> 32 CR5.H7
003	12 CR3.H4 --> 33 CR5.H6
004	12 CR3.H4 --> 34 CR5.H5
005	13 CR3.H5 --> 31 CR5.H8
006	13 CR3.H5 --> 32 CR5.H7
007	13 CR3.H5 --> 33 CR5.H6
008	13 CR3.H5 --> 34 CR5.H5
009	14 CR3.H6 --> 31 CR5.H8
010	14 CR3.H6 --> 32 CR5.H7
011	14 CR3.H6 --> 33 CR5.H6
012	14 CR3.H6 --> 34 CR5.H5
013	15 CR3.H7 --> 31 CR5.H8
014	15 CR3.H7 --> 32 CR5.H7
015	15 CR3.H7 --> 33 CR5.H6
016	15 CR3.H7 --> 34 CR5.H5

Die eigentliche Datenbank DB3 wurde (nach vergeblichen Vorversuchen mit den Programmen MATHEMATICA[197] und MAPLE[198] mit Hilfe eines in der Programmiersprache C[1] geschriebenen Programms aufgebaut. Dieses Programm 7.3.9 csv_2_distdb.c, das Teile der Bibliothek bkutil (Anhang 7.3.10) verwendet, bearbeitet nacheinander alle 225 RBR. Zuerst liest das Programm die energy-Datei und die map-Datei des RBR ein, dann zeilenweise die entsprechende csv-Datei. In seiner einfachsten Funktion bestimmt das Programm dann den maximalen und den minimalen Abstand jeder Anker-Paarung jedes RBR. Für jedes RBR speichert das Programm diese Werte, zusammen mit der Beschreibung der verwendeten Anker in einer db-Datei ab.

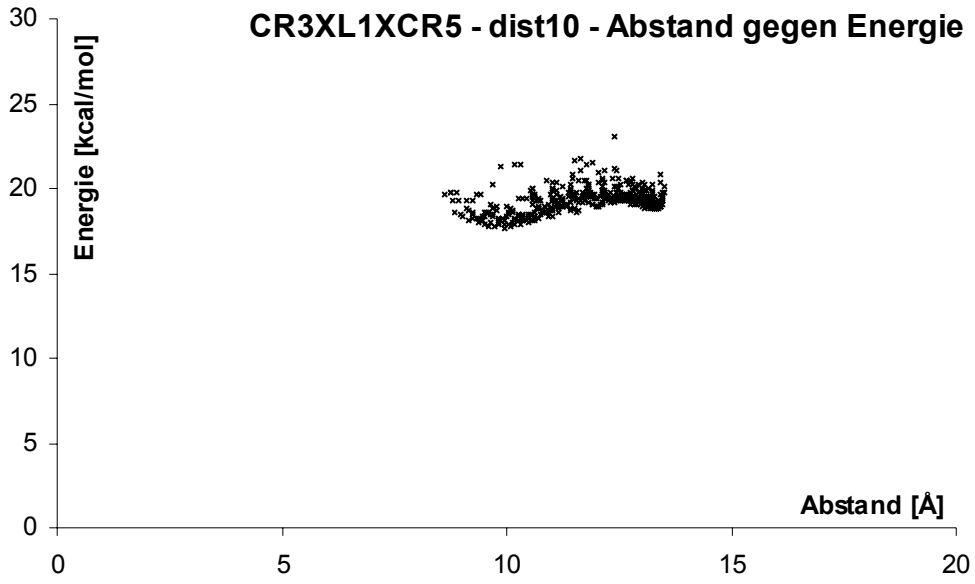


Abbildung 40: Auftragung der Energie der Rotamere der RBR CR3XL1XCR5 gegen den Abstand, den sie zwischen den Atomen mit den *atom IDs* "14" und "32" aufspannen (vgl. Tabelle 3).

Tabelle 4: db-Datei der systematischen Konformationssuche der RBR CR3XL1XCR5. Unter Berücksichtigung aller Rotamere dieses RBR wurde für die 16 in der ersten und vierten Spalte beschriebenen Ankerpaare (vgl. Tabelle 3) aufgelistet, welchen Abstand sie minimal (d_{min}) bzw. maximal (d_{max}) überbrücken können. Eine graphische Auftragung der Werte für die in Zeile 10 angegebene Ankerpaarung befindet sich in Abbildung 40.

Nr.	d_{min}	d_{max}	Beschreibung der Anker
1	4.785	9.575	12 CR3.H4 --> 31 CR5.H8
2	5.935	10.845	12 CR3.H4 --> 32 CR5.H7
3	5.625	10.355	12 CR3.H4 --> 33 CR5.H6
4	3.565	8.325	12 CR3.H4 --> 34 CR5.H5
5	5.785	11.795	13 CR3.H5 --> 31 CR5.H8
6	6.625	13.055	13 CR3.H5 --> 32 CR5.H7
7	6.135	12.635	13 CR3.H5 --> 33 CR5.H6
8	4.945	10.605	13 CR3.H5 --> 34 CR5.H5
9	7.785	12.315	14 CR3.H6 --> 31 CR5.H8
10	8.635	13.525	14 CR3.H6 --> 32 CR5.H7
11	7.375	13.445	14 CR3.H6 --> 33 CR5.H6
12	5.835	11.545	14 CR3.H6 --> 34 CR5.H5
13	8.875	10.885	15 CR3.H7 --> 31 CR5.H8
14	9.855	11.995	15 CR3.H7 --> 32 CR5.H7
15	8.225	12.205	15 CR3.H7 --> 33 CR5.H6
16	6.135	10.525	15 CR3.H7 --> 34 CR5.H5

Die db-Dateien für alle 225 RBR wurden mit Hilfe des Shell-Skriptes 7.3.11 multidb_2_database.sh zu einer großen Tabelle, der Datenbank DB3 zusammengefügt.

Für jedes der 3515 Paare von Ankeratomen, dessen Abstände während der systematischen Konformationssuche (3.1.4) vermessen wurden, waren in dieser Tabelle der minimal und der maximal überbrückte Abstand aufgelistet. Außerdem enthielt die Tabelle Informationen über das verwendete RBR, die minimalen und maximalen Energiewerte seiner Rotamere, sowie die *atom IDs* der entsprechenden Anker-Atome.

Tabelle 5: Auszug aus der Datenbank DB3. In Spalte 1 ist der Name des RBR angegeben, von dem die jeweilige Ankerpaarung stammt. In den Spalten E_{min} und E_{max} sind die jeweiligen Energiegrenzen dieses RBR angegeben (vgl. Tabelle 2). Die vierte Spalte enthält eine laufende Nummer. Die restlichen Spalten wurden aus den jeweiligen db-Dateien übernommen und sind in der Überschrift zu Tabelle 4 erläutert.

	E_{min}	E_{max}	lfd.Nr.	lfd.Nr.	d_{min}	d_{max}	Anker-Atome
CR0XL0XCR0	22.43	31.53	1	1	2.085	4.815	7 CR0.H3 --> 21 CR0.H3
CR0XL0XCR0	22.43	31.53	2	2	2.225	4.895	7 CR0.H3 --> 22 CR0.H4
CR0XL0XCR0	22.43	31.53	3	3	4.505	6.175	7 CR0.H3 --> 23 CR0.H5
CR0XL0XCR0	22.43	31.53	4	4	4.495	5.525	7 CR0.H3 --> 24 CR0.H6
CR0XL0XCR0	22.43	31.53	5	5	2.375	4.965	8 CR0.H4 --> 22 CR0.H4
CR0XL0XCR0	22.43	31.53	6	6	4.645	6.285	8 CR0.H4 --> 23 CR0.H5
CR0XL0XCR0	22.43	31.53	7	7	4.635	5.645	8 CR0.H4 --> 24 CR0.H6
CR0XL0XCR0	22.43	31.53	8	8	6.915	8.045	9 CR0.H5 --> 23 CR0.H5
CR0XL0XCR0	22.43	31.53	9	9	6.855	7.535	9 CR0.H5 --> 24 CR0.H6
CR0XL0XCR0	22.43	31.53	10	10	6.685	7.105	10 CR0.H6 --> 24 CR0.H6
CR0XL0XCR1	12.53	15.54	11	1	2.195	4.735	7 CR0.H3 --> 21 CR1.H1
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
CR8XL4XCR8	39.20	74.32	3515	3	10.255	23.685	21 CR8.H4 --> 74 CR8.H4

3.1.6 Energieschnitte

Basis für den Aufbau der Datenbank DB3 in Abschnitt 3.1.5 war die gesamte DB2, also die kompletten Ausgaben der systematischen Konformationssuchen in Abschnitt 3.1.4. Somit waren nur die Rotamere aussortiert worden, bei denen es zu starken Verletzungen der van-der-Waals-Radien gekommen war (vgl. Abschnitt 1.8 und 3.1.4).

In Abbildung 40 ist zu sehen, dass die dort abgebildeten "Abstand gegen Energie"-Punkte einen weiten Bereich abdecken. Interessant sind aber vor allem die Punkte mit geringer Energie. Da die Abstandsbereiche der verschiedenen Ankerpaare weit überlappen, wurde entschieden, von jedem Ankerpaar nur den unteren, energiegünstigen Teil der "Abstand gegen Energie"-Punktmenge zu verwenden. Für jedes RBR sollten nur die günstigsten Rotamere zur Bestimmung der "Abstandsgrenzen" berücksichtigt werden. Bei welcher Energie der Schnitt

zwischen den Rotameren, die noch berücksichtig werden sollten, und denen, die als zu ungünstig verworfen werden sollten, anzusetzen war, sollte im Folgenden ermittelt werden.
Das in Abschnitt 3.1.5 bereits erwähnte Programm csv_2_distdb.c wurde hierfür mit der Funktion ausgestattet, nur die Zeilen der csv-Datei (Rotamere) für die Bestimmung des maximalen und des minimalen Abstandes der Anker-Paarungen zu berücksichtigen, deren Energie innerhalb eines definierten Intervalls über der minimalen Energie dieses RBR lag.

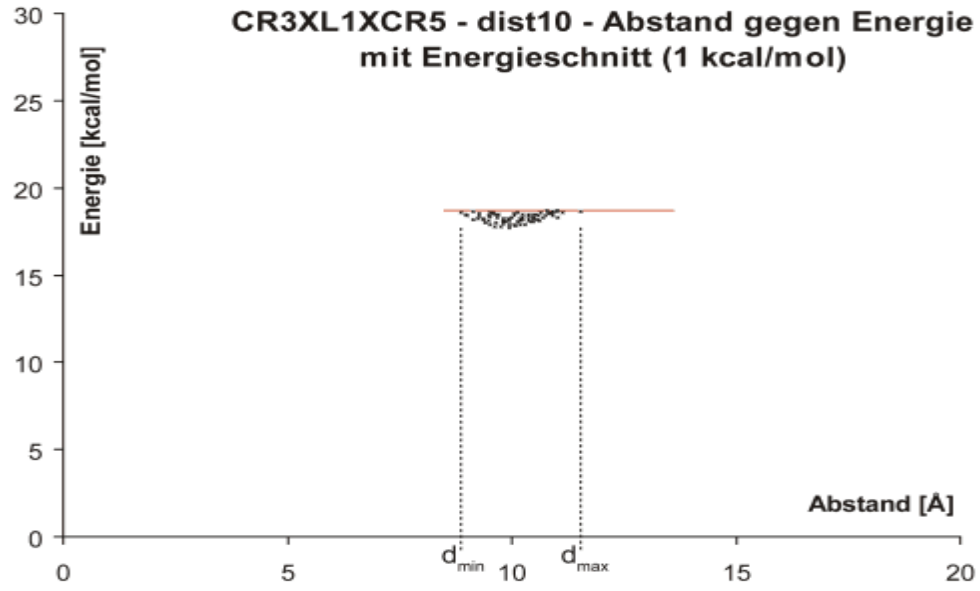


Abbildung 41: Auftragung der Energie der Rotamere der RBR CR3XL1XCR5 gegen den Abstand, den sie zwischen den Atomen mit den *atom IDs* "14" und "32" aufspannen (vgl. Tabelle 3). Im Unterschied zu Abbildung 40 sind hier nur noch die Rotamere berücksichtigt, die sich unterhalb des Energieschnittes (hier als rote Linie dargestellt) bei 18.69 kcal/mol (E_{min} + 1 kcal/mol) befinden. Hierdurch liegen die Abstandsgrenzen d_{min} und d_{max}, die dieses Ankerpaar noch überbrücken kann näher beieinander.

Das Energielimit x wird in absoluten kcal/mol angegeben, und auf die für dieses RBR festgestellte minimale Energie (E_{min}) addiert.

$$newE_{max} = E_{min} + x$$

Formel 5: Formel für den Energieschnitt. Nur Rotamere, deren Energie kleiner oder gleich newE_{max} ist, werden weiter berücksichtigt.

Eine weitere Option, bei der das Energielimit prozentual angegeben werden kann (Formel 6), wurde zwar programmiert und getestet, fand aber keine Anwendung. Da hier die Energie des Rotamers mit der höchsten Energie (E_{max}) in die Formel einging, wurde die Selektion stark

durch einzelne hochenergetische Konformere beeinflusst. Da es mit dieser Methode nicht möglich war, reproduzierbar nur energiegünstige Rotamere zu selektieren, wurde sie nicht weiter verwendet.

$$newE_{\max} = E_{\min} + \left(\frac{E_{\max} - E_{\min}}{100} \right) * x$$

Formel 6: Formel für die Verwendung eines prozentualen Energieschnittes. Unter Verwendung dieser Formel für den Energieschnitt wurde keine reproduzierbare Selektion von Rotameren geringer Energie erzielt. Die Formel wurde nicht weiter verwendet. Der Energieschnitt wurde nach Formel 5 durchgeführt.

Die csv-Dateien der RBR enthielten zwischen drei und 500000 Rotamere (vgl. Abschnitt 3.1.4). Da der gewählte Energieschnitt sowohl für RBR mit vielen Rotameren, als auch für RBR mit wenigen Rotameren gute Ergebnisse bringen sollte, wurden exemplarisch sechs RBR ausgewählt, die gut die Gesamtheit aller RBR repräsentieren (Abbildung 42).

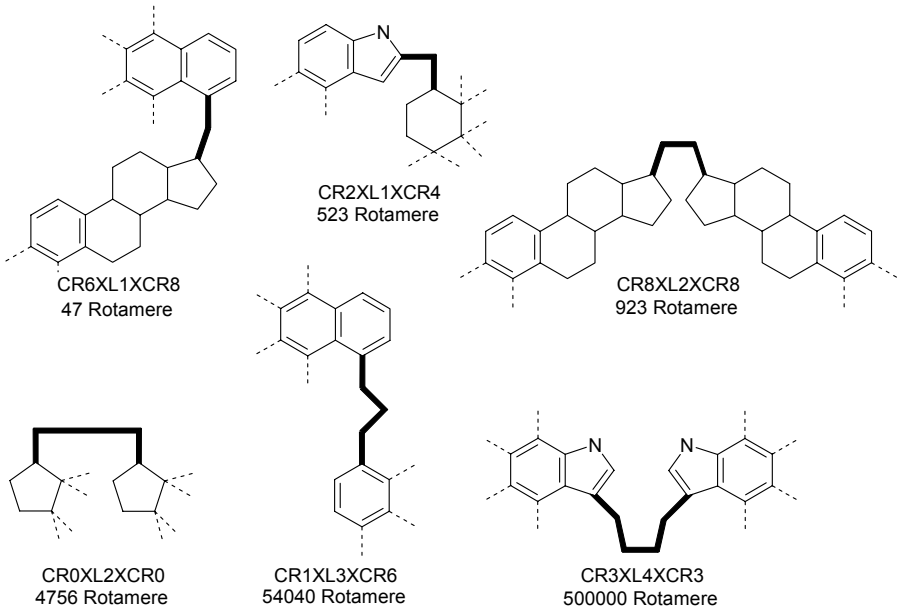


Abbildung 42: Die sechs RBR, die zur Bestimmung des günstigsten Energieschnittes ausgewählt wurden. Die Brücken sind als fett gedruckte Bindungen, die Anker als gestrichelte Bindungen dargestellt.

Jeweils für das erste Ankerpaar jedes dieser RBR wurde das Programm csv_2_distdb.c mit acht verschiedenen Energielimits (0.25, 0.5, 0.75, 1, 2, 3, 5, 10 kcal/mol) gestartet. In Tabelle 6 ist jeweils der minimale und der maximale überspannte Abstand, sowie die

Differenz [E_{max} - E_{min}] (in der Tabelle als *range* bezeichnet) aufgelistet. In Abbildung 43 ist die *range* gegen das als Parameter übergebene Energielimit aufgetragen.

RBR	Energie schnitt	distance #1		
		range	d_{min}	d_{max}
CR6XL1XCR8	0.25	0.000	11.335	11.335
47 Rotamere	0.5	0.000	11.335	11.335
	0.75	0.000	11.335	11.335
	1	1.130	10.775	11.905
	2	5.400	10.775	16.175
	3	5.400	10.775	16.175
	5	5.590	10.775	16.365
	10	5.770	10.775	16.545
CR2XL1XCR4	0.25	3.050	5.325	8.375
523 Rotamere	0.5	3.230	5.145	8.375
	0.75	3.260	5.115	8.375
	1	3.410	4.965	8.375
	2	3.640	4.735	8.375
	3	3.640	4.735	8.375
	5	3.710	4.665	8.375
	10	3.710	4.665	8.375
CR8XL2XCR8	0.25	0.340	22.725	23.065
923 Rotamere	0.5	0.520	22.545	23.065
	0.75	0.520	22.545	23.065
	1	0.660	22.455	23.115
	2	0.990	22.125	23.115
	3	2.580	20.545	23.125
	5	4.280	18.855	23.135
	10	7.900	15.235	23.135

RBR	Energie schnitt	distance #1		
		range	d_{min}	d_{max}
CR0XL2XCR0	0.25	0.110	5.725	5.835
4756 Rotamere	0.5	2.440	4.305	6.745
	0.75	2.900	3.995	6.895
	1	2.920	3.995	6.915
	2	3.280	3.685	6.965
	3	3.460	3.525	6.985
	5	4.160	2.825	6.985
	10	4.640	2.345	6.985
CR1XL3XCR6	0.25	1.970	5.225	7.195
54040 Rotamere	0.5	3.040	4.935	7.975
	0.75	3.040	4.935	7.975
	1	3.190	4.935	8.125
	2	4.720	4.935	9.655
	3	5.530	4.615	10.145
	5	6.320	4.615	10.935
	10	6.860	4.125	10.985
CR3XL4XCR3	0.25	0.610	3.605	4.215
500000 Rotamere	0.5	0.670	3.545	4.215
	0.75	1.420	3.125	4.545
	1	1.460	3.115	4.575
	2	3.030	2.705	5.735
	3	4.790	2.465	7.255
	5	6.180	2.215	8.395
	10	7.030	2.055	9.085

Tabelle 6: Analyse der systematischen Konformationssuchen der in Abbildung 42 dargestellten RBR. Für jedes RBR wurden Energieschnitte bei 0.25, 0.5, 0.75, 1, 2, 3, 5 und 10 kcal/mol durchgeführt und für jeden dieser Energieschnitte analysiert, welchen Einfluss er auf den minimalen (d_{min}) bzw. maximalen (d_{max}) Abstand, den ein Ankerpaar an dem jeweiligen RBR überspannen kann, hat. In der Spalte "*range*" ist jeweils die Differenz d_{max} - d_{min} aufgeführt. Eine graphische Auswertung dieser Daten folgt in Abbildung 43.

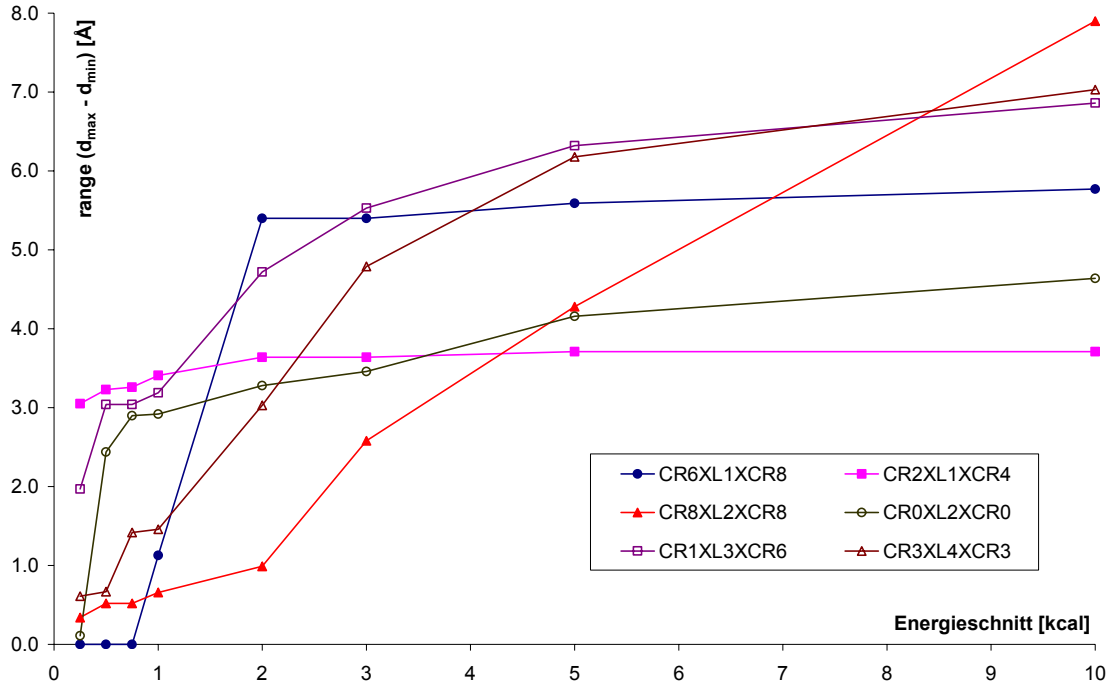


Abbildung 43: Auftragung des Energieschnittes, bis zu dem Rotamere der RBR in Abbildung 42 für die Bestimmung der Abstandsgrenzen d_{min} und d_{max} (vgl. Tabelle 6) berücksichtigt wurden, gegen die Differenz zwischen dem maximalen und dem minimalen überspannten Abstand des ersten Ankerpaars des RBR (hier als "*range*" bezeichnet). Je höher der Wert für "*range*", desto breiter der Bereich, für den diese Ankerpaarung das RBR eingesetzt werden kann. Je höher der Energieschnitt, desto mehr energetisch ungünstige Rotamere werden berücksichtigt.

Zur Auswahl eines geeigneten Energieschnittes wurde nach einem Wert für x (vgl. Formel 5) gesucht, bei dem für alle betrachteten RBR repräsentative aber nicht zu viele Rotamere berücksichtigt wurden. Ein zu kleiner Wert für x bedingte, dass manche RBR nur durch ihr günstigstes Rotamer repräsentiert wurden - ein Vorgehen, das der Grobheit des Modells und der zu erwartenden Flexibilität der Liganden nicht gerecht wird. Ein zu hoch angesetzter Energieschnitt würde für eine Anfrage zu viele Antworten ergeben, insbesondere zu viele Antworten, die von energetisch vergleichsweise ungünstigen Rotameren stammen.

Die fünf zuvor gewählten RBR mit vielen Rotameren zeigen im Bereich knapp unter 1 kcal/mol ein Plateau in ihrem Kurvenverlauf. Nur die Kurve des RBR CR6XL1XCR8, dessen csv-Datei nur 47 Rotamere enthält, zeigt solch einen Verlauf nicht. Sie befindet sich aber auch bei 1 kcal/mol weder bei einem Entfernungsbereich von 0 (also bei dem Zustand, in dem nur ein Rotamer verwendet wird), noch bei dem maximalen Abstandsbereich für dieses RBR, der bei 5.77 Å liegt. Daher wurde als Energieschnitt 1 kcal/mol gewählt.

Dieser Energieschnitt wurde auf alle 225 Rotamer-Dateien angewendet. Von den ursprünglich 28.2 Millionen Rotameren, deren Energie um bis zu 35.12 kcal/mol über der Energie des günstigsten Rotamers ihres RBR lag, wurden alle aussortiert, deren Energie höher als E_{min} + 1 kcal/mol war. Unterhalb des Energieschnittes lagen 18470 Rotamere, die die Datenbank DB4 bildeten (vgl. auch Abbildung 97). Aus den db-Dateien mit engeren Abstandsgrenzen wurde mit Hilfe des Shell-Skriptes 7.3.11 multidb_2_database.sh die Datenbank DB5 aufgebaut.

Tabelle 7: Auszug aus der Datenbank DB5. Die Spalten entsprechen denen in Tabelle 5. Die zusätzliche Spalte new_E_{max} enthält den jeweiligen Energieschnitt dieses RBR, also den Energiewert, bis zu dem Rotamere noch für die Bestimmung von d_{min} und d_{max} berücksichtigt wurden. Die Spalten d_{min} und d_{max} enthalten hier nun deutlich sichtbar näher beieinander liegende Werte als in Tabelle 5, die ohne Energieschnitt erzeugt wurde.

	E_{min}	E_{max}	new_E_{max}	lfd.Nr.	lfd.Nr.	d_{min}	d_{max}	Anker-Atome
CR0XL0XCR0	22.43	31.53	23.43	1	1	3.915	4.465	7 CR0.H3 --> 21 CR0.H3
CR0XL0XCR0	22.43	31.53	23.43	2	2	2.915	4.895	7 CR0.H3 --> 22 CR0.H4
CR0XL0XCR0	22.43	31.53	23.43	3	3	5.445	5.975	7 CR0.H3 --> 23 CR0.H5
CR0XL0XCR0	22.43	31.53	23.43	4	4	4.585	5.525	7 CR0.H3 --> 24 CR0.H6
CR0XL0XCR0	22.43	31.53	23.43	5	5	2.375	4.765	8 CR0.H4 --> 22 CR0.H4
CR0XL0XCR0	22.43	31.53	23.43	6	6	4.915	6.285	8 CR0.H4 --> 23 CR0.H5
CR0XL0XCR0	22.43	31.53	23.43	7	7	4.635	5.455	8 CR0.H4 --> 24 CR0.H6
CR0XL0XCR0	22.43	31.53	23.43	8	8	7.465	7.945	9 CR0.H5 --> 23 CR0.H5
CR0XL0XCR0	22.43	31.53	23.43	9	9	6.895	7.535	9 CR0.H5 --> 24 CR0.H6
CR0XL0XCR0	22.43	31.53	23.43	10	10	6.695	6.955	10 CR0.H6 --> 24 CR0.H6
CR0XL0XCR1	12.53	15.54	13.53	11	1	2.195	4.675	7 CR0.H3 --> 21 CR1.H1
…	…	…		…	…			…
…	…	…		…	…			…
…	…	…		…	…			…
CR8XL4XCR8	39.20	74.32	40.2	3515	3	23.505	23.685	21 CR8.H4 --> 74 CR8.H4

Um die Anzahl an *hits* für eine gegebene Distanz vorherzusagen, wurde das Programm 7.3.12 dist_2_count.sh verwendet, das zusätzlich zur gesuchten Entfernung die Angabe einer maximalen Abweichung erlaubt. Diese maximale Abweichung wurde auf 0.2 Ångström gesetzt und berücksichtigt,

- dass in den meisten Fällen kein Rotamer exakt die gesuchte Entfernung widerspiegelt,
- dass die RBR flexibel sind und sehr viel mehr Konformationen einnehmen könne, als durch die Abtastung in 10 Grad-Schritten generiert wurden und
- dass sich Liganden den sterischen Ansprüchen der Bindungstasche anzupassen vermögen (*induced fit* des Liganden).

Bei einem Suche mit 0.2 Ångström maximaler Abweichung nach einem Rotamer, das eine Entfernung von 6 Ångström überspannt werden alle Bausteine als *hits* ausgegeben, für die "6 Ångström" innerhalb des Intervalls d_{min} - 0.2 bis d_{max} + 0.2 Ångström liegt (vgl. Tabelle 7). Von den ersten elf in Tabelle 7 gezeigten Datenbankeinträgen werden demnach lediglich der dritte und der sechste als *hits* zurückgegeben, denn nur sie können energiegünstig eine Entfernung von 5.8 Ångström bis 6.2 Ångström überbrücken. Das Skript dist_2_count.sh führt eine Suche mit anschließender Zählung durch und gibt nur das Ergebnis der Zählung aus. Diese Daten wurden genutzt, um die das Profil der Datenbank DB5 in Abbildung 44 zu erzeugen.

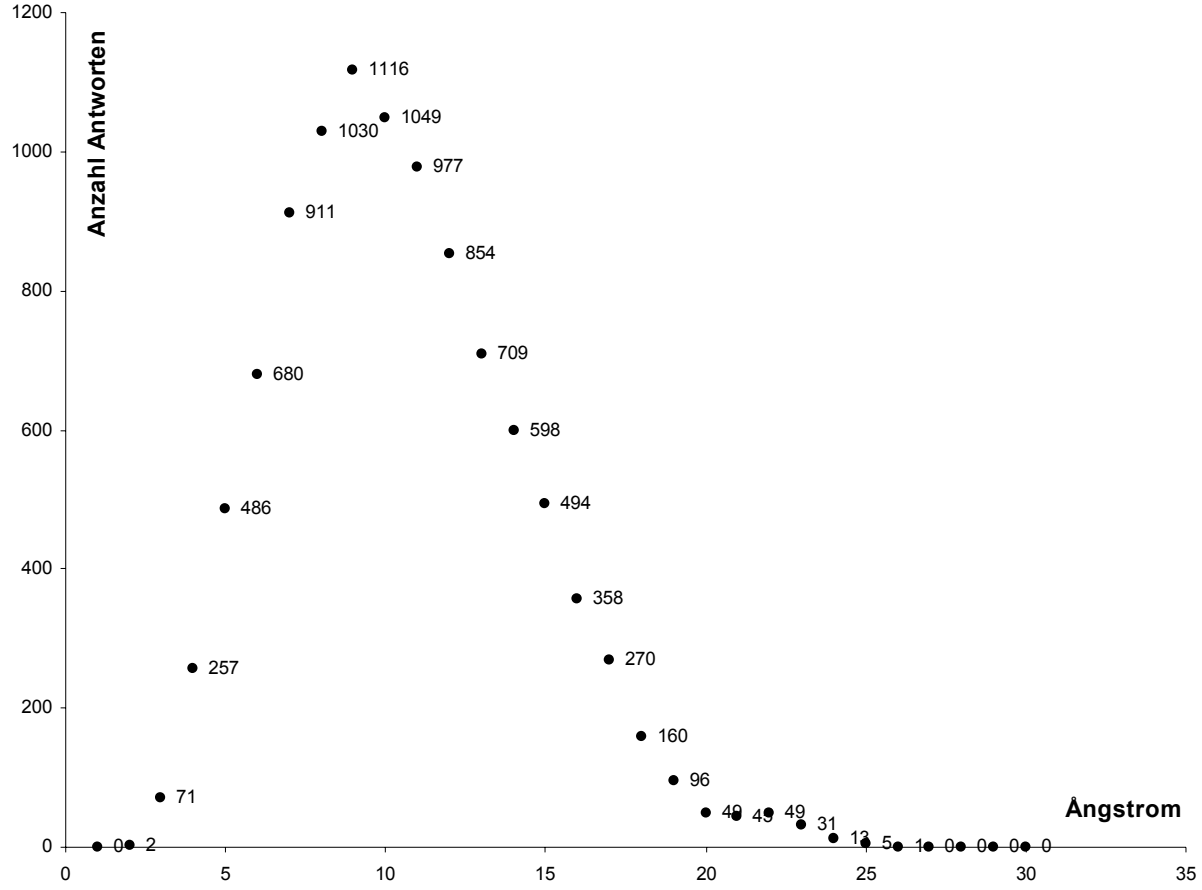


Abbildung 44: Auftragung der Anzahl an *hits*, die die Datenbank DB5 für Pharmakophor-Distanzen von 1 bis 30 Ångström liefert. Die meisten *hits* (1116) werden erhalten bei einer Anfrage mit einer Distanz von 9 Ångström (ermittelt mit dem Programm dist_2_count.sh und einer "*allowed deviation*" von 0.2 Å).

Die meisten *hits* gab die Datenbank bei einer gesuchten "*distance*" von 9 Ångström zurück. Im Bereich von 6 bis 14 Ångström gab die Datenbank für jede Anfrage über 500 *hits* zurück. Typische Entfernungen innerhalb von Pharmakophoren liegen zwischen vier und

20 Ångström.[199-201] Für diesen Bereich ist die Datenbank hervorragend geeignet. Einen Überblick über die in der Datenbank vorhandenen Bausteine und die von ihnen überspannten Entfernungen gibt Abbildung 45.

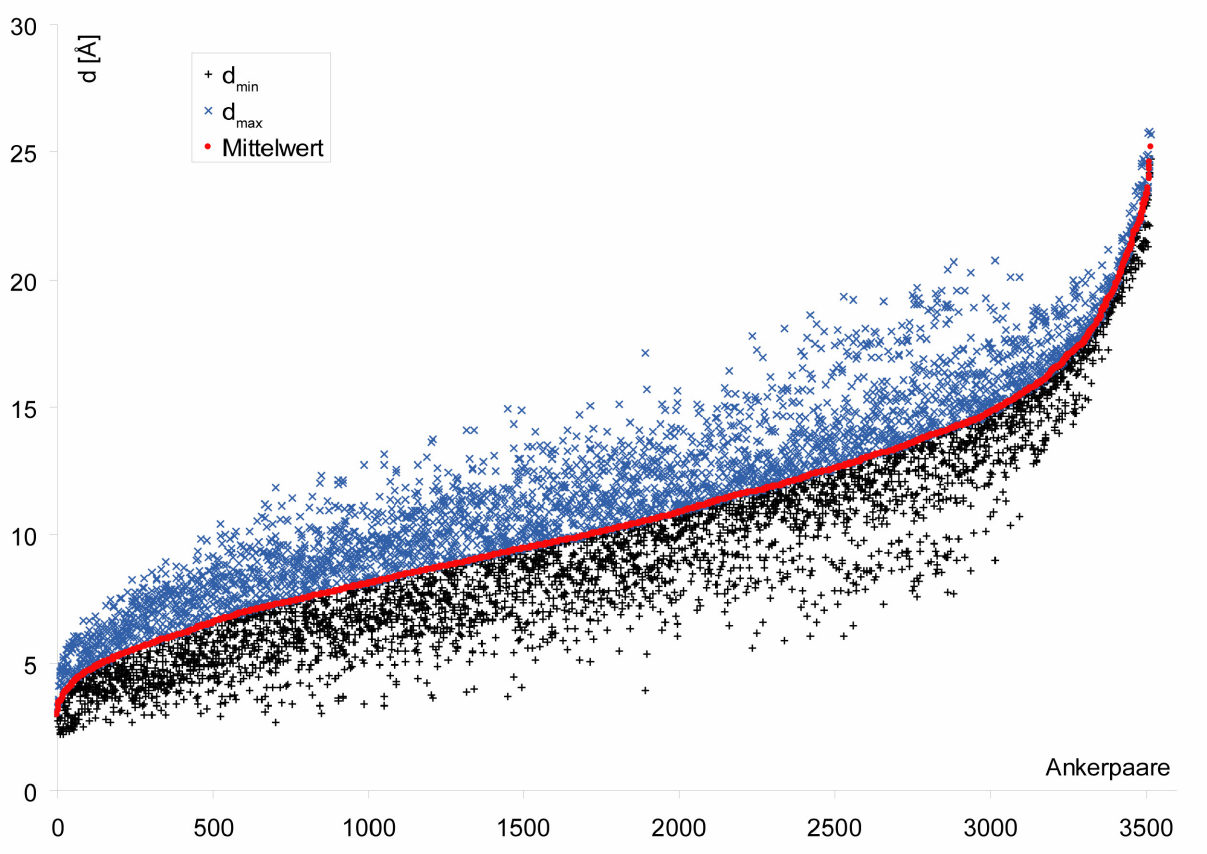


Abbildung 45: Auftragung des minimal überspannten Abstandes (d_{min}), des maximal überspannten Abstandes (d_{max}) und des Mittelwertes der beiden für alle 3515 Ankerpaare. Auf der y-Achse sind die Abstände und auf der x-Achse die einzelnen Ankerpaare aufgetragen. Zur Erhöhung der Übersichtlichkeit sind die Ankerpaare hier nach dem Mittelwert der überspannten Abstände sortiert.

3.1.7 Interface-Skripte für die Datenbank DB5

Um später auf diese Datenbank zugreifen zu können, wurden zwei Skripte geschrieben (7.3.13 und 7.3.14).

Das erste ist das Shell-Skript 7.3.13 dist_2_rotamerlist.sh. Es erlaubt (wie dist_2_count.sh) die Angabe der gesuchten Entfernung und der maximalen Abweichung. Dann wird die csv-Datei des entsprechenden RBR durchsucht. Von allen Rotameren, die innerhalb des Energielimits liegen, wird das ausgewählt, bei dem die Entfernung zwischen den Ankeratomen der gesuchten Entfernung entspricht. Existiert ein solches Rotamer nicht, so wird das ausgewählt, bei dem die Differenz der Entfernung zur gesuchten Entfernung

möglichst gering ist. Für dieses Rotamer wird die Zeile aus der csv-Datei, erweitert um den Namen des RBR und die Anzahl der zu modifizierenden Torsionswinkel ausgegeben. Diese Ausgabe dient dann als Datenbasis für das SPL-Skript 7.3.14 rotamerlist_2_manymol2s.spl. Dieses wird in SYBYL gestartet. Für jede Zeile der Ausgabe wird aus der SYBYL-Datenbank das entsprechende RBR geladen. Dann werden die Torsionswinkel der jeweiligen Brückenbindung auf die aus der csv-Datei ausgelesenen Werte gesetzt. Für das Rotamer, dessen Torsionswinkel den Angaben aus der Ausgabe von dist_2_rotamerlist.sh entsprechen wird anschließend der Abstand zwischen den beiden Ankern gemessen. Das Molekül wird dann als mol2-Datei abgespeichert, wobei die Herkunft der Datei dadurch nachvollziehbar wird, dass der Dateiname aus dem Namen des RBR, der Entfernung zwischen den beiden Ankeratomen und den *atom IDs* der Anker-Atome zusammengesetzt wird.
Jede Zeile der Ausgabe entspricht einem *hit*, den dist_2_rotamerlist.sh in der Datenbank DB5 gefunden hat. Für jeden *hit* erzeugt rotamerlist_2_manymol2s.spl ein Molekül *in silico*.

3.2 Rekonstruktion des GP120-CD4-Pharmakophors

Als erste pharmakologisch relevante Anwendung der Datenbank DB5 wurde die Interaktion des humanen CD4 mit dem GP120 des HIV (vgl. Abschnitt 1.9.1) gewählt. Zu dieser Interaktion lagen sowohl NMR-spektroskopische als auch röntgenkristallographische Informationen vor. Die Informationen über ein diskontinuierliches Bindungsepitop wurden von Wülfken[34] 2001 mit Hilfe der STD-NMR-Spektroskopie ermittelt. Aus diesen Studien war bekannt (vgl. Seite 36), dass das Dekapeptid ^{425}NMWQKV^{430}-G-^{123}TPL^{125} aus dem GP120 an CD4 bindet. Im Besonderen das Indol-System des Trp (auf eine Nummerierung der Aminosäuren des Dekapeptides wird im Folgenden verzichtet, da sie eindeutig sind), die α- und ε-Protonen des Lys, sämtliche Protonen des Val, die β- und γ-Protonen des Thr, die β-Protonen des Pro und die Methylgruppen des Leu vermitteln die Bindung.

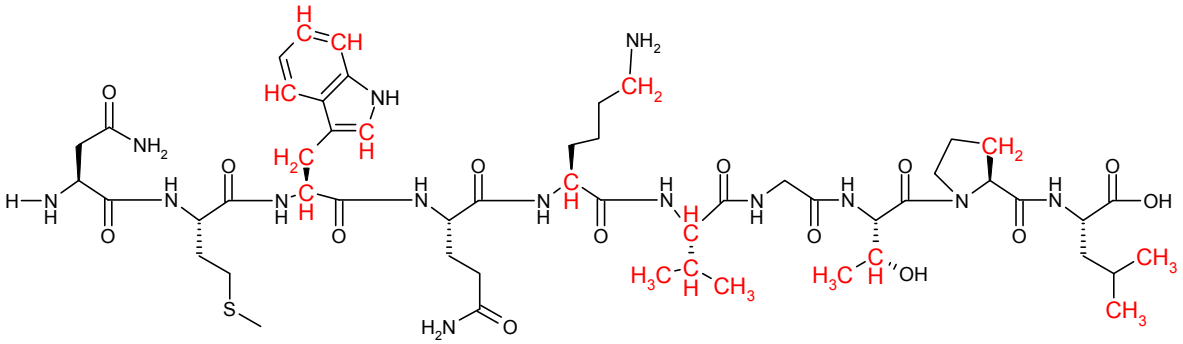


Abbildung 46: Das von Wülfken[34] mit Hilfe der STD-HSQC-Spektroskopie ermittelte Bindungsepitop des vom GP120 abgeleiteten Dekapeptides NMWQKVGTPL. Die in der Abbildung rot gekennzeichneten Atome zeigten einen starken STD-Effekt und sind somit wahrscheinlich direkt an der Bindung dieses Peptides an CD4 beteiligt.

Die Informationen über die dreidimensionale Anordnung dieser Gruppen im Raum kann theoretisch sowohl aus *transferred* NOESY-NMR-Studien, als auch aus röntgenkristallographischen Daten erhalten werden. Da für diese Wechselwirkung keine ausreichend hochaufgelöste *transferred* NOESY-NMR-Studie vorlag, wurden die 3D-Informationen mit Hilfe der Röntgenstruktur von Kwong *et al.*[188] ermittelt.

Anhand der Ergebnisse der STD-NMR-Untersuchungen, der Virusinhibitions-Assays und der Röntgenstrukturanalyse kann als gesichert angesehen werden, dass der Bindungsmodus des Dekapeptides NMWQKVGTPL dem der entsprechenden Aminosäuren des GP120 entspricht.[34;188]

3.2.1 Bearbeitung der Röntgenstruktur

Kwong *et al.* gelang es 1998 ein partiell deglykosyliertes und gekürztes GP120 mit zwei Domänen des CD4 und einem gegen GP120 gerichteten Antikörper zu kokristallisieren (vgl. Abbildung 18).[188]

Die Röntgenstruktur mit einer Auflösung von 2.5 Å wurde unter dem Zugriffscode 1gc1 aus der *Brookhaven Protein Data Bank*[56;57] geladen. Mit Hilfe von SYBYL wurden aus der Struktur alle nicht zum Komplex gehörenden Atome, alle Wassermoleküle und der Antikörper entfernt. Da Wasserstoffatome in der Röntgenstrukturanalyse nicht aufgelöst werden (vgl. Abschnitt 1.5.3), wurden diese anschließend mit SYBYL hinzugefügt. Analog zu den Arbeiten von Wülfken[34] wurden alle Aminosäuren des GP120 bis auf die zehn oben genannten entfernt und dann die beiden verbleibenden Sequenzabschnitte, das Heptapeptid ^{425}NMWQKVG^{431} und das Tripeptid ^{123}TPL^{125} verknüpft. Hierzu wurde eine Bindung

zwischen dem Carbonyl-Sauerstoffatom des Gly und dem amidischen Stickstoffatom des Thr geknüpft. Unter Fixierung der Aminosäuren Lys und Pro wurde eine Energieminimierung des Tripeptides V-G-T nach AAV 1b durchgeführt. Die so erhaltene Struktur des Dekapeptid-CD4-Komplexes, die aufgrund von Überlappungen der nachträglich hinzugefügten Wasserstoffatome miteinander energetisch sehr ungünstig war, wurde anschließend nach AAV 1c über 1000 Schritte energieminimiert. Hierbei wurden Gasteiger-Marsili-Ladungen und eine Dielektrizitätskonstante von 1 verwendet. Um die Struktur in einer ähnlichen Konformation wie die Struktur von Kwong *et al.* zu erhalten und um Vergleichbarkeit zum biologischen System zu gewährleisten wurde diese Minimierung in einer Wasserbox durchgeführt.

Im Folgenden wurden die, von Wülfken per STD-NMR-Spektroskopie identifizierten, Wechselwirkungen von Aminosäuren des Dekapeptides mit Aminosäuren des CD4 berücksichtigt. Die Seitenkette des Trp interagiert mit Ser42 und Phe43. Die geladene Seitenkette des Lys wechselwirkt ionisch mit der Carboxylgruppe des Asp63. Diese beiden Interaktionen wurden durch Modifikation der entsprechenden Torsionswinkel des Dekapeptides berücksichtigt. Die von Wülfken beschriebene Interaktion des Thr und Leu mit Ser60 und Leu61, sowie die "Einbettung" der Seitenkette des Val durch Trp62, Ser42, Phe43, Arg59, Ser60, Asp63 und Ser23 waren schon ohne manuelle Modifikation gut gegeben.

Die modifizierte Struktur wurde nach AAV 1c über 1000 Schritte in einer Wasserbox energieminimiert.

3.2.2 Vermessung des Pharmakophors

Der Ligand NMWQKVGTPL und der Rezeptor CD4 wurden für die weitere Verwendung in DOCK (ab Abschnitt 3.3.2) im mol2-Format[202] gespeichert.

Um einen Teil des Pharmakophors mit Bausteinen aus der Datenbank DB5 nachahmen zu können, wurden zunächst fünf Ankeratome im Dekapeptid definiert. Basierend auf den STD-Ergebnissen von Wülfken (Abbildung 46) waren dies das aromatisches Hε3 des Trp, ein Hε des Lys ein Hγ des Val, ein Hγ des Thr und ein Hδ des Leu (siehe Abbildung 47 und Tabelle 8).

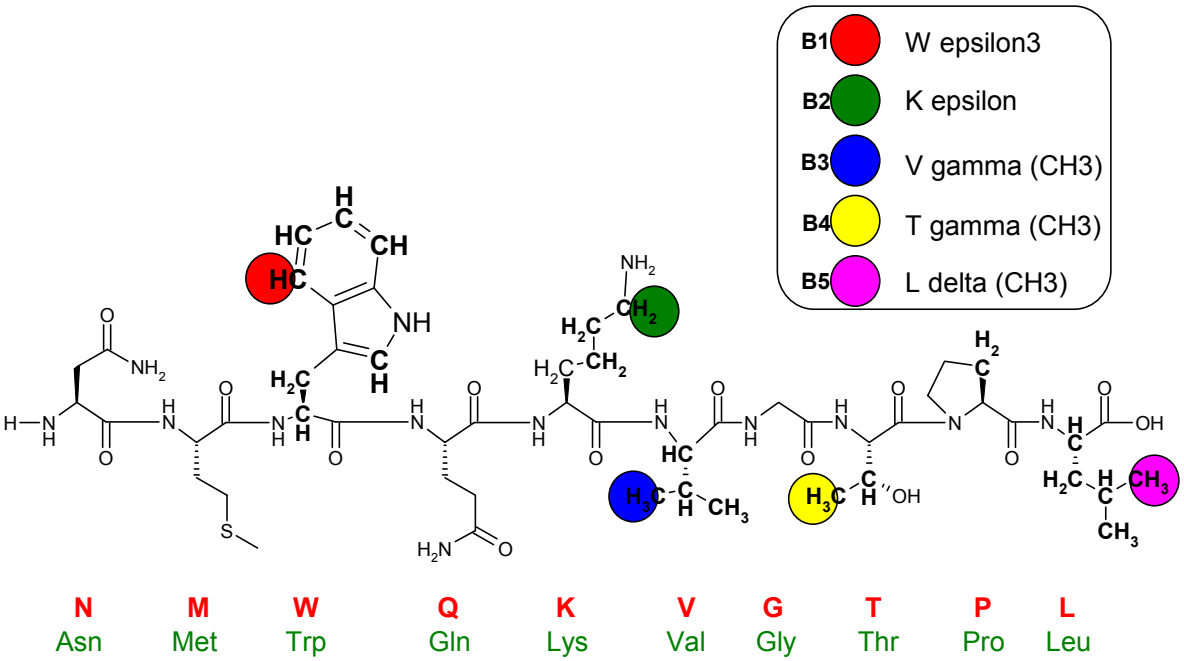


Abbildung 47: Bindungsepitop des Dekapeptides NMWQKVGTPL zum CD4. Die fett gedruckten Atome vermitteln die Bindung. Die fünf farbigen Kreise bezeichnen die fünf Protonen, die als Repräsentanten der bindenden Gruppen ausgewählt und als Anker B1 bis B5 definiert wurden.

Tabelle 8: Nähere Beschreibung der fünf Ankeratome im NMWQKVGTPL (vgl. Abbildung 47). Die eigentlichen Anker sind die als Pharmakophoratom bezeichneten Wasserstoffatome. Zur eindeutigen Bezeichnung der Atome ist zusätzlich in der Spalte "Gerüstatom" das Kohlenstoffatom mit angegeben, an das sie kovalent gebunden sind.

Anker	Aminosäure	Pharmakophoratom	Gerüstatom
B1	W - Tryptophan	Hε3	Cε3
B2	K - Lysin	Hε	Cε
B3	V - Valin	Hγ	Cγ
B4	T - Threonin	Hγ	Cγ
B5	L - Leucin	Hδ	Cδ

Die zehn Abstände zwischen diesen fünf Atomen wurden mit SYBYL vermessen. Die Ergebnisse sind in Tabelle 9 und Abbildung 49 dargestellt.

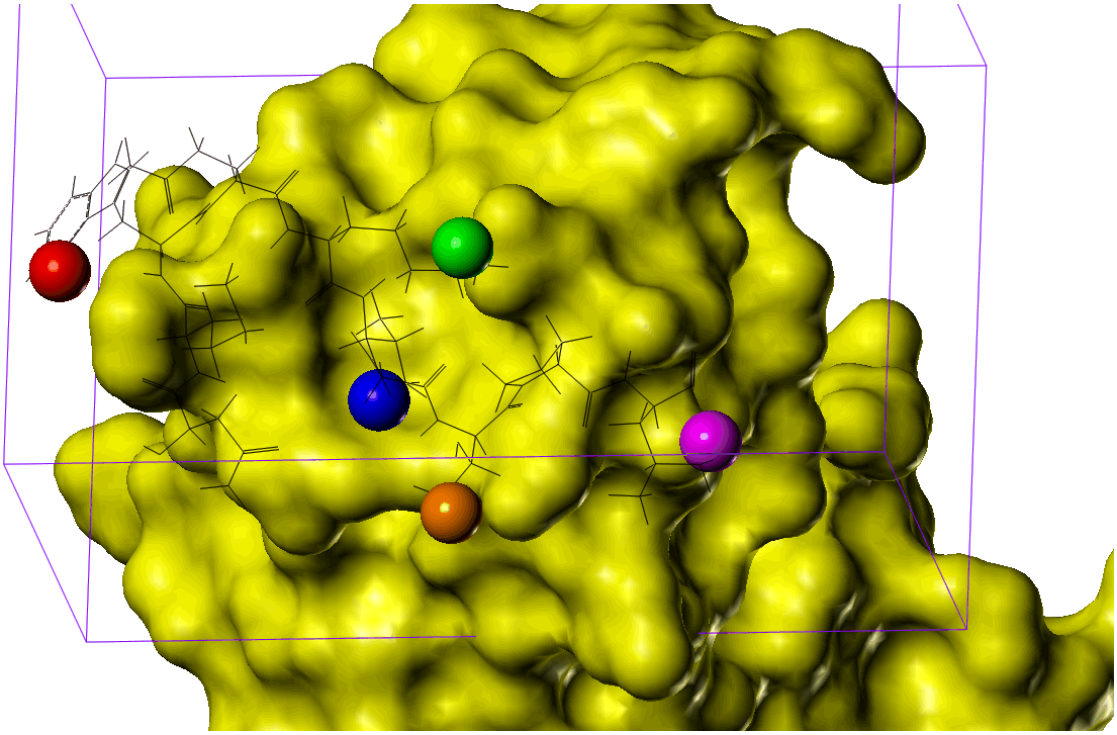


Abbildung 48: Darstellung der fünf Ankeratome B1 bis B5 (farbige Kugeln, Farbcode entsprechend Abbildung 47) im Dekapeptid NMWQKVGTPL (schwarze Linien). Die Connolly-Oberfläche[100] des CD4 ist gelb dargestellt, die *box* für das spätere *docking* als violetter Kasten.

Exemplarisch wurden die Anker B2 und B5, also das Hε des Lysins und das Hδ des Leucins ausgewählt. Der Abstand zwischen ihnen beträgt 11.477 Å. Dieser wurde im folgenden Abschnitt für die Suche in der Datenbank DB5 verwendet.

Tabelle 9: Abstände zwischen den in Abbildung 47 definierten und Abbildung 48 dargestellten Ankeratomen im Dekapetid NMWQKVGTPL. Das Ankerpaar B2-B5 ist fett gedruckt, da es im Folgenden für die Suche in der Datenbank verwendet wurde.

Anker-Paar	Entfernung
B1-B2	15.138
B1-B3	11.940
B1-B4	16.816
B1-B5	23.437
B2-B3	8.345

Anker-Paar	Entfernung
B2-B4	9.207
B2-B5	**11.477**
B3-B4	7.043
B3-B5	11.801
B4-B5	9.818

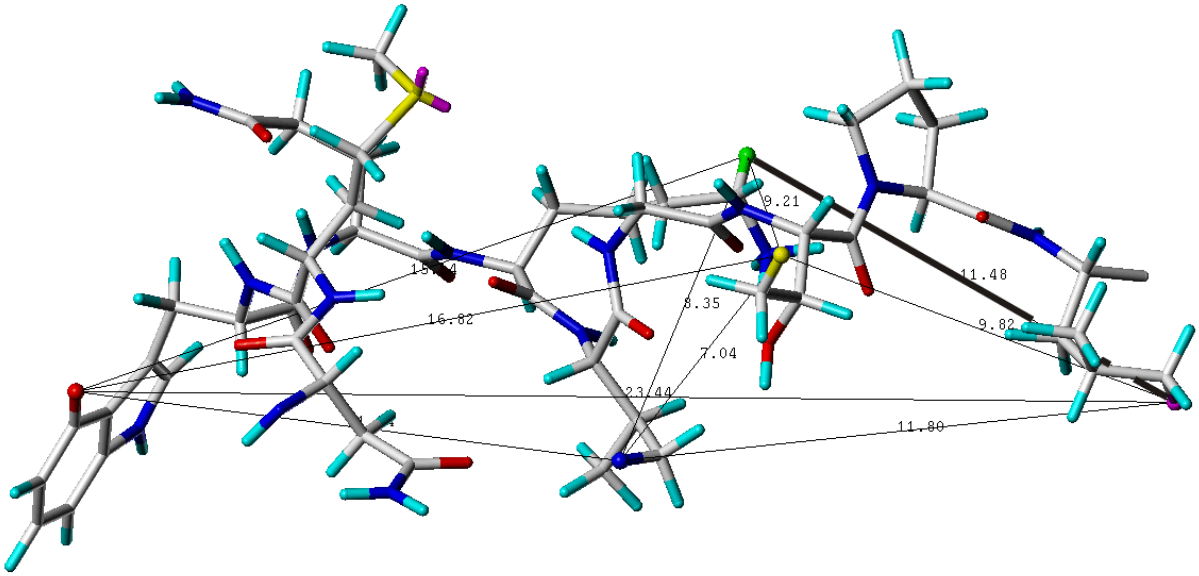


Abbildung 49: Darstellung aller Entfernungen zwischen den als Kugeln dargestellten Ankeratomen B1 (rot), B2 (grün), B3 (blau) B4 (gelb) und B5 (magenta) im an CD4 (nicht dargestellt) gebundenen Dekapeptid NMWQKVGTPL. Der Rest des Peptides ist "*by atom color*" eingefärbt (vgl. Seite V). Die Strecke zwischen den Ankern B2 und B5 ist fett gedruckt, da sie im Folgenden für die Suche in der Datenbank verwendet wurde.

3.3 Auswahl virtueller Leitstrukturen

Die Entfernung von 11.477 Å, die an diesem Punkt als einziges Kriterium für die Suche in der Datenbank DB5 verwendet werden sollte, repräsentiert die tatsächliche chemische Situation im Pharmakophor nur ungenau und unvollständig. Durch die Reduktion aller Anforderungen der Bindungstasche auf den Abstand zwischen zwei Protonen wird das Problem extrem vereinfacht. Informationen über weitere sterische und elektrostatische Anforderungen und mögliche Einschränkungen sind in diesem Konzept nicht enthalten. Auch täuscht die Angabe des "Suchabstandes" mit drei Nachkommastellen über die Grobheit des Modells hinweg. Die Bausteine, die aufgrund dieser Eingabedaten gefunden werden, können nur Anregungen für die weitere Wirkstoffentwicklung sein. Auch nach ihrer chemischen Funktionalisierung in Abschnitt 3.4 sind sie bestenfalls für dieses *in silico*-Modell optimiert. *In vitro*-Assays werden durch *in silico*-Verfahren wie dieses selbstverständlich nicht vollständig zu ersetzen sein, ebenso wenig wie der chemische Sachverstand des Forschers.

3.3.1 Suche in der Datenbank

Für das Ankerpaar B2-B5 (vgl. Abbildung 47) wurde im vorherigen Abschnitt eine Entfernung von 11.477 Å gemessen. Diese wurde zur Suche in der Datenbank DB5 genutzt. Um der Grobheit des Modells gerecht zu werden, wurde eine *allowed deviation* von 0.2 Å verwendet.

Es wurden 943 Ankerpaare gefunden, die in der Lage sind, energiegünstig den Abstand von 11.477 Å zu überspannen. Die chronologisch ersten drei Ankerpaare, die von verschiedenen RBR stammen, sind exemplarisch in Tabelle 10 dargestellt.

Tabelle 10: Auszug aus den 943 Ankerpaaren aus der Datenbank DB5 auf die Anfrage nach Bausteinen, die energiegünstig die Distanz von 11.477 Å (± 0.2 Å) überspannen können. Da hier noch kein *ranking* existiert, sind chronologisch die ersten drei Ankerpaare aufgelistet, die von verschiedenen RBR stammen.

	E_{cut}	E_{min}	E_{max}	new_E_{max}	lfd.Nr.	lfd.Nr.	d_{min}	d_{max}	Anker-Atome
CR0XL0XCR7	1k	32.89	39.43	33.89	112	10	11.625	11.645	8 CR0.H4 --> 35 CR7.H25
CR0XL0XCR8	1k	30.15	36.97	31.15	136	2	11.295	11.325	7 CR0.H3 --> 35 CR8.H4
CR0XL1XCR5	1k	12.10	19.80	13.10	222	10	9.255	11.535	9 CR0.H5 --> 31 CR5.H7

Das Skript dist_2_rotamerlist.sh selektiert nicht nur die passenden Ankerpaare, es durchsucht auch die csv-Dateien, die die Rotamere der jeweiligen RBR enthalten, nach dem Rotamer, das den gesuchten Abstand am besten erfüllt. Existiert kein Rotamer, welches exakt den gesuchten Abstand überspannt, so wählt das Skript das Rotamer, das, innerhalb der Energiegrenzen des Energieschnittes die geringste Abweichung von der gesuchten Distanz aufweist. Als Beispiele für die Ergebnisse dieses Schrittes sind in Abbildung 50 die drei RBR aus Tabelle 10 mit den von ihren ausgewählten Rotameren überspannten Distanzen dargestellt.

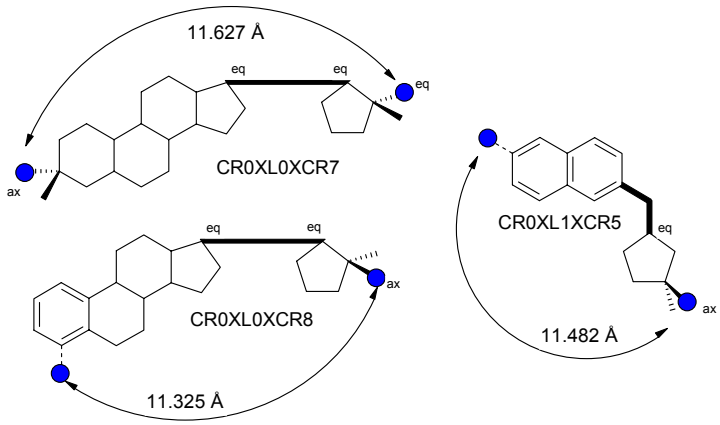


Abbildung 50: Abbildung je eines Rotamers für jedes der drei Ankerpaare aus Tabelle 10 und somit dreier *hits* auf die Anfrage nach Bausteinen, die energiegünstig eine Entfernung von 11.477 Å (± 0.2 Å) überspannen können. Die mit blauen Kreisen markierten Atome sind die Ankeratome, die diese Entfernung aufspannen.

3.3.2 *Ranking* der initialen *hits*

Die Liste der *hits* nach dem initialen Schritt, der Suche in der Datenbank DB5, wird immer relativ groß sein (vgl. Abbildung 44). Da die chemische Funktionalisierung zeitintensiv ist und in diesem System nicht vollautomatisch vom Computer erledigt werden kann, sollte nach der initialen Suche eine Verringerung der Anzahl der *hits* erfolgen. Im Falle der vorliegenden Arbeit wurde dies mit Hilfe des Programms DOCK[126] und der 3D-Struktur des Rezeptors durchgeführt. Für Fälle, in denen keine 3D-Struktur des Rezeptors vorliegt, sind andere Möglichkeiten des *ranking* und der Verringerung der Anzahl der *hits* denkbar. Mögliche Kriterien für die Selektion von *hits* sind zum Beispiel die gewünschte Flexibilität, die Molmasse, das Vorhandensein oder Nicht-Vorhandensein bestimmter Ringsysteme oder auch QSAR-Studien.

3.3.3 Vorbereitung des *docking*

Da in diesem Fall eine 3D-Rezeptorstruktur (vgl. Seite 69) vorlag, konnte diese, in Verbindung mit dem Programm DOCK, für die weitere Selektion der *hits* verwendet werden.

Das *docking* wurde im Wesentlichen nach der Kurzanleitung im DOCK-Handbuch[163] durchgeführt. Auf die Abweichungen davon wird im Folgenden näher eingegangen.

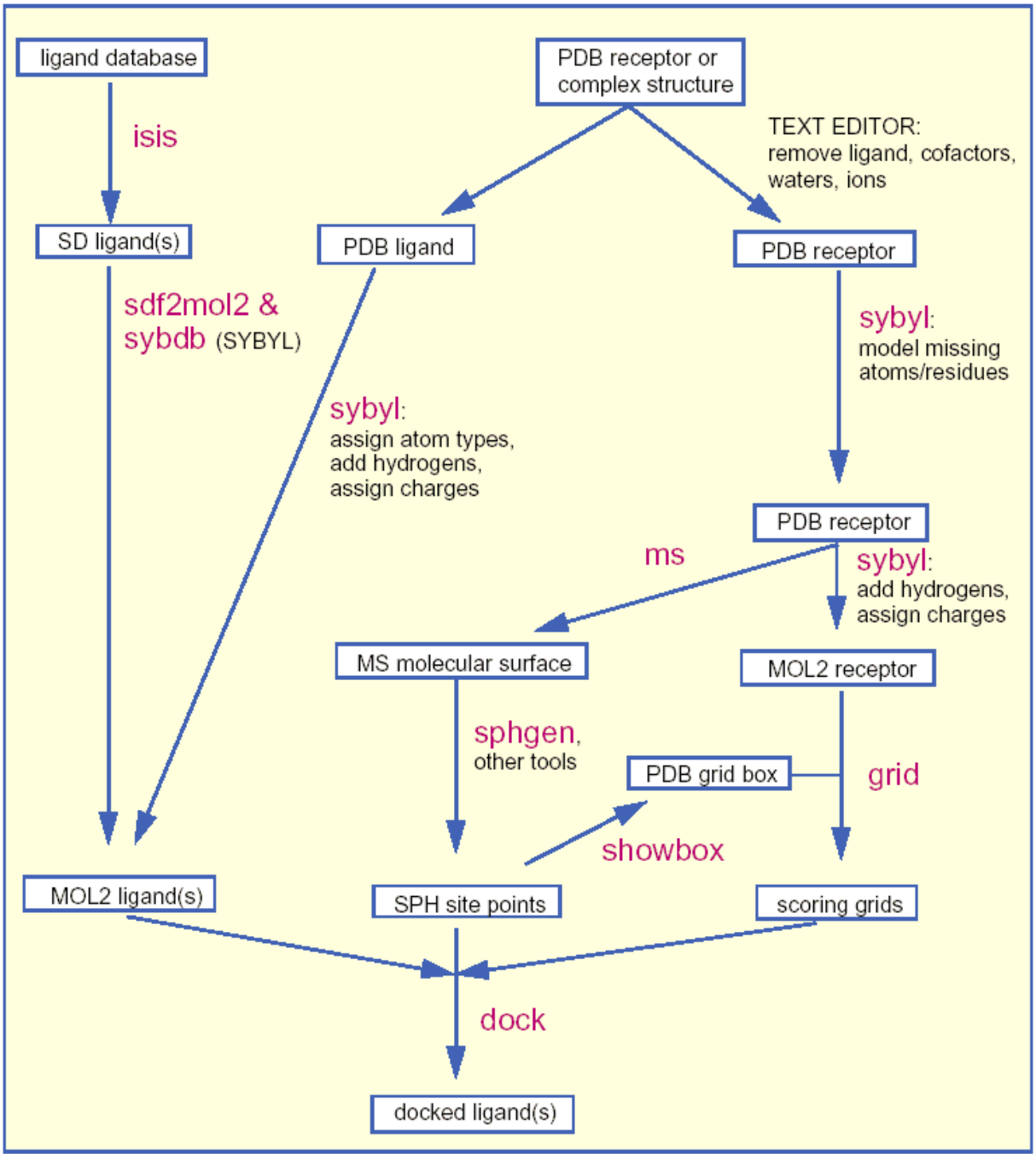


Abbildung 51: Flussdiagramm aus der Bedienungsanleitung zu Dock4.[163] Es ist gezeigt, wie man von einer Datenbank von Liganden (oben links) und einer Röntgenstruktur des Rezeptors (oben mittig) zu gedockten Liganden (unten mittig) kommt. Weiße Kästen mit blauem Rand beschreiben Dateien. Die anzuwendenden Programme sind in magentafarbener Schrift angegeben.

Der so genannte PDB-Rezeptor wurde aus dem in Abschnitt 3.2.1 erzeugten Rezeptor (vgl. Seite 69) erzeugt, indem alle Wasserstoffatome, alle Atome des GP120 bzw. Dekapeptides und alle Ladungen entfernt wurden.

Der so genannte mol2-Rezeptor wurde aus dem in Abschnitt 3.2.1 erzeugten Rezeptor erzeugt, indem zuerst alle Atome des GP120 bzw. Dekapeptides und alle Ladungen entfernt wurden. Anschließend wurden Ladungen nach dem Gasteiger-Marsili-Modell[203] hinzugefügt.
Die Rezeptoroberfläche wurde mit dem Programm AUTOMS für den gesamten Rezeptor erzeugt.
Das Programm SPHGEN wurde verwendet, um einen Cluster von *spheres* zu erzeugen, die DOCK später für die Platzierung des Liganden benötigt. Von den 39 erzeugten *sphere clusters*, die an verschiedensten Stellen der Oberfläche des CD4 lokalisiert waren, wurde einer, der im Bereich der Bindungsstelle des Dekapeptides lag, ausgewählt und manuell modifiziert. Gelöscht wurden *spheres*, die weit entfernt von der Region lagen, in der das Dekapeptid bindet. In Bereichen, in denen zwar das Dekapeptid bindet, der ausgewählte Cluster aber keine *spheres* enthielt, wurden *spheres* aus anderen Clustern hinzugefügt. Der resultierende *sphere cluster*, der 40 *spheres* enthielt, ist in Abbildung 52 dargestellt. Für das *constrained docking* wurden dem *sphere cluster* zwei weitere *spheres* hinzugefügt, an den Koordinaten, an denen die Ankeratome B2 und B5 des Dekapeptides lokalisiert waren. Diese beiden *spheres*, die in der *sphere*-Datei als "*critical spheres*" (vgl. Abschnitt 1.7.4) gekennzeichnet wurden, sind in Abbildung 52 grün (Lys) und magentafarben (Leu) dargestellt.

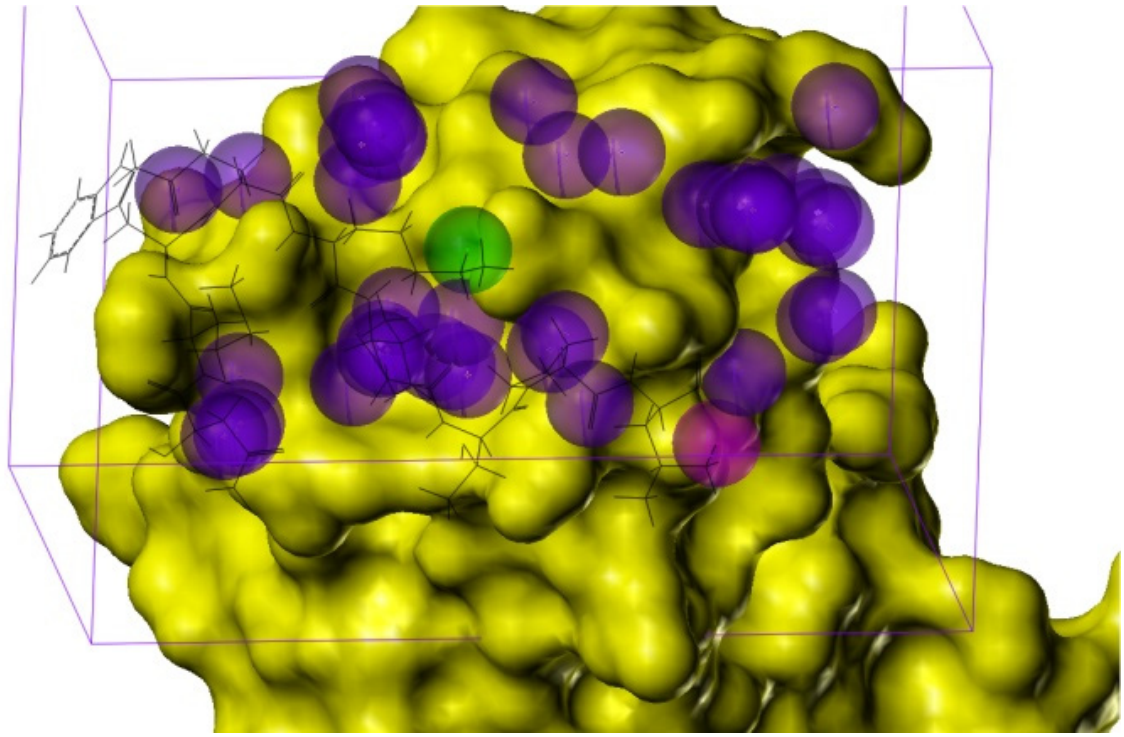


Abbildung 52: Abbildung des *sphere cluster* für das *constrained docking*. Die 40 von SPHGEN generierten *spheres* sind als violette Kugeln dargestellt. Die beiden zusätzlich hinzugefügten *critical spheres* sind grün (Bildmitte, am Lysin des Dekapeptides) und magenta (unten rechts im Bild, am Leucin des Dekapeptides) dargestellt. Das Dekapeptid ist als schwarze Linien, die Oberfläche des CD4 in gelb und die *box* für das *constrained docking* als violetter Kasten dargestellt.

Das *grid*, welches DOCK dazu nutzt, das spätere *scoring* zu beschleunigen, ist ein virtuelles Gitter, das den gesamten Raum der Bindungsregion füllt. Für jeden Gitterpunkt werden im Vorwege bereits die elektrostatischen und sterischen Einflüsse des Rezeptors auf ein Atom an diesem Punkt berechnet und abgespeichert. Für das *scoring* stehen diese dann bereits zur Verfügung. Das *scoring-grid* wurde mit den Programmen SHOWBOX und GRID erzeugt. Die entsprechende Parameter-Datei ist in Anhang 7.4 abgedruckt.

Erzeugung von *ligand spheres*

Für das *constrained docking*, bei dem die Ankeratome des zu dockenden Liganden exakt an den Positionen der Ankeratome B2 und B5 des Dekapeptides fixiert werden sollten, wird für jeden zu dockenden Liganden eine *sphere*-Datei *(sphere_file)* benötigt, in der *spheres* an den beiden zu verwendenden Ankeratomen dieses Liganden definiert sind. Diese sph-Dateien wurden mit 7.3.15 ligand_2_ligandcenter.spl automatisiert direkt aus den entsprechend benannten mol2-Dateien der Liganden erzeugt.

Um automatisiert einen Liganden nach AAV 2 (*constrained docking*) docken zu können und gleichzeitig die dafür benötigte ligand.sph-Datei zu erzeugen, wurde das Shell-Skript 7.3.16 dock_single_ligand_to_B2B5.sh verwendet. Dieses wiederum wird von einem weiteren Skript (7.3.17 dock_all_in_dir_to_B2B5.sh) aufgerufen, welches dazu genutzt wurde, alle Liganden, deren mol2-Dateien sich innerhalb eines Verzeichnisses befinden, nach AAV 2 an CD4 zu docken. Für jeden gedockten Liganden wird eine log-Datei der Ausgabe von DOCK erzeugt.

3.3.4 *Docking* und Ergebnisse

Für die 943 *hits* aus Schritt 3.3.1 wurden mit Hilfe von rotamerlist_2_manymol2s.spl mol2-Dateien erzeugt. Diese sollten mit Hilfe von dock_all_in_dir_to_B2B5.sh gemäß AAV 2 an den Rezeptor gedockt werden. Dieses *docking* war, aufgrund der großen Anzahl an Liganden und aufgrund von Beschränkungen der verwendeten Software (der Begrenzung des für Argumente und Umgebungsvariablen reservierten Adressraums) nur mit dem Skript 7.3.18 dock_struct_ligandsx_to_B2B5.sh möglich. Für jeden Liganden wurde hierbei eine dock.in-Datei erzeugt. Das Templat, auf dem alle diese Dateien beruhen, ist in Tabelle 25 abgedruckt.

Die Analyse der *dockings* erfolgte mit Hilfe des Skriptes 7.3.19 analyze_dockings_in_dir.sh. Es zeigte sich, dass von den 943 ersten *hits* 660 ein *docking* ergaben und 283 nicht (vgl. Tabelle 11). Die 283 Moleküle, die sich gemäß AAV 2 nicht docken ließen, wurden verworfen. Alle anderen 660 *hits* wurden weiter untersucht.

Tabelle 11: Ausschnitt aus den Ergebnissen analyze_dockings_in_dir.sh. Nach der Auflistung der 283 Moleküle, die kein *docking* ergaben, sind energiesortiert die 660 *hits* aufgelistet, die sich docken ließen.

NULL ORIENTATIONS FOR :
dock_CR0XL1XCR7_10.967_8_39.log, 0, 1000.00
dock_CR0XL1XCR7_11.736_8_40.log, 0, 1000.00
dock_CR0XL1XCR7_12.061_8_38.log, 0, 1000.00
...
dock_CR6XL3XCR6_9.853_14_41.log, 0, 1000.00
SUCCESSFUL DOCKINGS :
dock_CR3XL4XCR5_11.551_15_41.log, 2, -10.05
dock_CR3XL1XCR5_11.522_14_33.log, 2, -7.70
dock_CR5XL3XCR8_11.707_15_47.log, 2, -6.84
...
dock_CR6XL1XCR7_11.396_14_44.log, 2, 1500928.00

Für jeden der 660 *hits*, die erfolgreich gedockt werden konnten ("bestätigte *hits*"), hatte DOCK zwei Orientierungen abgespeichert. Diese Orientierungen unterschieden sich vor allem darin, welches der Ankeratome des RBR welche der zu besetzenden "*critical spheres*" B2 und B5 belegte. Beispielhaft sind in Abbildung 53 und Abbildung 54 die beiden Orientierungen für das RBR CR3XL4XCR5_11.551_15_41 angegeben, das insgesamt das *docking* mit der günstigsten Energie (-10.05 kcal/mol) ergab.

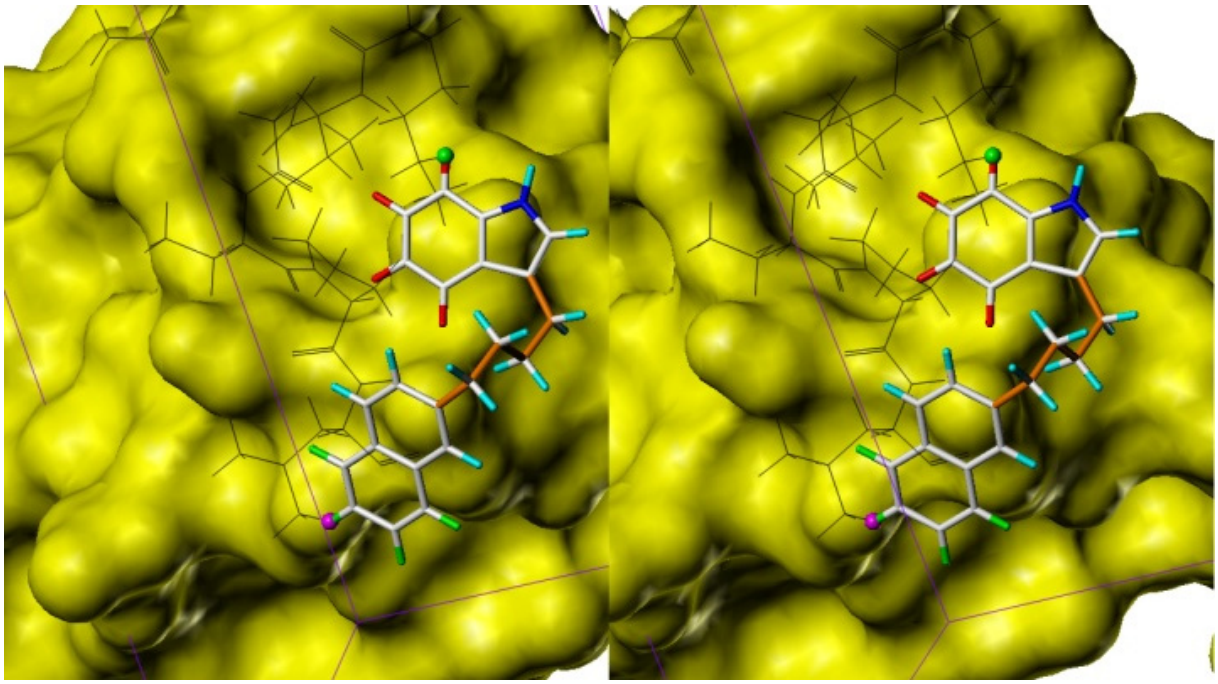


Abbildung 53: Erste Orientierung aus dem *constrained docking* des RBR CR3XL4XCR5_11.551_15_41. Farben entsprechend Abbildung 35 und Seite V. Diese Orientierung erreichte eine Energie von -10.05 kcal/mol. *Crossed* Stereodarstellung.

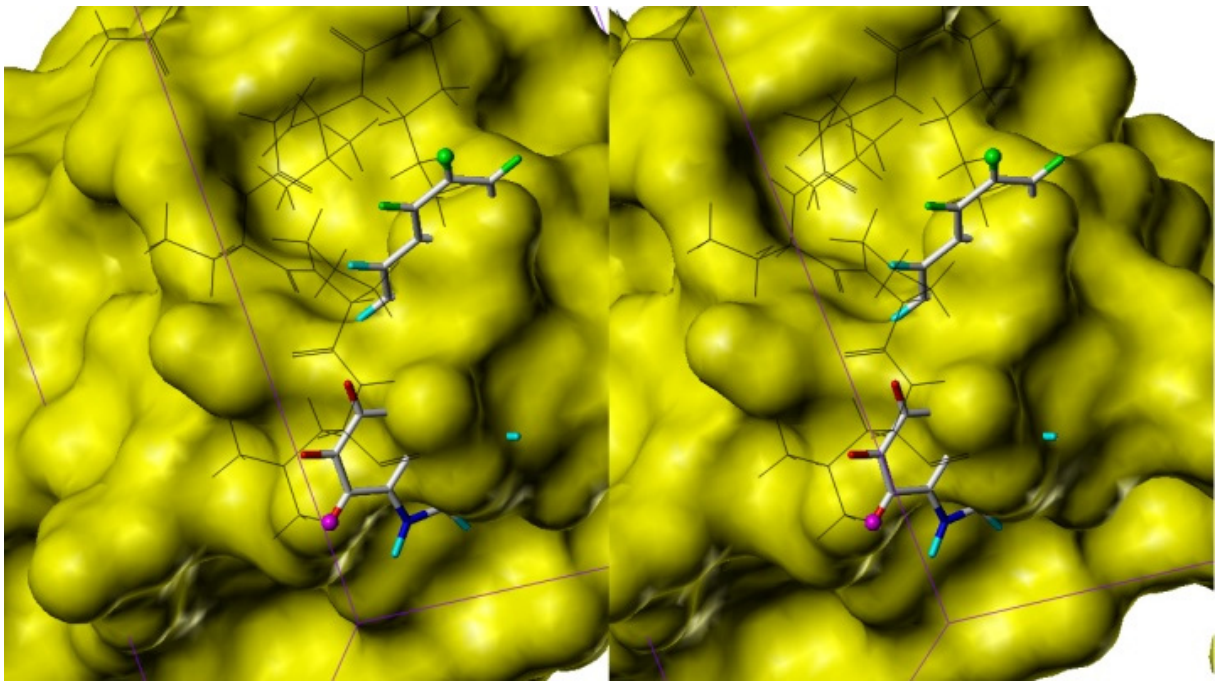


Abbildung 54: Zweite Orientierung aus dem *constrained docking* des RBR CR3XL4XCR5_11.551_15_41. Farben entsprechend Abbildung 35 und Seite V. Diese Orientierung wurde von DOCK mit einer Energie von 4187562 kcal/mol bewertet. Die Energie ist so hoch, weil Atome des Liganden mit Atomen des Rezeptors überlagern. Die jeweils zweiten Orientierungen hatten oft sehr hohe Energien und wurden im Rahmen dieser Arbeit nicht weiter verwendet.

Für die 660 *hits*, die erfolgreich gedockt werden konnten, wurde mittels analyze_dockings_in_dir.sh ein *ranking* erstellt. Hierfür wurden jeweils die Energiewerte der günstigeren der beiden Orientierungen verwendet.
Die Ergebnisse der 660 erfolgreich gedockten *hits* sind in Tabelle 12 und Abbildung 55 und Abbildung 56 zusammengefasst.

Tabelle 12: Übersicht über die Energien der *constrained dockings* der 943 *hits*.

943 gedockt, davon 283 ohne Erfolg 660 mit Erfolg (günstigste Energie = -10.05 kcal/mol)	
E < 0 kcal/mol	84
0 ≤ E ≤ 100 kcal/mol	102
100 ≤ E ≤ 1000 kcal/mol	77
1000 ≤ E ≤ 10000 kcal/mol	82
10000 ≤ E ≤ 100000 kcal/mol	150
100000 ≤ E ≤ 1500928 kcal/mol	165

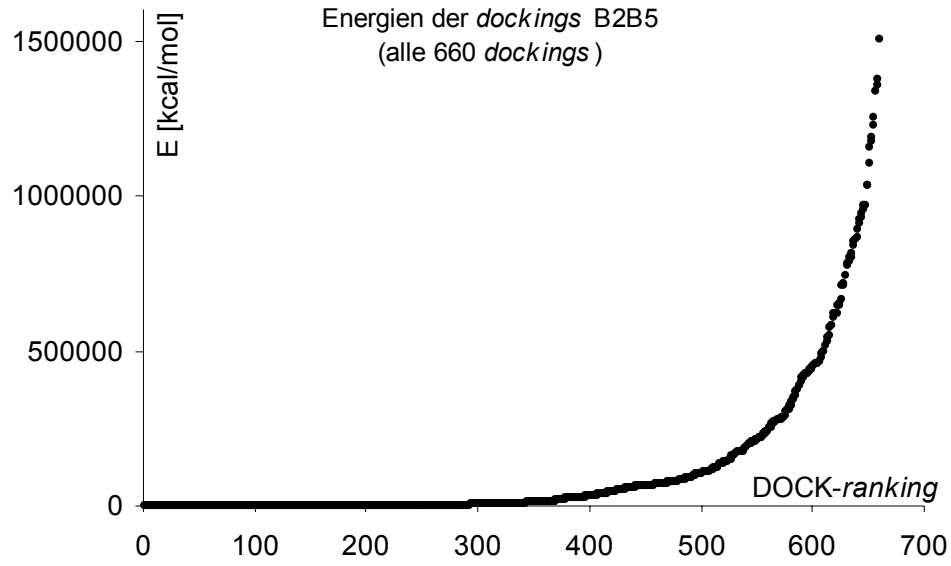


Abbildung 55: Auftragung der in Tabelle 12 zusammengefassten Energiewerte der *constrained dockings* der 660 *hits* an CD4. Eine Vergrößerung des Bereiches der besten 100 DOCK-*rankings* folgt in Abbildung 56.

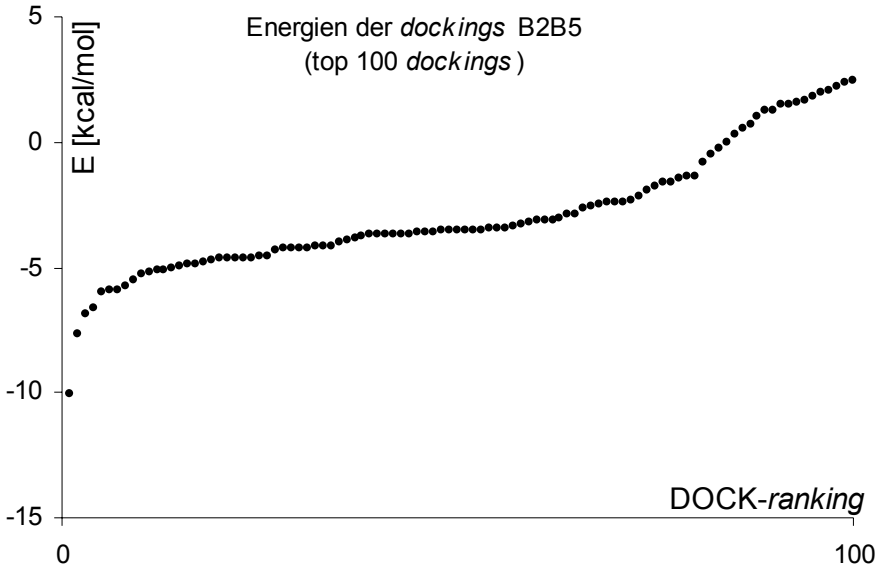


Abbildung 56: Auftragung der Energiewerte der besten 100 *constrained dockings* der 660 *hits* an CD4. Die Energien aller 660 *dockings* sind in Abbildung 55 aufgetragen.

3.3.5 Beurteilung nach Variabilität in den Ringen

Um ein möglichst repräsentatives Spektrum der *hits* weiter verfolgen zu können, wurden fünf der TOP10 bestätigten *hits* als Leitstrukturgerüste für das weitere Vorgehen ausgewählt. Die TOP10 sind in Abbildung 57 abgebildet, ihre Energien in Tabelle 13 aufgelistet.

Tabelle 13: Auflistung der TOP10 des *constrained docking* der 660 *hits* der Suche nach Rotameren, die energiegünstig die Entfernung von 11.477 Å überspannen können.

gedockter Ligand	Energie [kcal/mol]
CR3XL4XCR5_11.551_15_41	-10.05
CR3XL1XCR5_11.522_14_33	-7.70
CR5XL3XCR8_11.707_15_47	-6.84
CR4XL4XCR5_11.464_13_42	-6.63
CR1XL4XCR7_11.474_07_50	-5.99
CR2XL4XCR4_11.478_12_42	-5.92
CR4XL4XCR5_11.475_13_40	-5.88
CR5XL2XCR7_11.400_16_48	-5.72
CR4XL4XCR4_11.477_13_40	-5.53
CR3XL2XCR7_11.558_14_45	-5.26

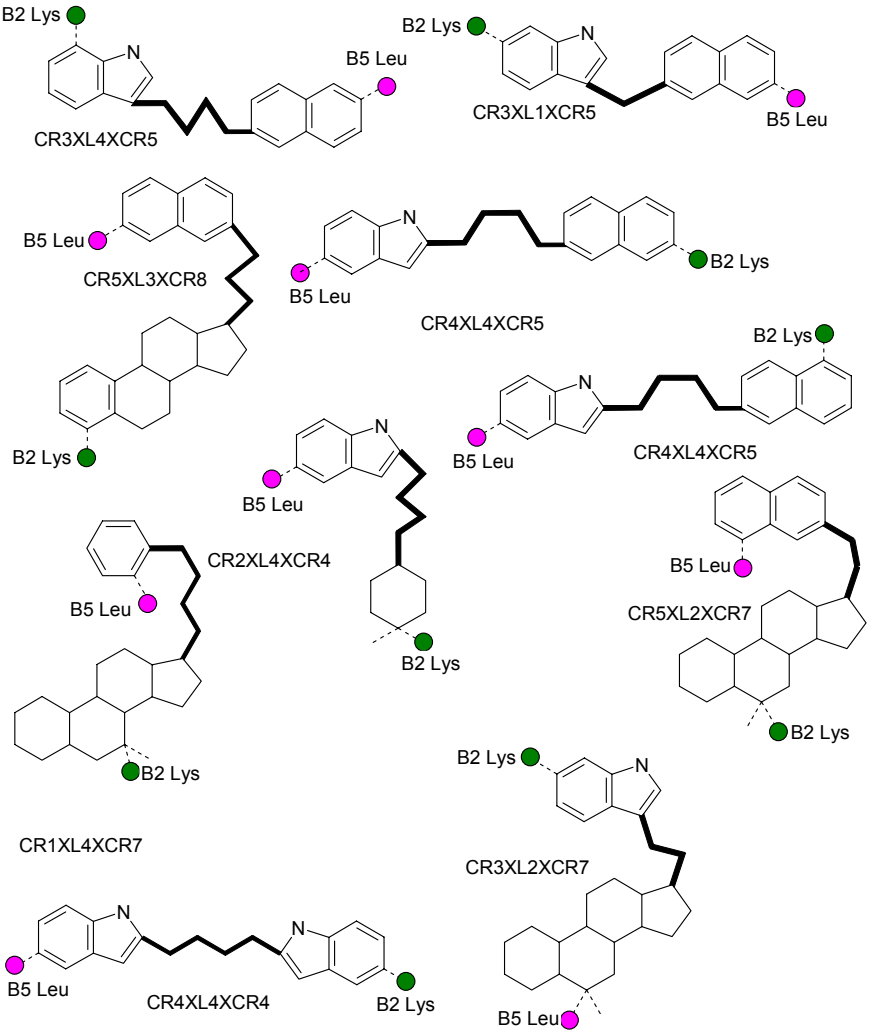


Abbildung 57: Die TOP10 des *constrained docking* der 660 *hits* der Suche nach Rotameren, die energiegünstig die Entfernung von 11.477 Å überspannen können. Die Liganden docken in ihrer energiegünstigeren Orientierung (vgl. Abbildung 53 und Abbildung 54) mit dem hier magentafarben markierten Proton an der Stelle, wo sich der Anker B5, also das Hδ des Leucins befindet. Das hier grün markierte Proton dockt in dieser Orientierung dann am Anker B2, der Position des Hε des Lysins (vgl. Tabelle 8).

Bei einem visuellen Vergleich der RBR fiel auf, dass CR4XL4XCR5 (viertbeste Energie) dem RBR mit der besten Energie (CR3XL4XCR5) sehr ähnlich ist. Beide RBR bestehen aus den gleichen Ringsystemen und der gleichen Brücke, nur der Verknüpfungspunkt der Brücke am Indol-Ringsystem ist um eine Position verschoben. Da die Untersuchung von sowohl CR4XL4XCR5 als auch CR3XL4XCR5 wahrscheinlich zu sehr ähnlichen Ergebnissen geführt hätte, wurde auf die weitere Untersuchung von CR4XL4XCR5 verzichtet, und stattdessen noch der *hit* mit der sechstbesten Energie (CR2XL4XCR4) mit in Schritt 3.4 übernommen.

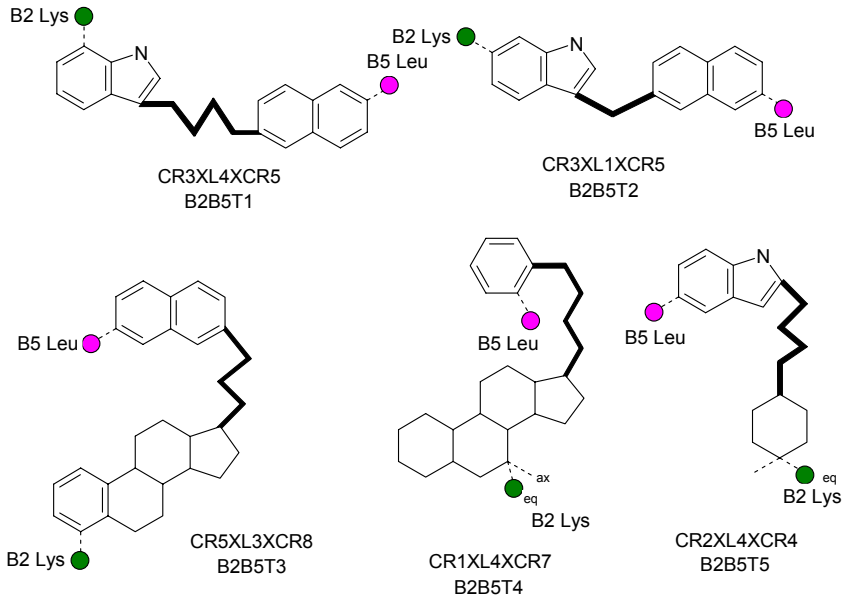


Abbildung 58: Die fünf RBR, die zur chemischen Funktionalisierung in Abschnitt 3.4 ausgewählt wurden (TOP5). Angegeben ist jeweils die Bezeichnung, die sich aus der *in silico*-Kombinatorik ergeben hatte und die im Folgenden verwendete Bezeichnung, die den Rang des *hit*, bewertet nach Energie angibt (so ist B2B5T2 der *hit*, der unfunktionalisiert beim *constrained docking* an die Anker B2 und B5 die zweitbeste Energie ergeben hatte).

3.4 Funktionalisierung der ausgewählten Gerüste

Die Datenbank DB5 enthält nur unfunktionalisierte RBR. Reale Wirkstoffe enthalten eine Vielzahl von funktionellen Gruppen.[32] Diese "*sidechains*" wurden bisher in dieser Arbeit ignoriert. Sie wurden im Ansatz von Bemis und Murcko (vgl. Abschnitt 1.3.1) entfernt, als die Wirkstoffe der *Comprehensive Medicinal Chemistry database* in Gerüste, Ringsysteme, Brücken und Seitenketten aufgeteilt wurden. Für die Konstruktion der RBR in Abschnitt 3.1 wurden nur die Analysen über Ringsysteme und Brücken (*linker*) angewandt. Seitenketten wurden bis zu diesem Abschnitt vollständig ignoriert. Des Weiteren wurden auch Heteroatome innerhalb der Ringsysteme und der Brücken bisher ignoriert.

Durch das Hinzufügen von Seitenketten und funktionellen Gruppen an den Ringsystemen und durch Heteroatom-Substitiution innerhalb der RBR ist es möglich, die Bindungseigenschaften der jeweiligen RBR zu verbessern. Hinzugefügte unpolare Reste können in hydrophobe Taschen des Rezeptors binden. Zwischen geladenen Gruppen des Rezeptors und neu hinzugefügten geladenen Gruppen am RBR können Ionenbindungen entstehen etc. Solche Verbesserungen der Bindung sind nur zu erwarten, wenn beim Hinzufügen von funktionellen Gruppen Informationen verwendet werden können, die über die die Pharmakophor-Distanz

hinausgehen. Als Quelle für solche, die Bindung näher beschreibenden Informationen kommen zum Beispiel Kristallstrukturen, QSAR-Studien und *transferred* NOESY-NMR-Studien in Frage, und somit genau die Techniken, die auch zur Beschreibung des Pharmakophors verwenden werden können. Unter Zuhilfenahme dieser Informationen kann eine Optimierung der Liganden *in silico* stattfinden, bevor erste *in vitro*-Tests oder ein HTS durchgeführt werden.

Wenn keine Röntgenkristallstruktur vorliegt, muss sich die Einführung funktioneller Gruppen allein an der Funktionalität des Liganden, der als zur Ermittlung des Pharmakophors gedient hatte, orientieren. Für die Abschätzung der Bindungsenergien kann dann allein die Ähnlichkeit zu diesem Liganden herangezogen werden.

Im vorliegenden Fall existiert sowohl eine Röntgenkristallstruktur,[188] die im Rahmen dieser Arbeit bereits verwendet wurde (Abschnitt 3.2) und ein "Musterligand" (das Dekapeptid NMWQKVGTPL). Die Funktionalisierung der RBR erfolgte im Rahmen dieser Arbeit anhand der Struktur des biologisch aktiven Liganden NMWQKVGTPL.[34] Die funktionellen Gruppen wurden so eingeführt, dass sie im Dekapeptid vorhandene Strukturmotive nachahmen oder konservativ variieren sollten. In wieweit diese Nachahmung jeweils erfolgreich war, und welche funktionelle Gruppe an welcher Position die Bindungseigenschaften wie beeinflusst, wurde *in silico* durch *constrained docking* an die CD4-Röntgenstruktur von Kwong *et al.* untersucht. Die auf diesen *docking*-Untersuchungen basierende Energiebewertung wurde zur Auswahl günstiger Einzelmodifikationen und zur Kombination der jeweils zwei günstigsten Modifikationen in einem neuen Liganden verwendet. Die Bindungseigenschaften der doppelt funktionalisierten RBR wurden anhand von *unconstrained docking*-Experimenten abgeschätzt. Im Folgenden sind dieses Schritte für die in Abschnitt 3.3.5 ausgewählten Leitstrukturgerüste beschrieben. Da ab diesem Schritt der Arbeit auch ionische Wechselwirkungen betrachtet werden, wurden ab hier alle *docking*-Experimente mit Molekülen durchgeführt, deren Partialladungen nach dem Modell von Gasteiger *et al.*[203] berücksichtigt wurden.

3.4.1 Funktionalisierung des Leitstrukturgerüstes B2B5T1

An dem nach AAV 2 gedockten Leitstrukturgerüst wurden vier verschiedene Funktionalisierungen am Anker B5Leu und sieben Funktionalisierungen am Anker B2Lys durchgeführt. Wie auch bei allen anderen Funktionalisierungen wurden die Änderungen der Atomtypen, das Löschen und Hinzufügen von Atomen und die Berechnung der

Partialladungen in einem SPL-Skript zusammengefasst. Dies erleichterte das Auffinden von Fehlern und erhöhte die Reproduzierbarkeit. Exemplarisch ist in Anhang 7.3.20 das Skript B2B5T1_L01K00_15_41.col zur Erzeugung von B2B5T1_L01K00 abgedruckt. In diesem Skript wurde zuerst das Ergebnis des *docking* des unmodifizierten Liganden B2B5T1_L00K00_15_41 geladen und danach das Wasserstoffatom an der 8-Position des Naphtylringes durch ein sp^3-hybridisiertes Kohlenstoffatom ersetzt. Anschließend wurden die drei neuen Wasserstoffatome an diesem Kohlenstoffatom hinzugefügt. Nach der Berechung von Gasteiger-Marsili-Ladungen[203] wurde das Molekül unter seinem neuen Namen B2B5T1_L01K00_15_41 wieder abgespeichert.
Jedes einfach modifizierte Leitstrukturgerüst wurde nach AAV 2 energiebewertet.

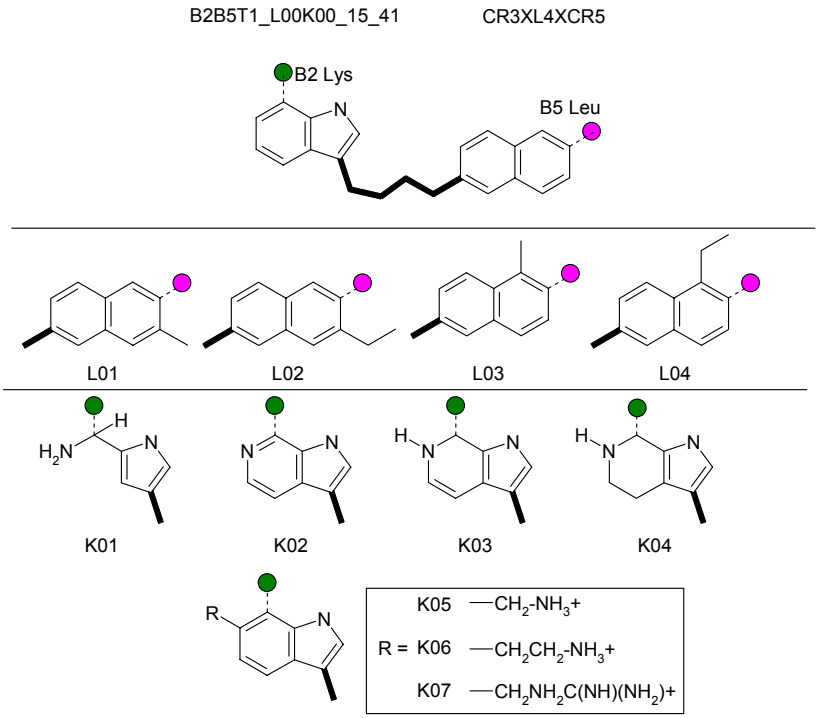


Abbildung 59: Die am RBR CR3XL4XCR5, durchgeführten *in silico*-Funktionalisierungen. Oben: Das unfunktionalisierte RBR. Mitte: Die vier Modifikationen am Ringsystem CR5 zur Nachahmung des Bindungsbeitrages des Leucins. Unten: Die sieben Funktionalisierungen am Ringsystem CR3 zur Nachahmung der Bindungseigenschaften des Lysins. Die fett gedruckte Bindung am jeweiligen Verknüpfungspunkt repräsentiert hier jeweils die gesamte Brücke und das jeweils andere Ringsystem in seiner unmodifizierten Form.

Tabelle 14: Energien der *constrained dockings* der in Abbildung 59 dargestellten einfach funktionalisierten RBR. Die Energie des unmodifizierten RBR L00K00 war mit -9.50 kcal/mol etwas ungünstiger als im ersten *docking* (Tabelle 13), da nun, im Gegensatz zum ersten *docking* die Partialladungen des Moleküls berücksichtigt wurden. Die Coulomb-Energie dieses *docking* von L00K00 beträgt +0.55 kcal/mol.

Ligand	Energie
L00K00	-9.50
L01K00	-9.82
L02K00	-10.03
L03K00	-10.09
L04K00	-10.34
L00K01	-14.23

Ligand	Energie
L00K02	NA
L00K03	-9.12
L00K04	-1.01
L00K05	-16.01
L00K06	4.95
L00K07	62.26

Top scorer waren L04 und K05. Die am L04 neu eingefügte Ethylgruppe passte sich gut in die unpolare Region des Rezeptors ein, in der im Dekapeptid Cβ und Cγ des Leucins liegen.
Auf der Seite des Lysins wurde die ionische Wechselwirkung mit dem Asp63 am besten durch das sterisch am wenigsten anspruchsvoll substituierte Indol K05 erfüllt. Sowohl die Einführung größerer Substituenten an dieser Position (K06, K07) als auch die Aufhebung des Indol-Systems (K01-K04) resultierte in ungünstigeren Energiewerten.
Das doppelt modifizierte B2B5T1_L04K05 zeigte nach AAV 2 bewertet einen Energiewert von -16.85 kcal/mol. Diese Energie setzte sich zusammen aus -11.66 kcal/mol van-der-Waals-Energie und -5.19 kcal/mol Coulomb-Energie. Da im Vergleich dazu B2B5T1_L00K00 eine Energie von -10.05 (van-der-Waals) und +0.55 (Coulomb) gezeigt hatte, wurde deutlich, dass hier vor allem die bessere Nachahmung der ionischen Wechselwirkung des Lysins mit dem Asp63 die Bindungsenergie verbessert hat.
Da die Unterschiede in den Energiewerten der verschiedenen Leucin-Modifikationen eher gering waren (die Werte reichten von -9.05 bis -10.34 kcal/mol, vgl. Tabelle 14), war auch der Effekt des Wechsels von L00 nach L04 gering (0.84 kcal/mol).

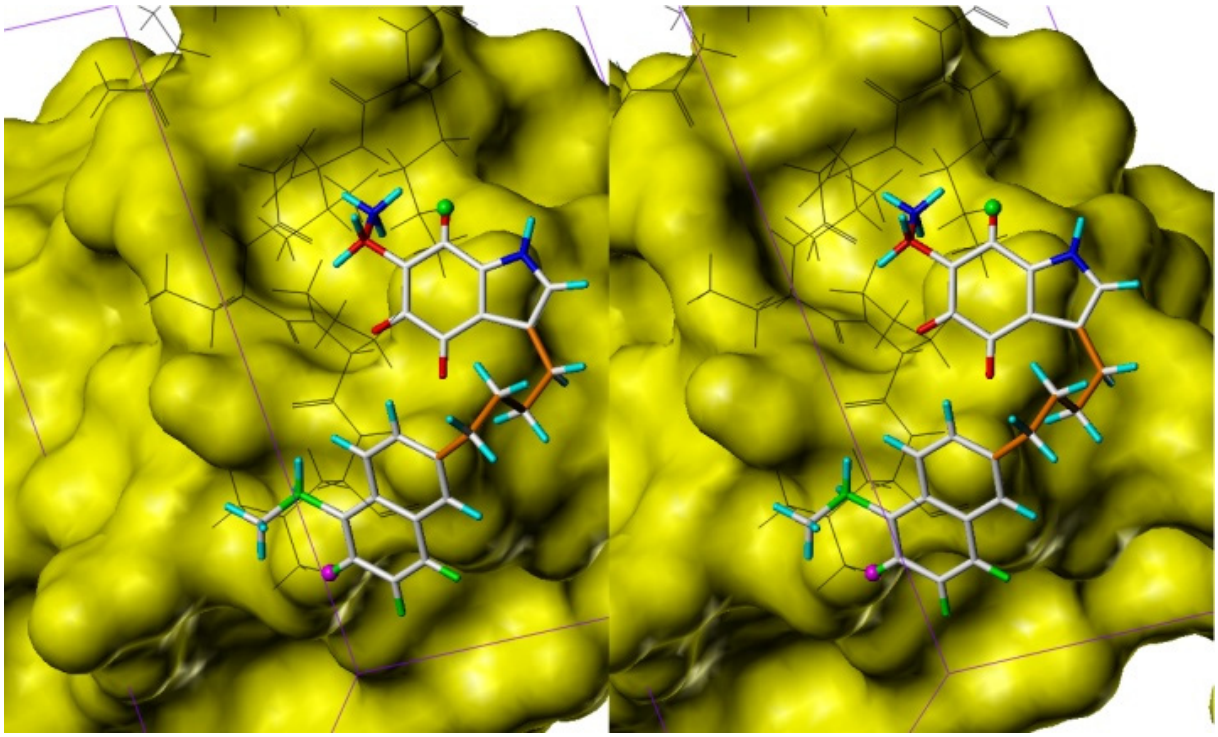


Abbildung 60: ***Crossed*** **Stereodarstellung des** ***constrained docking*** **des Leitstrukturmotivs B2B5T1_L04K05, das eine Energie von -16.85 kcal/mol erreichte. Die Farben sind entsprechend Abbildung 35 und Seite V gewählt.**

3.4.2 Funktionalisierung des Leitstrukturgerüstes B2B5T2

Das nach AAV 2 gedockte Leitstrukturgerüst wurde vierfach am Anker B5Leu und elffach am Anker B2Lys modifiziert. Hierbei stellte sich die Frage, wie die freie Drehbarkeit von neu angeknüpften Bindungen in den Substitutionen K01, K06-K11, L02 und L04 berücksichtigt werden sollte. Es wurde entschieden, für jede dieser Varianten einzeln alle möglichen Rotamere zu generieren und zu docken. Im Anhang 7.3.21 ist exemplarisch das SPL-Skript B2B5T2_L00K07X_14_33.col gezeigt, mit dem die 1296 Rotamere des RBR B2B5T2_L00K07 (Abbildung 61) erzeugt wurden.

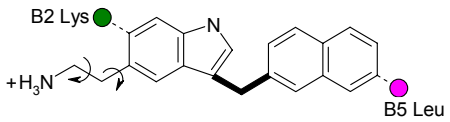


Abbildung 61: Das Leitstrukturgerüst B2B5T2, einfach funktionalisiert gemäß Abbildung 62 zum B2B5T2_L00K07. Die optimalen Torsionswinkel der mit Pfeilen gekennzeichneten rotierbaren Bindungen der neu hinzugefügten Aminoethylgruppe werden mit Hilfe der Skripte B2B5T2_L00K07X_14_33.col und B2B5T2_LXXKXX_torsionscan.sh ermittelt.

Mit Hilfe des Skriptes 7.3.22 B2B5T2_LXXKXX_torsionscan.sh wurden alle diese Rotamere nach AAV 2 gegen CD4 gedockt, ebenso wie die Rotamere für die anderen Substitutionen. Das Skript erzeugte für jeden Liganden eine zusammenfassende Datei, aus der dann das Skript 7.3.23 B2B5T2_move_best_rotamers.sh die jeweils besten Rotamer selektierte. Dieses Vorgehen des systematischen Absuchens des Konformationsraumes in 10 Grad-Schritten mit anschließender Selektion des energiegünstigsten Rotamers wurde als AAV 3 in den Anhang aufgenommen und für die folgenden *hits* ebenfalls angewendet.

Für B2B5T2_L00K10 zeigte sich, dass die optimale Bindungsenergie von -9.24 kcal/mol bei Torsionswinkeln von 110 Grad und 160 Grad erreicht wurde. Das Rotamer wurde mit B2B5T2_L00K10X110X160_14_33 bezeichnet und im Folgenden als Repräsentation für die Substitution K10 weiterverwendet.

Unter Verwendung der torsionsoptimierten RBR für K01, K06-K11, L02 und L04 ergaben sich beim *docking* nach AAV 2 die in Tabelle 15 dargestellten Energiewerte. In der Tabelle sind ebenfalls die jeweils eingestellten Torsionswinkel an den Substituenten mit freier Drehbarkeit aufgelistet.

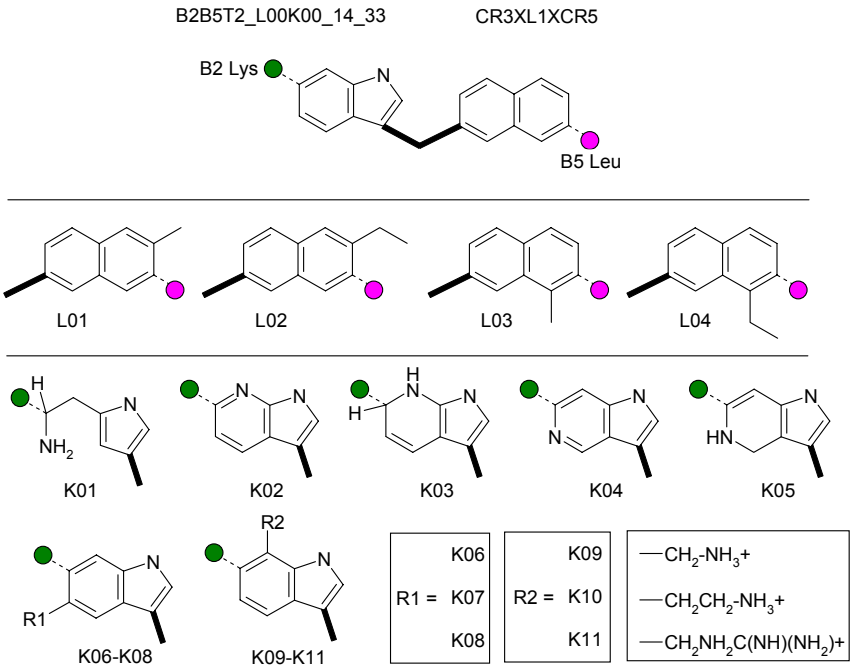


Abbildung 62: Die am Leitstrukturgerüst CR3XL1XCR5 durchgeführten *in silico*-Funktionalisierungen. Es wurden vier auf der Leucin-Seite und elf auf der Lysin-Seite durchgeführt. Die Energiewerte sind in Tabelle 15 aufgelistet.

Tabelle 15: Energien der *constrained dockings* der in Abbildung 8 dargestellten einfach funktionalisierten RBR des als B2B5T2_L00K00 bezeichneten Leitstrukturgerüstes CR3XL1XCR5. In der Spalte "Torsion" sind die nach AAV 3 ermittelten günstigsten Torsionswinkel der rotierbaren Bindungen der neu eingeführten funktionellen Gruppen aufgelistet.

Ligand	Torsion	Energie
L00K00		-7.28
L01K00		-7.48
L02K00	310	-7.79
L03K00		3.88
L04K00	220	-4.96
L00K01	220/40	-14.80
L00K02		NA
L00K03		-8.13

Ligand	Torsion	Energie
L00K04		NA
L00K05		-4.89
L00K06	320	-9.90
L00K07	310/10	-12.09
L00K08	320	-8.08
L00K09	100	-9.71
L00K10	110/160	-9.24
L00K11	100	-7.22

Bei den Leucin-Modifikationen waren die Energiewerte für L03 und L04 deutlich ungünstiger als die für L01 und L02, die sich wiederum nur wenig von dem Energiewert für das unmodifizierte L00 unterschieden. Wahrscheinlich lag eine ungünstige sterische

Wechselwirkung zwischen L03/L04 und dem Ser60 des CD4 vor (Abbildung 63). Für die Doppelmodifizierungen wurden sowohl L02 als auch L00 weiterverfolgt.

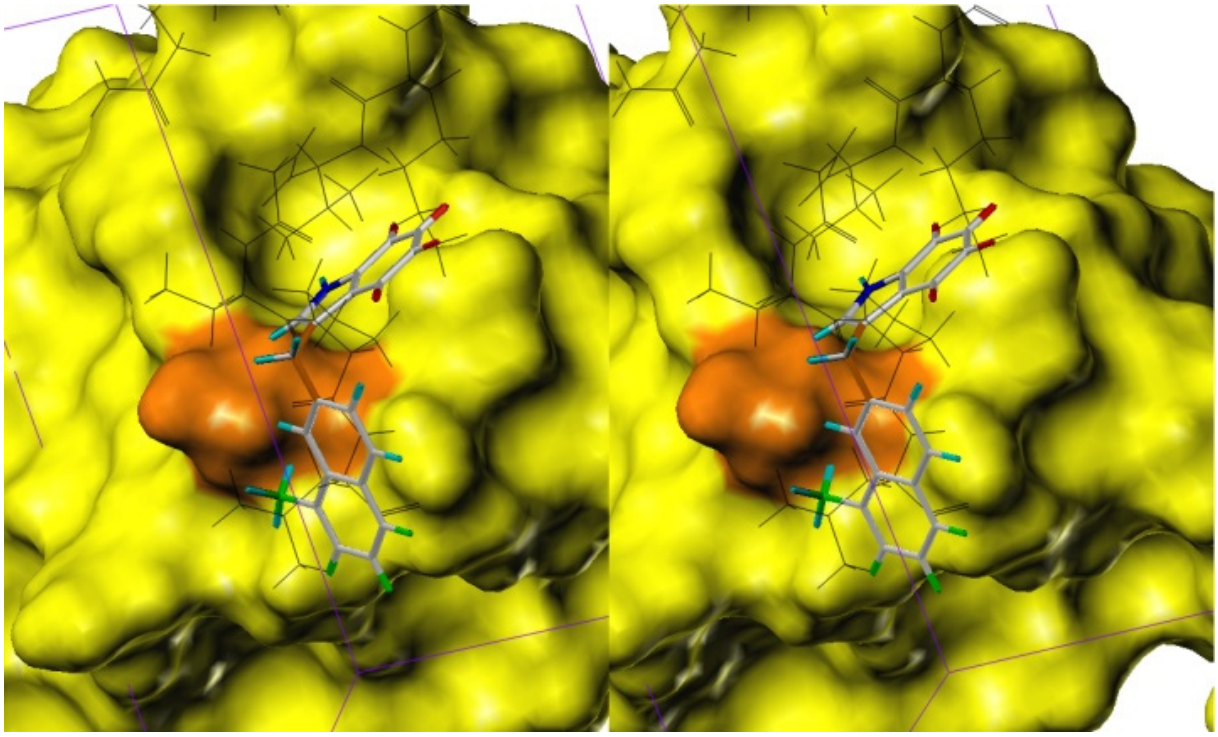


Abbildung 63: Die Funktionalisierungen L03 und L04 auf der Leucin-Seite (vgl. Abbildung 62) sind ungünstig, da die neu eingeführten Alkylgruppen zu nah an die Rezeptoroberfläche, insbesondere an die in dieser Abbildung orange eingefärbte Aminosäure Ser60, hineinragen. Die *crossed* Stereodarstellung zeigt das *constrained docking* des einfach funktionalisierten Liganden B2B5T2_L03K00. Die grün/cyan gefärbte Methylgruppe am zweiten Ringsystem CR5 in der Bildmitte kommt dem Ser60 des CD4 zu nah. Die Farben sind entsprechend Abbildung 35 und Seite V gewählt.

Bei den Lysin-Modifikationen war der *top scorer* K01, bei dem, ein Pyrrol-System anstatt des Indol-Ringsystems vorliegt. Zusätzlich zu dieser recht stark verändernden Substitution wurde ein konservativ substituiertes Indol-Derivat (K06-K11) weiterverfolgt. Hier zeigte sich, dass die jeweiligen Modifikationen an R1 günstigere Energiewerte als an R2 ergaben (K07 besser als K10, K08 besser als K11 und K06 besser als K09). Die Strukturen, bei denen der Indolring selbst modifiziert wurde (K02-K05), docken schlecht, bis auf K03.

L02 ist der *top scorer* für die Leucin-Modifikation. K01 brachte den besten Energiewert für die Lysin-Modifikation, aber da auch K07 einen sehr guten Energiewert erbrachte, wurden diese beiden Lysin-Modifikationen weiterverfolgt.

Anstatt eines doppelt modifizierten Liganden wurden in diesem Fall zwei Strukturen (B2B5T2_L00K01 und B2B5T2_L02K07) *in silico* konstruiert. Somit sollte sowohl der

Effekt der Ethylgruppe am L02 abgeschätzt werden, als auch eine unkonventionelle und eine konventionelle Modifikation der Lysin-Seite weiterverfolgt werden. Beide Leitstrukturgerüste wurden nach AAV 2 *constrained* gegen CD4 gedockt und ergaben Energiewerte von -14.80 kcal/mol (L00K01) und -12.41 kcal/mol (L02K07).

Im *constrained docking* des Leitstrukturgerüstes B2B5T2_L00K01 (Abbildung 64) wurde das Ankeratom des Liganden (in Abbildung 62 mit einem grünen Kreis markiert) nicht mehr exakt auf der Position des Ankers B2Lys des Dekapeptides (in Abbildung 64 mit einer grünen Kugel markiert) platziert, sondern 0.517 Å davon entfernt. In diesem Fall befindet sich das Ankeratom durch den Ringabbau in einer flexiblen Seitenkette. Daher wird durch die Seitenkettenkonformationssuche bereits dem *unconstrained docking* vorgegriffen und eine Anpassung der Position des Ankers an die Gegebenheiten der Bindungsregion erlaubt. Die Bindungsenergie des Liganden konnte durch diese Schritte im Vergleich zur Energie des unmodifizierten Liganden um über 7 kcal/mol verbessert werden.

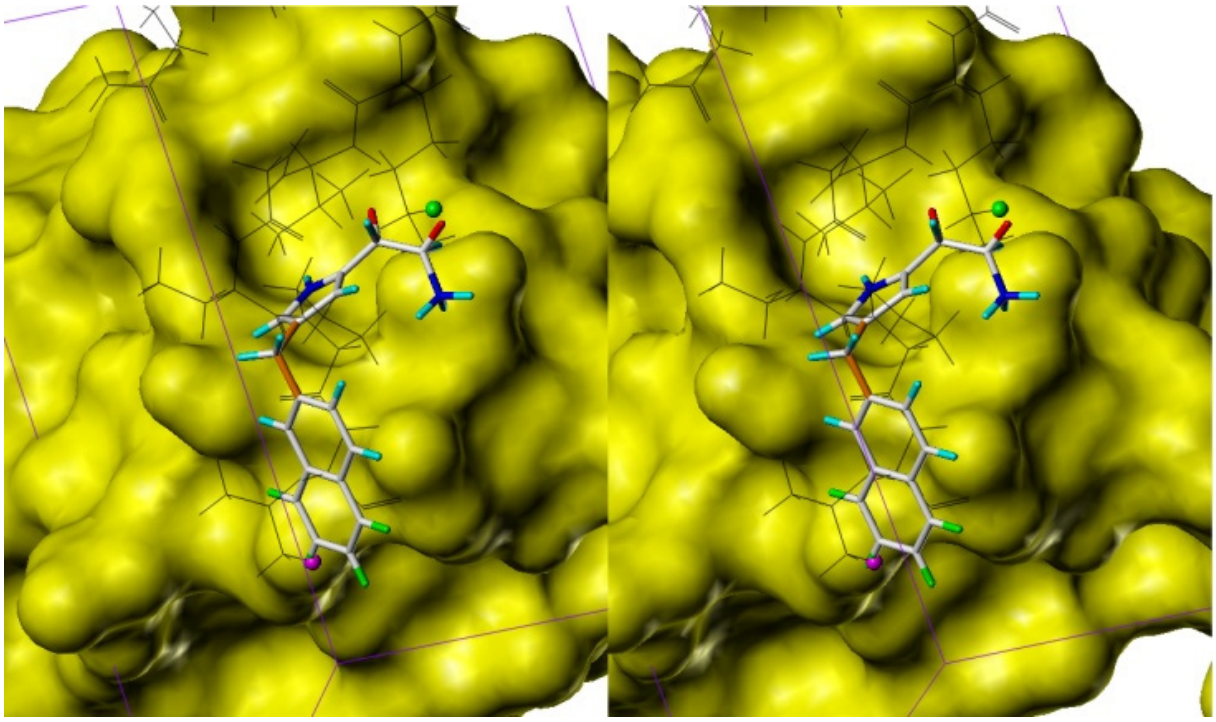


Abbildung 64: *Crossed* Stereodarstellung des *constrained docking* des Leitstrukturmotivs B2B5T2_L00K01, das eine Energie von -14.80 kcal/mol erreichte.

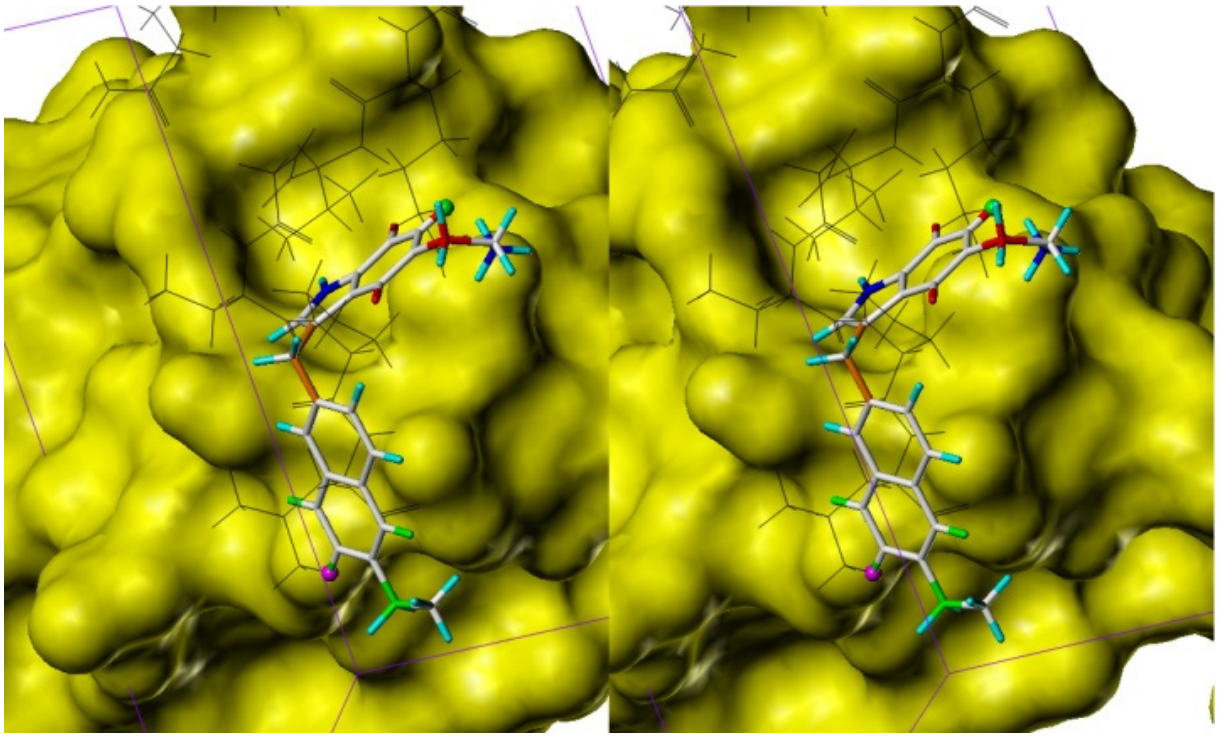


Abbildung 65: ***Crossed*** **Stereodarstellung des** ***constrained docking*** **des Leitstrukturmotivs B2B5T2_L02K07, das eine Energie von -12.41 kcal/mol erreichte.**

3.4.3 Funktionalisierung des Leitstrukturgerüstes B2B5T3

Für das RBR mit dem drittbesten Energiewert wurden vier Funktionalisierungen am B5Leu und vier am B2Lys durchgeführt. Für die Funktionalisierungen, bei denen eine oder mehrere der neuen Bindungen freie Drehbarkeit aufwiesen (L02, L04, K02, K03, K04), wurde wieder gemäß AAV 3 analysiert, welches Rotamer das günstigste war. Diese Rotamere wurden dann gemeinsam mit den einfacheren Funktionalisierungen L01, L03 und K01 nach AAV 2 gegen CD4 gedockt. Die Ergebnisse dieses *docking* sowie die günstigsten Torsionswinkel sind in Tabelle 16 aufgelistet.

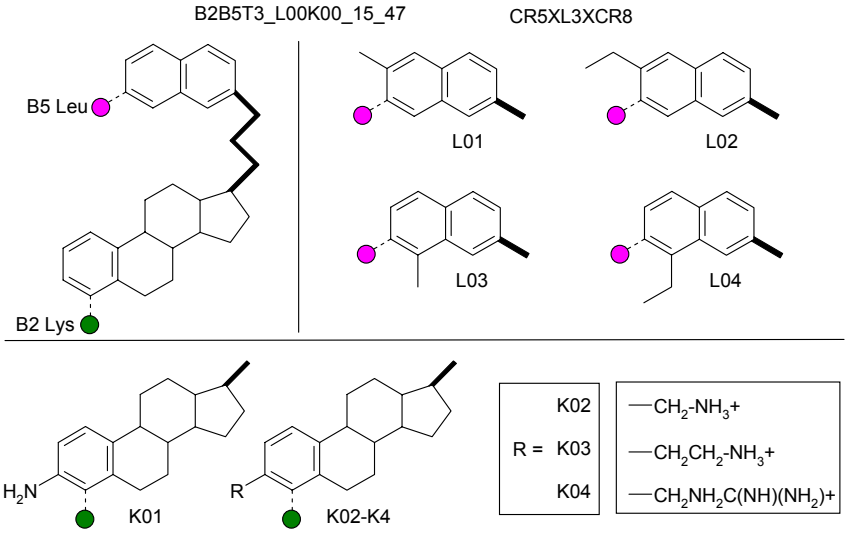


Abbildung 66: Am Leitstrukturgerüst B2B5T3 (CR5XL3XCR8) wurden vier *in silico*-Funktionalisierungen auf der Leucin-Seite und fünf auf der Lysin-Seite durchgeführt. Die Energiewerte sind in Tabelle 16 aufgelistet.

Tabelle 16: Energien der *constrained dockings* der in Abbildung 66 dargestellten einfach funktionalisierten RBR des Leitstrukturgerüstes CR5XL3XCR8. In der Spalte "Torsion" sind die nach AAV 3 ermittelten energiegünstigsten Torsionswinkel der rotierbaren Bindungen der neu eingeführten funktionellen Gruppen L02, L04, K02, K03 und K04 angegeben.

Ligand	Torsion	Energie
L00K00		-6.46
L01K00		-6.53
L02K00	170	-6.78
L03K00		-3.03
L04K00	250	0.12

Ligand	Torsion	Energie
L00K01		-11.84
L00K02	140	-13.02
L00K03	140/10	-14.33
L00K04	20	-7.90

Top scorer waren L02 (mit einem Torsionswinkel von 170 Grad) und K03 (mit Torsionswinkeln von 140 Grad und 10 Grad). Die Modifikationen L03 und L04 waren deutlich ungünstiger, da sie direkt in den Rezeptor hineinragten (Abbildung 67).

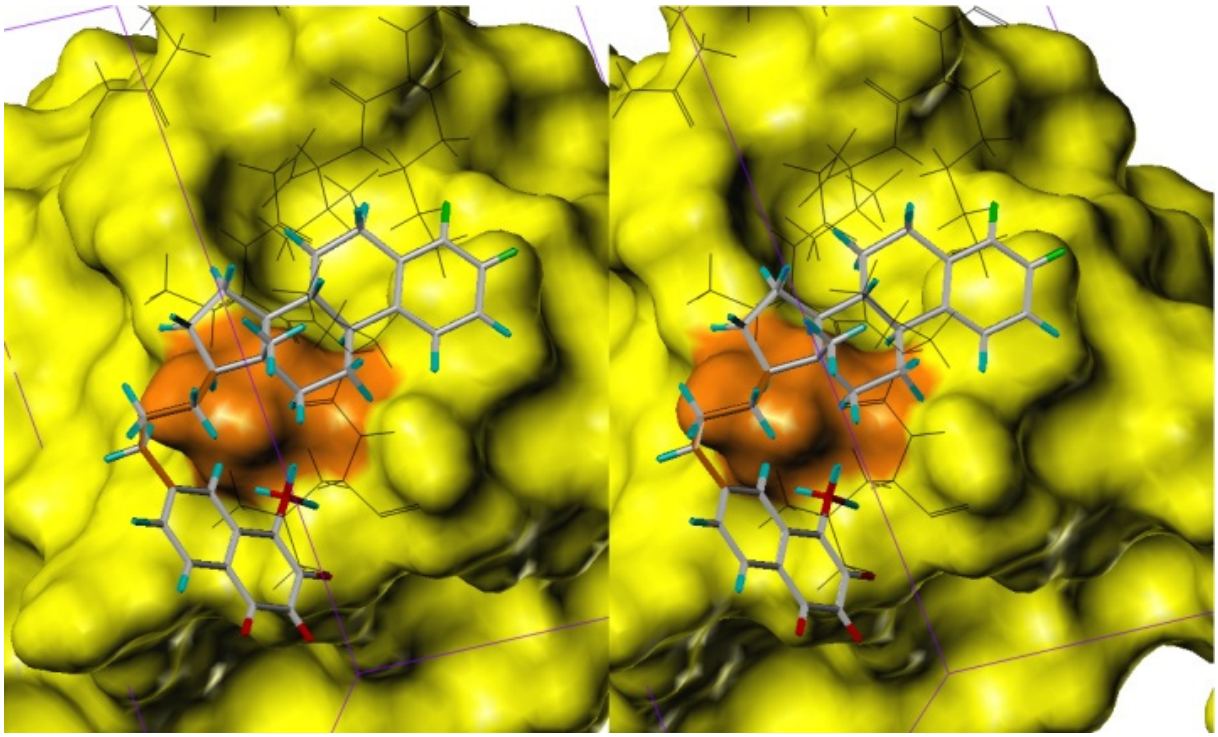


Abbildung 67: Die Funktionalisierungen L03 und L04 auf der Leucin-Seite (vgl. Abbildung 66) sind ungünstig, da die neu eingeführten Alkylgruppen zu nah an die Rezeptoroberfläche, besonders an die in dieser Abbildung orange eingefärbte Aminosäure Ser60 hineinragen. Die *crossed* Stereodarstellung zeigt das *constrained docking* des einfach funktionalisierten Liganden B2B5T3_L03K00. Die rot/cyan gefärbte Methylgruppe am ersten Ringsystem CR5 in der Bildmitte kommt dem Ser60 des CD4 zu nah. Farben entsprechend Abbildung 35 und Seite V.

Als doppelt modifizierter Ligand wurde B2B5T3_L02K03 konstruiert, wobei die für die Einzelmodifikationen optimierten Torsionswinkel eingestellt wurden.

Die Energiebewertung dieses Liganden nach AAV 2 durch *constrained docking* ergab -14.65 kcal/mol.

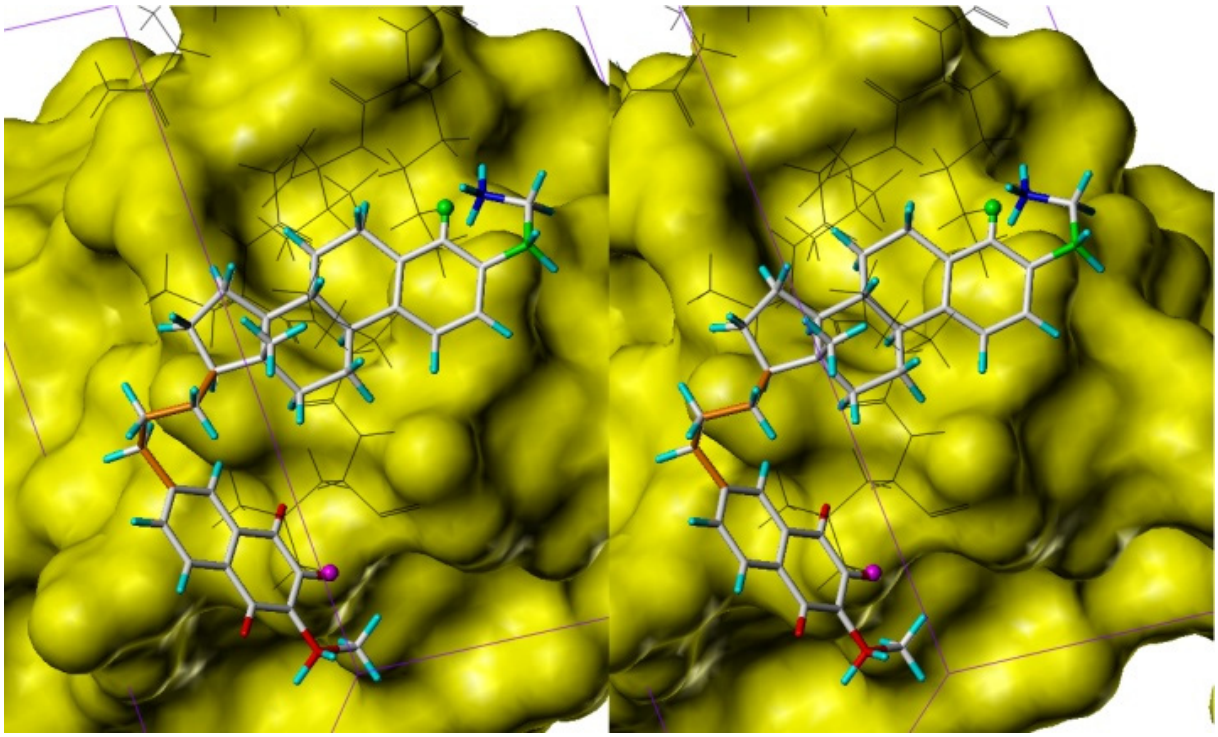


Abbildung 68: ***Crossed*** **Stereodarstellung des** ***constrained docking*** **des Leitstrukturmotivs B2B5T3_L02K03, das eine Energie von -14.65 kcal/mol erreichte. Die Abbildung zeigt, dass dieses Leitstrukturmotiv trotz der günstigen Bindungsenergie im** ***constrained docking*** **nur sehr bedingt zur Nachahmung des hier als schwarze Linien dargestellten Dekapeptides geeignet ist, da der große Ligand nur in einer engen Faltung die beiden als grüne und magentafarbene Kugeln dargestellten** ***critical spheres*** **B2Lys und B5Leu erreichen kann.**

In der Abbildung 68 ist zu erkennen, dass dieses Leitstrukturgerüst keineswegs die kürzeste Verbindung zwischen den beiden Punkten B2Lys und B5Leu überbrückt, sondern einen weiten Bogen beschreibt. Im späteren *unconstrained docking* (Abschnitt 3.5.5) zeigte sich, dass dieses RBR zur Nachahmung der Bindung des Lys und Leu des NMWQKVGTPL eher zu groß war. In allen durchgeführten *dockings* kam es zu Anordnungen in einem weiten Bogen auf der Oberfläche mit in jedem Fall ungünstiger innerer Energie des Liganden.

3.4.4 Funktionalisierung des Leitstrukturgerüstes B2B5T4

Für den *hit* B2B5T4 wurden zwei Funktionalisierungen am B5Leu und vier am B2Lys durchgeführt. Die günstigsten Rotamere wurden nach AAV 3 ausgewählt. Im Folgenden sind die nach AAV 2 ermittelten Energiewerte aufgeführt, die anschließend für das *ranking* der möglichen Funktionalisierungen genutzt wurden.

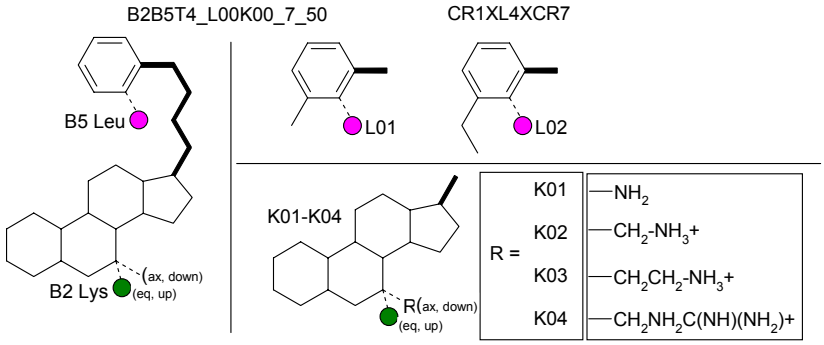


Abbildung 69: Die Einzel-Funktionalisierungen die *in silico* am Leitstrukturgerüst B2B5T4 durchgeführt wurden. Gezeigt sind zwei Modifikationen der Leucin-Seite und vier der Lysin-Seite. Die Energien der *dockings* folgen in Tabelle 17.

Tabelle 17: Energien und günstigste Torsionswinkel der *constrained dockings* der in Abbildung 69 dargestellten einfach funktionalisierten RBR des Leitstrukturgerüstes CR1XL4XCR7.

Ligand	Torsion	Energie
L00K00		-6.21
L01K00		-6.22
L02K00	310	-6.43

Ligand	Torsion	Energie
L00K01		-8.43
L00K02	200	-8.56
L00K03	200/10	-10.53
L00K04	210	-6.73

Top scorer waren hier K03 (Torsionswinkel 200 Grad und 10 Grad) und L02 (Torsionswinkel 310 Grad). Als doppelt modifizierter Ligand wurde B2B5T4_L02K03 mit entsprechenden Torsionswinkeln *in silico* konstruiert. Das *docking* dieses Liganden nach AAV 2 ergab eine Energie von -10.65 kcal/mol.

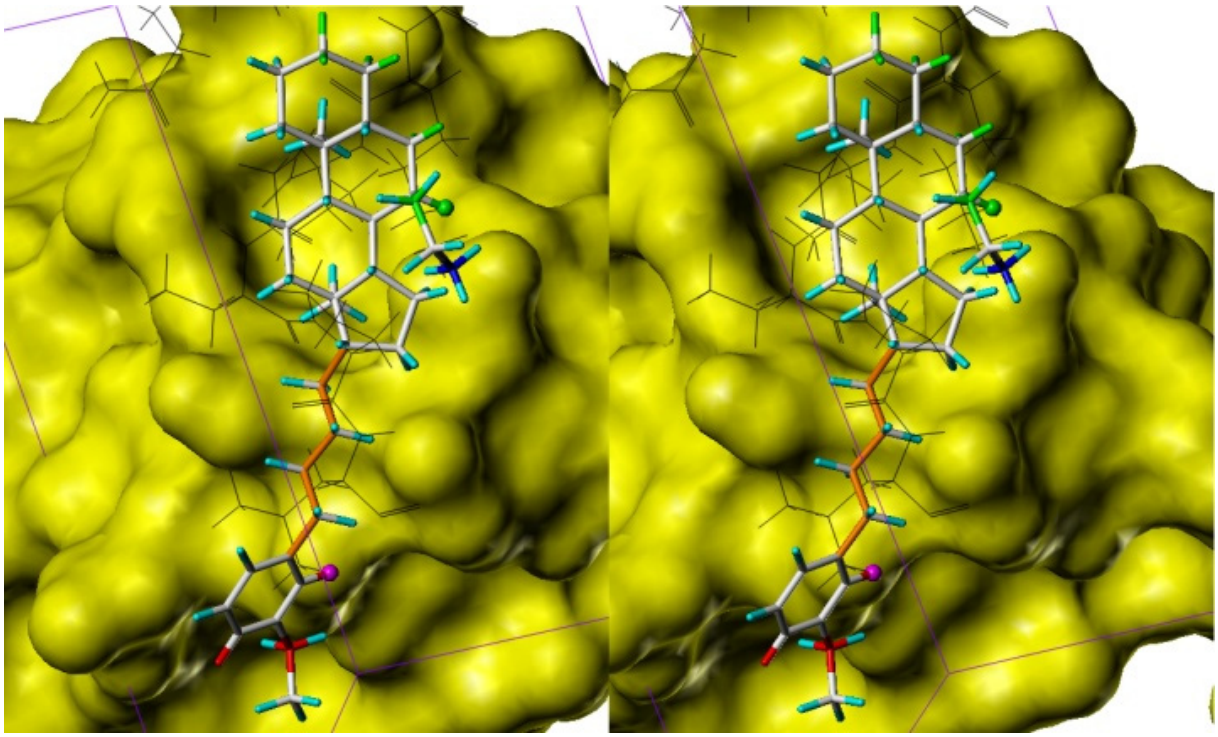


Abbildung 70: ***Crossed*** **Stereodarstellung des** ***constrained docking*** **des Leitstrukturmotivs B2B5T4_L02K03, das eine Energie von -10.65 kcal/mol erreichte. Die Torsionswinkel der drehbaren Bindungen der neu eingeführten Gruppen entsprechen den nach AAV 3 ermittelten Werten (310 Grad bei L02, 200 Grad und 10 Grad bei K03).**

Die Abbildung 70 zeigt die sterische Hinderung der 9-Position des Steroides bzw. der dort verknüpften Aminoethylgruppe. Im späteren freien *docking* (Abschnitt 3.5.6) kam es nie zu einer günstigen Coulomb-Wechselwirkung mit dem Asp63 des CD4, ohne dass es gleichzeitig an vielen Stellen zu Energieverlusten aufgrund geringer Abstände zwischen Atomen des sperrigen Steroid-Gerüstes und des Rezeptors kam.

3.4.5 Funktionalisierung des Leitstrukturgerüstes B2B5T5

Am Liganden B2B5T5 wurden jeweils vier verschiedene Modifikationen am Leucin und am Lysin durchgeführt. Für die Funktionalisierung mit frei drehbaren Bindungen wurden die optimalen Torsionswinkel nach AAV 3 bestimmt. Die Einzel-Modifikationen wurden nach AAV 2 energiebewertet. Aus diesen Energiewerten wurde ein *ranking* erstellt.

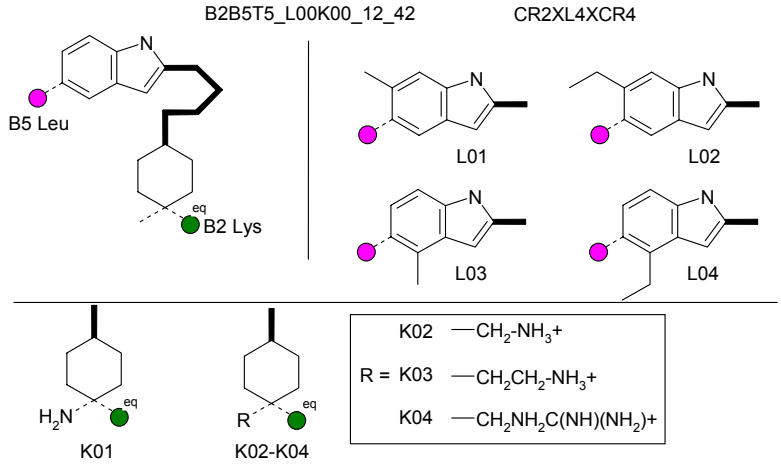


Abbildung 71: Am als B2B5T5 bezeichneten Leitstrukturgerüst CR2XL4XCR4 wurden je vier *in silico*-Funktionalisierungen zur Nachahmung des Leucins und des Lysins durchgeführt und nach AAV 2 und AAV 3 energiebewertet.

Tabelle 18: Energien und günstigste Torsionswinkel der in Abbildung 71 dargestellten Einzelmodifikationen des Leitstrukturgerüstes B2B5T5_L00K00.

Ligand	Torsion	Energie
L00K00		-5.90
L01K00		854.28
L02K00	210	80.86
L03K00		-6.05
L04K00	60	-6.45

Ligand	Torsion	Energie
L00K01		41.66
L00K02	230	106.28
L00K03	230/230	106.79
L00K04	150	2287.59

Nach AAV 2 bewertet erwiesen sich alle Lysin-Modifikationen als sehr ungünstig (positive Bindungsenergien von 42 bis 2000 kcal/mol). Dies lag daran, dass der Cyclohexylring parallel zur Oberfläche des Asp63 des CD4 lag (Abbildung 72).

Jede Vergrößerung des sterischen Anspruchs des Liganden in der axialen 4-Position führte zu einer sterischen Behinderung. Es kann davon ausgegangen werden, dass die Bindungsenergie in der Realität dadurch nicht so stark verschlechtert wird, denn das ganze Gerüst des CR2XL4XCR4 ist sehr flexibel und der Cyclohexylring könnte sich wahrscheinlich so weit aus der Ebene drehen, dass ein Substituent an seiner Unterseite nicht mehr diese ungünstige Wechselwirkung hervorrufen würde. Da im Folgenden nach AAV 4 (*unconstrained docking*) energiebewertet werden sollte, wurde K01, der sterisch am wenigsten anspruchsvolle geladene Substituent, weiterverfolgt.

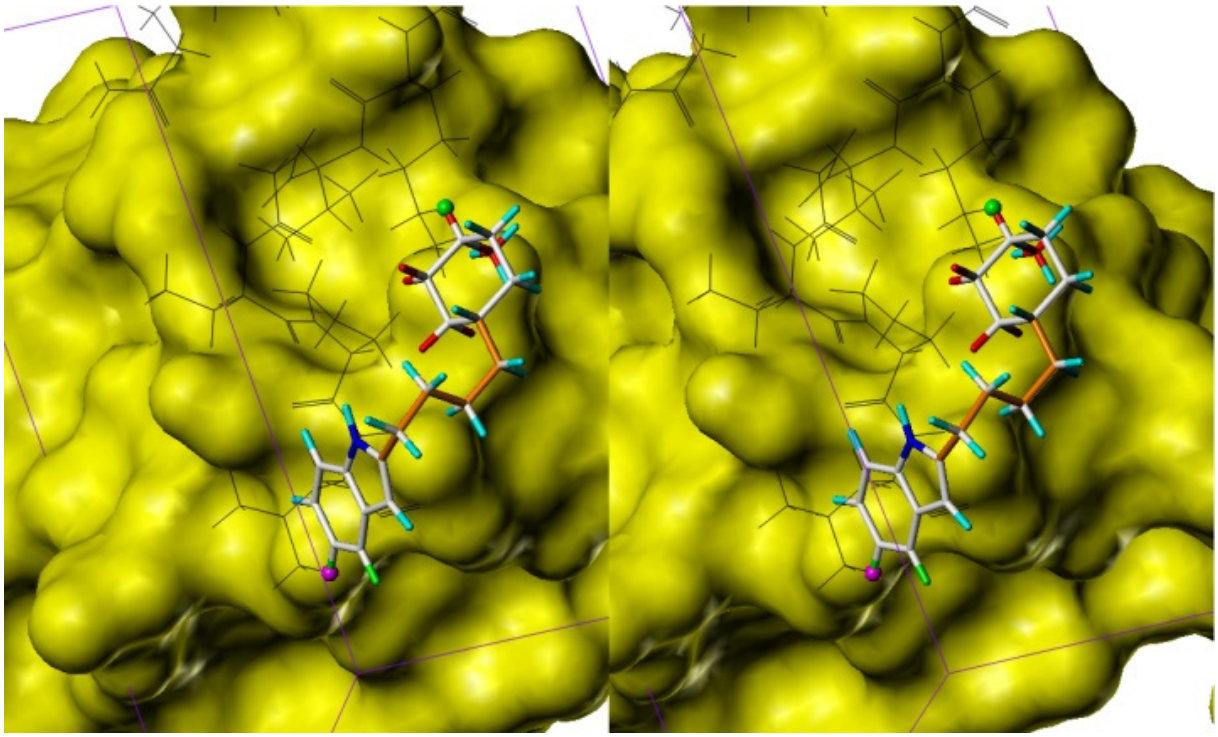


Abbildung 72: ***Crossed*** **Stereodarstellung des** ***constrained docking*** **des einfach modifizierten Leitstrukturgerüstes B2B5T5_L00K01. Die Energie ist mit 854.28 kcal/mol extrem hoch, da die neu eingeführte Methylgruppe am Cyclohexylring CR2 (oben rechts im Bild) ungünstig mit der Aminosäure Asp63 des CD4 (Oberfläche in gelb dargestellt) wechselwirkt.**

Bei den Leucin-Modifikationen war der *top scorer* L04. Dieses bindet interessanterweise außerhalb des Bereiches, in dem das Leu des NMWQKVGTPL bindet (Abbildung 48) und ragt in eine benachbarte Bindungstasche hinein. Solche unerwarteten *dockings* bieten immer wieder interessante Ansatzpunkte, um im Rahmen des rationalen Wirkstoffdesigns neue Wechselwirkungen mit dem Rezeptorprotein auszunutzen.
Für das weitere Vorgehen wurde B2B5T5_L04K01 *in silico* konstruiert.

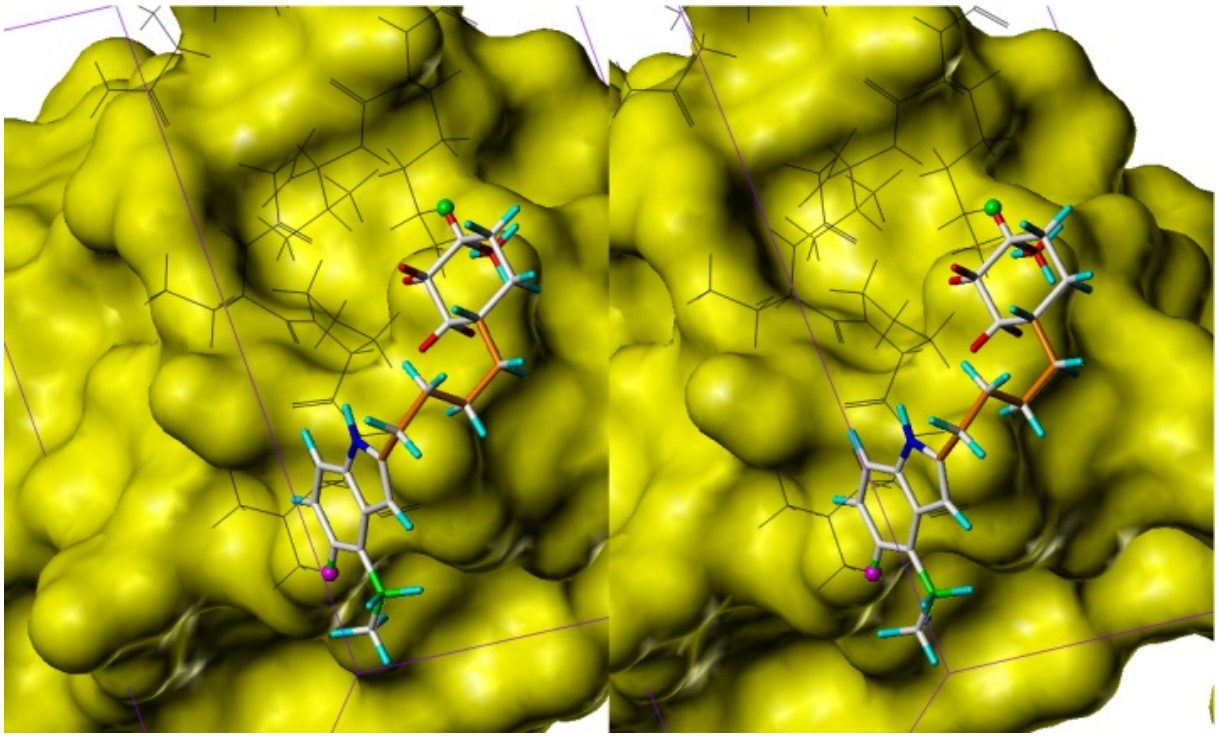


Abbildung 73: *Crossed* Stereodarstellung des *constrained docking* des Leitstrukturmotives B2B5T5_L04K01. Dieses *docking* ergab nur eine Energie von +41.32 kcal/mol, da die neu eingeführte Methylgruppe (rot/cyan dargestellt, im Bild oben rechts) und die Aminosäure Asp63 des CD4 ungünstig wechselwirken (vgl. Abbildung 92 für die günstigeren Ergebnisse des *unconstrained docking*). Unten in der Bildmitte ist zu sehen, wie die neu eingeführte Ethylgruppe (weiß/grün/cyan) in eine andere Region des Rezeptors (Oberfläche des CD4 in gelb dargestellt) ragt, als die entsprechende Isopropylgruppe des Leucins des Dekapeptides (schwarze Linien).

Die Energiebewertung dieses Liganden nach AAV 2 durch *constrained docking* ergab +41.32 kcal/mol. Diese war positiv, weil im *constrained docking* der Substituent am Cyclohexylring ungünstig mit der Oberfläche des Rezeptors wechselwirkt. Es wurde davon ausgegangen, dass diese ungünstige Wechselwirkung durch Drehung des Ringes aus seiner Position aufgehoben werden könnte. Dies sollte sich im anschließenden *unconstrained docking* bestätigen.

3.5 Unconstrained docking der funktionalisierten Leitstrukturmotive

Das *constrained docking* nach AAV 2, mit dem das primäre *ranking* der *hits* durchgeführt und das auch bei der Funktionalisierung der RBR für die Energiebewertung benutzt wurde, fixiert die jeweils zwei Ankeratome der Liganden an ihren Positionen. Dies wurde im Rahmen dieser Arbeit durchgeführt, um zu zeigen, dass eine Optimierung der primären *hits* auch ohne Vorhandensein einer Rezeptorstruktur möglich ist. Im realen biologischen System existiert

kein Potential, dass die Liganden exakt an dieser Position hält. Daher sind die Energiewerte, die das *constrained docking* liefert, auch lediglich als grobe Anhaltspunkte für die tatsächliche Bindungsenergie zu sehen. Um die Ergebnisse dieses Prozesses zu überprüfen, waren weitere Studien notwendig.

Im Verlauf des *rational drug design* ohne Rezeptorstruktur wären nun *in vitro*-Assays durchgeführt worden, um die prognostizierten Bindungseigenschaften der auf diesem Wege entwickelten Liganden zu überprüfen. Da in diesem speziellen Fall aber eine Rezeptorstruktur (vgl. Abschnitt 3.2.1) vorlag, wurden die Bindungseigenschaften der optimiert funktionalisierten RBR gemäß AAV 4 durch *unconstrained docking* an CD4 ermittelt.

3.5.1 Vorbereitung des *unconstrained docking*

Bei ersten Versuchen die in Abschnitt 3.4 optimierten Leitstrukturmotive ohne die Verwendung von "ligand_centers" (Punkte an den Ankern B2 und B5, die DOCK nutzt, um den Liganden exakt zu positionieren, vgl. Abschnitt 1.7.4) an den Rezeptor CD4 zu docken, waren die Liganden von DOCK weit über die gesamte Oberfläche des Rezeptors verteilt angeordnet worden. Sie befanden sich dabei unter anderem in Regionen, die bei der CD4-GP120-Interaktion durch andere Aminosäuren des GP120 belegt sind (besonders im Bereich Asn_{425} / Trp_{427}), und auch in Bereichen, die außerhalb der GP120-CD4-Interaktionsfläche liegen. Dies zeigte, dass die für das *constrained docking* definierten Abmessungen der Bindungsregion (4 Å um das Dekapeptid in jede Richtung) für das *unconstrained docking* zu groß waren. Auch das Fehlen einer tiefen Bindungstasche, welche die Platzierung der Liganden allein durch ihre Geometrie hätte lenken können, trug sicher zu der weiten Verteilung der Liganden bei.

Für die neue Definition der Bindungsregion wurden einige der unter 3.3.3 erzeugten Dateien verwendet, so der PDB- und der mol2-Rezeptor und die auf dem Rezeptor erzeugte Oberfläche. Zur Definition der Rezeptor-*spheres* wurde wieder das Programm SPHGEN verwendet. Aus dreien der erzeugten Cluster von *spheres* wurden insgesamt 21 *spheres* ausgewählt, die im Bereich der Bindung der Aminosäuren Lys_{429} bis Leu_{125} lagen. Der Rezeptor-*sphere cluster* ist in Abbildung 74 gezeigt.

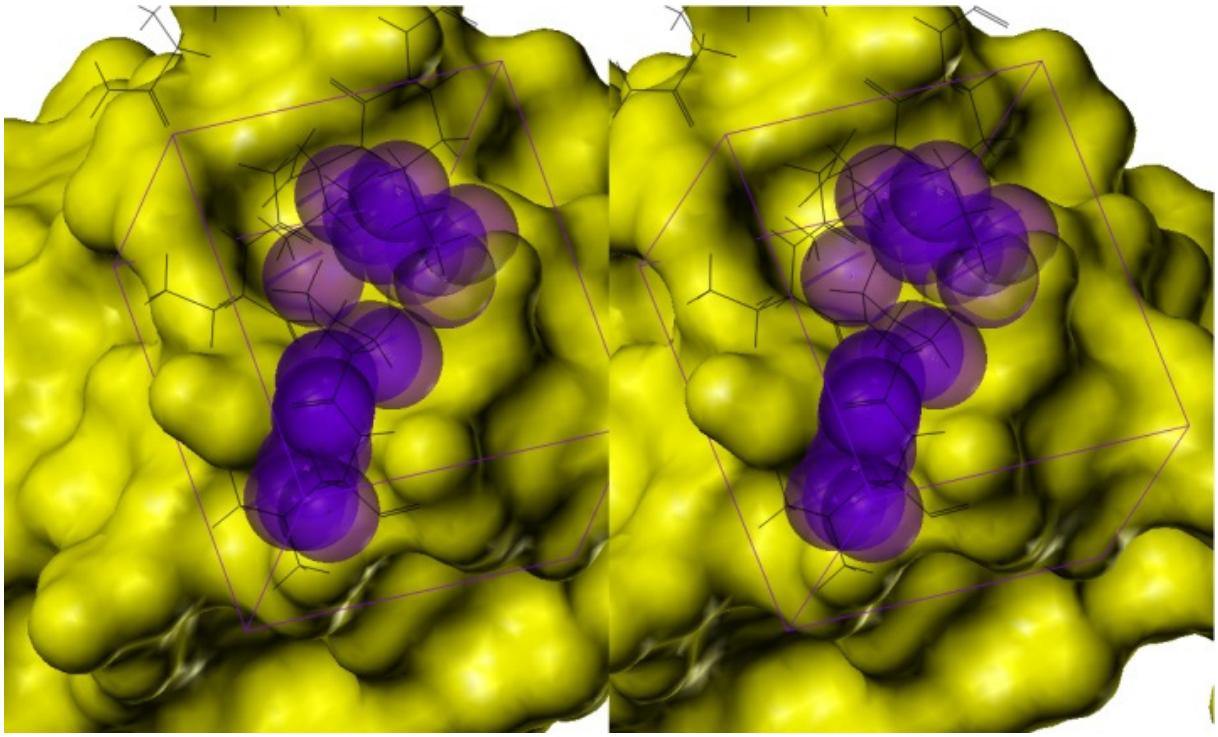


Abbildung 74: *Crossed* Stereodarstellung der 21 als violette Kugeln dargestellten *spheres* des *sphere cluster* für das *unconstrained docking*. Die Oberfläche des CD4 ist gelb, die *box* für das *unconstrained docking* ist als violetter Kasten und das Dekapeptid NMWQKVGTPL ist als schwarze Linien dargestellt. Im Vergleich mit Abbildung 52 ist deutlich zu erkennen, dass die *box* für das *unconstrained docking* verkleinert wurde.

Für die Erzeugung des *scoring grid* wurden überwiegend die Einstellungen aus Abschnitt 3.3.3 verwendet, lediglich der "*extra margin to also be enclosed*" wurde auf von 4 auf 2 Å verringert. Da die Punkte dieses *grid* nicht exakt an den gleichen Stellen lagen wie die des *grid*, das für das *constrained docking* in Abschnitt 3.3 und 3.4 verwendet worden war, werden identische Orientierungen von Liganden bei der Bewertung mit den beiden unterschiedlichen *grids* energetisch etwas verschieden beurteilt. Um zu klären, welchen Einfluss die Verwendung des neuen *grid* auf die Bindungsenergien hat, wurden die unmodifizierten TOP5 aus Abschnitt 3.3 zum Vergleich mit dem neuen *grid* gemäß AAV 2 energiebewertet. Die Energiewerte sind im Vergleich zu denen, die mit dem alten *grid* erzeugt wurden, in Tabelle 19 aufgelistet. Da die Unterschiede mit maximal 5.6 % relativ gering waren, wurden sie im Folgenden vernachlässigt.

Tabelle 19: Vergleich zwischen Energien, die für identische Orientierungen der unmodifizierten Leitstrukturgerüste (vgl. Abbildung 58) mit den beiden im Verlauf dieser Arbeit verwendeten *grids* (vgl. "*force field on a grid*", Abschnitt 1.7.4) ermittelt wurden. Da sich die Energien durch den Wechsel des *grid* nur um maximal 5.6 % änderten, wurde dieser Effekt im Folgenden vernachlässigt.

	Energie erstes *grid* [kcal/mol]	Energie zweites *grid* [kcal/mol]	Energie-differenz [kcal/mol]	Energie-differenz [%]
B2B5T1	-10.05	-9.50	-0.55	5.5
B2B5T2	-7.70	-7.82	0.12	1.6
B2B5T3	-6.84	-6.46	-0.38	5.6
B2B5T4	-5.99	-6.21	0.22	3.7
B2B5T5	-5.92	-5.90	-0.02	0.3

3.5.2 DOCK-Parameter

Um den realen Bedingungen der Bindung möglichst nahe zu kommen, wurden die Liganden für das *docking* als flexibel angesehen. Im Rahmen einer "*anchor search*" selektierte der DOCK-Algorithmus jeweils zunächst ein möglichst großes und starres Ringsystem, das dann möglichst energiegünstig in der Bindungsregion gedockt wurde. Sobald hierfür eine günstige Orientierung gefunden worden war, wurde der Rest des Liganden angeknüpft und entsprechend der energetischen Gegebenheiten der Bindungsregion flexibel angeordnet. Für die endgültige Platzierung des Liganden führte DOCK noch eine Minimierung über maximal 100 Schritte durch.

Beim *unconstrained docking* der optimierten Leitstrukturmotive zeigte sich, dass dem Parameter random_seed, einer Zahl, mit der der DOCK-Zufallszahlengenerator initialisiert wird,[126] große Bedeutung zukam. Bei Angabe verschiedener Werte für dieses random_seed wurde bei sonst identischen Parametern der gleiche Ligand komplett verschieden gedockt. In Abbildung 75 ist das exemplarisch für die beiden random_seeds 02 und 03 beim *unconstrained docking* von B2B5T2_L04K07 gezeigt.

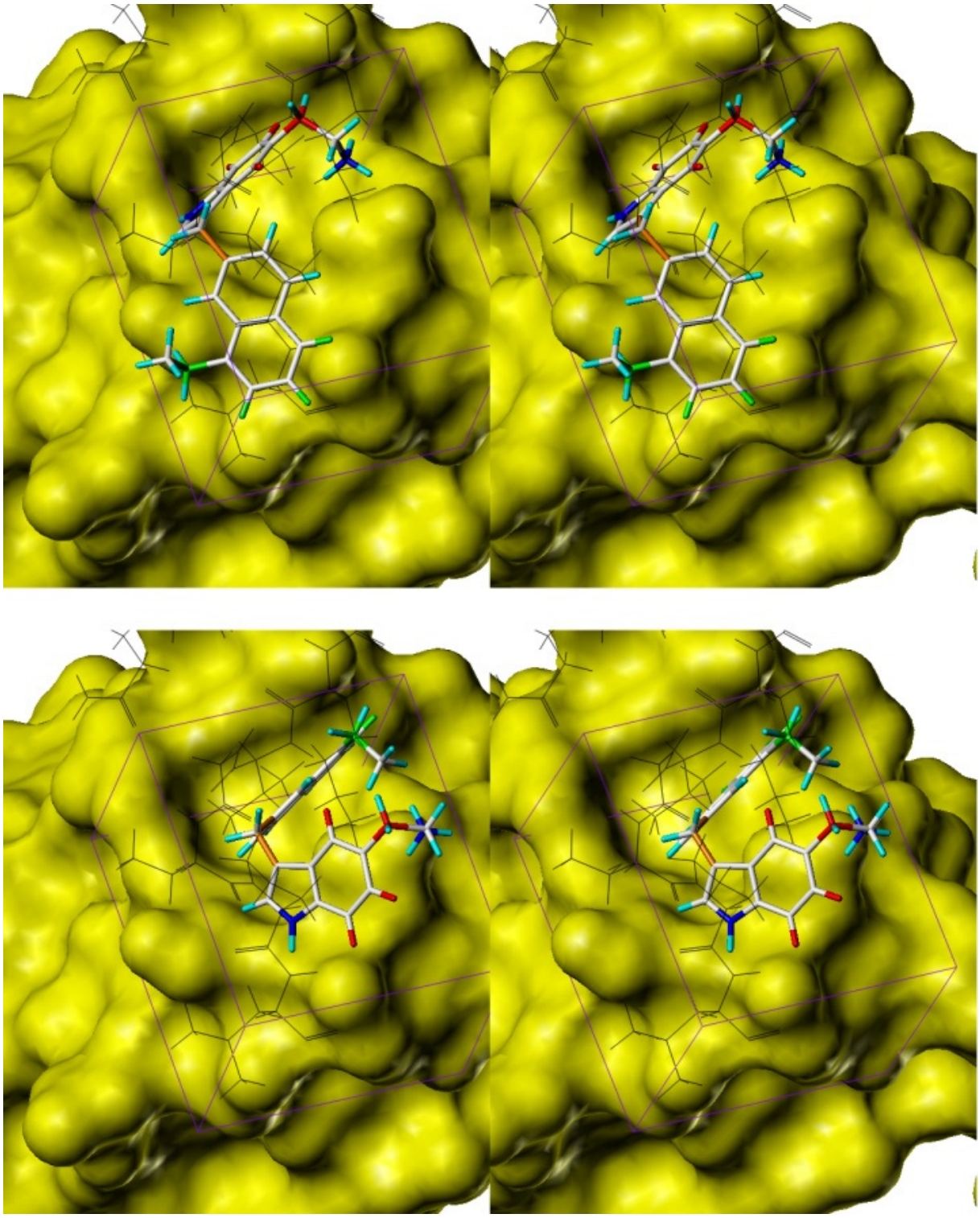


Abbildung 75: ***Crossed*** **Stereodarstellungen gemäß Abbildung 35 und Seite V. Exemplarisch ist der Einfluss des Parameters random_seed auf das Ergebnis des** ***unconstrained dockings*****, am Liganden B2B5T2_L04K07 gezeigt. Bis auf das random_seed sind die Parameter der DOCK-Läufe, die zu der im oberen bzw. unteren Bild gezeigten Orientierung geführt haben, identisch. Die Energien der gezeigten** ***dockings*** **betragen -14.18 kcal/mol. bzw. -14.79 kcal/mol.**

Es wurde entsprechend eines Vorschlages aus dem DOCK-Handbuch[163] vorgegangen und eine große Anzahl von *dockings* unter Variation des random_seed durchgeführt. Hierfür wurde für jeden zu dockenden Liganden ein Skript geschrieben, dass das *docking* dieses Liganden unter Variation des random_seed-Parameters durchführte. Es wurden jeweils die Werte für das random_seed von 1 bis 30 genutzt. Exemplarisch für die Skripte befindet sich dock_B2B5T5_L04K01_random_seed_variation.sh im Anhang 7.3.24. Alle auf diesem Wege erzeugten *dockings* mit den günstigsten Energiewerten wurden visuell inspiziert.

Nicht in allen Fällen wurde die gedockte Orientierung mit dem günstigsten Energiewert verwendet. Teilweise wurden bei diesen einzelne günstige Wechselwirkungen mit dem Rezeptor nicht mehr realisiert, sondern der Ligand war zum Beispiel mit nur einer Seite an den Rezeptor gebunden. Bei anderen *dockings* befand sich der Ligand in einem etwas anderen Bereich als das Dekapeptid. Solche Orientierungen können im Verlauf des *rational drug designs* wertvolle Hinweise auf weitere günstige Interaktionsmöglichkeiten liefern. Auf die Auswertung solcher Hinweise wurde in diesem Fall jedoch verzichtet, da hier das Ziel nicht primär die Entwicklung neuer Liganden für das CD4, sondern ein *proof of concept* der Methode war. Dass solche unvorhergesehenen Orientierungen auftreten, war zu erwarten, da die Bindung hier nicht innerhalb einer Bindungstasche mit hohen Potentialgrenzen sondern eher in einer Art flachen Bindungsmulde stattfindet.

Im Folgenden ist für jeden Liganden die ausgewählte Konformation ("bestes *docking*") gezeigt und angegeben, mit welchem Wert für random_seed sie erzeugt wurde. Der Vollständigkeit halber ist ebenfalls das random_seed angegeben, mit dem das energiegünstigste *docking* erreicht wurde.

3.5.3 *Unconstrained docking* des Leitstrukturmotives B2B5T1_L04K05

Für den Liganden B2B5T1_L04K05 wurde mit den in Tabelle 26 angegebenen DOCK-Parametern gemäß AAV 4 *docking* unter Variation des random_seed durchgeführt. Die Ergebnisse sind in Tabelle 20 und Abbildung 77 dargestellt.

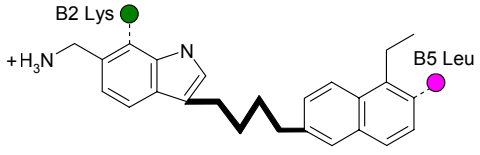


Abbildung 76: Das Leitstrukturmotiv B2B5T1_L04K05.

Tabelle 20: Energiewerte des *unconstrained docking* des Leitstrukturmotives B2B5T1_L04K05. Das *docking* mit der günstigsten Energie wurde mit einem random_seed von 29 erreicht. Diese Orientierung wich jedoch weit von dem Bereich des Dekapeptides NMWQKVGTPL ab. Die beste Orientierung des *docking* mit einem random_seed von 04 erreichte mit -16.26 kcal/mol eine ähnlich günstige Energie.

B2B5T1_L04K05	random_seed	Energie [kcal/mol]
bestes *docking*	04	-16.26
energiegünstigstes *docking*	29	-17.00

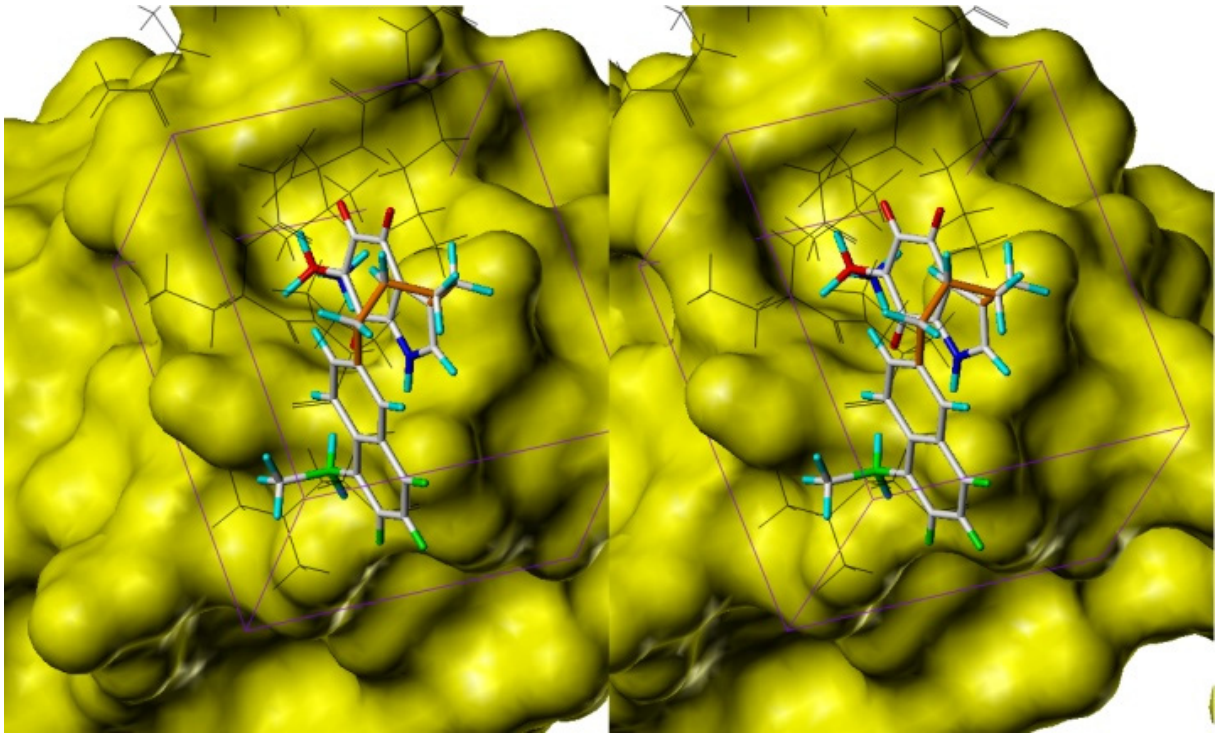


Abbildung 77: *Crossed* Stereodarstellung der günstigsten Orientierung des *unconstrained docking* von B2B5T1_L04K05, erzeugt mit einem Wert für random_seed von 04. Diese Orientierung hat eine Energie von -16.26 kcal/mol.

Um einen Überblick über die Energiewerte der einzelnen Optimierungsstufen des Leitstrukturgerüstes B2B5T1 zu geben, sind diese in Abbildung 78 aufgetragen. Nachdem die Energie durch die Kombination der beiden günstigsten Einzelmodifikationen im *constrained docking* erwartungsgemäß mit -16.85 kcal/mol besser als die der Einzelmodifikationen wird, verschlechtert sie sich im letzten Schritt, dem *unconstrained docking*, unerwartet etwas und steigt auf -16.26 kcal/mol. Diese geringe Verschlechterung der Bindungsenergie um 0.6 kcal/mol liegt jedoch innerhalb der durch die Verwendung des neuen *grid* entstandenen Abweichungen.

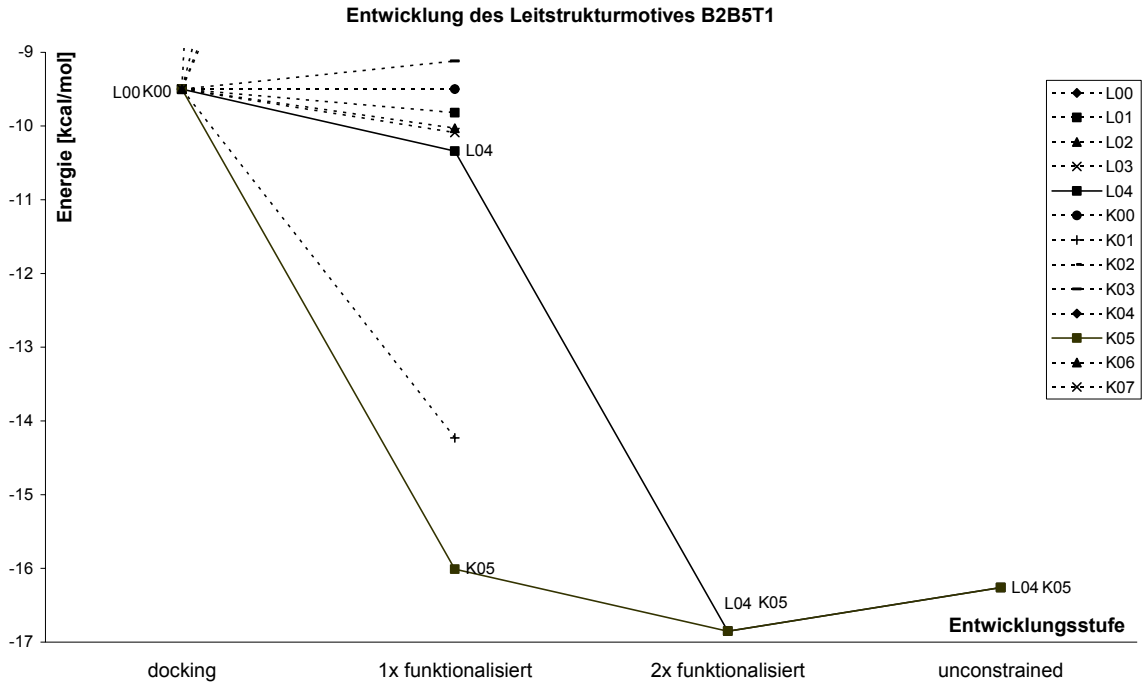


Abbildung 78: Übersicht über die Entwicklung des Leitstrukturmotives B2B5T1. Auf der x-Achse ist die jeweilige Entwicklungsstufe aufgetragen. Sie führt von dem ersten *docking* des unmodifizierten Liganden L00K00 (ganz links), über die vielen einzelnen Energiewerte der Einzelmodifikationen ("1x funktionalisiert"), das *constrained docking* der Kombination der beiden *top scorer* L04 und K05 ("2x funktionalisiert") bis zum *unconstrained docking* dieses doppelt modifizierten Liganden mit random_seed 04 (vgl. Tabelle 20). Die Energie verbessert sich vom unmodifizierten über die Einzelmodifikationen bis hin zum doppelt modifizierten Leitstrukturmotiv. Der nur geringe Anstieg der Energie im letzen Schritt von -16.85 kcal/mol auf -16.26 kcal/mol zeigt wie gut die Methode der Optimierung des durch das *constrained docking* fixierten Liganden hier funktioniert hat.

3.5.4 *Unconstrained docking* der Leitstrukturmotive B2B5T2_L00K01 und B2B5T2_L02K07

Bei diesem *hit* war entschieden worden, zwei doppelt funktionalisierte Liganden zu konstruieren (siehe Abschnitt 3.4.2). Im *constrained docking* hatte B2B5T2_L00K01 einen Energiewert von -14.80 kcal/mol und B2B5T2_L02K07 einen Wert von -12.41 kcal/mol ergeben.

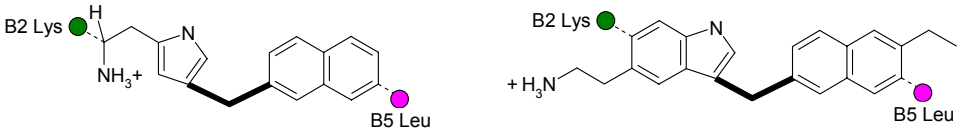


Abbildung 79: Die Leitstrukturmotive B2B5T2_L00K01 (links) und B2B5T2_L02K07 (rechts).

Im *unconstrained docking* nach AAV 4 ergaben sich Energiewerte von -18.04 kcal/mol (L00K01) und -16.99 kcal/mol (L02K07), in beiden Fällen deutliche Verbesserungen der Energie um drei bzw. um vier kcal/mol.

Tabelle 21: Energie und random_seed der günstigsten *unconstrained dockings* des doppelt modifizierten Leitstrukturmotives B2B5T2_L00K01.

B2B5T2_L00K01	random_seed	Energie [kcal/mol]
bestes *docking*	01	-18.04
energiegünstigstes *docking*	26	-18.09

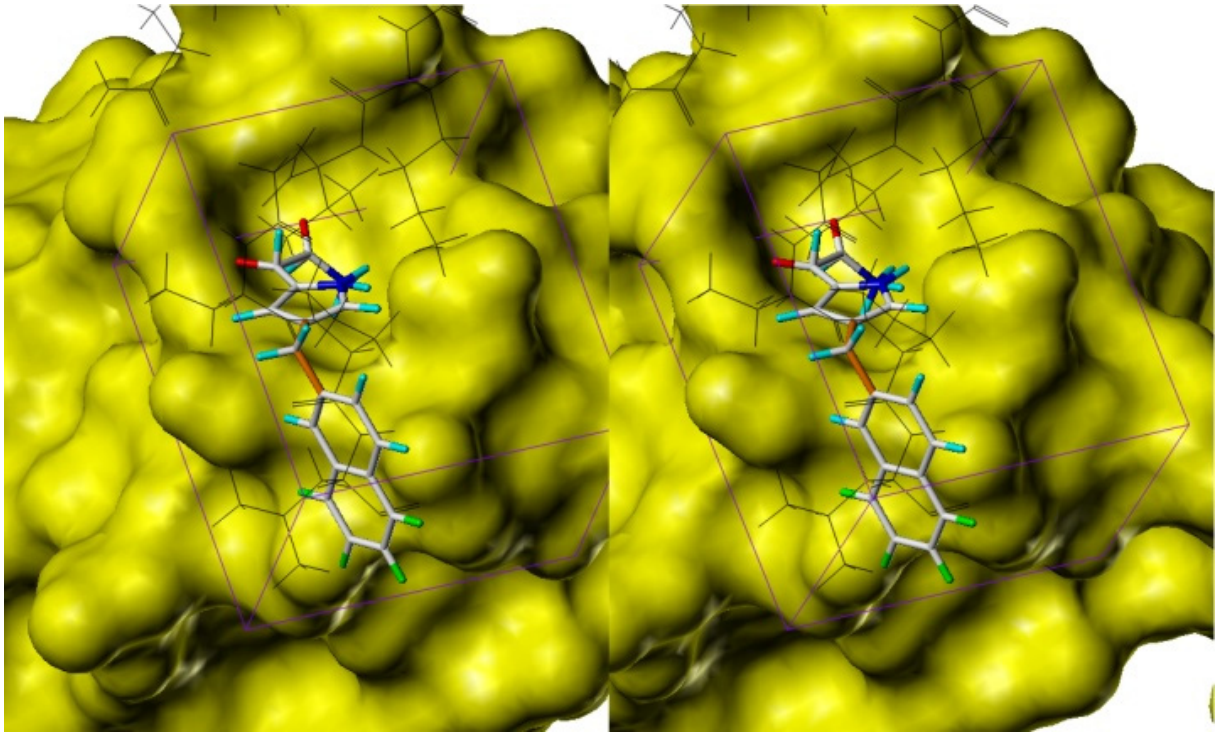


Abbildung 80: *Crossed* Stereodarstellung der günstigsten Orientierung des *unconstrained docking* von B2B5T2_L00K01, erzeugt mit einem Wert für random_seed von 01. Diese Orientierung hat eine Energie von -18.04 kcal/mol.

Tabelle 22: Energie und random_seed der günstigsten *unconstrained dockings* des doppelt modifizierten Leitstrukturmotives B2B5T2_L02K07.

B2B5T2_L02K07	random_seed	Energie [kcal/mol]
bestes *docking*	11	-16.99
energiegünstigstes *docking*	25	-17.70

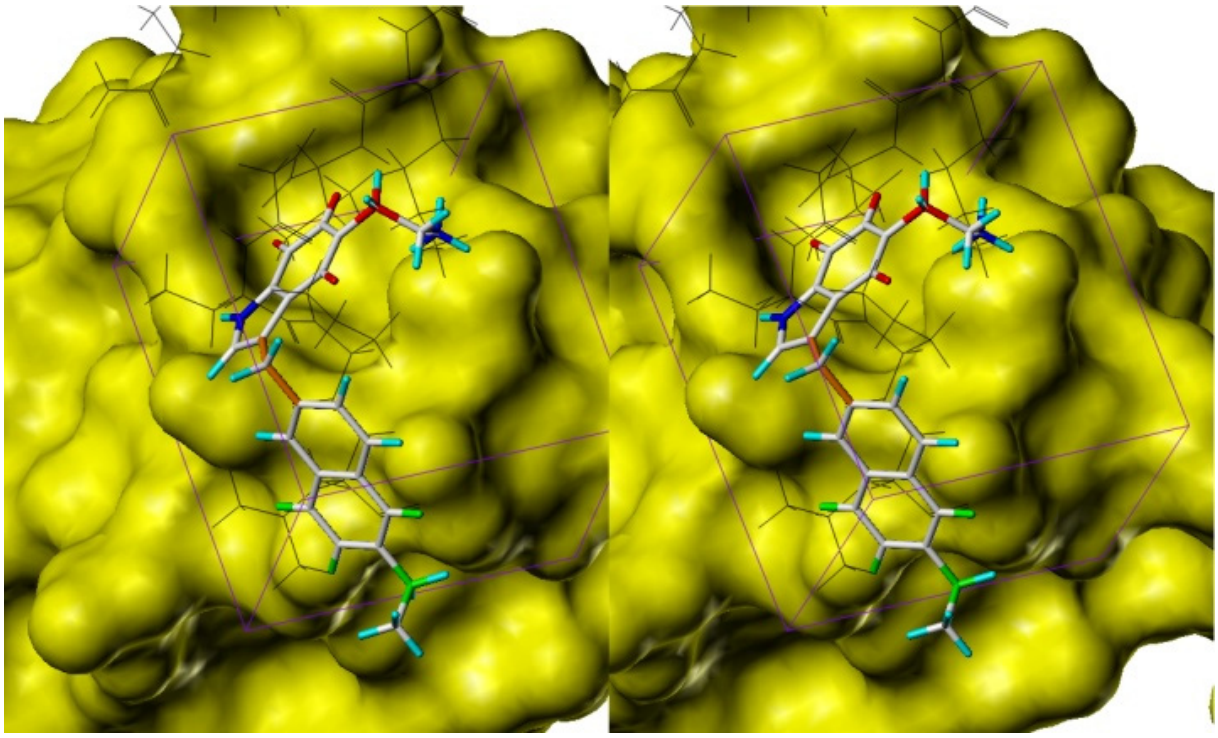


Abbildung 81: *Crossed* Stereodarstellung der günstigsten Orientierung des *unconstrained docking* von B2B5T2_L02K07, erzeugt mit einem Wert für random_seed von 04. Diese Orientierung hat eine Energie von -16.99 kcal/mol.

Die Energiewerte der einzelnen Optimierungsstufen der beiden Leitstrukturgerüste B2B5T2 sind in Abbildung 82 aufgetragen. Die Abbildung zeigt, wie sich die Energiewerte entsprechend den Erwartungen von den einfach modifizierten über die doppelt modifizierten *constrained* gedockten bis zu den doppelt modifizierten *unconstrained* gedockten Liganden stetig verbessert haben.

Hier bestätigte sich, was in Abschnitt 3.4.2 aufgrund der visuellen Inspektion der *constrained dockings* angenommen wurde: Auf der Lysin-Seite bewährte sich die unkonventionelle Ersetzung des Indols durch ein Pyrrol. Auf der Leucin-Seite wurde der Raum, den die Isopropylgruppe des Leucins eingenommen hatte, durch das Ringsystem des CR5 bereits gut

ausgefüllt, so dass durch die zusätzliche Ethylfunktion am B2B5T2_L02K07 kein zusätzlicher Energiegewinn mehr erzielt werden konnte.

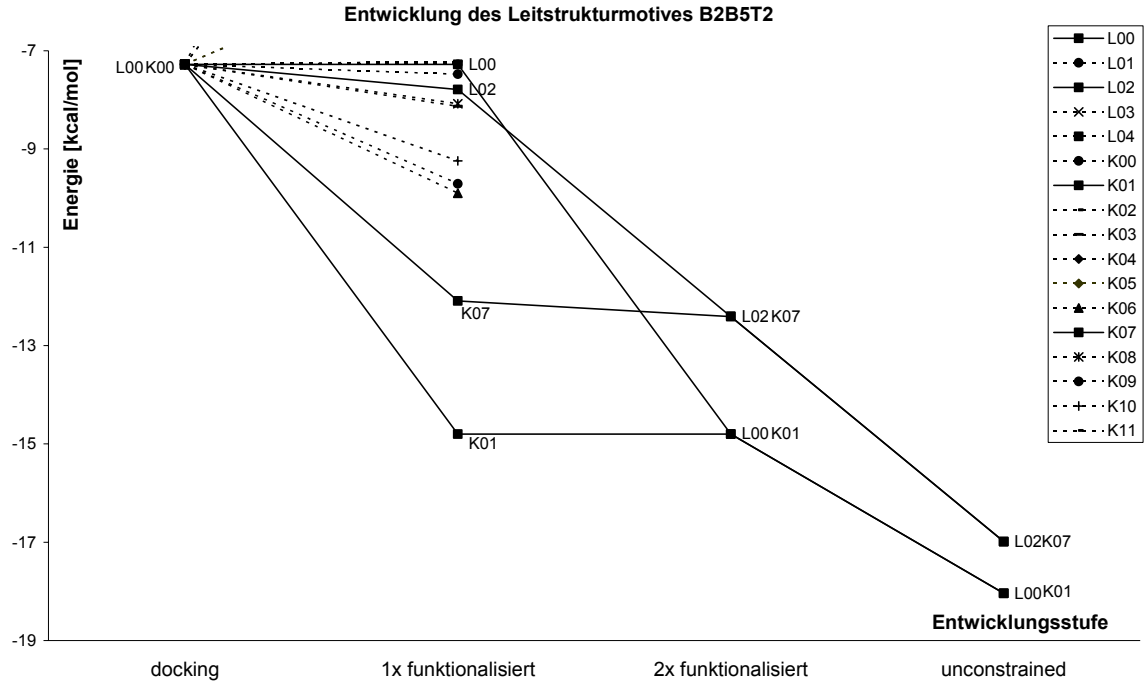


Abbildung 82: Übersicht über die Entwicklung des Leitstrukturmotives B2B5T2. Es wurde eine ungewöhnlichere Variante L00K01 und eine konservativere Variante L02K07 bearbeitet. Für beide verbessert sich die *docking*-Energie kontinuierlich vom unmodifizierten über die Einzelmodifikationen bis hin zum doppelt modifizierten Leitstrukturmotiv. Dass sich die Energie für beide im letzen Schritt vom *constrained docking* zum *unconstrained docking* der doppelt modifizierten Liganden nochmals so stark verbessert (in einem Fall sogar unter -18 kcal/mol), zeigt wie erfolgreich die in dieser Arbeit entwickelte Methode in diesem Fall war.

3.5.5 *Unconstrained docking* des Leitstrukturmotives B2B5T3_L02K03

Als doppelt modifizierter Ligand war hier B2B5T3_L02K03 konstruiert worden.

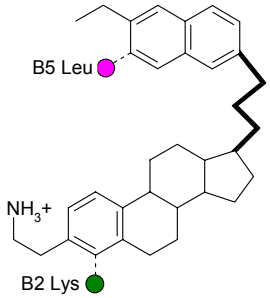


Abbildung 83: Das Leitstrukturmotiv B2B5T3_L02K03.

Im *unconstrained docking* dieses Liganden gegen CD4 bestätigte sich die Annahme, dass der Ligand für diese "Aufgabe" zu groß und sperrig war. Die beste erreichte Energie (random_seed 00) von -13.21 kcal/mol war schon über eine Kilokalorie schlechter, als die -14.65 kcal/mol, die sich im *constrained docking* ergeben hatten. Die erste Orientierung die in etwa die erwartete Anordnung einnahm (random_seed 24) hatte nur noch eine Energie von -12.89 kcal/mol.

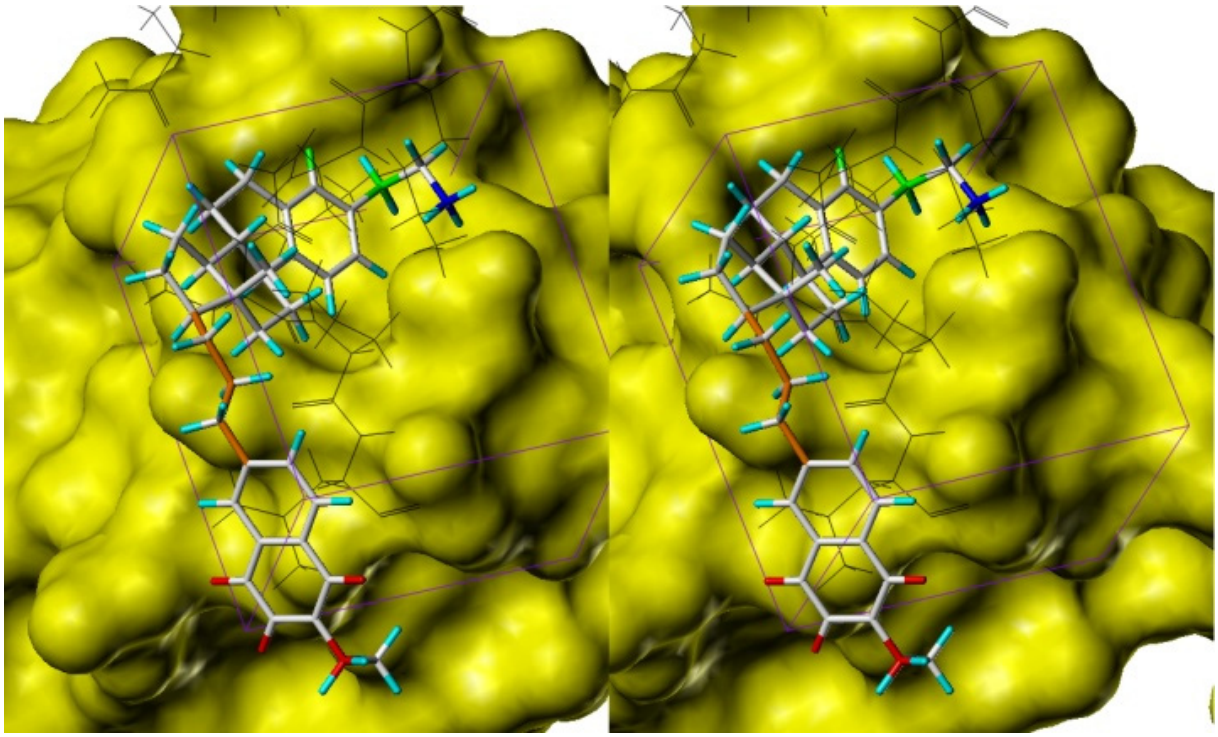


Abbildung 84: *Crossed* Stereodarstellung der energiegünstigsten Orientierung des *unconstrained docking* von B2B5T3_L02K03, erzeugt mit einem Wert für random_seed von 00. Diese Orientierung wurde nicht weiter verwendet, da sich die neu eingeführte Ethylgruppe an CR5 (rot/weiß/cyan dargestellt) nicht mehr im Bereich der Isopropylgruppe des Leucins des Dekapeptides (schwarze Linien) befindet, sondern frei in den Raum gerichtet ist.

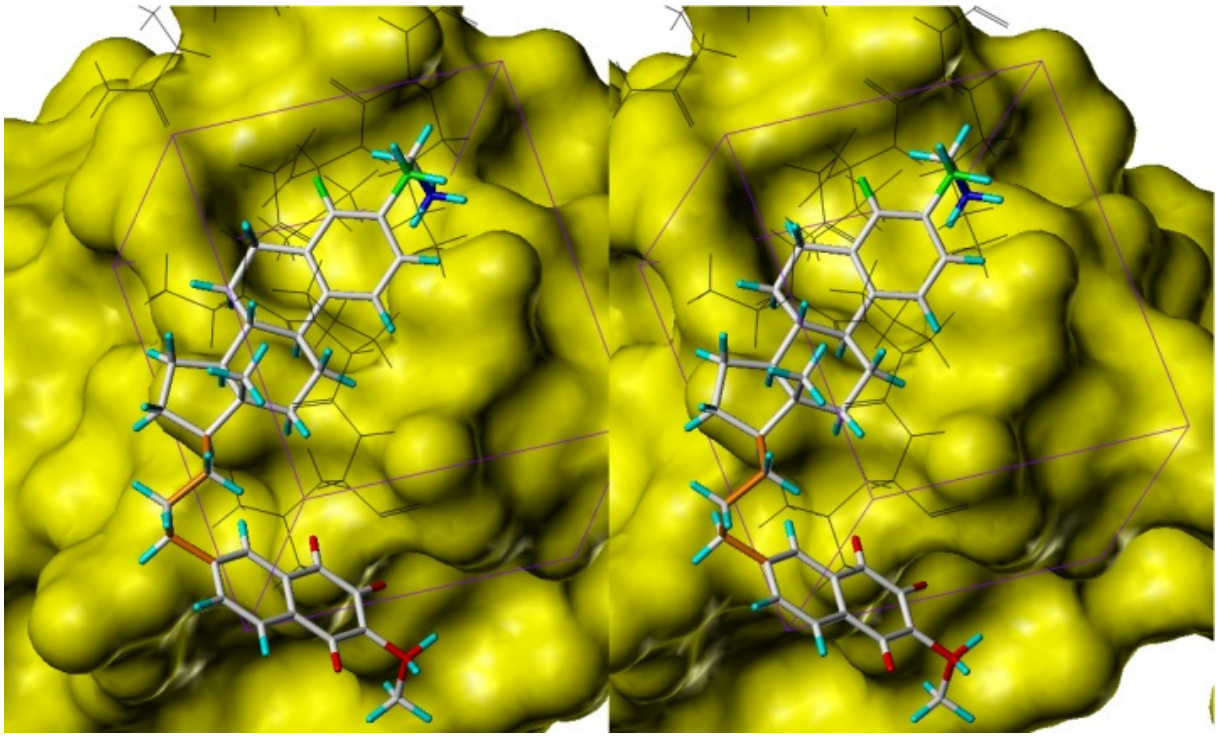


Abbildung 85: Diese Orientierung aus dem *unconstrained docking* des Liganden B2B5T3_L02K03 entstand mit einem random_seed von 24. Auch hier verlässt das CR5-Ringsystem den Bereich des Leucins. Die Orientierung wurde trotzdem als die beste Nachahmung des Leucins ausgewählt, da zumindest die Protonen an der 2- und 3-Position des CR5(naphtyl)-Systems im Bereich des Leucins liegen.

Um einen Überblick über die Energiewerte der einzelnen Optimierungsstufen des Leitstrukturgerüstes B2B5T3 zu geben, sind diese in Abbildung 86 aufgetragen. Die Verschlechterung der Energiewerte des Liganden im *unconstrained docking* um fast 2 kcal/mol ist in der Grafik deutlich zu erkennen.

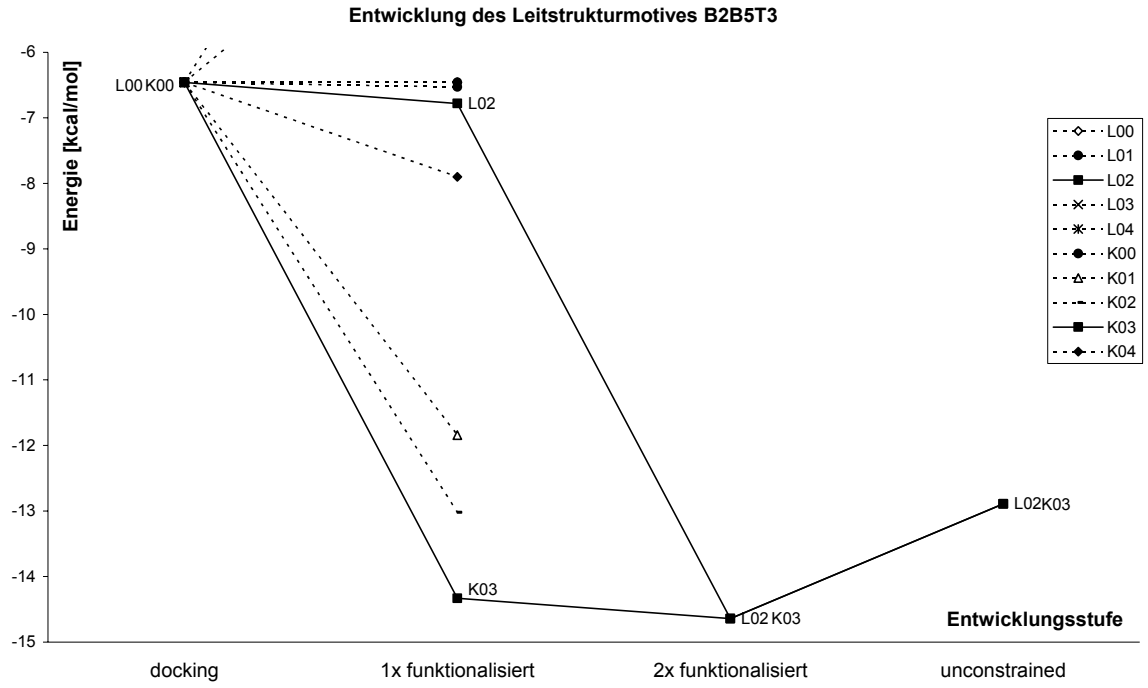


Abbildung 86: Übersicht über die Entwicklung des Leitstrukturmotives B2B5T3. Zu Beginn der Ligandenentwicklung (vgl. Abbildung 68) fiel bereits auf, dass der Ligand aufgrund seines CR8(steroid)-Systems sehr sperrig und für die Nachahmung der Lys-Leu-Wechselwirkung eventuell nicht geeignet ist. Die Optimierung im *constrained docking*, bei der das Steroid-Gerüst nicht bewegt wurde, zeigte hier einen normalen Energieverlauf. Im *unconstrained docking* des doppelt modifizierten Liganden steigt die Energie jedoch deutlich an, da sich das voluminöse Steroid-Gerüst an vielen Stellen ungünstig und sperrig der Rezeptoroberfläche nähert.

3.5.6 *Unconstrained docking* des Leitstrukturmotives B2B5T4_L02K03

Auch dieses RBR enthielt ein Steroid-Gerüst (CR7). Anhand der Ergebnisse des *constrained docking* war entschieden worden (vgl. Abschnitt 3.4.4), das doppelt modifizierte Leitstrukturmotiv B2B5T4_L02K03 zu konstruieren.

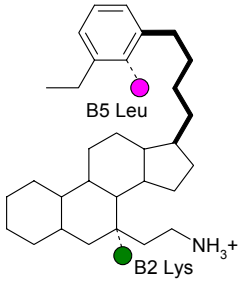


Abbildung 87: Das Leitstrukturmotiv B2B5T4_L02K03.

In den ersten *dockings* in Abschnitt 3.4.4 hatte sich bereits gezeigt, dass die 9-Position dieses Steroides, an der die Substitutionen zur Nachahmung der teminalen Aminogruppe des Lys_{429} durchgeführt wurden, sterisch schlecht zugänglich war. Dies bestätigte sich im *unconstrained docking*. Ausgehend von einer Energie von -10.65 kcal/mol aus dem *constrained docking*, wurde im *unconstrained docking* bestenfalls eine Energie von -2.57 kcal/mol erreicht.

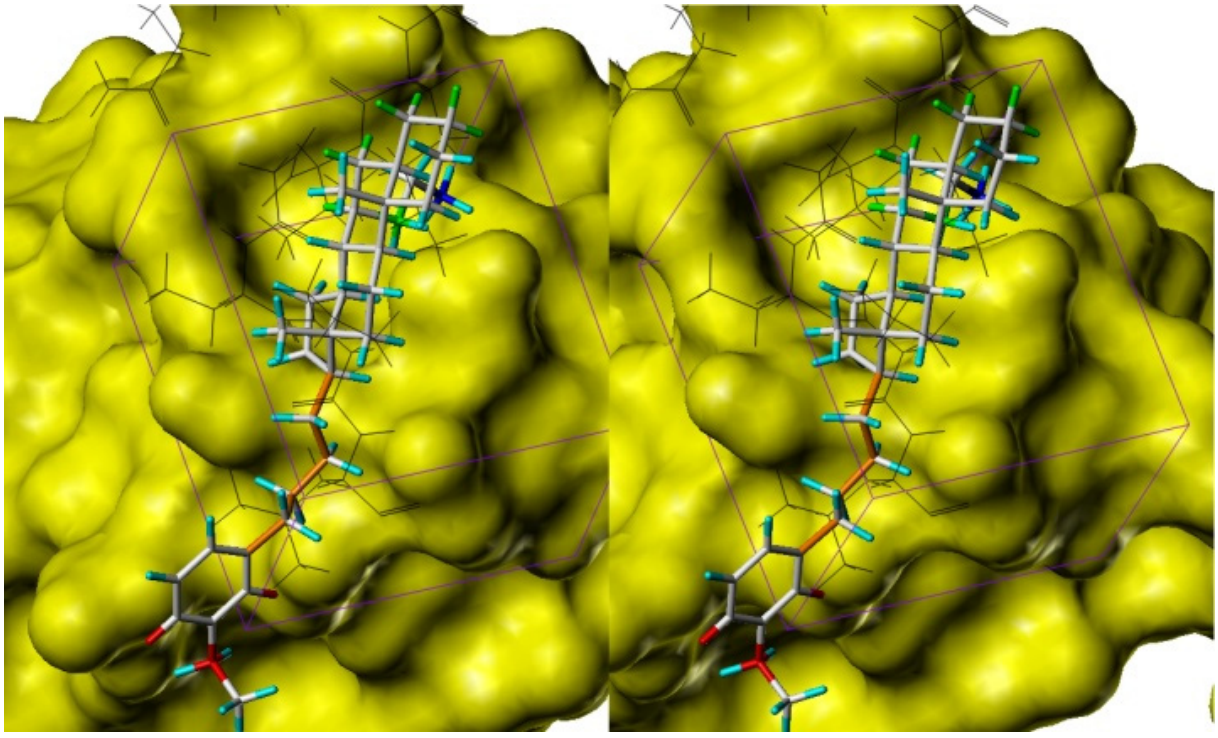


Abbildung 88: *Crossed* Stereodarstellung der mit -2.57 kcal/mol energiegünstigsten Orientierung des *unconstrained docking* von B2B5T4_L02K03, erzeugt mit einem Wert für random_seed von 01. Um die Coulomb-Interaktion zwischen der an der 9-Position des Steroides angeknüpften Aminoethylgruppe (oben rechts, blau/cyan/weiß) und der Carboxylatfunktion des Asp63 zu ermöglichen, wurde der unpolare Substituent, der das Leucin nachahmen soll (unten links, rot/cyan/weiß), aus der Region herausgeschoben, in der sich das Dekapeptid (schwarze Linien) befunden hatte.

In der Abbildung ist zu sehen, dass das Gerüst des Liganden viel zu groß für die gestellte Aufgabe (Mimickry der Wechselwirkung des Lys_{429}/Leu_{123}) war.

Auch die Auftragung der Energiewerte der einzelnen Optimierungsstufen der Leitstruktur B2B5T4 in Abbildung 89 zeigt deutlich, dass die Methode in diesem Fall an ihre Grenzen stößt. Die Energie des Liganden, der aus den Optimierungsschritten im *constrained docking* hervorgegangen ist, wird beim Schritt zum *unconstrained docking* um 8 kcal/mol schlechter. Daran ist zu erkennen, dass die Optimierung im *constrained* Zustand bei einem so sperrigen Baustein für eine vergleichsweise räumlich kleine Pharmakophorregion nicht mehr gut geeignet ist.

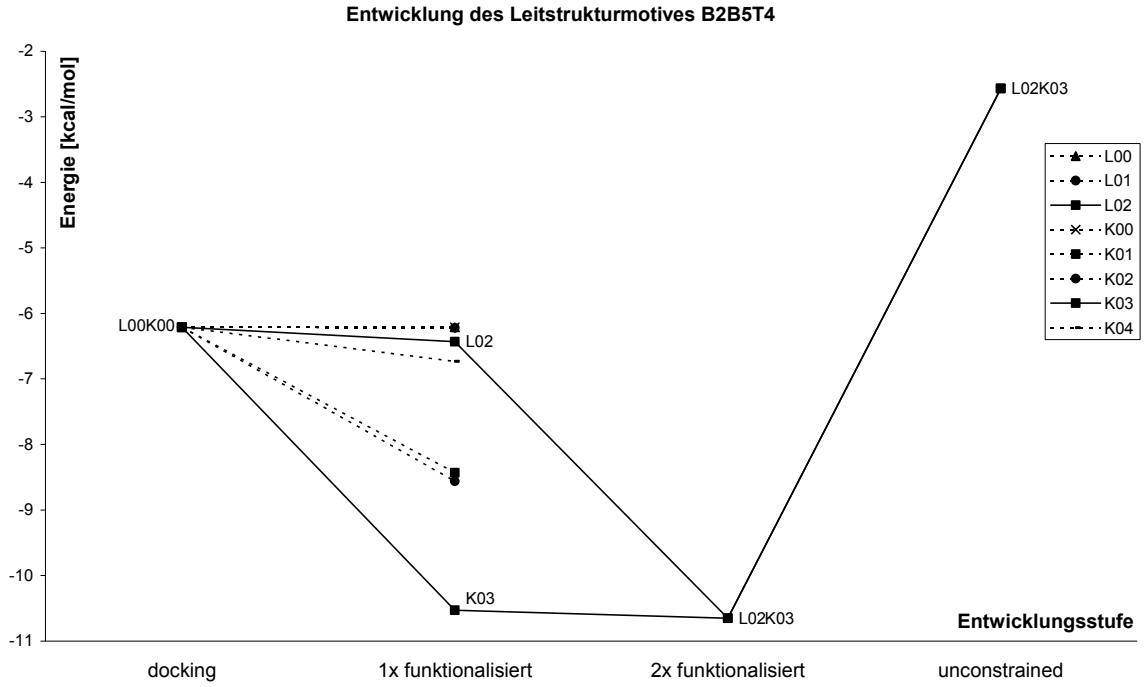


Abbildung 89: Übersicht über die Entwicklung des Leitstrukturmotives B2B5T4. Der Energieverlauf ist über die nach AAV 2 bewerteten *dockings* normal (vgl. Abbildung 78 und Abbildung 82). Im *unconstrained docking* des doppelt modifizierten Liganden jedoch stieg die Energie um über 8 kcal/mol an, da sich das voluminöse Steroid-Gerüst an vielen Stellen ungünstig und sperrig der Rezeptoroberfläche näherte.

3.5.7 Fazit zu den Steroid-Bausteinen und -Gerüsten

Für das Bindungsepitop der Wechselwirkung des Lysins und des Leucins aus dem NMWQKVGTPL mit dem CD4 waren RBR mit Steroid-Anteilen (CR7 und CR8, vgl. Abbildung 29) ungeeignet. Die zu überbrückende Entfernung von B2Lys nach B5Leu betrug

(vgl. Tabelle 9) 11.477 Å. Das Steroid CR8 als Grundgerüst, gemessen von einem C-Atom an C3 bis zu einem C-Atom an C-17, überspannte 11.288 Å.

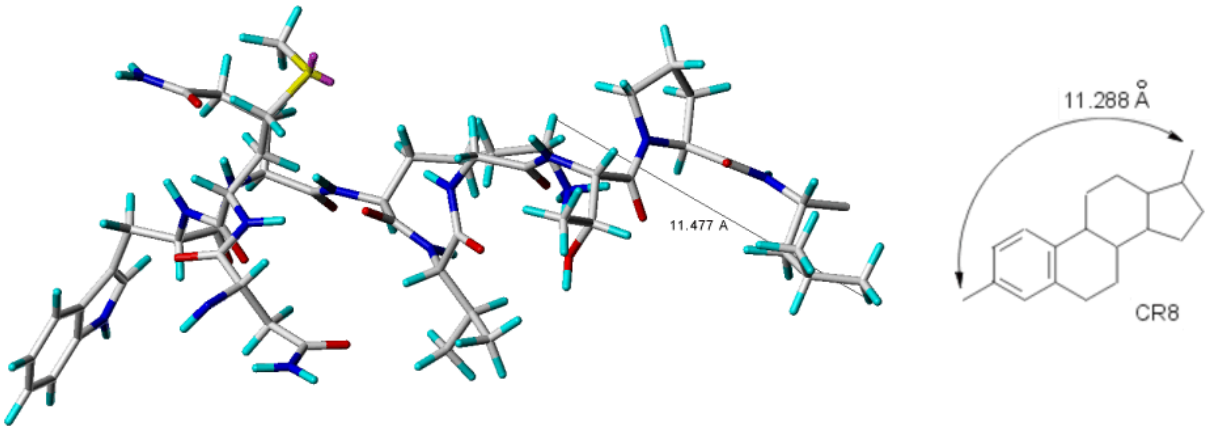


Abbildung 90: Links: Eine *capped sticks*-Darstellung des Dekapeptides NMWQKVGTPL, eingefärbt "*by atom color*". Als dünne schwarze Linie ist die Strecke von 11.477 Å dargestellt, der Abstand des Hε des Lysin (B2) zum Hδ des Leucin (B5). Rechts: Strukturformel des Ringsystems CR8. Dieses spannt zwischen einer Methylgruppe an C-3 und einer Methylgruppe an C-17 eine Entfernung von 11.288 Å auf. Die Abbildung verdeutlicht, dass Liganden die einen Steroid-Anteil enthalten, zu groß und unflexibel sind, um die B2-B5-Wechselwirkung nachzuahmen.

Für wesentlich größere Entfernungen könnten Liganden mit Steroid-Anteil wieder interessant werden, da sie eine starre und somit entropisch günstige Möglichkeit bieten, einen großen Teil der Entfernung zwischen den beiden zu verbrückenden Zentren zu überspannen. Die sterischen Schwierigkeiten mit der Substitution in der 9-Position von CR7 legen nahe, nur noch die gut zugänglichen Positionen 3, 4 und 17 zu verwenden.

3.5.8 *Unconstrained docking* des Leitstrukturmotives B2B5T5_L04K01

Für diesen Liganden war in den *constrained docking*-Experimenten der doppelt modifizierte Ligand B2B5T5_L04K01 entwickelt worden. Seine Bindungsenergie war mit +41.32 kcal/mol sehr schlecht.

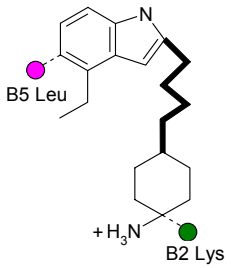


Abbildung 91: Das Leitstrukturmotiv B2B5T5_L04K01.

Es war zu vermuten, dass der Energieverlust durch die ungünstige sterische Wechselwirkung zwischen dem Substituenten am Cyclohexylring des Liganden und der Carboxylgruppe des Asp63 des CD4 sich leicht durch Drehung einiger Bindungen in der Brücke würde aufheben lassen. Daher war der Ligand trotzdem in die Stufe des *unconstrained docking* übernommen worden. Es war zu erwarten, dass es dann zu einer energetisch sehr günstigen Interaktion zwischen den beiden gegensätzlich geladenen Gruppen kommen sollte.
Im *unconstrained docking* gemäß AAV 4 bestätigte sich diese Annahme. Der Ligand erreichte eine Energie von -16.43 kcal/mol (random_seed 01) wobei der Cyclohexylring ein Stück aus der Parallelität zur Proteinoberfläche gekippt wurde. Die Aminogruppe an C-3 blieb weiterhin in unmittelbarer Nähe zur Carboxylgruppe des Asp63.

Tabelle 23: Energie und random_seed der günstigsten *unconstrained dockings* des doppelt modifizierten Leitstrukturmotives B2B5T5_L04K01.

B2B5T5_L04K01	random_seed	Energie [kcal/mol]
bestes *docking*	01	-16.43
energiegünstigstes *docking*	10	-16.83

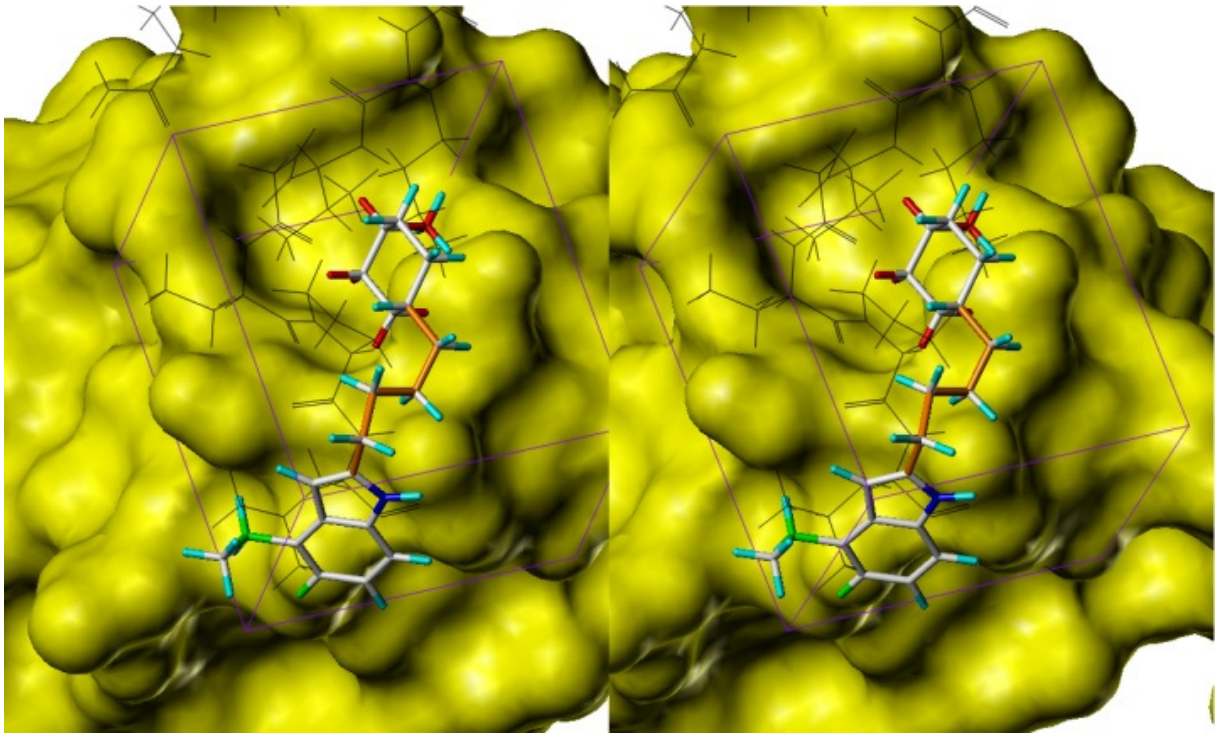


Abbildung 92: *Crossed* Stereodarstellung der günstigsten Orientierung des *unconstrained docking* von B2B5T5_L04K01, erzeugt mit einem Wert für random_seed von 01. Diese Orientierung hat eine Energie von -16.43 kcal/mol.

Für dieses Leitstrukturgerüst B2B5T5 ist ein Überblick über die Energiewerte der einzelnen Optimierungsstufen in Abbildung 93 gegeben.

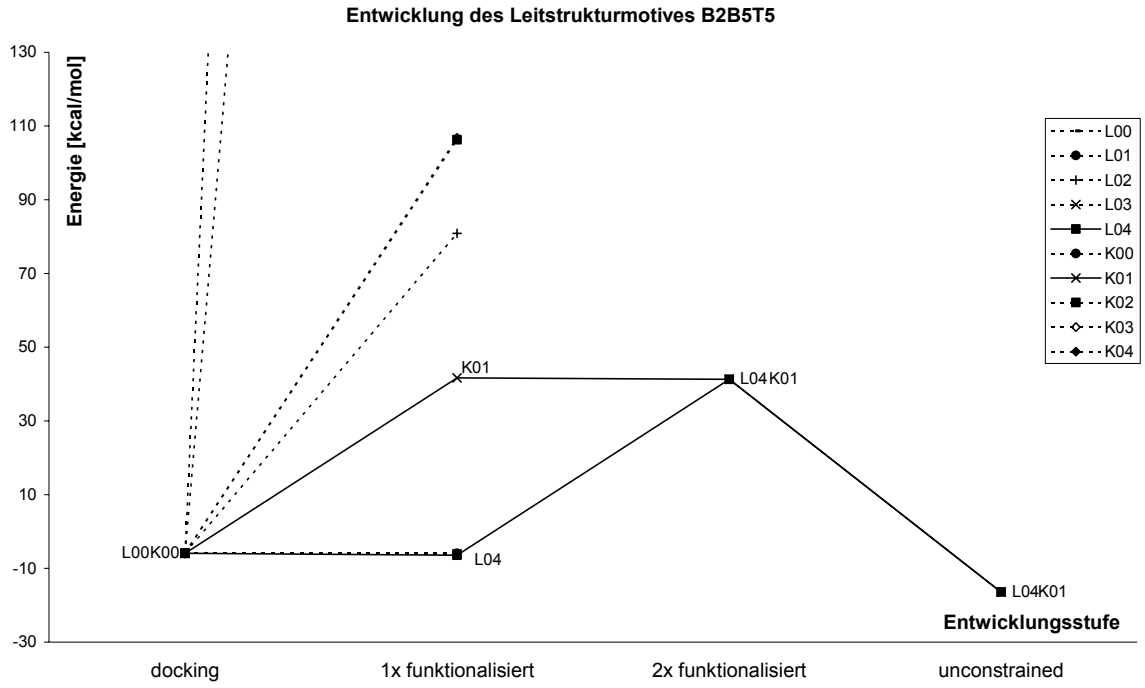


Abbildung 93: Energieverlauf der Entwicklung des Leitstrukturmotives B2B5T5_L04K01. Da der CR2-Cyclohexylring parallel zur Oberfläche des Rezeptors lag (vgl. Abbildung 72), kam es bei jeder Modifikation auf der Lysin-Seite zu einem starken Anstieg der Energie weit in den positiven Bereich hinein. Die günstigen Modifikationen auf der Leucin-Seite verliefen im *constrained docking* nahezu energieneutral. Auch nach der Kombination der beiden ausgewählten Einzelmodifikationen L04 und K02 schien das Konstrukt immer noch kein guter Binder zu sein (2 x funktionalisiert, Energie über +41 kcal/mol). I. Im *unconstrained docking* kam es jedoch zu einer Drehung des Cyclohexylringes und damit zu einer Absenkung der Energie auf unter -16 kcal/mol.

Wie zu sehen ist, veränderte die Leucin-Modifikationen beinahe nichts an der Energie der Liganden, aber die Modifikationen auf der Lysin-Seite erhöhte die Energie drastisch bis in den positiven Bereich. Auch die Energie des doppelt modifizierten Liganden L04K01 war mit +41 kcal/mol noch sehr ungünstig. Erst im *unconstrained docking* konnte sich der Cyclohexylring aus der Ebene drehen (vgl. Abbildung 92) und das System erreichte eine günstigere Energie als die des unmodifizierten Liganden.

3.6 Überblick über die durchgeführte Ligandenentwicklung

Die in den vorhergehenden Abschnitten dargestellten Weiterentwicklungen der fünf Leitstrukturgerüste durch Funktionalisierung sind in Abbildung 90 graphisch zusammengefasst. Hier ist zu sehen, dass für die drei RBR, die keinen Steroid-Anteil

enthielten, über den Entwicklungsprozess eine deutliche Verbesserung der vorhergesagten Bindungsenergie erreicht wurde.

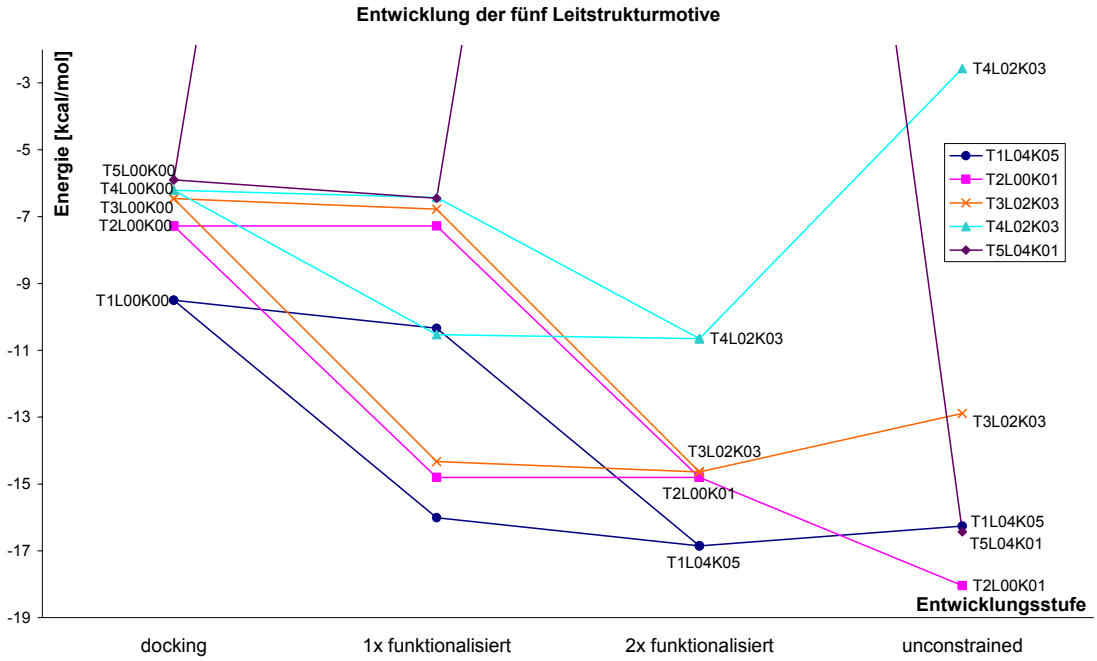


Abbildung 94: Überblick über die Energien der Entwicklungsschritte der fünf Leitstrukturmotive. Aus dem unfunktionalisierten Ligand B2B5T1_L00K00 (hier als T1L00K00 bezeichnet und in dunkelblau markiert), der im ersten *constrained docking* mit -9.50 kcal/mol den niedrigsten Energiewert erreichte, ging mit B2B5T1_L04K05 (hier als T1L04K05 bezeichnet) ein sehr energiegünstiges Leitstrukturmotiv hervor. Das Leitstrukturmotiv T2 (magentafarben markiert) erreichte eine noch niedrigere Energie von -18 kcal/mol. Die beiden Liganden die einen Steroid-Anteil enthielten (B2B5T3, in orange markiert) und B2B5T4 (in cyan markiert) erreichten ungünstigere bis sehr viel ungünstigere Energiewerte. Der Energiewert der Lysin-Modifikation des B2B5T5 (violett gekennzeichnet) liegt weit im positiven Bereich (vgl. Abbildung 93) und ist in dieser Abbildung daher nicht dargestellt.

Die größte Verbesserung der Energie gelang bei B2B5T2. Hier wurde durch die optimierte Funktionalisierung gleichzeitig die mit -18.04 kcal/mol insgesamt günstigste Energie erreicht. Eine ebenfalls große Verbesserung der Energie wurde bei B2B5T5 erreicht. Hier hatten sich im *constrained docking* alle lediglich auf der Lysin-Seite funktionalisierten RBR als energetisch sehr ungünstig gezeigt. Die in Abschnitt 3.4.5 angestellten Überlegungen über die wahrscheinlichen Ursachen dieser ungünstigen Energien und Möglichkeiten zu ihrer Behebung bestätigten sich im *unconstrained docking*. Die Energie konnte im Vergleich zum unmodifizierten Liganden B2B5T5 um 10.53 kcal/mol auf einen Wert von -16.43 kcal/mol verbessert werden. Die ungünstige sterische Wechselwirkung von Substituenten auf der

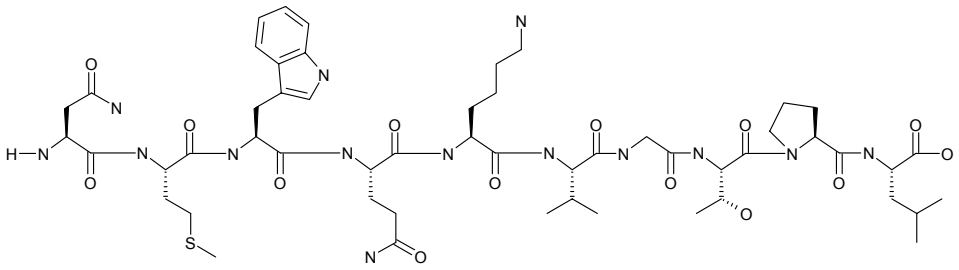
Lysin-Seite, die teils zu Energiewerten über 100 kcal/mol geführt hatten, konnten durch die Rotation des Cyclohexylringes verhindert werden.

Für den *top scorer* der initialen Suche (B2B5T1) bei dem das unmodifizierte RBR bereits eine Energie von -10.05 kcal/mol erreicht hatte, konnte durch Funktionalisierung eine Verbesserung um 6.76 kcal/mol auf -16.26 kcal/mol erreicht werden.

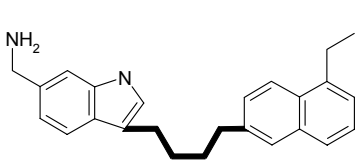
Für die beiden RBR, die ein Steroid als eines ihrer Ringsysteme enthielten (B2B5T3 und B2B5T4), konnten keine günstigen Energien erzielt werden. Wie in Abschnitt 3.5.7 bereits dargelegt, sind diese Bausteine für das gegebene Problem sterisch zu anspruchsvoll. Im RBR B2B5T3, bei dem das Lysin durch funktionelle Gruppen an der relativ gut zugänglichen 3- bzw. 4-Position des Steroides mimikriert werden sollte, wurden noch etwas bessere Bindungsenergien erreicht als bei B2B5T4, in welchem die lysinartigen Funktionalisierungen an der sterisch stark abgeschirmten 9-Position des Steroides lokalisiert waren.

3.7 Vergleich der entwickelten Liganden mit dem Dekapeptid

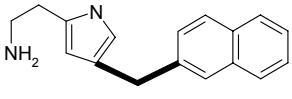
Die drei Leitstrukturmotive, die im *unconstrained docking* die besten Energien erreichten, sind in Abbildung 95 dargestellt. Ihre Bindungsmodi an CD4 sind in Abbildung 96 gezeigt.



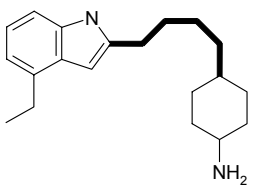
Dekapeptid NMWQKVGTPL
E = +28.52 kcal/mol M = 1173.41 g/mol



B2B5T1 L04K05
E = -16.26 kcal/mol M = 356.52 g/mol



B2B5T2 L00K01
E = -18.04 kcal/mol M = 250.35 g/mol



B2B5T5 L04K01
E = -16.43 kcal/mol M = 298.48 g/mol

Abbildung 95: Das Dekapeptid NMWQKVGTPL sowie die drei daraus entwickelten Leitstrukturmotive, die die besten Bindungsenergien zum CD4 erreichten. Zu jeder Struktur ist ihre mit DOCK bestimmte Bindungsenergie sowie ihre Molmasse angegeben.

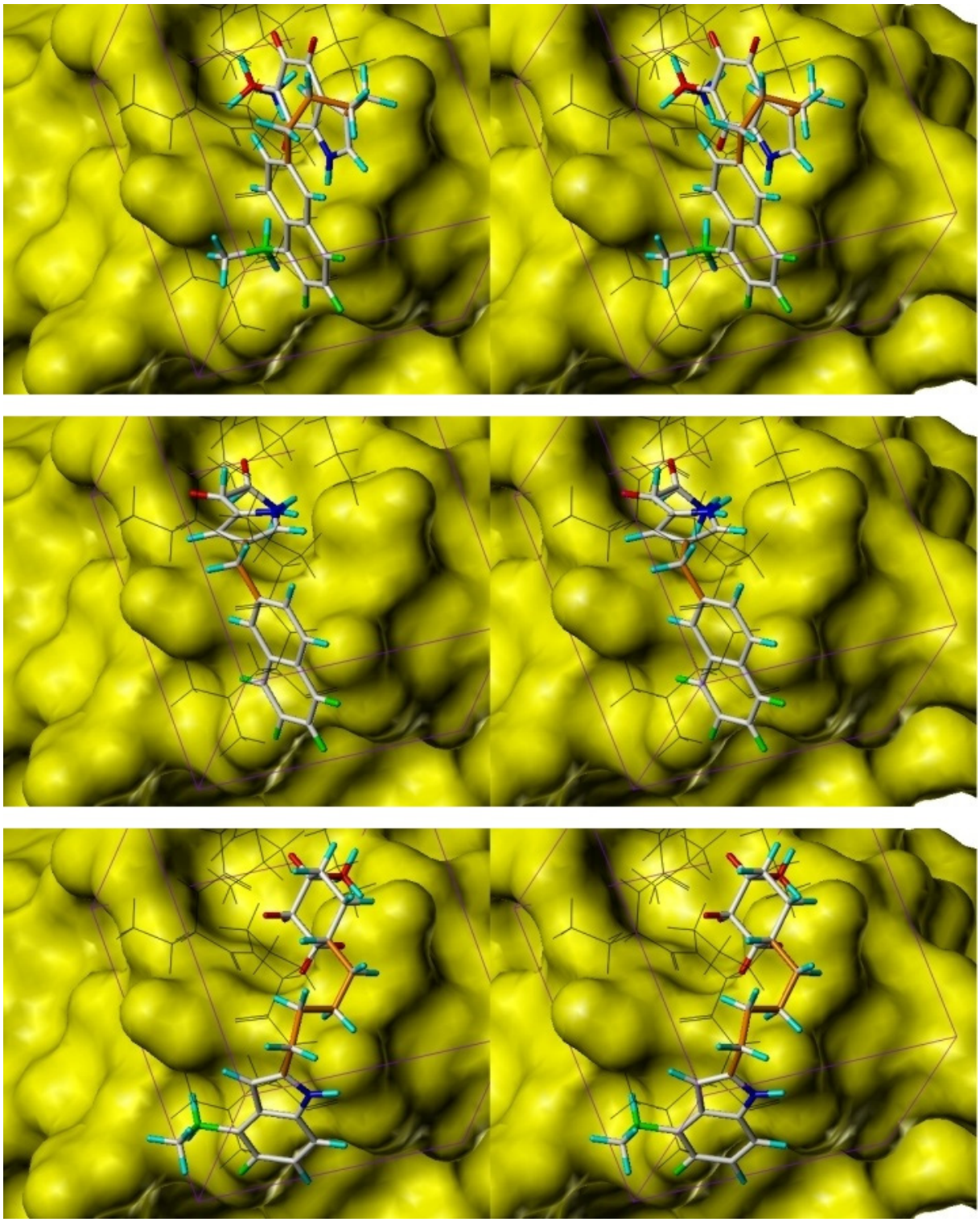


Abbildung 96: Übersicht der drei besten in dieser Arbeit entwickelten Liganden. Mit -18.04 kcal/mol erreichte B2B5T2_L00K01 (Mitte) die beste Energie. Die Energien von B2B5T1_L04K05 (oben) und B2B5T5_L04K01 (unten) sind mit -16.26 kcal/mol und -16.43 kcal/mol etwas geringer.

Um einen Bezugspunkt für die mit DOCK ermittelten Energien der hier entwickelten Liganden zu erhalten, wurde die Energie der Bindung des Dekapeptids NMWQKVGTPL bestimmt. Das Dekapeptid war (vgl. Seite 69) aus der Röntgenstruktur von Kwong *et al.*[188]

entwickelt worden und hatte als Ausgangspunkt für die Entwicklung der Leitstrukturmotive gedient.

Unter Verwendung der in Tabelle 24 abgedruckten Parameter wurde mit DOCK die Bindungsenergie des Dekapeptids zum Rezeptor CD4 bestimmt. Die Parameter "flexible_ligand = no" und "orient_ligand = no" legten hierbei fest, das Peptid weder translatorisch bewegt noch flexibel behandeln wurde. Die von DOCK berechnete Energie für die Bindung des Dekapeptids an CD4 betrug +28.52 kcal/mol. Diese Energie ist wesentlich ungünstiger als die guten Energien der *dockings* der unmodifizierten RBR (vgl. Abschnitt 3.3.4).

Tabelle 24: Auflistung der Parameter, die zur Energiebewertung der Orientierung des Dekapeptides NMWQKVGTPL zum CD4 verwendet wurden. Eine ausführliche Erklärung der einzelnen Parameter befindet sich im DOCK-Handbuch[163] ab Seite 23.

flexible_ligand	no	chemical_score	no
orient_ligand	no	energy_score	yes
score_ligand	yes	atom_model	a
minimize_ligand	no	vdw_scale	1
multiple_ligands	no	electrostatic_scale	1
intermolecular_score	yes	score_grid_prefix	
gridded_score	yes	../grid/grid	
grid_version	4	vdw_definition_file	
contact_score	no	/usr/user/dock/vdw.defn	

Die Energien der drei in Abbildung 96 gezeigten Leitstrukturmotive sind mit -16 bis -18 kcal/mol deutlich günstiger. Die Molmasse der neu entwickelten Liganden betragen zwischen 250 und 356 g/mol. Damit liegen sie deutlich unterhalb der von Lipinsky[16] vorgeschlagenen Obergrenze von 500 g/mol und um den Faktor 3 bis 4 unter der Molmasse der Dekapeptids (M = 1173 g/mol).

3.8 Schematischer Überblick über die wichtigsten Schritte der Arbeit

Die folgenden beiden Abbildungen geben einen Überblick über die wichtigsten Schritte dieser Arbeit.

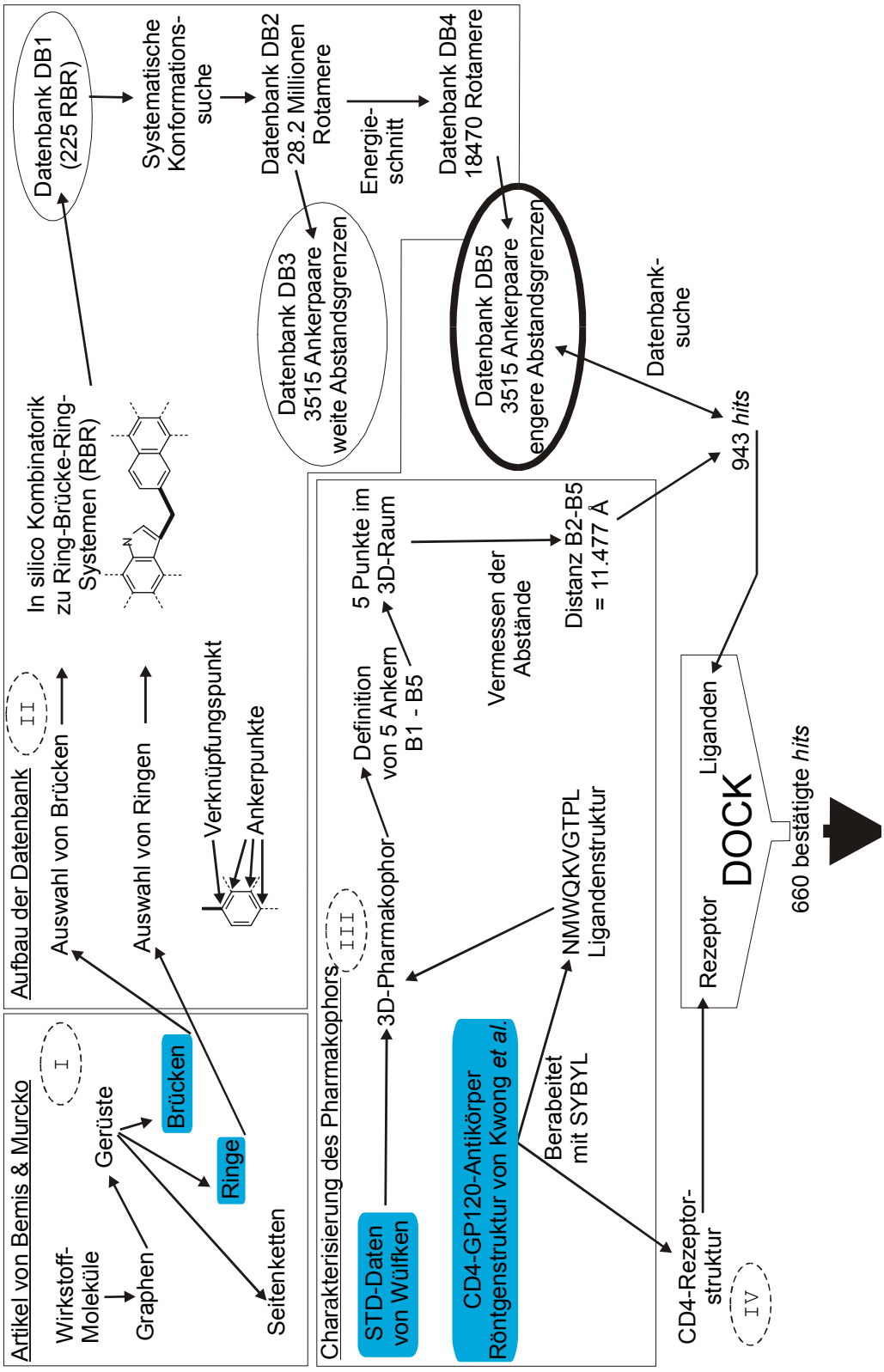


Abbildung 97: Überblick über die wichtigsten Schritte der vorliegenden Arbeit (Teil 1 von 2). Ausgehend von den Untersuchungen von Bemis und Murcko (I) wurde eine Datenbank aufgebaut (II) und auf eine Fragestellung aus dem HIV-Infektionsmechanismus angewendet (III). Die erste Sortierung der *hits* erfolgte anhand der Frage, ob sie sich mit dem Programm DOCK an das *target* CD4 docken ließen oder nicht .

Abbildung 97 zeigt, wie ausgehend von der Analyse heutiger Wirkstoffmoleküle durch Bemis und Murcko (I in Abbildung 97) Ringe und Brücken ausgewählt wurden (II in Abbildung 97), aus denen dann durch *in silico*-Kombinatorik die Ring-Brücke-Ring-Systeme (RBR) der Datenbank DB1 erzeugt wurden. Mittels einer anschließenden systematischen Konformationssuche wurde hieraus die Datenbank DB2 mit 28.2 Millionen Rotameren erzeugt. Die Datenbank DB3 enthielt eine Auflistung der Distanzbereiche, die die einzelnen Ankerpaare der RBR in Datenbank DB2 überbrücken konnten. Da hierbei jedoch auch die energetisch ungünstigen Rotamere berücksichtigt worden waren, wurde durch einen Energieschnitt der Datenbank DB2 die Datenbank DB4 erzeugt, die nur noch 18470 Rotamere in energetisch günstiger Konformation enthielt. Auch für diese wurde eine Analyse der Distanzbereiche, die die einzelnen Ankerpaare überbrücken können, durchgeführt. Diese bildete die Grundlage für die Datenbank DB5.

Angewendet wurde die Datenbank DB5 auf eine Fragestellung zum HIV-Infektionsmechanismus. Basierend auf einer Röntgenstruktur von CD4 und GP120 (III in Abbildung 97) und auf STD-NMR-Daten von Wülfken wurde ein 3D-Pharmakophor konstruiert. Die Distanz zwischen den Ankern B2 und B5 in diesem Pharmakophor wurde exemplarisch ausgewählt, mit 11.477 Å ausgemessen und als Basis für eine Datenbanksuche in DB5 gewählt.

Die Datenbanksuche ergab 943 *hits*, und demnach 943 Ankerpaarungen an RBR, die in energiegünstiger Konformation in der Lage sein sollten, die Pharmakophorsuche "d = 11.477 Å" zu erfüllen.

Diese 943 *hits* wurde mit dem Programm DOCK gegen einen aus der Röntgenstruktur entwickelten CD4-Rezeptor gedockt (IV in Abbildung 97 und Abbildung 98). Hierdurch wurde eine Reduzierung auf die 660 bestätigte *hits* erreicht, deren *docking* erfolgreich verlaufen war.

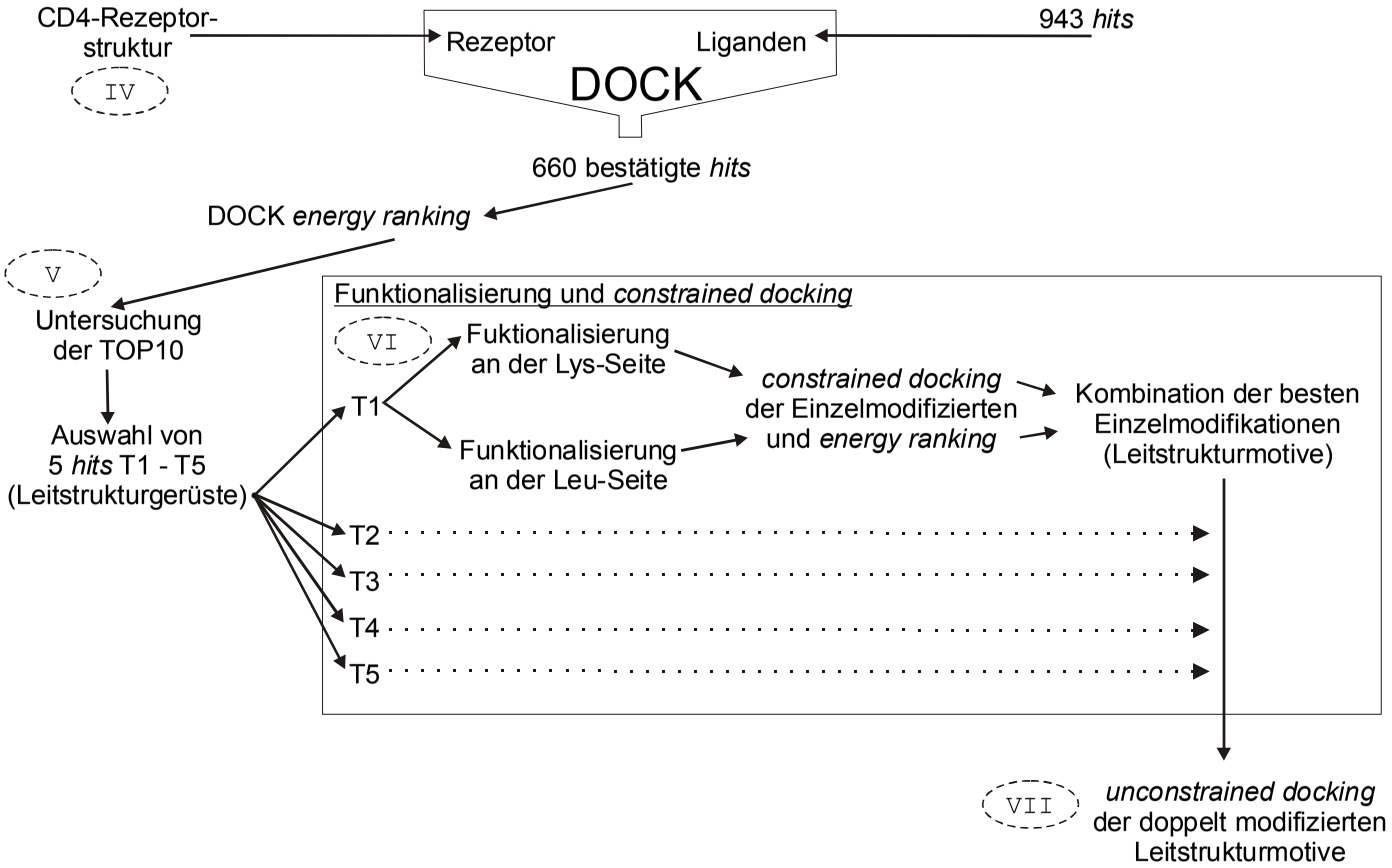


Abbildung 98: Überblick über die wichtigsten Schritte der vorliegenden Arbeit (Teil 2 von 2). Die mit DOCK vorgefilterten Liganden (IV) wurden anhand eines *energy ranking* auf zehn reduziert. (V) Aus diesen wurden fünf für die chemische Funktionalisierung *in silico* (VI) ausgewählt. Die Funktionalisierung wurde einzeln an den beiden Seiten der Liganden durchgeführt und über Energiewerte aus dem *constrained docking* gesteuert. Die hieraus kombinierten doppelt modifizierten Leitstrukturmotive wurden einer Energiebewertung im freien *docking* unterzogen (VII).

Anschließend wurden die 660 bestätigten *hits* mit dem in DOCK eingebauten AMBER Kraftfeld energiebewertet und nach diesem *ranking* sortiert. Die TOP10, die zehn *hits* mit den günstigsten Energien, wurden visuell inspiziert (V in Abbildung 98) und auf fünf Leitstrukturgerüste (T1 bis T5) reduziert.

Jedes dieser Leitstrukturgerüste wurde *in silico* chemisch funktionalisiert (VI in Abbildung 98), sowohl an der Seite, die beim *docking* (IV) an dem Anker B2Lys lokalisiert gewesen war, als auch an der Seite, die am Anker B5Leu befunden hatte. Die einzelnen Modifikationen am Anker B2Lys und B5Leu wurden am als starr und unbeweglich betrachteten Leitstrukturgerüst mit Hilfe von DOCK energiebewertet (*constrained docking*). Die jeweils günstigsten Einzelmodifikationen wurden zu doppelt modifizierten Leitstrukturmotiven zusammengeführt, die dann sowohl nach der Methode des *constrained docking*, als auch durch freies *docking* an CD4 analysiert wurden (VII in Abbildung 98).

Die Anwendung hierarchischer Filter in dieser Arbeit führte zu einer drastischen Reduzierung der Verbindungen, die rechenzeitaufwendig gedockt oder visuell inspiziert werden mussten (Abbildung 99).

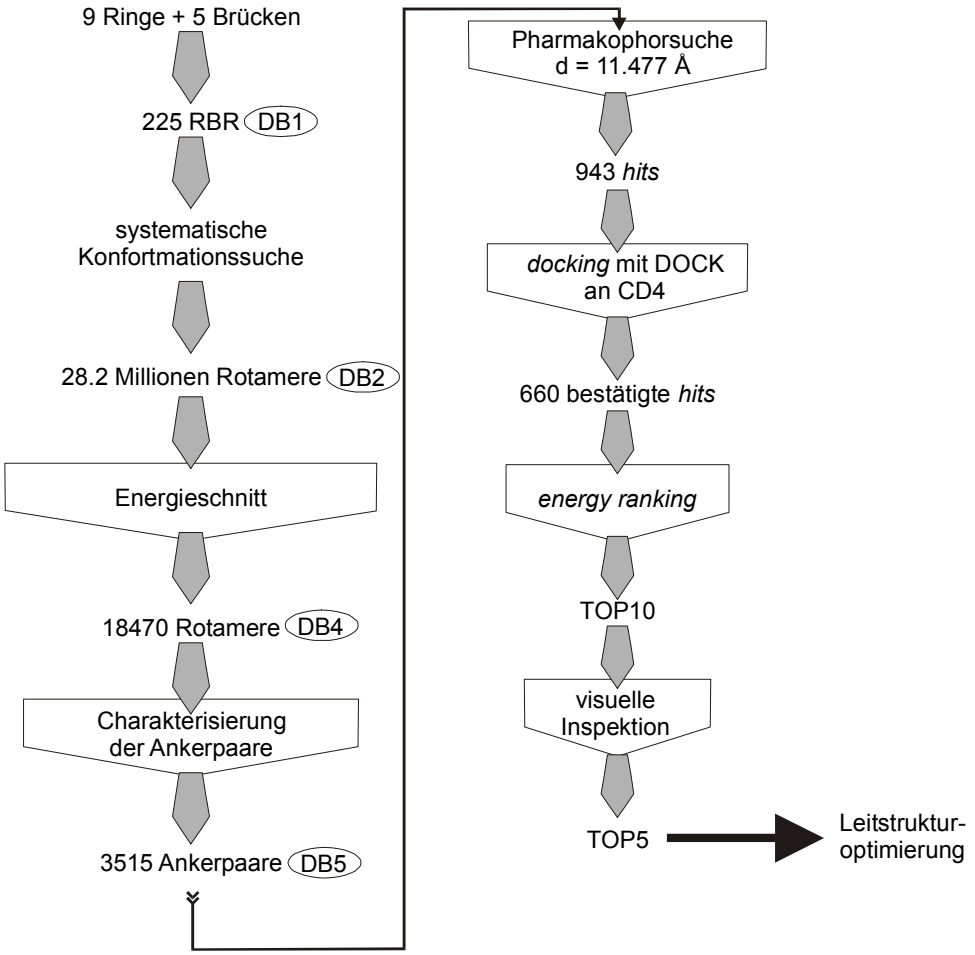


Abbildung 99: Übersicht über die hierarchischen Filter, die beim Aufbau der Datenbank DB5 (linke Seite der Abbildung) und bei ihrer Anwendung auf die in dieser Arbeit behandelte Pharmakophorsuche genutzt wurden.

3.9 Ausblick

Für die in dieser Arbeit aufgebaute Datenbank von Ring-Brücke-Ring-Systemen (RBR) wurde nur ein sehr kleiner Ausschnitt der Ringsysteme, die in heutigen Wirkstoffen vorkommen, verwendet. Da sich im Rahmen dieser Arbeit gezeigt hat, dass die sterisch sehr anspruchsvollen Steroid-Gerüste für zumindest dieses Problem ungeeignet waren, wäre es reizvoll, das Repertoire an Ringen, aus denen die Datenbank konstruiert wird, zu modifizieren. Das Steroid CR7 trägt mit seinen acht Ankeratomen ganz erheblich zum Umfang der Datenbank bei. Dieses Gerüst zu entfernen würde die Datenbank und damit den Rechenaufwand in beinahe allen Schritten erheblich reduzieren. Um die Varianz der Bausteine zu erhöhen, wäre die Einführung eines Ringsystems mit einem Siebenring denkbar.

Nach neuer Auswahl der Ringe müssten die Schritte *in silico*-Kombinatorik, Minimierung, systematische Konformationssuche, Export in das csv-Format und Aufbau der Datenbank wiederholt werden, bevor eine neue Suche durchgeführt werden könnte.

Interessant wäre es auch, dem Nutzer der Datenbank ein Programm zur Verfügung zu stellen, mit dem er eigene Bausteine in das System einspeisen könnte. Damit ein solches Programm für eine breite Anwenderschaft nutzbar würde, sollte seine Entwicklung begleitet sein vom Verfassen eines ausführliches Handbuches.

Großes Potential könnte die hier entwickelte Methode entfalten, wenn es gelänge, eine Kooperation mit etablierten Softwarepaketen, hier bieten sich SYBYL von Tripos und DOCK von der *University of California* an, aufzubauen.

Die in dieser Arbeit vorgeschlagenen neuen Liganden zur Inhibition der CD4-GP120-Wechselwirkung ahmen die Wechselwirkung des Peptides "Lys_{429}-Leu_{125}" nach. Zum einen wäre es interessant, einen oder mehrere dieser Liganden zu synthetisieren und ihre Bindungseigenschaften *in vitro* mittels SPR- und/oder NMR-Studien zu testen. Zum anderen ließe sich sowohl die Bindungsenergie als auch die Bindungsspezifität der Liganden wahrscheinlich noch steigern, wenn zusätzlich die Wechselwirkung des Trp_{427} mit Ser42 und Phe43 mimikriert werden würde. Der Abstand zwischen Trp (B1) und Lys (B2) beträgt 15.138 Å. Dieser könnte als Basis einer neuen Suche nach einer geeigneten Leitstruktur dienen, die nach chemischer Funktionalisierung mit einem der in dieser Arbeit vorgeschlagenen Liganden fusioniert werden könnte. Da das Molekulargewicht der hier vorgeschlagenen bifunktionellen Liganden jeweils bei etwa 300 g/mol liegt, ist anzunehmen, dass auch ein trifunktioneller Ligand "Trp_{427}-Lys_{429}-Leu_{125}"-Mimetikum noch unterhalb der von Lipinsky[16] vorgeschlagenen Obergrenze von 500 g/mol liegen wird. Ebenfalls interessant wäre es, den Effekt einer Fixierung der günstigsten Konformation der Brückenbindungen, beispielsweise durch Mehrfachbindungen oder verbrückende Ringstrukturen zu erforschen.

Interessant wäre es weiterhin, die Desolvatationsenergien für die Liganden und den Rezeptor zu berücksichtigen. Dafür bietet sich ein Programm wie HYDREN[204;205] an, das bereits von Shoichet *et al.* 1999 mit gutem Erfolg mit DOCK kombiniert wurde.[160] Auch in der neuesten Version von DOCK sind Solvatationsterme berücksichtigt.

4 Zusammenfassung

Ein zentrales Problem bei der Entwicklung neuer Wirkstoffe ist der Schritt von einer Pharmakophor-Hypothese zu einer geeigneten Leitstruktur. Liegt eine 3D-Rezeptorstruktur vor, werden üblicherweise *in silico docking*-Methoden[98] verwendet, andernfalls Methoden des *de novo design*[11] oder QSAR.[85]

In der vorliegenden Arbeit wurde ein fragmentbasierter Ansatz mit Methoden des *de novo design*, der Pharmakophorsuche und des *docking* entwickelt, um den Prozess der Leitstruktursuche zu unterstützen. Ein besonderer Vorteil der hier entwickelten Methode liegt darin, dass sie auch angewendet werden kann, wenn keine Röntgenkristallstruktur des Rezeptors oder des Rezeptor-Ligand-Komplexes vorliegt. Häufig kann dennoch die Konformation des Liganden im gebundenen Zustand mit Hilfe der *transferred* NOESY-NMR-Spektroskopie, ermittelt werden. Das Bindungsepitop des Liganden kann mit der STD NMR Spektroskopie bestimmt werden. Basierend auf diesen beiden Informationen lässt sich ein ausschließlich Liganden-basiertes Verfahren konzeptionieren.

Aufbauend auf den Arbeiten von Bemis und Murcko[25] über die Bausteine heutiger Wirkstoffe wurden neun Ringsysteme und fünf aliphatische Brücken als Grundlage für die Konstruktion von 225 Ring-Brücke-Ring-Systemen (RBR) ausgewählt. Die Verknüpfung erfolgte automatisiert mit Hilfe eines Skriptes in der SYBYL-Skriptsprache SPL. An jedem Ringsystem wurden Ankerpunkte definiert, die für die weitere Verknüpfung dienen können. Information über Atomtypen wurde verworfen und vorerst nur die Geometrien der Gerüste betrachtet, da das erste Ziel die Erzeugung von geometrisch passenden Liganden ist. Um die Flexibilität dieser Moleküle zu berücksichtigen, wurde für jedes der Konformationsraum abgetastet. Hierfür wurde eine systematische Konformationssuche aller Torsionswinkel der Brückenbindungen in 10 Grad-Schritten durchgeführt. Alle so erzeugten 28.2 Millionen Rotamere wurden in einer Datenbank für die Leitstruktursuche abgelegt. Für jede Paarung interessanter Anker-Atome an einem Ring-Brücke-Ring-System wurde berechnet, welchen minimalen und maximalen Abstand diese beiden Atome in allen Rotameren dieses Systems einnehmen, deren Energie unterhalb eines Schwellenwertes lag. Hierfür wurde in der Programmiersprache C ein Programm entwickelt. Als günstigster Energieschwellenwert wurde 1 kcal/mol oberhalb der Energie des jeweils energiegünstigsten Rotamers ausgewählt.

Für 3515 Paarungen von Ankeratomen an Ring-Brücke-Ring-Systemen wurden die so ermittelten günstigen Abstände in einer Datenbank erfasst. Um einen übersichtlichen Zugriff

auf diese Datenbank zu ermöglichen, wurden verschiedene Zugriffs-Skripte in UNIX-Shell-Skripten und in SPL geschrieben.

Um die Datenbank an einem realen Problem zu testen, wurde die Wechselwirkung zwischen dem menschlichen CD4 und dem GP120 des HIV modelliert. Die hierfür benötigten 3D-Koordinaten der Bindungspartner wurden, aufbauend auf einer von Kwong *et al.* veröffentlichten Röntgenstruktur[188] mit Methoden des *molecular modeling* bestimmt. Informationen über das die Bindung vermittelnde Epitop waren von Wülfken 2001 mit Hilfe der STD-NMR-Spektroskopie[42] ermittelt worden.[34]

Als Ankerpunkte der zu modellierenden Wechselwirkung wurde jeweils ein Wasserstoffatom des $Lysins_{429}$ und des $Leucins_{125}$ aus dem GP120 gewählt. Der zwischen diesen beiden Atomen gemessenen Distanz von 11.5 Ångström entsprachen 943 Bausteine aus der Datenbank. Durch eine Energiebewertung mit dem DOCK-Programmpaket[206] konnten 283 der *hits* verworfen und für die restlichen 660 *hits* ein *ranking* erstellt werden.

Fünf der bestplatzierten Liganden des ersten *energy ranking* wurden *in silico* chemisch funktionalisiert. Nachdem die jeweils an einer Stelle funktionalisierten Liganden mit DOCK energiebewertet waren, wurden die energiegünstigsten Modifikationen in jeweils einem Molekül zusammengeführt. Diese Moleküle wurden *unconstrained* mit DOCK gegen CD4 gedockt und ihre Energie mit der der unfunktionalisierten *hits* verglichen.

Durch die Einführung funktioneller Gruppen konnten die Bindungsenergien der Liganden um bis zu 10 kcal/mol verbessert werden. Der Ligand mit der günstigsten Bindungsenergie (B2B5T2_L00K01) war hierbei zusammengesetzt aus einem Naphthalin-Ringsystem, das über eine Brücke mit zwei Bindungen mit einem 1H-Pyrrol verknüpft war, das mit einer Ethylaminogruppe funktionalisiert war. Die Bindungsenergie dieses Liganden im betrug im *unconstrained docking* -18 kcal/mol.

Die erzeugten Gerüste konnten erfolgreich so erzeugt und modifiziert werden, dass sie von DOCK so platziert wurden, dass sie in der Pharmakophorregion eine verbesserte Bindungsenergie erreichten.

Um diese sehr viel versprechenden *in silico* prognostizierten Bindungsenergien *in vitro* zu verifizieren und zu neuen, nicht-peptidischen CD4-Liganden zu kommen, werden im Arbeitskreis einige der hier vorgeschlagenen Verbindungen synthetisiert und untersucht werden.

Mit der in dieser Arbeit entwickelten Datenbank konnte erfolgreich ein neues Werkzeug zur *in silico*-Ideengenerierung in der Leitstruktursuche etabliert werden.

5 Summary

A critical point in modern drug design is the step from a pharmacophore hypothesis to a suitable lead structure. *In silico* docking methods[98] are used if the 3D structure of the receptor is available. In other cases one can resort to *de novo* design[11] or QSAR concepts.[85]

To support lead structure generation, a fragment based approach has been developed in the present work. It is based on a combination of *de novo* design, pharmacophore searching and docking.

A special advantage of the method developed here is that it can be applied to cases where x-ray structures of the receptor or of a complex of the receptor with a ligand are not available. The conformation of the ligand in the bound state can be obtained by transferred NOESY NMR, while the binding epitope can be elucidated by STD NMR. Based on these a procedure depending exclusively on ligand data of the can be developed.

Based on Bemis' and Murcko's publication[25] on the scaffolds of modern drugs, nine ring systems and five linkers have been selected as building blocks. In an *in silico* combinatorial process, controlled by a script in the SYBYL-programming-language SPL, these building blocks have been combined to form 225 ring-linker-ring systems.

On each ring system anchor atoms were defined for further linking. Since the primary aim has been the construction of ligands with suitable steric properties, all information on atom types was discarded.

To account for the high flexibility of the molecules, the conformational space for each was sampled by a systematic search along all torsion angles of each linker in steps of 10 degrees. The resulting 28.2 million rotamers were stored in a database. For all rotamers below an energy-cutoff of 1 kcal/mol, the minimum and maximum distance between pairs of anchor atoms was calculated and stored in a database to look-up potential leads. For these calculations, a program in the programming language C was developed. For an easy access to the database, appropriate tools in UNIX-shellscript and the SPL were developed.

To test the database-concept on a real problem, the interaction of human CD4 and HIV-GP120 was modeled. To obtain 3D-coordinates, methods of molecular modeling were applied to an x-ray-structure published by Kwong et al.[188] The binding epitope in question had been identified by Wülfken using STD-NMR[42] in 2001.[34]

Two hydrogens of GP120, one of the ε-CH_2-group of $Lysine_{429}$ and one of the δ-methyl-group of $Leucine_{125}$ were chosen as key functional groups for the GP120-CD4-interaction. Their distance was 11.5 Ångstrom. A database-search with this distance yielded 943 primary hits.

Using the DOCK program package[206] and a newly developed method named "constrained docking" 283 of these hits were rejected and an energy-ranking of the remaining 660 hits was obtained.

The five top scoring ligands were chemically functionalized *in silico*.

The ligands were modified at one of the two interaction-sites and were ranked with DOCK. The most favorable modifications at each site were combined in one molecule. As a proof of concept these molecules were docked "unconstrained" to CD4 and their binding energies were compared to the energies of the unmodified hits.

The addition of functional groups to the ligands improved the binding energies by up to 10 kcal/mol. The top scoring ligand (B2B5T2_L00K01) was composed of a naphthalene ring system which was linked with a two-bond-linker to an 1H-pyrrol which was modified by an ethylamino-group. The binding energy of this ligand was -18 kcal/mol.

This approach was successful in creating and modifying the scaffolds in a way that DOCK could place them in the pharmacophore-region with improved binding energies.

To verify these promising *in silico* results *in vitro* and to reach new nonpepdidic CD4-binding-ligands, some of the ligands proposed here will be synthesized and analyzed in the research group of Prof. Meyer in the near future.

The database that has been developed in this work is a new tool for *in silico* idea generation in lead structure design.

6 Experimenteller Teil

6.1 Verwendete Hard- und Software

Hardware

Handelsübliche PCs (IA32 Prozessor)

Silicon Graphics Workstations (Octane2/R14000, Octane2/R12000, Octane1/R10000)

Software

DOCK (SPHGEN, AUTOMS, GRID, SHOWBOX) http://dock.compbio.ucsf.edu

SYBYL *Molecular Modelling Package*, Version 7.0; Tripos, Inc.: St.Louis, MO, USA

Programmer Studio für Windows, Version 4.1.3, Whisper Technology, Surrey, UK

Tripos Bookshelf 7.0, Tripos Inc., St. Louis, MO, USA

SCIFINDER SCHOLAR 2006, *American Chemical Society*

MIPSpro C-Compiler, Version 7.3.1.3m

6.2 Allgemeine Arbeitsvorschriften

6.2.1 AAV 1 Energieminimierung

Das im mol2-Format vorliegende Molekül wird in SYBYL mit den Standard-Einstellungen des Tripos-Kraftfeldes[75] energieminimert (vgl. Abschnitt 1.6.2). Hierbei wird eine der folgenden drei Varianten angewendet:

a) Über 10000 Schritte, ohne Partialladungen, mit einer Dielektrizitätskonstante von 1.

b) Über 1000 Schritte, mit Gasteiger-Marsili-Partialladungen[203] und einer Dielektrizitätskonstante von 4.

c) Nach Solvatation des Komplexes in drei Lagen TIP3-Wasser,[76] über 1000 Schritte, mit Gasteiger-Marsili-Partialladungen und einer Dielektrizitätskonstante von 1.

6.2.2 AAV 2 *Constrained docking*

Für einen nach dem Muster Name_*ID1*_*ID2*.mol2 (*ID1* und *ID2* bezeichnen die *atom IDs* der Wasserstoffatome, die an die Positionen der Anker B2Lys und B5Leu gedockt werden sollen) benannten Liganden werden zunächst automatisiert mit Hilfe von 7.3.15 ligand_2_ligandcenter.spl eine ligand_centers-Datei erzeugt. Anschließend wird eine dock.in-Datei mit folgenden Einstellungen erzeugt (siehe Tabelle 25):

Tabelle 25: Auflistung der Parameter, die in allen dock.in-Dateien des *constrained docking* identisch waren. Eine ausführliche Erklärung der einzelnen Parameter befindet sich im DOCK-Handbuch[163] ab Seite 23.

flexible_ligand	no	check_degeneracy	no
orient_ligand	yes	reflect_ligand	no
score_ligand	yes	critical_points	yes
minimize_ligand	no	multiple_points	yes
multiple_ligands	no	chemical_match	no
random_seed	0	intermolecular_score	yes
match_receptor_sites	yes	gridded_score	yes
random_search	no	grid_version	4
ligand_centers	yes	bump_filter	no
automated_matching	no	contact_score	no
maximum_orientations		chemical_score	no
5000		energy_score	yes
write_orientations	yes	atom_model	a
rank_orientations	yes	vdw_scale	1
rank_orientation_total	10	electrostatic_scale	1
nodes_minimum	1	score_grid_prefix	
nodes_maximum		../grid/grid	
1000		vdw_definition_file	
distance_tolerance		/usr/user/dock/vdw.defn	
0.25		receptor_site_file	
distance_minimum	2	../struct/receptor_spheres_for_B2B5.sph	

Die Werte für die Parameter ligand_atom_file, ligand_center_file und ligand_energy_file werden jeweils individuell festgelegt.

Hierbei wird der Ligand als starr behandelt (Flexibilität wird gegebenenfalls im Voraus durch eine systematische Seitenketten-Konformationssuche berücksichtigt) und durch DOCK relativ zum Rezeptor neu angeordnet. Hierfür werden statt der Schweratome des Liganden die im ligand_center_file definierten ligand_centers verwendet. Durch die Nutzung von critical_points und ligand_centers wird der Ligand in eine Position gezwungen, in der die beiden Atome mit den *atom IDs ID1* und *ID2* die Positionen der beiden mit B2 und B5 bezeichneten Wasserstoffatome des Dekapeptides einnehmen (vgl. Abbildung 48). Von den 5000 erzeugten Orientierungen werden anhand einer auf einem *grid* berechneten Energiebewertung die besten zehn Orientierungen gespeichert. Die Gewichtung von van-der-Waals- und elektrostatischen Parametern sowie die Definition des *grid* entsprechen den Standardeinstellungen. Eine Variation des random_seed erfolgt in diesem Fall nicht.

Anschließend wird das Programm DOCK mit dieser Parameter-Datei aufgerufen und die Ausgabe zu Dokumentationszwecken in eine log-Datei umgeleitet.
Die Ergebnisse des *docking* finden sich anschließend im ligand_energy_file, das sowohl die Koordinaten der energiegünstigsten Orientierungen, als auch die von DOCK ermittelten Energiewerte der Orientierungen enthält.

6.2.3 AAV 3 Systematische Seitenketten-Konformationssuche

Diese Arbeitsvorschrift dient der Berücksichtigung der konformationellen Flexibilität der neu hinzugefügten funktionellen Gruppen bei modifizierten Liganden, bei denen diese Gruppen Rotations-Freiheitsgrade haben die die Bindung an den Rezeptor beeinflussen können.
Das SPL-Skript, das den jeweiligen Liganden aus dem unmodifizierten *hit* entwickelt (vgl. B2B5T1_L01K00_15_41.col in Anhang 7.3.20), ist in diesem Fall um einen zweiten Teil erweitert. Dieser Teil (siehe Anhang 7.3.21 B2B5T2_L00K07X_14_33.col) führt eine systematische Konformationssuche an den entsprechenden Bindungen durch. Unabhängig von der inneren Energie des Liganden erzeugt dieses Skript in 10 Grad-Schritten alle Rotamere an der oder den relevanten Bindungen. Für Liganden mit einer neuen rotierbaren Bindung (z.B. nach Hinzufügen einer Ethylgruppe in B2B5T2_L02K00) erzeugt das Skript somit 36 Rotamere. Bei zwei neuen rotierbaren Bindungen, beispielsweise nach Hinzufügen einer Aminoethylgruppe in B2B5T2_L00K07 entstehen 36^2, also 1296 Rotamere. Die Rotamere werden unter eindeutig identifizierenden Namen (z.B. B2B5T2_L00K07X310X100_14_33 für das Rotamer bei dem die erste rotierte Bindung einen Torsionswinkel von 310 Grad hat, die zweite einen Torsionswinkel von 100 Grad) abgespeichert und dann mit Skript 7.3.22 und DOCK starr und *constrained* nach AAV 2 an den Rezeptor gedockt. Die energiegünstigsten Rotamere werden mit Skript 7.3.23 identifiziert.

6.2.4 AAV 4 *Unconstrained docking*

Der zu dockende Ligand muss in keinem bestimmten Format benannt sein, da in diesem Fall das Skript ligand_2_ligandcenter.spl nicht benötigt wird. Es wird eine dock.in-Datei mit den folgenden Einstellungen erzeugt:

Tabelle 26: Auflistung der Parameter, die in allen dock.in-Dateien des *unconstrained docking* identisch waren. Der Parameter random_seed variierte von 1 bis 30. Eine ausführliche Erklärung der einzelnen Parameter befindet sich im DOCK-Handbuch[163] ab Seite 23.

flexible_ligand	yes	repulsive_exponent	12
orient_ligand	no	score_grid_prefix	
score_ligand	yes	../grid_B2B5soft/grid_20082005	
minimize_ligand	yes	flex_definition_file	
multiple_ligands	no	/usr/user/dock/flex.defn	
intermolecular_score	yes	vdw_definition_file	
gridded_score	yes	/usr/user/dock/vdw.defn	
grid_version	4	flex_drive_file	
contact_score	no	/usr/user/dock/flex_drive.tbl	
chemical_score	no	anchor_size	1
energy_score	yes	configurations_per_cycle	5
atom_model	a	torsion_minimize	yes
vdw_scale	1	reminimize_layer_number	2
electrostatic_scale	1	minimize_anchor	no
anchor_search	yes	reminimize_anchor	no
multiple_anchors	yes	reminimize_ligand	yes
write_partial_structures	no	energy_minimize	yes
torsion_drive	yes	initial_translation	1
clash_overlap	0.5	initial_rotation	0.1
write_conformations	yes	initial_torsion	10
write_conformation_total	1	maximum_iterations	100
intramolecular_score	yes	energy_convergence	0.1
energy_cutoff_distance	10	maximum_cycles	2
distance_dielectric	yes	cycle_convergence	1
dielectric_factor	4	energy_termination	1
attractive_exponent	6	random_seed	1

Die Parameter ligand_atom_file und ligand_energy_file werden jeweils individuell festgelegt. Der Ligand wird hier als flexibel angenommen, und diese Flexibilität wird entsprechend der DOCK-anchor_search behandelt (vgl. Abschnitt 1.7.4). Durch die Auswahl von "anchor_size = 1" ist es DOCK möglich, die Konstruktion des Liganden in der Bindungsregion mit jedem beliebigen Fragment des Liganden als Anker zu beginnen. Wie viele Orientierungen DOCK insgesamt erzeugt, hängt hier von der Anzahl der Fragmente, die DOCK als Anker ausprobiert und von der Anzahl der rotierbaren Bindungen im Molekül ab. Um die Rechenzeit gering zu halten ist der Parameter configurations_per_cycle, der definiert, wie detailliert DOCK den Konformationsraum absuchen soll, hier mit 5 angesetzt. Nach dem vollständigen Aufbau eines Liganden erfolgt eine Energieminimierung (vgl. Abschnitt 1.6.2) innerhalb des AMBER Kraftfeldes. Abschließend werden die Orientierungen energiebewertet und die energiegünstigste wird im ligand_energy_file gespeichert.

Das *docking* mit den oben beschriebenen Einstellungen wird 30 Mal durchgeführt, wobei für jedes *docking* der Parameter random_seed um eins erhöht wird. Diese Variation wird automatisiert durch Skripte durchgeführt. Exemplarisch ist in Abschnitt 7.3.24 das Skript dock_B2B5T5_L04K01_random_seed_variation.sh für den Liganden B2B5T5_L04K01 abgedruckt.

7 Anhang

7.1 Glossar englischer Begriffe

Wo die Verwendung deutscher Begriffe unüblich ist, wurden in der vorliegenden Arbeit folgende englischen Begriffe genutzt.

anchor	Anker. Hier: Atom, das an einer bestimmten Position festgehalten wird.
atom ID	Eindeutige, ganzzahlige Identifizierungsnummer eines Atoms im Programm SYBYL.
box	Quaderförmiger Bereich, in dem das Programm DOCK versucht, die Liganden zu platzieren.
constrained docking	*Docking* unter sehr strikten Rahmenbedingungen. Hier: *Docking*, bei dem von vorneherein feststeht, an welchen Punkten die zwei Ankeratome des Liganden sich befinden müssen. Hierfür wurden *critical points* verwendet.
critical	Entscheidend. *Critical points* sind im Rahmen dieser Arbeit Punkte im dreidimensionalen Raum, die der gedockte Liganden unbedingt sättigen muss.
docking	Das Platzieren von Liganden in Bindungstaschen.
file	Datei.
frameworks	Grundgerüste (vgl. Abbildung 2).
grid	Gitter. Hier: Ein virtuelles Gitter von Punkten in der Bindungsregion, für die der Beitrag des Rezeptors zur Energieberechnung bereits vor dem *docking* bestimmt wird.
hit	Treffer. Hier: Ergebnisse der Suche in einer Datenbank.

induced fit	Vorgang bei der Ausbildung eines Rezeptor-Ligand-Komplexes, bei dem die Liganden-Bindungsstelle erst ausgebildet wird, während der Ligand an den Rezeptor bindet.
lead	Leitstruktur. Mögliche Vorstufe eines potentiellen Arzneistoff-Kandidaten.
linker	Brücke. Hier: Flexible Kette von Atomen, die zwei Molekülteile miteinander verbindet.
molecular modeling	Die Untersuchung molekularer Fragestellungen mit Hilfe von Computersimulationen.
random seed	Zahl, mit der ein Zufallszahlengenerator (z.B. im Programm DOCK) initialisiert wird. Bei Auswahl verschiedener Werte für das *random seed* erzeugt der Zufallszahlengenerator verschieden Abfolgen quasi-zufälliger Zahlen.
range	Bereich, Streubreite, Strecke.
ranking	Rangfolge. Ergibt eine Methode eine Vielzahl von Ergebnissen, so kann ein *ranking* nach geeigneten Sortierkriterien (z.B. nach dem Energiewert) helfen, schnell die günstigsten Ergebnisse auszuwählen.
resonance	Resonanz. Hier: Die kernmagnetischen Resonanzen eines Proteins.
scoring	Bewerten von Ergebnissen nach mathematischen Kriterien. Die *score* eines Ergebnisses bestimmt seinen Platz im *ranking*.
scoring functions	Mathematische Funktion mit der Ergebnisse bewertet werden. Hier meist eine empirische Funktion zur Berechnung von freien Bindungsenthalpien.
screening	Verfahren, bei dem ein Pool von Stoffen auf ihre Tauglichkeit als Medikament systematisch geprüft werden.
shapes	Formen. Hier: Die grundlegende sterische Struktur von Molekülen die von Bemis *et al.* beschrieben wird.[25]
spheres	Sphären. Hier: Kugelförmige Bereiche, die Zentren möglicher Wechselwirkung zwischen Ligand und Rezeptor beschreiben.

target	Wirkstoff-Zielverbindung. Biomolekül, das Angriffspunkt für ein (zukünftiges) Medikament ist. Das Target in der vorliegenden Arbeit ist das humane Protein CD4.
top scorer	Ergebnis, das in einem vorangegangenen *ranking* die beste *score* erzielt hat. Hier: Die chemische Modifikation an einem Molekül, die, verglichen mit allen anderen getesteten chemischen Modifikationen, den größten Energiegewinn erzielte.
unconstrained docking	*Docking* unter weitgehend offenen Rahmenbedingungen. Hier: *Docking*, bei dem keine critical_points oder ligand_centers verwendet werden, bei dem also nicht festgelegt ist wo die Ankeratome des Liganden platziert werden müssen.

7.2 Definitionen

Die folgenden Auflistung enthält die im Rahmen dieser Arbeit getroffenen Definitionen und Bezeichnungen. Ein schematischer Überblick über die wichtigsten Schritte der vorliegenden Arbeit befindet sich als Referenz in Abschnitt 3.8.

Anker, Ankerpunkte	Ankerpunkte sind die Atome, die sich an den entscheidenden Punkten des Pharmakophors befinden oder befinden sollen. Ankeratome werden an den Ringsystemen in Abschnitt 3.1.1 definiert und sind in den Abbildungen mit gestrichelten Bindungen gekennzeichnet.
Asp63, D63 (Beispiel)	In dieser Schreibweise, mit normal formatiertem Index, sind die Aminosäuren des CD4 bezeichnet.
B1 - B5	Ankeratome am Dekapeptid NMWQKVGTPL. Die fünf an den Punkten B1 bis B5 lokalisierten Wasserstoffatome definieren in der vorliegenden Arbeit exemplarisch den 3D-Pharmakophor der CD4-GP120-Wechselwirkung.

B2B5T2_L00K01_14_33 (Beispiel)	Am Anker B2Lys modifiziertes Leitstrukturgerüst B2B5T2. Die Leucin-Seite ist unmodifiziert (L00), die Art der Modifikation der Lysin-Seite (K01) ergibt sich aus Abbildung 62. Die Zahlen 14 und 33 bezeichnen die *atom IDs* der Ankeratome.
B2B5T2_L00K07X310 X100_14_33 (Beispiel)	Am Anker B2Lys modifiziertes Leitstrukturgerüst. Die dort substituierte Aminoethylgruppe (vgl. Abbildung 61) hat zwei rotierbare Bindungen, für die die optimalen Torsionswinkel nach AAV 3 zu 310 und 100 Grad bestimmt wurden.
bestätigte *hits*	Die Datenbanksuche in DB5 ergab 943 *hits*. Nur 660 ließen sich mit DOCK *constrained* an CD4 docken. Dies sind die 660 bestätigten *hits*. Die 283 anderen wurden verworfen.
Brücke	Eine Brücke im Sinne dieser Arbeit ist eine flexible Kette von einer bis fünf frei drehbaren Kohlenstoff-Kohlenstoff-Einfachbindungen. Im Programmcode zu dieser Arbeit (Abschnitt 7.3) ist synonym auch das englische Wort "*linker*" für Brücken verwendet.
CR0 - CR8	Ringsysteme (Abbildung 26, Abbildung 28, Abbildung 29).
CR1XL2XCR3 (Beispiel)	RBR, das aus dem Ringsystem CR1, der Brücke L2 und dem Ringsystem CR3 kombiniert wurde.
DB1	Erste Datenbank, die RBR enthält (siehe auch Abbildung 97).
DB2, DB4	Aus DB1 durch systematische Konformationssuche der RBR erzeugte Datenbanken, die Rotamere enthalten (siehe auch Abbildung 97).
DB3, DB5	Aus DB2 und DB4 erzeugte Datenbanken, die eine Auflistung von Ankerpaaren an den RBR in DB1 zusammen mit Angaben darüber, welche Distanzen diese Ankerpaare überbrücken können (siehe auch Abbildung 97), enthalten.
doppelt funktionalisierte RBR	Die doppelt funktionalisierten RBR sind die optimierten Leitstrukturmotive der vorliegenden Arbeit (Schritt VI und VII in Abbildung 98). Sie sind die Kandidaten für Synthese und Bindungs-Assays.
Gerüste, Brücken, Ringe und Seitenketten	Elemente von Wirkstoffmolekülen (siehe Abbildung 1).

L0 - L4	Brücke mit null bis vier C-Atomen (siehe Abbildung 33).
Lys_{429}, K_{429} (Beispiel)	In dieser Schreibweise, mit tief gestelltem Index, sind die Aminosäuren des GP120 beziehungsweise die Aminosäuren des daraus abgeleiteten Dekapeptides NMWQKVGTPL bezeichnet.
NMWQKVGTPL	Von Wülfken aus dem GP120 entwickeltes Dekapeptid der Sequenz Asparagin-Methionin-Tryptophan-Glutamin-Lysin-Valin-Glycin-Threonin-Prolin-Leucin.
RBR	Mit RBR sind im Rahmen dieser Arbeit die in Abschnitt 3.1.3 durch *in silico*-Kombinatorik erzeugten Ring-Brücke-Ring-Systeme abgekürzt. Im Programmcode zu dieser Arbeit (Abschnitt 7.3) ist synonym auch das Wort "cuff" für RBR verwendet.
T1 - T5	Leitstrukturgerüst das beim *docking* in Schritt IV (Abbildung 98) die beste bis fünftbeste Energie erreichte (siehe auch TOP5).
TOP10	Die TOP10 sind die zehn *hits*, die im *energy ranking* der 660 bestätigten *hits* die ersten zehn Plätze belegten.
TOP5	Die TOP5, auch als Leitstrukturgerüste bezeichnet, wurden in Schritt 3.3.5 aus den TOP10 ausgewählt. Die TOP5 wurden als B2B5T1 bis B2B5T5 oder T1 bis T5 bezeichnet (siehe auch T1 - T5).
Verknüpfungspunkte	Die Atome an einem Ringsystem, an denen im Laufe der *in silico*-Kombinatorik Brücken verknüpft werden. Verknüpfungspunkte werden in Abschnitt 3.1.1 definiert und sind in den Abbildungen mit fett gedruckten Bindungen gekennzeichnet.

7.3 Programme und Skripte

Im Folgenden sind die in dieser Arbeit verwendeten Programme und Skripte abgedruckt. Die Art des Quellcodes ist jeweils auch an der Dateiendung zu erkennen. Spl-Dateien sind in der SYBYL *Programming Language* geschrieben, sh-Dateien sind Shell-Skripte, und h- und c-Dateien sind Quellcode in der Programmiersprache C.

7.3.1 combine_all.spl

```
     1: ### combine_all.spl
     2: ### Skript zum Verknuepfen von Ring-Linker-Ring-Systemen
     3: ### Entwicklungsstand v9 18.06.2004 Boris Kroeplien
     4: ###
     5:
     6:
     7:
     8:
     9:
    10: setvar GLOBAL!bkl_logfile %cat("bkl_logfile_"
%FILE_CLEAN_FILENAME("%time()"))
    11:
    12:
    13:
    14:
    15:
    16: uims define macro logbkl sybylbasic YES
    17: echo "$1" >> $GLOBAL!bkl_logfile
    18: .
    19:
    20:
    21:
    22:
    23:
    24: uims define expression_generator bklvalidpart YES
    25:
    26: ### ring or linker must already be loaded in area $1
    27: ### function returns "none" if
    28: ### 1.) the molecule-name does not include an X
    29: ### 2.) dummy-atoms are present
    30: ### 3.) atom 1 is not a C.3
    31: ### 4.) atom 1 has not exactly one neighbour
    32: ### then, if
    33: ### 5.) the set {BKC_ANCHORS} has members
    34: ### the function returns "ring"
    35: ### if {BKC_ANCHORS} has no members, but
    36: ### 6.) atom 2 is a C.3
    37: ### 7.) atom 2 has only one neighbour
    38: ### the function returns "linker"
    39:
    40: LOCALVAR answer
    41:
    42: if %not(%eq($# 1))
    43:   echo "Usage : %bklvalidpart(area_expr)"
    44:   return
    45:  else
    46:    if %not( %atoms( $1 ) )
    47:      echo "Mol Area $1 is empty"
    48:      echo "Usage : bklvalidpart(area_expr)"
    49:      return
    50:     endif
    51:  endif
    52:
    53: setvar answer "ring"
    54:
    55: # test, if in $1 is at all valid
    56:
    57: if %POS("X" %mol_info($1 name))
    58:   setvar answer "none"
    59:   GOTO EarlyExit
    60:  endif
    61: if %atoms($1((<Du>)))
    62:   setvar answer "none"
    63:   GOTO EarlyExit
    64:  endif
    65: if %not(%streql("%ATOM_INFO($1(1) type)" "C.3") )
    66:   setvar answer "none"
    67:   GOTO EarlyExit
    68:  endif
    69: if %neq(%count(%ATOM_INFO($1(1) neighbors)) 1)
    70:   setvar answer "none"
    71:   GOTO EarlyExit
    72:  endif
    73:
    74: # test BKC_ANCHORS. If it has members, the molecule is a ring
```

```
    75:
    76: if %atoms($1({BKC_ANCHORS}) m)
    77:   setvar answer "ring"
    78:   GOTO EarlyExit
    79:  endif
    80:
    81: # now in $1 is no valid ring, it may still be a valid linker
    82: setvar answer "none"
    83: if %streql("%ATOM_INFO($1(2) type)" "C.3")
    84:   if %eq(%count(%ATOM_INFO($1(2) neighbors)) 1)
    85:     setvar answer "linker"
    86:    endif
    87:  endif
    88:
    89: # return either "linker" or "none". "ring" would have been returned
above.
    90:
    91: EarlyExit:
    92:
    93: %return($answer)
    94:
    95: .
    96:
    97:
    98:
    99:
   100:
   101: uims define macro bklcombine sybylbasic YES
   102: logbkl "_____________________________________"
   103: logbkl "Start : %time()"
   104: logbkl "___________bklcombine_____________"
   105:
   106: # setvar cgq_old $cgq_timeout
   107: # setvar cgq_timeout 0
   108:
   109: setvar ring1
   110: setvar ring2
   111: setvar linker
   112:
   113:   SWITCH $#
   114:
   115:     CASE 0)
   116:       setvar ring1[0]
"/usr/user12/bkroep/sybyl/cuffs/databases/ringsA.mdb"
   117:       setvar ring2[0]
"/usr/user12/bkroep/sybyl/cuffs/databases/ringsC.mdb"
   118:       setvar linker[0]
"/usr/user12/bkroep/sybyl/cuffs/databases/linkerB.mdb"
   119:       setvar outputdb
"/usr/user12/bkroep/sybyl/cuffs/databases/output.mdb"
   120:     ;;
   121:     CASE 3)
   122:       setvar ring1[0] $1
   123:       setvar ring2[0] $2
   124:       setvar linker[0] $3
   125:       setvar outputdb
"/usr/user12/bkroep/sybyl/cuffs/databases/output.mdb"
   126:     ;;
   127:     CASE 4)
   128:       setvar ring1[0] $1
   129:       setvar ring2[0] $2
   130:       setvar linker[0] $3
   131:       setvar outputdb $4
   132:     ;;
   133:     CASE)
   134:       %dialog_message( ERROR "usage $0              //            $0
<ringA.mdb><ringB.mdb><linkerC.mdb>              //              $0
<ringA.mdb><ringB.mdb><linkerC.mdb> <output.mdb>" "usage of the macro $0 is
possible without any parameters, with three parameters or with four parameters "
) >$NULLDEV
   135:       logbkl "Early program terminiation, wrong number of commandline-
options"
   136:       return
   137:     ;;
   138:   ENDSWITCH
   139:
   140: # falls die gleiche Ring-Datenbank fuer ring1 und ring2 geoeffnet wurde
   141: # falls also (ring1[0] == ring2[0]) wuerde einfaches Kombinieren auch
```

```
   142: # doppelte Cuff-Systeme erzeugen. Um dies zu verhindern, werden dann
   143: # mit Hilfe der Variablen ringcounter einige ring2 uebersprungen.
   144: # Damit bei verschiedenen ring-datenbanken nichts ausgelassen wird
   145: # wird dann nicht mitgezaehlt, welche Ringe schon kombiniert wurden.
   146: setvar identical_rings 0
   147:
   148: if %streql($ring1[0] $ring2[0])
   149:   setvar identical_rings 1
   150:   logbkl "identical databases for 'ring1' and 'ring2' detected. Creating
only unique combinations"
   151:  else
   152:    setvar identical_rings 0
   153:    logbkl "non identical databases for 'ring1' and 'ring2' detected.
Creating all combinations"
   154:  endif
   155:
   156: # check all necessary databases for contents
   157: # ist das database create ein sicherheitsrisiko ?
   158:
   159: database open $ring1[0] READONLY
   160: if %eq( %count( %database(*) ) 0 )
   161:   %dialog_message( ERROR "No molecules in database" "No Molecules" )
>$NULLDEV
   162:   logbkl "Early program terminiation, empty database for ring1"
   163:   return
   164:  else
   165:    logbkl "Database for Ring1 : $ring1[0]"
   166:  endif
   167:
   168: if %not(%eq($identical_rings 1))
   169:   database open $ring2[0] READONLY
   170:   if %eq( %count( %database(*) ) 0 )
   171:     %dialog_message( ERROR "No molecules in database" "No Molecules" )
>$NULLDEV
   172:     logbkl "Early program terminiation, empty database for ring2"
   173:     return
   174:    else
   175:      logbkl "Database for Ring2 : $ring2[0]"
   176:    endif
   177:    else
   178:      logbkl "Database for Ring2 : $ring2[0]"
   179:    endif
   180:
   181: database open $linker[0] READONLY
   182: if %eq( %count( %database(*) ) 0 )
   183:   %dialog_message( ERROR "No molecules in database" "No Molecules" )
>$NULLDEV
   184:   logbkl "Early program terminiation, empty database for linker"
   185:   return
   186:  else
   187:    logbkl "Database for Linker : $linker[0]"
   188:  endif
   189: if %not(%file_exists($outputdb))
   190:   DATABASE CREATE $outputdb
   191:  endif
   192: logbkl "Database for Output of Cuff-Systems : $outputdb"
   193:
   194: # die folgenden 3x19 Zeilen liessen sich gut in einer Funktion mit
   195: # call_by_reference-uebergabe der parameter &ring1 &ring2 %linker
   196: # vereinfachen.
   197:
   198: DATABASE OPEN $ring1[0] READ
   199: setvar ring1[1] %count(%database(*))
   200: setvar count 2
   201: logbkl "Checking contents of $ring1[0]"
   202: for i in %database(*)
   203:   DATABASE GET $i M1
   204:   if %streql(%bklvalidpart(m1) "ring")
   205:     setvar ring1[$count] $i
   206:     logbkl " $i is a valid %bklvalidpart(m1)"
   207:     setvar count %math($count +1)
   208:    else
   209:      logbkl " $i is not a valid ringsystem, skipped"
   210:      setvar ring1[1] %math($ring1[1] -1)
   211:    endif
   212:  endfor
   213: if %eq($count 2)
   214:   logbkl " No valid ringsystems in $ring1[0]. Program aborted"
```

```
   215:   return
   216:  endif
   217: database close
   218:
   219: ###
   220: ### hier waere eine abfrage, ob identische Datenbanken vorliegen
sinnvoll, hat aber nicht funktioniert.
   221: ### insofern wird in dem fall jetzt halt auch die zweite (gleiche)
database nochmal geprueft.
   222: ###
   223: # logbkl "DEBUG : identical_rings : $identical_rings"
   224: # if %eq($identical_rings 1)
   225: DATABASE OPEN $ring2[0] READ
   226: setvar ring2[1] %count(%database(*))
   227: setvar count 2
   228: logbkl "Checking contents of $ring2[0]"
   229: for i in %database(*)
   230:   DATABASE GET $i M1
   231:   if %streql(%bklvalidpart(m1) "ring")
   232:     setvar ring2[$count] $i
   233:     logbkl " $i is a valid %bklvalidpart(m1)"
   234:     setvar count %math($count +1)
   235:    else
   236:      logbkl " $i is not a valid ringsystem, skipped"
   237:      setvar ring2[1] %math($ring2[1] -1)
   238:    endif
   239:  endfor
   240: if %eq($count 2)
   241:   logbkl " No valid ringsystems in $ring2[0]. Program aborted"
   242:   return
   243:  endif
   244: database close
   245: # else
   246: #   setvar ring2[1] $ring1[1]
   247: #   for count in %range(2 %math($ring2[1]+1))
   248: #     setvar ring2[$count] $ring1[$count]
   249: #    endfor
   250: # endif
   251:
   252: DATABASE OPEN $linker[0] READ
   253: setvar linker[1] %count(%database(*))
   254: setvar count 2
   255: logbkl "Checking contents of $linker[0]"
   256: for i in %database(*)
   257:   DATABASE GET $i M1
   258:   if %streql(%bklvalidpart(m1) "linker")
   259:     setvar linker[$count] $i
   260:     logbkl " $i is a valid %bklvalidpart(m1)"
   261:     setvar count %math($count +1)
   262:    else
   263:      logbkl " $i is not a valid linker-system, skipped"
   264:      setvar linker[1] %math($linker[1] -1)
   265:    endif
   266:  endfor
   267: if %eq($count 2)
   268:   logbkl " No valid linkersystems in $linker[0]. Program aborted"
   269:   return
   270:  endif
   271: database close
   272:
   273: setvar ringcounter 0
   274: setvar cuffcounter 1
   275: setvar cuff_system
   276:
   277: clearscreen
   278:
   279: for r1 in %range(2 %math($ring1[1]+1))
   280:   for li in %range(2 %math($linker[1]+1))
   281:     for r2 in %range(%math(2 + $ringcounter) %math($ring2[1]+1))
   282:       setvar cuff_system[$cuffcounter][1] $ring1[$r1]
   283:       setvar cuff_system[$cuffcounter][2] $linker[$li]
   284:       setvar cuff_system[$cuffcounter][3] $ring2[$r2]
   285:       logbkl "$cuff_system[$cuffcounter]"
   286:       setvar cuffcounter %math($cuffcounter + 1)
   287:      endfor
   288:    endfor
   289:   setvar ringcounter %math($ringcounter + $identical_rings)
   290:  endfor
```

```
   291:
   292: # in den Variablen cuff_system[x][1-3] sind nun fuer eine Kombination x
   293: # das 1.Ringsystem[1], der Linker[2] und das 2.Ringsystem[3] abgelegt.
   294: # nun muessen diese Teile in m1-m3 geladen und kombiniert werden.
   295: # dafuer lauft diese Schleife ueber alle cuff_systems[$cuffcounter]
   296:
   297: logbkl "total number of cuffs to be joined : %math( $cuffcounter - 1 )"
   298: for cuffcounter in %range(1 %math($cuffcounter - 1 ))
   299:   DATABASE OPEN $ring1[0] READ
   300:
   301: ###
   302: ### hier beginnt die Arbeit mit den tatsaechlichen Molekuelteilen,
   303: ### vorher wurden nur Namensteile bearbeitet, kombiniert ...
   304: ###
   305:
   306: # load first ring
   307: DATABASE GET $cuff_system[$cuffcounter][1] m1
   308:
   309: # delete any previous definition of anchors_a
   310: if %atoms(m1({ANCHORS_A}) m)
   311:   REMOVE LOCAL_SET M1(ANCHORS_A) > $NULLDEV
   312:  endif
   313:
   314: # replace the set bkc_anchors with the set anchors_a
   315: DEFINE STATIC_SET ATOM M1({BKC_ANCHORS}) ANCHORS_A """"
   316: REMOVE LOCAL_SET M1(BKC_ANCHORS) > $NULLDEV
   317:
   318: # load second ring
   319: DATABASE OPEN $ring2[0] READ
   320: DATABASE GET $cuff_system[$cuffcounter][3] m2
   321:
   322: # hier muss die info des sets bkc_anchors in Du-Atomen bewahrt werden
...
   323: # die Schreibweise zu finden war sehr zeitaufwendig
   324: for dummieatom in %atoms(m2({BKC_ANCHORS}) m)
   325:   MODIFY ATOM TYPE $dummieatom Du > $NULLDEV
   326:  endfor
   327: REMOVE LOCAL_SET M2(BKC_ANCHORS) > $NULLDEV
   328:
   329: # load linker
   330: DATABASE OPEN $linker[0] READ
   331: DATABASE GET $cuff_system[$cuffcounter][2] m3
   332:
   333: # join parts (erster teil, unkompliziert)
   334: JOIN m3(2) m2(1)
   335:
   336: ###
   337: ### join parts (zweiter teil, kompliziert)
   338: ### if both connecting atoms [m1(1) m3(1)] are not <C.3>, the resulting
bond-type
   339: ###  might be ambigous. so is has to be set to 1
   340: ###
   341:
   342: setvar forcebondtype ""
   343: setvar neighbor_in_m1 %atom_info(M1(1) neighbors)
   344: setvar neighbor_in_m3 %atom_info(M3(1) neighbors)
   345:
   346: if %streql("%atom_info(M1($neighbor_in_m1) type)" "C.2")
   347:   logbkl "neigbour of m1(1) <c.2>"
   348:   if %streql("%atom_info(M3($neighbor_in_m3) type)" "C.2")
   349:     JOIN m1(1) m3(1) 1
   350:    else
   351:      JOIN m1(1) m3(1)
   352:    endif
   353:   else
   354:    JOIN m1(1) m3(1)
   355:  endif
   356:
   357: # delete leftovers
   358: ZAP m2
   359: ZAP m3
   360:
   361: # delete any previous definition of anchors_b
   362: if %atoms(m1({ANCHORS_B}) m)
   363:   REMOVE LOCAL_SET M1(ANCHORS_B) > $NULLDEV
   364:  endif
   365:
   366: # repair atom-types, restore set B
```

```
   367: DEFINE STATIC_SET ATOM M1((<Du>)) ANCHORS_B """"
   368: for dummieatom in %atoms(m1({ANCHORS_B}) m)
   369:   MODIFY ATOM TYPE $dummieatom H > $NULLDEV
   370:  endfor
   371:
   372: # now determine which bonds are rotatable and place them in $bond_list
   373: setvar atom_list %sln_search2d( M1 Any[!f;NOT=OH,NH2,CH3,SH,C(F)(F)F]-
[!r]Any[!f;NOT=OH,NH2,CH3,SH,C(F)(F)F] NoDup Norm 1 2 )
   374: setvar bond_list ""
   375: for i in $atom_list
   376:   setvar atoms %set_unpack( $i )
   377:   setvar origin %arg( 1 $atoms )
   378:   setvar target %arg( 2 $atoms )
   379:   setvar bond %bonds( %cat( "$origin" "=" "$target" ) )
   380:   setvar bond_list $bond_list $bond
   381:  endfor
   382:
   383: # now create the bond_expression 'M1(((15)+19)+22)'
   384: setvar bondexpr  "M1"
   385: for i in $bond_list
   386:   setvar bondexpr %cat($bondexpr "(" )
   387:  endfor
   388: for i in $bond_list
   389:   setvar bondexpr %cat($bondexpr $i ")+" )
   390:  endfor
   391: setvar bondexpr %substr($bondexpr 1 %math(%strlen($bondexpr) - 1))
   392:
   393: # delete any previous definition of BKL_ROTATABLE
   394: if %atoms(m1({BKL_ROTATABLE}) m)
   395:   REMOVE LOCAL_SET M1(BKL_ROTATABLE) > $NULLDEV
   396:  endif
   397: DEFINE STATIC_SET BOND $bondexpr BKL_ROTATABLE """"
   398:
   399: # color the features of the cuff
   400: COLOR BOND $bondexpr ORANGE
   401: COLOR ATOM M1({ANCHORS_A}) RED
   402: COLOR ATOM M1({ANCHORS_B}) GREEN
   403:
   404: # save r-l-r to database
   405: DATABASE CLOSE
   406: DATABASE OPEN $outputdb UPDATE
   407: setvar newmoleculename %cat($cuff_system[$cuffcounter][1] "X"
$cuff_system[$cuffcounter][2] "X" $cuff_system[$cuffcounter][3])
   408: MODIFY MOLECULE NAME M1 $newmoleculename > $NULLDEV
   409:
   410: if %database(*)
   411:   if %set_member($newmoleculename %SET_CREATE(%database(*)))
   412:     DATABASE ADD "M1" REPLACE
   413:    else
   414:      DATABASE ADD "M1"
   415:    endif
   416:  else
   417:    DATABASE ADD "M1"
   418:  endif
   419:
   420: logbkl "$newmoleculename written to database "
   421: ZAP M1
   422:
   423: ### now one cuff is finished, continue with the next
cuff_system[$cuffcounter]
   424:  endfor
   425:
   426: logbkl "__________bklcombine_____________"
   427: logbkl "End   : %time()"
   428: logbkl "______________________________________"
   429:
   430: # setvar cgq_timeout $cgq_old
   431:
   432: .
   433:
   434:
   435:
   436:
   437:
   438:
   439: # bklcombine
   440:
```

7.3.2 minimize_all.spl

```
### minimize_all.spl
### Skript zum Minimieren aller Molekuele einer Datenbank
### Entwicklungsstand v1 07.06.2004 Boris Kroeplien
###

setvar bkl_logfile %cat("bkl_logfile_" %FILE_CLEAN_FILENAME("%time()"))




uims define macro logbkl sybylbasic YES
echo "$1" >> $bkl_logfile
.




uims define macro bklminimize sybylbasic YES

logbkl "_______________________________________"
logbkl "Start : %time()"
logbkl "___________bklminimize_____________"

setvar iterations_max 10000
setvar cgq_old $cgq_timeout
setvar cgq_timeout 0

  SWITCH $#

    CASE 0)
      setvar cuffdb
"/usr/user12/bkroep/sybyl/cuffs/databases/output.mdb"
    ;;
    CASE 1)
      setvar cuffdb $1
    ;;
    ;;
    CASE)
      %dialog_message( ERROR "usage $0              //            $0
<cuff-
databse.mdb>" "usage of the macro $0 is possible without any parameters
or with
a databasename " ) >$NULLDEV
      logbkl "Early program terminiation, wrong number of commandline-
options"
      return
    ;;
  ENDSWITCH

# check database for contents
database open $cuffdb READONLY
if %eq( %count( %database(*) ) 0 )
  %dialog_message( ERROR "No molecules in database" "No Molecules" )
>$NULLDEV
  logbkl "Early program terminiation, empty database"
  return
 else
   logbkl "Database for minimization : $cuffdb"
 endif

logbkl "Maximum number of iterations per Cuff-System : $iterations_max"

# setup for the minimization-engine
SETVAR TAILOR!MAXIMIN2!MINIMIZATION_METHOD POWELL
SETVAR TAILOR!MAXIMIN2!TERMINATION_OPTION GRADIENT
SETVAR TAILOR!MAXIMIN2!MIN_ENERGY_CHANGE  0.050
SETVAR TAILOR!MAXIMIN2!RMS_GRADIENT  0.050
SETVAR TAILOR!MAXIMIN2!RMS_DISPLACEMENT  0.001
SETVAR TAILOR!MAXIMIN2!MAXIMUM_ITERATIONS $iterations_max
SETVAR TAILOR!FORCE_FIELD!FF_CHOICE TRIPOS
SETVAR TAILOR!FORCE_FIELD!PARAMETER_SET Tripos
SETVAR TAILOR!FORCE_FIELD!NON_BONDED_CUTOFF 8.000000
SETVAR TAILOR!FORCE_FIELD!DIELECTRIC_CONSTANT 1.000000
```

```
 71: SETVAR TAILOR!FORCE_FIELD!DIELECTRIC_FUNCTION distance
 72: SETVAR TAILOR!FORCE_FIELD!REVIEW_HS_AND_LPS DO_NOT_CHANGE
 73: SETVAR TAILOR!MAXIMIN2!LIST_TERMS NO
 74: SETVAR TAILOR!FORCE_FIELD!ONE_FOUR_SCALING 1.000000
 75: SETVAR TAILOR!FORCE_FIELD!HBOND_RAD_SCALING 0.700000
 76:
 77: DATABASE OPEN $cuffdb UPDATE
 78: for i in %database(*)
 79:   DATABASE GET "$i" m1
 80:   logbkl "$i loaded from database "
 81:   ENERGY M1 Electrostatics IGNORE_ELECTROSTATICS Interesting M1(*-(0))
 82: Boundary_Conditions IGNORE_PBCS List DONE
 83:   setvar energie_vorher $ENERGY_TOTAL
 84:   MAXIMIN2 M1 Electrostatics IGNORE_ELECTROSTATICS Interesting M1(*-(0))
 85: Boundary_Conditions IGNORE_PBCS List DONE INTERACTIVE
 86:   logbkl "$i minimized $energie_vorher -> $ENERGY_TOTAL"
 87:   DATABASE ADD "M1" REPLACE
 88:   logbkl "$i written to database "
 89:  endfor
 90: database close
 91:
 92: logbkl "__________bklminimize_____________"
 93: logbkl "End   : %time()"
 94: logbkl "______________________________________"
 95:
 96: setvar cgq_timeout $cgq_old
 97:
 98: .
 99:
100:
101:
102:
103:
104: # bklminimize
105:
```

7.3.3 setupsearch_all.spl

```
  1: ### setupsearch_all.spl
  2: ### Skript zum Vorbereiten der syst.Suche aller Molekuele einer Datenbank
  3: ### Entwicklungsstand v11 14.07.2004 Boris Kroeplien
  4: ###
  5:
  6: setvar bkl_logfile %cat("bkl_logfile_" %FILE_CLEAN_FILENAME("%time()"))
  7:
  8:
  9:
 10:
 11:
 12: uims define macro logbkl sybylbasic YES
 13: echo "$1" >> $bkl_logfile
 14: .
 15:
 16:
 17:
 18:
 19:
 20: uims define macro bklsearchall sybylbasic YES
 21:
 22: logbkl "______________________________________"
 23: logbkl "Start : %time()"
 24: logbkl "___________bklsearchall_____________"
 25:
 26: setvar bkl_angle_increment 10
 27: setvar bkl_energy_limit 9999
 28:
 29: logbkl "Parameters : "
 30: logbkl " Angle-Increment $bkl_angle_increment"
 31: logbkl " Energy-limit $bkl_energy_limit"
 32:
 33: setvar cgq_old $cgq_timeout
 34: setvar cgq_timeout 0
 35:
 36: ### handle the command-line-parameters
 37:
```

```
    38:   SWITCH $#
    39:     CASE 0)
    40:       setvar cuffdb "/usr/user12/bkroep/sybyl/cuffs/databases/output.mdb"
    41:     ;;
    42:     CASE 1)
    43:       setvar cuffdb $1
    44:     ;;
    45:     ;;
    46:     CASE)
    47:       %dialog_message( ERROR "usage $0              //            $0 <cuff-
databse.mdb>" "usage of the macro $0 is possible without any parameters or with a databasename
" ) >$NULLDEV
    48:       logbkl "Early program terminiation, wrong number of commandline-options"
    49:       return
    50:     ;;
    51:   ENDSWITCH
    52:
    53: ### check database for contents
    54: database open $cuffdb READONLY
    55: if %eq( %count( %database(*) ) 0 )
    56:   %dialog_message( ERROR "No molecules in database" "No Molecules" ) >$NULLDEV
    57:   logbkl "Early program terminiation, empty database"
    58:   return
    59:  else
    60:   logbkl " Database for systematic-searches : $cuffdb"
    61:  endif
    62: database close
    63:
    64: ### loop over database, load cuff_systems
    65: DATABASE OPEN $cuffdb UPDATE
    66: for current_cuff in %database(*)
    67:
    68:   DATABASE GET "$current_cuff" m1
    69:   logbkl "$current_cuff loaded from database "
    70: ### creating a filename (and dir) for the search
    71:   setvar search_dataset %cat($current_cuff "_search")
    72:   setvar search_dataset %basename($search_dataset)
    73:   if %file_exists($search_dataset)
    74:     setvar deleted  %file_delete($search_dataset)
    75:     logbkl " $deleted file $search_dataset existed. Will be rewritten"
    76:    endif
    77: # diese 2 zeilen wieder aktivieren, wenn doch in directories geschrieben werden soll
    78: #  dann aber muessen nach dem ende des netbatch-runs alle files
    79: #  CRaXLbXCRcX_search.* nach CRaXLbXCRcX_search/ verschoben werden
    80: #  dabei werden dann *.inp und *.sta ueberschrieben
    81: # setvar dir_made %file_make_dir($search_dataset)
    82: # setvar search_dataset %cat($search_dataset/ $search_dataset)
    83:
    84:   logbkl " Filenames for search-data : $search_dataset"
    85:
    86: ### setting up the search-engine
    87:   SET AUTOSAVE OFF
    88:   SEARCH SETUP M1 ROTATABLE_BONDS DEFINE M1({BKL_ROTATABLE}) |^
    89:   SEARCH SETUP M1 ROTATABLE_BONDS STATUS INACTIVE M1({BKL_ROTATABLE}) ^
    90:   SEARCH SETUP M1 ENERGY ENERGY $bkl_energy_limit NO_ELECTROSTATICS |^
    91:   setvar rotbondid 0
    92:   for bond_id in %bonds({BKL_ROTATABLE})
    93:     setvar rotbondid %math($rotbondid + 1)
    94:     SEARCH SETUP M1 ANGLES INCREMENT $rotbondid $bkl_angle_increment |^
    95:     SEARCH SETUP M1 ANGLES RANGES $rotbondid 0 359 |||^
    96:    endfor
    97: ### if (ring1 == ring2) take only unique positions into the distance-map
    98: ### identical_rings will be used as increment in the later creation of the distance-
map
    99:   setvar lefstring
   100:   setvar middlelinker
   101:   setvar rightring
   102:   setvar identical_rings
   103:   setvar teile $current_cuff
   104:   setvar leftring %split("X" $teile)
   105:   setvar middlelinker %split("X" "")
   106:   setvar rightring %split("X" "")
   107:   if %streql($leftring $rightring)
   108:     logbkl " identical rings for 'ring1' and 'ring2' detected. only uniqe
positionpairs will be in distance-map "
   109:     setvar identical_rings 1
   110:    else
   111:      setvar identical_rings 0
```

```
   112:      logbkl " non-identical rings for 'ring1' and 'ring2' detected. all positionpairs
will be in distance-map "
   113:    endif
   114: ### construct the string, needed for the definition of the distance-map
   115:   setvar mapstring "0.01"
   116:   setvar mapdimensions 0
   117:   setvar explainstring "ColumnTitles will be : "
   118:   setvar position_A 0
   119:   for part_a in %atoms({ANCHORS_A})
   120:     setvar position_A %math($position_A +1)
   121:     setvar position_B 0
   122:     for part_b in %atoms({ANCHORS_B})
   123:       setvar position_B %math($position_B +1)
   124:       if %eq($identical_rings 1)
   125:         if %lt(%math($position_B - $position_A) 0)
   126:           continue
   127:          endif
   128:        endif
   129:       setvar pair_ab %cat("DEFINE $part_a $part_b")
   130:       setvar mapstring %cat("$mapstring $pair_ab")
   131:       setvar mapdimensions %math($mapdimensions + 1)
   132:       setvar explainstring %cat("$explainstring DistDim $mapdimensions : [")
   133:       setvar explainstring %cat("$explainstring %atom_info(M1($part_a) ID FULLNAME)")
   134:       setvar explainstring %cat("$explainstring --> %atom_info(M1($part_b) ID
FULLNAME)]")
   135:      endfor
   136:    endfor
   137: ### set up the distance-map
   138:   SEARCH SETUP M1 DISTANCE_MAP NONE ^
   139:   SEARCH SETUP M1 DISTANCE_MAP CREATE_ONLY $mapstring | ^
   140:   logbkl " creating a $mapdimensions-dimensional distance-map"
   141:   logbkl " $explainstring"
   142: ### set up the remaining search-parameters and HOLD the search
   143:   SEARCH SETUP M1 ANGLES OUTPUT_STATUS RECORD | ^
   144:   SEARCH SETUP M1 ROTATABLE_BONDS STATUS ACTIVE M1({BKL_ROTATABLE}) ^
   145:   SEARCH SETUP M1 ANGLES REFERENCE ZEROED |^
   146:   SEARCH SETUP m1 DO_SEARCH $search_dataset sgi12.chemie.uni-hamburg.de HOLD_FOR_LATER
   147: #   SEARCH SETUP m1 DO_SEARCH $search_dataset sgi12.chemie.uni-hamburg.de RUN_IT_NOW
   148:   SET AUTOSAVE ON
   149:   logbkl "$current_cuff prepared for search "
   150:   DATABASE ADD "M1" REPLACE
   151:   logbkl "$current_cuff written to database "
   152: ### now continue with next $current_cuff from database
   153:  endfor
   154:
   155: database close
   156:
   157: logbkl "__________bklsearchall___________"
   158: logbkl "End   : %time()"
   159: logbkl "_____________________________________"
   160:
   161: setvar cgq_timeout $cgq_old
   162:
   163: .
   164:
   165:
   166:
   167:
   168:
   169:
   170: # bklsearchall
   171:
```

7.3.4 startsearches_all.spl

```
     1: ### startsearches_all.spl
     2: ### Skript zum Starten des syst.Suchen aller Molekuele einer Datenbank
     3: ### Entwicklungsstand v3 14.07.2004 Boris Kroeplien
     4: ###
     5:
     6: setvar bkl_logfile %cat("bkl_logfile_" %FILE_CLEAN_FILENAME("%time()"))
     7:
     8:
     9:
    10:
```

```
    11:
    12: uims define macro logbkl sybylbasic YES
    13: echo "$1" >> $bkl_logfile
    14: .
    15:
    16:
    17:
    18:
    19:
    20: uims define macro bklstartsearches sybylbasic YES
    21:
    22: logbkl "_____________________________________"
    23: logbkl "Start : %time()"
    24: logbkl "________bklstartsearches___________"
    25:
    26: setvar cgq_old $cgq_timeout
    27: setvar cgq_timeout 0
    28: setvar intervallmin 1
    29:
    30:   SWITCH $#
    31:     CASE 0)
    32:       setvar cuffdb "/usr/user12/bkroep/sybyl/cuffs/databases/output.mdb"
    33:     ;;
    34:     CASE 1)
    35:       setvar cuffdb $1
    36:     ;;
    37:     ;;
    38:     CASE)
    39:       %dialog_message( ERROR "usage $0              //            $0 <cuff-
databse.mdb>" "usage of the macro $0 is possible without any parameters or with a databasename
" ) >$NULLDEV
    40:       logbkl "Early program terminiation, wrong number of commandline-options"
    41:       return
    42:     ;;
    43:   ENDSWITCH
    44:
    45: # check database for contents
    46: database open $cuffdb READONLY
    47: if %eq( %count( %database(*) ) 0 )
    48:   %dialog_message( ERROR "No molecules in database" "No Molecules" ) >$NULLDEV
    49:   logbkl "Early program terminiation, empty database"
    50:   return
    51:  else
    52:   logbkl "Database for searches : $cuffdb"
    53:  endif
    54:
    55: ### Die aktuelle Zeit feststellen und zerlegen
    56: setvar teile %time()
    57: setvar timestring_WEEK %split(" " "$teile")
    58: setvar timestring_MONTH %split(" " "")
    59: setvar timestring_DAY %split(" " "")
    60: setvar timestring_TIME %split(" " "")
    61: setvar timestring_YEAR %split(" " "")
    62: setvar timestring_HOUR %split(":" "$timestring_TIME")
    63: setvar timestring_MINUTE %split(":" "")
    64: setvar DDMMMYYYY %cat("$timestring_DAY $timestring_MONTH $timestring_YEAR")
    65: setvar nextminute $timestring_MINUTE
    66: setvar nexthour $timestring_HOUR
    67:
    68: ### da das Skript noch ein paar Minuten laufen wird, sollen die ersten Suchen erst in
2 Minuten beginnen
    69: setvar nextminute %math($nextminute + 3 )
    70: if %gteq($nextminute 60)
    71:   setvar nextminute %math($nextminute - 60)
    72:   setvar nexthour %math($nexthour +1)
    73:   if %gteq($nexthour 24)
    74:     echo("Program can not be run around midnight, sorry. Try again later")
    75:     exit
    76:    endif
    77:  endif
    78:
    79: DATABASE OPEN $cuffdb UPDATE
    80: for i in %database(*)
    81:   DATABASE GET "$i" m1
    82:   logbkl "$i loaded from database "
    83:
    84:   setvar search_dataset %cat($i "_search")
    85:   setvar search_dataset %basename($search_dataset)
```

```
    86:
    87: # diese zwei Zeilen muessen wieder aktiviert werden, falls die Suchergebnisse
    88: # in eigenen Unterverzeichnissen abgelegt werden sollen
    89: # Auswertung ist erst moeglich, wenn ale files zu einer Suche in einer dir sind.
    90: # setvar dir_made %file_make_dir($search_dataset)
    91: # setvar search_dataset %cat($search_dataset/ $search_dataset)
    92:
    93:   SET AUTOSAVE OFF
    94: # als test hat geklappt :
    95: # SEARCH SETUP M1 DO_SEARCH dumdidumdidum sgi12.chemie.uni-hamburg.de RUN_IT_NOW
    96:
    97: # sofortiges Ausfuehren ging mit :
    98: # SEARCH SETUP M1 DO_SEARCH $search_dataset sgi12.chemie.uni-hamburg.de RUN_IT_NOW
    99:
   100:   setvar nextminute %math($nextminute + $intervallmin)
   101:   if %gteq($nextminute 60)
   102:     setvar nextminute %math($nextminute - 60)
   103:     setvar nexthour %math($nexthour +1)
   104:       if %gteq($nexthour 24)
   105:         echo("Program can not be run around midnight, sorry. Try again later")
   106:         exit
   107:        endif
   108:    endif
   109:
   110:   setvar nextstarttime %cat("$DDMMMYYYY $nexthour $nextminute")
   111:
   112:   logbkl "systematic search will be started on $i at $nextstarttime"
   113:
   114:   if %FILE_EXISTS($search_dataset.log)
   115:     logbkl "files for search $search_dataset exist, overwriting ..."
   116:     SEARCH SETUP M1 DO_SEARCH $search_dataset "sgi12.chemie.uni-hamburg.de" START_TIME
$nextstarttime |
   117:    else
   118:      logbkl "no previous results for search $search_dataset present ..."
   119:      SEARCH SETUP M1 DO_SEARCH $search_dataset "sgi12.chemie.uni-hamburg.de"
START_TIME $nextstarttime |
   120:    endif
   121:
   122:   SET AUTOSAVE ON
   123:   DATABASE ADD "M1" REPLACE
   124:   logbkl "$i written to database "
   125:  endfor
   126:
   127: database close
   128:
   129: logbkl "________bklstartsearches___________"
   130: logbkl "End   : %time()"
   131: logbkl "_____________________________________"
   132:
   133: logbkl "_____________________________________"
   134: logbkl "Start : %time()"
   135: logbkl "______status of the netbatch_______"
   136:
   137: netbatch list jobs >> $bkl_logfile
   138: wait 15
   139: echo %system("at -l") >> $bkl_logfile
   140: wait 15
   141:
   142: logbkl "______status of the netbatch_______"
   143: logbkl "End   : %time()"
   144: logbkl "_____________________________________"
   145:
   146: setvar cgq_timeout $cgq_old
   147:
   148: .
   149:
   150:
   151:
   152:
   153:
   154: # bklstartsearches
   155:
```

7.3.5 go_all.spl

```
     1: ### go_all.spl
     2: ### Skript das verschieden Skripte nacheinander ausfuehrt. s.unten ...
     3: ### Entwicklungsstand v1 17.07.2004 Boris Kroeplien
     4: ###
     5:
     6: TAKE /usr/user12/bkroep/sybyl/spl/combine_all.spl
     7: TAKE /usr/user12/bkroep/sybyl/spl/minimize_all.spl
     8: TAKE /usr/user12/bkroep/sybyl/spl/setupsearch_all.spl
     9: TAKE /usr/user12/bkroep/sybyl/spl/startsearches_all.spl
    10:
    11: setvar ring_database "/usr/user12/bkroep/sybyl/cuffs/databases/rings_CR0_CR8.mdb"
    12: setvar link_database "/usr/user12/bkroep/sybyl/cuffs/databases/linker_L0_L4.mdb"
    13: setvar cuff_database
"/usr/user12/bkroep/sybyl/cuffs/databases/cuffs_CR0_CR8_L0_L4.mdb"
    14:
    15: bklcombine \
    16:   $ring_database \
    17:   $ring_database \
    18:   $link_database \
    19:   $cuff_database
    20:
    21: bklminimize \
    22:   $cuff_database
    23:
    24: bklsearchall \
    25:   $cuff_database
    26:
    27: bklstartsearches \
    28:   $cuff_database
    29:
```

7.3.6 export_dirs_csv.spl

```
     1: ### export_dirs_csv.spl
     2: ### Skript zum Exportieren der Ergebnisse von systematischen Suchen ins csv-format
     3: ### jetzt neu, basierend auf den files selbst, nicht mehr auf der Datenbank
     4: ### Entwicklungsstand 06.09.2004 BKL
     5: ###
     6:
     7: setvar bkl_logfile %cat("bkl_logfile_" %FILE_CLEAN_FILENAME("%time()"))
     8:
     9:
    10:
    11:
    12:
    13: uims define macro logbkl sybylbasic YES
    14: echo "$1" >> $bkl_logfile
    15: .
    16:
    17:
    18:
    19:
    20:
    21: uims define macro bklexportallcsv sybylbasic YES
    22:
    23: logbkl "_______________________________________"
    24: logbkl "Start : %time()"
    25: logbkl "________bklexportallcsv_____________"
    26:
    27: setvar cgq_old $cgq_timeout
    28: setvar cgq_timeout 0
    29:
    30: setvar currentdir %system("/usr/bin/pwd")
    31:
    32: for angfile in %system("/bin/ls *_search/*.ang")
    33:   setvar slashpos %pos("/" $angfile)
    34:   setvar search_dataset %substr($angfile "1" %math($slashpos - 1))
    35: # code to be run on the molecule to be added here ...
    36:   setvar analyzecommand %cat($search_dataset " $currentdir/" "$search_dataset" /
"$search_dataset" " " )
    37:   logbkl "opening $search_dataset"
    38:   SEARCH ANALYZE TABLE CREATE_NEW_TABLE $analyzecommand "$search_dataset" m1 ||
```

```
    39:   setvar csvfile %cat($search_dataset ".csv")
    40:   setvar exportcommand %cat(%basename($search_dataset) " * * CSV " $csvfile)
    41:   if %FILE_EXISTS($csvfile)
    42:     logbkl "  re-exporting $csvfile"
    43:     TABLE EXPORT $exportcommand
    44:    else
    45:      logbkl "     exporting $csvfile"
    46:      TABLE EXPORT $exportcommand
    47:    endif
    48:   TABLE CLOSE %basename($search_dataset) NO
    49:  endfor
    50:
    51: logbkl "________bklexportallcsv____________"
    52: logbkl "End   : %time()"
    53: logbkl "_____________________________________"
    54:
    55: setvar cgq_timeout $cgq_old
    56:
    57: .
    58:
    59:
    60:
    61:
    62:
    63: # bklexportallcsv
    64:
```

7.3.7 csv_2_energy_all.sh

```
     1: #!/bin/tcsh
     2: # csv_2_energy_all.sh
     3: # Arbeitstitel war "awkskript1"
     4: # Dieses Skript liest alle .csv-Files und speichert ihre minimalen
     5: # und maximalen Energiewerte in .energy-Files
     6:
     7: if ( $#argv != "0" ) then
     8:   echo "usage $0 <no parameters>"
     9:   exit
    10:  endif
    11:
    12: # echo "" >! ./energytemp
    13: # echo "" >! ./currentenergies
    14:
    15: foreach n (`ls -1 *.csv | awk 'BEGIN { FS=" " } {print $1}'`)
    16:   @ x = 0
    17:   @ x =`echo $n | awk 'BEGIN { FS="L" } {print $2}' | awk 'BEGIN { FS="X" } {print
$1}'`
    18:   @ x = $x + 2
    19:   cat $n | nawk 'BEGIN {FS =","}{print $"'"$x"'"}' | sort -n >! ./energytemp
    20:   head -1 ./energytemp >! ./currentenergies
    21:   tail -1 ./energytemp >> ./currentenergies
    22:   mv ./currentenergies `echo $n | awk 'BEGIN { FS="." } {print $1".energy"}'`
    23:  end
    24:
    25: exit
    26:
```

7.3.8 writelogs_all.spl

```
     1: ### writelogs_all.spl
     2: ### Skript zum Nachproduzieren aller Infos darueber, welche
     3: ### Distdim welchem Abstand in dem jeweiligen Molekuel entspricht.
     4: ### basierend auf der Datenbank der fertigen Cuffs
     5: ### Entwicklungsstand v11 14.07.2004 Boris Kroeplien
     6: ###
     7:
     8: setvar GLOBAL!bkl_logfile %cat("bkl_logfile_" %FILE_CLEAN_FILENAME("%time()"))
     9:
    10:
    11:
    12:
```

```
    13:
    14: uims define macro logbkl sybylbasic YES
    15: echo "$1" >> $bkl_logfile
    16: .
    17:
    18:
    19:
    20:
    21:
    22: uims define macro bklwritelogs sybylbasic YES
    23:
    24: logbkl "_____________________________________"
    25: logbkl "Start : %time()"
    26: logbkl "___________bklwritelogs_____________"
    27:
    28: setvar cgq_old $cgq_timeout
    29: setvar cgq_timeout 0
    30:
    31: ### handle the command-line-parameters
    32:   SWITCH $#
    33:     CASE 0)
    34:       setvar cuffdb "/usr/user12/bkroep/sybyl/cuffs/databases/output.mdb"
    35:     ;;
    36:     CASE 1)
    37:       setvar cuffdb $1
    38:     ;;
    39:     ;;
    40:     CASE)
    41:       %dialog_message( ERROR "usage $0              //            $0 <cuff-
databse.mdb>" "usage of the macro $0 is possible without any parameters or with a databasename
" ) >$NULLDEV
    42:       logbkl "Early program terminiation, wrong number of commandline-options"
    43:       return
    44:     ;;
    45:   ENDSWITCH
    46:
    47: ### check database for contents
    48: database open $cuffdb READONLY
    49: if %eq( %count( %database(*) ) 0 )
    50:   %dialog_message( ERROR "No molecules in database" "No Molecules" ) >$NULLDEV
    51:   logbkl "Early program terminiation, empty database"
    52:   return
    53:  else
    54:    logbkl " Database for systematic-searches : $cuffdb"
    55:  endif
    56: database close
    57: ### loop over database, load cuff_systems
    58: DATABASE OPEN $cuffdb UPDATE
    59: for current_cuff in %database(*)
    60:   DATABASE GET "$current_cuff" m1
    61:   logbkl "$current_cuff loaded from database "
    62: ### creating a filename (and dir) for the search
    63:   setvar search_dataset %cat($current_cuff "_search.map")
    64:   setvar search_dataset %basename($search_dataset)
    65:   if %file_exists($search_dataset)
    66:     setvar deleted  %file_delete($search_dataset)
    67:     logbkl " $deleted file $search_dataset existed. Will be rewritten"
    68:    endif
    69:   logbkl " Filenames for search-data : $search_dataset"
    70: ### if (ring1 == ring2) take only unique positions into the distance-map
    71: ### identical_rings will be used as increment in the later creation of the distance-
map
    72:   setvar lefstring
    73:   setvar middlelinker
    74:   setvar rightring
    75:   setvar identical_rings
    76:   setvar teile $current_cuff
    77:   setvar leftring %split("X" $teile)
    78:   setvar middlelinker %split("X" "")
    79:   setvar rightring %split("X" "")
    80:   if %streql($leftring $rightring)
    81:     logbkl " identical rings for 'ring1' and 'ring2' detected. only uniqe
positionpairs will be in distance-map "
    82:     setvar identical_rings 1
    83:    else
    84:      setvar identical_rings 0
    85:      logbkl " non-identical rings for 'ring1' and 'ring2' detected. all positionpairs
will be in distance-map "
```

```
    endif
### construct the string, needed for the definition of the distance-map
  setvar tempfilename %find_temp_filename(./tmp .tmp 12345)
  setvar mapstring "0.01"
  setvar mapdimensions 0
  setvar explainstring "ColumnTitles will be : "
  setvar position_A 0
  for part_a in %atoms({ANCHORS_A})
    setvar position_A %math($position_A +1)
    setvar position_B 0
    for part_b in %atoms({ANCHORS_B})
      setvar position_B %math($position_B +1)
      if %eq($identical_rings 1)
        if %lt(%math($position_B - $position_A) 0)
          continue
        endif
       endif
      setvar pair_ab %cat("DEFINE $part_a $part_b")
      setvar mapstring %cat("$mapstring $pair_ab")
      setvar mapdimensions %math($mapdimensions + 1)
      setvar explainstring %cat(" %atom_info(M1($part_a) ID FULLNAME)")
      setvar explainstring %cat("$explainstring --> %atom_info(M1($part_b) ID
FULLNAME)")
      echo "%printf("%03d %s" $mapdimensions "$explainstring")" >> $tempfilename
     endfor
   endfor
  echo $mapdimensions >> $search_dataset
  setvar tempvar %system("/sbin/cat $tempfilename >> $search_dataset")
  setvar tempvar %file_delete($tempfilename)
### set up the distance-map
  logbkl " creating a $mapdimensions-dimensional distance-map for $current_cuff"
### now continue with next $current_cuff from database
 endfor
database close

logbkl "___________bklwritelogs___________"
logbkl "End   : %time()"
logbkl "_____________________________________"

setvar cgq_timeout $cgq_old

.




# bklwritelogs

```

7.3.9 csv_2_distdb.c

```
/* dies ist csv_2_distdb, Version 0006*/


/*** einzubindende Header-Dateien ***/
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include <time.h>
  #include "bkutil.h"


/*** definierte Konstanten ***/
  /* siehe bkutil.h */


/*** definierte Variablentypen ***/

 struct DISTDIMTYPE{
   int id;
   char description[80];
```

```
 24:    double mindist;
 25:    double maxdist;
 26:    struct DISTDIMTYPE *nextdist;
 27:    };
 28:
 29:  struct CUFFTYPE{
 30:    int id;
 31:    char name[20];
 32:    struct CUFFTYPE *nextcuff;
 33:    };
 34:
 35:  struct MAINPOINERTYPE{
 36:    /*static*/
 37:    char energycut[20];
 38:    int nummcuffs;
 39:    struct CUFFTYPE *firstcuff;
 40:    struct CUFFTYPE *lastcuff;
 41:    /*dynamic*/
 42:    struct DISTDIMTYPE *firstdist;
 43:    struct DISTDIMTYPE *lastdist;
 44:    int numdistdims;
 45:    int numcolumns;
 46:    int energycolumn;
 47:    double e_min;
 48:    double e_max;
 49:    double new_e_max;
 50:    long int number_of_lines;
 51:    long int number_of_good_lines;
 52:    };
 53:
 54: /*** Funktionsprototypen ***/
 55:   void anleitung(void);
 56:   void read_allcuffs(char *sourcefilename, struct MAINPOINERTYPE *mainpointers);
 57:   void read_mapfile(char *sourcefilename, struct MAINPOINERTYPE *mainpointers);
 58:   void read_firstlinecsv(char *sourcefilename, struct MAINPOINERTYPE *mainpointers);
 59:   void read_energyfile(char *sourcefilename, struct MAINPOINERTYPE *mainpointers);
 60:   void read_fullcsv(char *sourcefilename, struct MAINPOINERTYPE *mainpointers);
 61:   void write_distdb(char *targetfilename, struct MAINPOINERTYPE *mainpointers);
 62:   void set_new_e_max(struct MAINPOINERTYPE *mainpointers);
 63:   void clear_dynamic_mainpointers(struct MAINPOINERTYPE *mainpointers);
 64:
 65: /*** Variablendeklaration vor main (GLOBAL) ***/
 66:   int DEBUG = 0; /* global */
 67:   FILE *in_stream; /* global */
 68:   struct MAINPOINERTYPE mainpointers; /* global */
 69:
 70:
 71:
 72:   /* ********** VVV ********** MAIN ********** VVV ********** */
 73:   /* ********** VVV ********** MAIN ********** VVV ********** */
 74:   /* ********** VVV ********** MAIN ********** VVV ********** */
 75:
 76: /*** main ***/
 77:  int main(int argc, char *argv[])
 78:    {
 79:    char dateiname[256];
 80:    char singlechar;
 81:    int i;
 82:    struct CUFFTYPE *currentcuffptr;
 83:    char mapfilename[256];
 84:    char csvfilename[256];
 85:    char energyfilename[256];
 86:    char distfilename[256];
 87:    time_t starttime, point1time, point2time, endtime;
 88:    /* int count;
 89:    double dummiedouble;
 90:    char pufferstring[256];*/
 91:
 92:    /* note the starttime*/
 93:    time(&starttime);
 94:
 95:    /* set default values (in case, no parameters are given) */
 96:    strcpy(dateiname, "allcuffs_list.txt");
 97:    strcpy(mainpointers.energycut,"3k");
 98:
 99:    /* Manage given commandline parameters */
100:    if (argc == 0)
101:      {
```

```
   102:      /* keine paramater */
   103:      }
   104:     else
   105:       {
   106:       /* Parse the command line arguments. The i loop starts at 1 because
   107:        * argument 0 is the name of the executable. */
   108:       for(i=1;i<argc;i++)
   109:         {
   110:         /* Parse the '-' switch arguments */
   111:         if (argv[i][0] == '-')
   112:           {
   113:           /* Look at the letter after the '-' */
   114:           switch(argv[i][1])
   115:             {
   116:             case 'i':
   117:               if ((!argv[i+1]) || (strstr(argv[i+1], "-")))
   118:                 {
   119:                 printf("Error: No input file specified. Please specify an input-file
or leave off the -i switch to use the default enery-limit allcuffs_list.txt. Exiting.\n");
   120:                 exit(1);
   121:                 }
   122:
   123:               if (strstr(argv[i+1], "*"))
   124:                 {
   125:                 printf("Error: Invalid input file name. Exiting.\n");
   126:                 exit(1);
   127:                 }
   128:
   129:               if (strlen(argv[i+1]) > 256)
   130:                 {
   131:                 printf("Error: Buffer overrun: The name of the input file must be less
than 256 characters long. Exiting.\n");
   132:                 exit(1);
   133:                 }
   134:                else
   135:                  {
   136:                  strcpy(dateiname, argv[i+1]);
   137:                  }
   138:               break;
   139:
   140:             case 'e':
   141:               if ((!argv[i+1]) || (strstr(argv[i+1], "-")))
   142:                 {
   143:                 printf("Error: No enery-limit specified. Please specify an enery-limit
or leave off the -e switch to use the default enery-limit. Exiting.\n");
   144:                 exit(1);
   145:                 }
   146:
   147:              singlechar = argv[i+1][strlen(argv[i+1])-1];
   148:              if ((singlechar != 'p') && (singlechar != 'k'))
   149:                {
   150:                printf("Error: Invalid format. Give energy-limit as 15p (percent) or as
3k (kcal). Exiting.\n");
   151:                exit(1);
   152:                }
   153:
   154:              if ( strlen(argv[i+1]) > 20)
   155:                {
   156:                printf("Error: Buffer overrun: The name of the output file must be less
than %d characters long. Exiting.\n",20);
   157:                exit(1);
   158:                }
   159:               else
   160:                 {
   161:                 strcpy(mainpointers.energycut,argv[i+1]);
   162:                 }
   163:
   164:              break;
   165:
   166:            default:
   167:              printf("Error: Unknown option %s, exiting.\n", argv[i]);
   168:              anleitung();
   169:              exit(1);
   170:
   171:             }/* end switch*/
   172:           }/* end (if '-') */
   173:         } /* end for*/
   174:      } /* end if argc > 0 */
```

```
   175:
   176:    read_allcuffs(dateiname, &mainpointers);
   177:
   178:    /* log_1 */
   179:    /* Befehlszeile, Datum, Zeit, dateiname */
   180:    printf("\n\n csv_distdb by BKL (2005) \n");
   181:    printf(" Time : %.24s.\n", ctime(&starttime));
   182:    for(i=0;i<argc;i++)
   183:      {
   184:      printf(" %s",argv[i]);
   185:      }
   186:    printf("\n");
   187:    printf(" Listenfile : %s (%5d cuffs von %s -
%s)\n",dateiname,mainpointers.nummcuffs,(*(mainpointers.firstcuff)).name,(*(mainpointers.lastc
uff)).name);
   188:
   189:    /* DEBUG while cuffs.next != null do  ...*/
   190:    currentcuffptr = mainpointers.firstcuff;
   191:    while(currentcuffptr != NULL)
   192:      {
   193:      /*note startime for this cuff*/
   194:      time(&point1time);
   195:
   196:      /* log_2 */
   197:      /* cuffid, cuffname  */
   198:      printf("\n   Cuff %5d (of %5d) : %s\n", (*currentcuffptr).id,
mainpointers.nummcuffs, (*currentcuffptr).name);
   199:
   200:      strcpy(mapfilename, (*currentcuffptr).name );
   201:      strcat(mapfilename, ".map");
   202:      read_mapfile(mapfilename, &mainpointers);
   203:      strcpy(energyfilename, (*currentcuffptr).name );
   204:      strcat(energyfilename, ".energy");
   205:      read_energyfile(energyfilename, &mainpointers);
   206:
   207:      set_new_e_max(&mainpointers);
   208:
   209:      /* log_3 */
   210:      /* energy */
   211:      printf("   Energy : min = %.3f, max = %.3f, cut = %s
(%.3f)\n",mainpointers.e_min, mainpointers.e_max, mainpointers.energycut,
mainpointers.new_e_max);
   212:
   213:      strcpy(csvfilename, (*currentcuffptr).name );
   214:      strcat(csvfilename, ".csv");
   215:      read_firstlinecsv(csvfilename, &mainpointers);
   216:
   217:      /* log_4 */
   218:      /* numdistdims, numangles  */
   219:      printf("   Number of Angles: %3d, Number of Distances :
%3d\n",(mainpointers.energycolumn -1),mainpointers.numdistdims);
   220:
   221:      strcpy(distfilename, (*currentcuffptr).name );
   222:      strcat(distfilename, ".db");
   223:
   224:      /* printf("Ausgangswerte (1.Zeile mit niedriger Energie)\n");
   225:      write_distdb("stdout", &mainpointers);*/
   226:
   227:      /* log_5 */
   228:      /* flenames */
   229:      printf("   csv-file : %s, output-file : %s\n",csvfilename,distfilename);
   230:
   231:      read_fullcsv(csvfilename, &mainpointers);
   232:
   233:      /* printf("\n Werte nach dem Einlesen aller Zeilen\n");
   234:      write_distdb("stdout", &mainpointers);*/
   235:
   236:      write_distdb(distfilename, &mainpointers);
   237:
   238:      /* printf("new : %f \n",mainpointers.new_e_max);*/
   239:
   240:      currentcuffptr = (*currentcuffptr).nextcuff;
   241:      /* clear_dynamic_mainpointers(&mainpointers);*/
   242:
   243:      /*note endtime for this cuff*/
   244:      time(&point2time);
   245:
   246:      /* log_6 */
```

```
   247:      /* # of lines, time_spent */
   248:      printf("   Number of Rotamers : %ld (%ld used), time spent : %.0f
seconds\n\n",mainpointers.number_of_lines,mainpointers.number_of_good_lines,
difftime(point2time,point1time));
   249:
   250:      }/* DEBUG end while nextcuff != NULL */
   251:
   252:    puts(" --- Calculation finished --- \n\n");
   253:
   254:    /* note the endtime*/
   255:    time(&endtime);
   256:
   257:    printf(" Total Time Spent %.0f seconds\n",difftime(endtime,starttime));
   258:
   259:    /* log_7 */
   260:    /* Datum, Zeit*/
   261:    printf(" csv_distdb by BKL (2005) \n");
   262:    printf(" Time : %.24s.\n", ctime(&endtime));
   263:    /*
   264:    printf("%d \n",mainpointers.nummcuffs );
   265:    printf("%d \n",(*(mainpointers.lastcuff)).id);
   266:    printf("%d \n",(*(mainpointers.firstcuff)).id);
   267:    printf("%s \n",(*(mainpointers).lastdist).description);
   268:    printf("%lf \n",(*(mainpointers).lastdist).maxdist);*/
   269:
   270:    printf("\n\n");
   271:    exit(1);
   272:
   273:    return(0);
   274:    } /* end main */
   275:
   276:   /* ********** ^^^ ********** MAIN ********** ^^^ ********** */
   277:   /* ********** ^^^ ********** MAIN ********** ^^^ ********** */
   278:   /* ********** ^^^ ********** MAIN ********** ^^^ ********** */
   279:
   280:
   281:
   282: /*** Funktionen ***/
   283:
   284:  void anleitung(void)
   285:   {
   286:   puts("\n csv_2_distdb (c) 2005 by BKL"); /*DEBUG, better argv0*/
   287:   puts("Usage : csv_2_distdb [...]");
   288:   puts("Usage : csv_2_distdb -i # (inputfile)");
   289:   puts("Usage : csv_2_distdb -e #p (energylimit_delta in %)");
   290:   puts("Usage : csv_2_distdb -e #k (energylimit_delta in kcal/mol)");
   291:   puts("\n");
   292:   exit(EXIT_FAILURE);
   293:   }
   294:
   295:
   296:
   297:  void read_allcuffs(char *sourcefilename, struct MAINPOINERTYPE *mainpointers)
   298:   {
   299:   extern FILE *in_stream;
   300:   struct CUFFTYPE *newcuffptr;
   301:   int count, forelastposition;
   302:
   303:   /*** DATEI OEFFNEN ***/
   304:   in_stream = fopen(sourcefilename, "r");
   305:   if (in_stream == NULL)
   306:     {
   307:     fprintf(stderr, "Fehler beim Oeffnen der Datei %s .\n",sourcefilename);
   308:     exit(EXIT_FAILURE);
   309:     }
   310:
   311:   /*** Einlesen der Cuff-Anzahl ***/
   312:   fscanf(in_stream, "%d", &((*mainpointers).nummcuffs));
   313:
   314:   /*** Einlesen des ersten Cuff-Eintrages ***/
   315:   newcuffptr = (struct CUFFTYPE *)malloc(sizeof(struct CUFFTYPE));
   316:   checkmalloc(newcuffptr);
   317:   (*newcuffptr).id = 1;
   318:
   319:   fgets((*newcuffptr).name, 20, in_stream); /* linefeed nach der anzahl der cuffs.*/
   320:   fgets((*newcuffptr).name, 20, in_stream);
   321:
```

```
   322:   /*in der folgenden zeile wird ein eventuelles linefeed aus dem string geloescht ...
in funct auslagern ?*/
   323:   forelastposition = (strlen((*newcuffptr).name)-1); if
((*newcuffptr).name[forelastposition] == 10) { (*newcuffptr).name[forelastposition] = 0; }
   324:
   325:   (*mainpointers).firstcuff = newcuffptr;
   326:   (*mainpointers).lastcuff = newcuffptr;
   327:
   328:   /* printf("Current Cuff is : %s \n",(*newcuffptr).name);*/
   329:
   330:   /*** Einlesen der weiteren Cuff-Eintraege ***/
   331:   for (count=2; count<=(*mainpointers).nummcuffs; count++)
   332:     {
   333:     newcuffptr = (struct CUFFTYPE *)malloc(sizeof(struct CUFFTYPE));
   334:     checkmalloc(newcuffptr);
   335:     (*newcuffptr).id = count;
   336:     fgets((*newcuffptr).name, 20, in_stream); /* liest aus Datei */
   337:
   338:     /*in der folgenden zeile wird ein eventuelles linefeed aus dem string geloescht
... in funct auslagern ?*/
   339:     forelastposition = (strlen((*newcuffptr).name)-1); if
((*newcuffptr).name[forelastposition] == 10) { (*newcuffptr).name[forelastposition] = 0; }
   340:
   341:     (*(*mainpointers).lastcuff).nextcuff = newcuffptr; /* verzeigert die liste */
   342:     (*mainpointers).lastcuff = newcuffptr; /* vermerkt das neue lastcuff */
   343:     } /* end (for count) */
   344:
   345:   (*(*mainpointers).lastcuff).nextcuff = NULL; /* beendet die liste */
   346:
   347:   fclose(in_stream);
   348:   } /* end read_allcuffs */
   349:
   350:
   351:
   352:  void read_mapfile(char *sourcefilename, struct MAINPOINERTYPE *mainpointers)
   353:   {
   354:   extern FILE *in_stream;
   355:   struct DISTDIMTYPE *newdistptr;
   356:   int count, forelastposition;
   357:
   358:   /*** DATEI OEFFNEN ***/
   359:   in_stream = fopen(sourcefilename, "r");
   360:   if (in_stream == NULL)
   361:     {
   362:     fprintf(stderr, "Fehler beim Oeffnen der Datei %s .\n",sourcefilename);
   363:     exit(EXIT_FAILURE);
   364:     }
   365:
   366:   fscanf(in_stream, "%d", &(*mainpointers).numdistdims);
   367:   /* printf("%d ",(*mainpointers).numdistdims );*/
   368:
   369:   /*** Einlesen des ersten Dist-Eintrages ***/
   370:   newdistptr = (struct DISTDIMTYPE *)malloc(sizeof(struct DISTDIMTYPE));
   371:   checkmalloc(newdistptr);
   372:
   373:   fgets((*newdistptr).description, 20, in_stream); /*Ueberspringen von junk, wohl
vorallem char 10*/
   374:   fscanf(in_stream, "%d",  &(*newdistptr).id);
   375:   fgets((*newdistptr).description, 80, in_stream);
   376:
   377:   /*in der folgenden zeile wird ein eventuelles linefeed aus dem string geloescht ...
in funct auslagern ?*/
   378:   forelastposition = (strlen((*newdistptr).description)-1); if
((*newdistptr).description[forelastposition] == 10) {
(*newdistptr).description[forelastposition] = 0; }
   379:
   380:   (*mainpointers).firstdist = newdistptr ;
   381:   (*mainpointers).lastdist = newdistptr ;
   382:
   383:   /*** Einlesen der weiteren Cuff-Eintraege ***/
   384:   for (count=2; count<=(*mainpointers).numdistdims; count++)
   385:     {
   386:     newdistptr = (struct DISTDIMTYPE *)malloc(sizeof(struct DISTDIMTYPE));
   387:     checkmalloc(newdistptr);
   388:     fscanf(in_stream, "%d",  &(*newdistptr).id); /*  entspricht :
(*newdistptr).id = count;  */
   389:     fgets((*newdistptr).description, 80, in_stream);/* liest aus Datei */
   390:
```

```
   391:     /*in der folgenden zeile wird ein eventuelles linefeed aus dem string geloescht
... in funct auslagern ?*/
   392:     forelastposition = (strlen((*newdistptr).description)-1); if
((*newdistptr).description[forelastposition] == 10) {
(*newdistptr).description[forelastposition] = 0; }
   393:
   394:     (*(*mainpointers).lastdist).nextdist = newdistptr; /* verzeigert die liste */
   395:     (*mainpointers).lastdist = newdistptr; /* vermerkt das neue lastdist */
   396:     } /* end (for count) */
   397:
   398:   (*(*mainpointers).lastdist).nextdist = NULL; /* beendet die liste */
   399:
   400:   fclose(in_stream);
   401:   } /* end read_mapfile */
   402:
   403:
   404:
   405:  void read_firstlinecsv(char *sourcefilename, struct MAINPOINERTYPE *mainpointers)
   406:   {
   407:   extern FILE *in_stream;
   408:   char singlechar;
   409:   char isfirst;
   410:   int count;
   411:   double dummiedouble,energy;
   412:   struct DISTDIMTYPE *currentdist;
   413:
   414:   /*** DATEI OEFFNEN ***/
   415:   in_stream = fopen(sourcefilename, "r");
   416:   if (in_stream == NULL)
   417:     {
   418:     fprintf(stderr, "Fehler beim Oeffnen der Datei %s .\n",sourcefilename);
   419:     exit(EXIT_FAILURE);
   420:     }
   421:
   422:   count = 0;
   423:   do /* Zeichenweise die erste Zeile einlesen */
   424:     {
   425:     singlechar = fgetc(in_stream);
   426:     if (singlechar ==',') /* Anzahl der ',' in count zaehlen */
   427:       {
   428:       count++;
   429:       }
   430:     }
   431:   while (singlechar != 10); /* Lesen bis Zeilenende */
   432:
   433:   (*mainpointers).numcolumns = count + 1;
   434:   (*mainpointers).energycolumn = (*mainpointers).numcolumns -
(*mainpointers).numdistdims;
   435:
   436:   /* printf("\ndebug cols : numcols %d, numdist %d, energycol
%d\n",(*mainpointers).numcolumns,(*mainpointers).numdistdims,(*mainpointers).energycolumn);*/
   437:
   438:   rewind(in_stream); /* Dateizeiger wieder auf Anfang */
   439:
   440:   isfirst = TRUE;
   441:   do
   442:     {
   443:     if (!(isfirst)) /* read until end of line */
   444:       {
   445:       do /* Zeichenweise die erste Zeile einlesen */
   446:         {
   447:         singlechar = fgetc(in_stream);
   448:         }
   449:       while (singlechar != 10); /* Lesen bis Zeilenende */
   450:       /* DEBUG puts("ueberlesen");*/
   451:       }
   452:     /* die winkel-spalten ueberlesen */
   453:     for (count=1; count<=((*mainpointers).energycolumn - 1); count++)
   454:       {
   455:       fscanf(in_stream, "%lf,", &dummiedouble);/*das komma im formatstring ist immens
wichtig ...*/
   456:       /* printf("Winkel %03d : %f \n",count, dummiedouble); */
   457:       }
   458:
   459:     /* die energy-spalte lesen */
   460:     fscanf(in_stream, "%lf", &energy);/* kein komma im formatstring  !!! */
   461:     isfirst = FALSE;
   462:     }
```

```
   463:   while (energy > (*mainpointers).new_e_max); /*liest, bis eine Zeile mit geringer
Energie gefunden wird.*/
   464:
   465:   /* DEBUG printf("Energie der initial verwendeten Zeile  : %f \n", energy);*/
   466:
   467:   currentdist = (*mainpointers).firstdist; /* count=1;*/
   468:
   469:   /* einlesen der Entfernungen der ersten Zeile, Zuweisen zu min und max*/
   470:   while (currentdist != NULL)
   471:     {
   472:     fscanf(in_stream, ",%lf", &dummiedouble);/*das komma im formatstring ist immens
wichtig ...*/
   473:     (*currentdist).mindist = dummiedouble;
   474:     (*currentdist).maxdist = dummiedouble;
   475:     currentdist = (*currentdist).nextdist;
   476:     /* if (count == 2)
   477:       printf("min&max (%03d) set to : %f \n",count, dummiedouble);
   478:       count++;*/
   479:     } /* (end while) *//* bis Listenende */
   480:
   481:   /* printf("\n\n Problem : es wurde immer 350.0000 0.0000 ausgegeben. Irgendwie kommt
er beim Einlesen nicht in der Zeile voran ... seit dem komma im formatstring ist das
gelöst.\n\n");*/
   482:
   483:   fclose(in_stream);
   484:   } /* end read_firstlinecsv*/
   485:
   486:
   487:
   488:  void read_energyfile(char *sourcefilename, struct MAINPOINERTYPE *mainpointers)
   489:   {
   490:   extern FILE *in_stream;
   491:
   492:   /*** DATEI OEFFNEN ***/
   493:   in_stream = fopen(sourcefilename, "r");
   494:   if (in_stream == NULL)
   495:     {
   496:     fprintf(stderr, "Fehler beim Oeffnen der Datei %s .\n",sourcefilename);
   497:     exit(EXIT_FAILURE);
   498:     }
   499:
   500:   fscanf(in_stream, "%lf", &((*mainpointers).e_min));
   501:   fscanf(in_stream, "%lf", &((*mainpointers).e_max));
   502:
   503:   /* printf("min : %f \n",(*mainpointers).e_min);
   504:   printf("max : %f \n",(*mainpointers).e_max);*/
   505:
   506:   fclose(in_stream);
   507:   } /* end read_energyfile */
   508:
   509:
   510:
   511:  void read_fullcsv(char *sourcefilename, struct MAINPOINERTYPE *mainpointers)
   512:   {
   513:   extern FILE *in_stream;
   514:   int count;
   515:   double energy, dummiedouble;
   516:   struct DISTDIMTYPE *currentdist;
   517:   char singlechar;
   518:
   519:   /*** DATEI OEFFNEN ***/
   520:   in_stream = fopen(sourcefilename, "r");
   521:   if (in_stream == NULL)
   522:     {
   523:     fprintf(stderr, "Fehler beim Oeffnen der Datei %s .\n",sourcefilename);
   524:     exit(EXIT_FAILURE);
   525:     }
   526:
   527:   (*mainpointers).number_of_lines = 0;
   528:   (*mainpointers).number_of_good_lines = 0;
   529:   /*** FILE "DURCHLESEN" ***/
   530:   while (! feof(in_stream))
   531:     {
   532:     (*mainpointers).number_of_lines++;
   533:     /* die winkel-spalten ueberlesen */
   534:     for (count=1; count<=((*mainpointers).energycolumn - 1); count++)
   535:       {
```

```
   536:       fscanf(in_stream, "%lf,", &dummiedouble);/*das komma im formatstring ist immens
wichtig ...*/
   537:       /* printf("Winkel %03d : %f \n",count, dummiedouble); */
   538:       }
   539:
   540:     /* die energy-spalte lesen */
   541:     fscanf(in_stream, "%lf", &energy);/* kein komma im formatstring  !!! */
   542:     /* printf("Energie : %f \n", energy);*/
   543:     if (energy > (*mainpointers).new_e_max)
   544:       { /* energie dieser zeile zu hoch, weiter zur nächsten zeile */
   545:       do /* Zeichenweise die erste Zeile einlesen */
   546:         {
   547:         singlechar = fgetc(in_stream);
   548:         }
   549:       while (singlechar != 10); /* Lesen bis Zeilenende */
   550:       /* puts("Zeile ueberlesen"); */
   551:       }
   552:      else
   553:        {  /* energie dieser zeile ist tief genug, also einlesen */
   554:        (*mainpointers).number_of_good_lines++;
   555:        currentdist = (*mainpointers).firstdist;
   556:        /**/count = 1;
   557:        while ( (currentdist != NULL) && (! feof(in_stream)) )
   558:          {
   559:          fscanf(in_stream, ",%lf", &dummiedouble);/*das komma im formatstring ist
immens wichtig ...*/
   560:          /* printf("D%02d : %f ",count, dummiedouble); */
   561:          /* max = (a>b) ?a :b; */
   562:          /* (*currentdist).mindist = ((*currentdist).mindist < dummiedouble) ?
(*currentdist).mindist : dummiedouble;
   563:             (*currentdist).maxdist = ((*currentdist).maxdist > dummiedouble) ?
(*currentdist).maxdist : dummiedouble;
   564:          */
   565:          if (dummiedouble < (*currentdist).mindist)
   566:            {
   567:
   568:            /* printf("min lowered from %f to
%f\n",(*currentdist).mindist,dummiedouble);*/
   569:
   570:            (*currentdist).mindist = dummiedouble;
   571:
   572:            }
   573:
   574:          if (dummiedouble > (*currentdist).maxdist)
   575:            {
   576:            /* if (count == 2)
   577:               printf("max raised from %f to %f (energy =
%f)\n",(*currentdist).maxdist,dummiedouble,energy);*/
   578:           (*currentdist).maxdist = dummiedouble;
   579:            }
   580:          /*
   581:          if (count == 1)
   582:          printf("%d count : %d min : %f, max : %f, dist: %f\n\n",count1, count,
(*currentdist).mindist,(*currentdist).maxdist,dummiedouble);
   583:          */
   584:          currentdist = (*currentdist).nextdist;
   585:          count++;
   586:          } /* (end while) bis Listenende */
   587:        } /* end else */
   588:      singlechar = fgetc(in_stream);
   589:      if ((singlechar != 10) && (!feof(in_stream)))
   590:        { /* nur linefeeds sollen ""beiseite gelesen" werden */
   591:        ungetc(singlechar,in_stream);
   592:        }
   593:     } /* ! feof(in_stream) */
   594:   fclose(in_stream);
   595:   } /* end read_fullcsv*/
   596:
   597:
   598:
   599:  void write_distdb(char *targetfilename, struct MAINPOINERTYPE *mainpointers)
   600:   {
   601:   /* int count;*/
   602:   struct DISTDIMTYPE *currentdist;
   603:   FILE *targetfile;
   604:
   605:   if (strcmp(targetfilename,"stdout")== 0)
   606:     targetfile= stdout;
```

```
   607:    else
   608:      {
   609:      targetfile = fopen(targetfilename, "w");
   610:      if (in_stream == NULL)
   611:        {
   612:        fprintf(stderr, "Fehler beim Oeffnen der Datei %s .\n",targetfilename);
   613:        exit(EXIT_FAILURE);
   614:        }
   615:      }
   616:   currentdist = (*mainpointers).firstdist;
   617:
   618:   /* count =1;*/
   619:   while (currentdist != NULL)
   620:     {
   621:     fprintf(targetfile, "%2d  %3.3f    %3.3f    %s\n", (*currentdist).id,
(*currentdist).mindist,(*currentdist).maxdist,(*currentdist).description);
   622:     currentdist = (*currentdist).nextdist;
   623:     /* count++;*/
   624:     } /* end else */
   625:   } /* end write_distdb */
   626:
   627:
   628:
   629:  void set_new_e_max(struct MAINPOINERTYPE *mainpointers)
   630:   {
   631:   int stringlaenge;
   632:   char energy_argstring[20];
   633:   double energy_argdouble;
   634:   double energyspan;
   635:
   636:   stringlaenge = (int)strlen((*mainpointers).energycut);
   637:
   638:   /* zahl vor dem p oder dem k in energy_argdouble speichern */
   639:   strncpy(energy_argstring,(*mainpointers).energycut,(stringlaenge - 1));
   640:   energy_argstring[stringlaenge-1]=0;
   641:   energy_argdouble = atof(energy_argstring);
   642:
   643:   /* feststellen, oder absolute kcal oder prozente gerechnet werden sollen*/
   644:   switch((*mainpointers).energycut[stringlaenge - 1])
   645:     {
   646:     case 'p':
   647:       energyspan = ((*mainpointers).e_max - (*mainpointers).e_min)*(energy_argdouble
/100);
   648:       break;
   649:
   650:     case 'k':
   651:       energyspan = energy_argdouble;
   652:       break;
   653:
   654:     default:
   655:       printf("Error: Unknown option for energy given : %c\n",
(*mainpointers).energycut[stringlaenge - 1]);
   656:       anleitung();
   657:       exit(1);
   658:     } /* end switch*/
   659:
   660:   (*mainpointers).new_e_max = (*mainpointers).e_min + energyspan;
   661:   }/* end set_new_e_max*/
   662:
   663:
   664:
   665:  void clear_dynamic_mainpointers(struct MAINPOINERTYPE *mainpointers)
   666:   {
   667:   (*mainpointers).firstdist = NULL;
   668:   (*mainpointers).lastdist = NULL;
   669:   (*mainpointers).numdistdims =0;
   670:   (*mainpointers).numcolumns =0;
   671:   (*mainpointers).energycolumn=0;
   672:   (*mainpointers).e_min=0;
   673:   (*mainpointers).e_max=0;
   674:   (*mainpointers).new_e_max=0;
   675:   (*mainpointers).number_of_lines=0;
   676:   }
   677:
   678:
   679:
   680: /* end file */
   681:
```

7.3.10 bkutil.h / bkutil.c

bkutil.h

```
 1: /* dies ist bkutil.h */
 2:
 3:
 4:
 5: /*** einzubindende Header-Dateien ***/
 6:
 7: /*** definierte Konstanten ***/
 8:   #ifndef NULL
 9:    #define NULL    0
10:   #endif
11:
12:   #ifndef JA
13:    #define JA      1
14:    #define NEIN    0
15:   #endif
16:
17:  #ifndef TRUE
18:   #define TRUE     1
19:   #define FALSE    0
20:  #endif
21:
22:  #ifndef PI
23:   #define PI 3.14159265
24:  #endif
25:
26:
27:
28: /*** definierte Variablentypen ***/
29:  struct MOLECULETYPE; /* in order to avoid first mentioning a struct in a prototype */
30:
31:
32:
33: /*** Funktionsprototypen ***/
34:  int bk_isdigit(char character);
35:  int bk_wrapped(char *stringvariable);
36:  double abs_double (double signednumber);
37:  char nonzero_double (double dummienumber);
38:  void checkmalloc (void *address);
39:  int num_fields(char* fullstring);
40:
41:
42:
43: /* end file */
44:
```

bkutil.c

```
 1: /* dies ist bkutil.c */
 2:
 3:
 4:
 5: /*** einzubindende Header-Dateien ***/
 6:   #include <stdio.h>
 7:   #include <stdlib.h>
 8:   #include <string.h>
 9:   #include "bkutil.h"
10:
11:
12:
13: /*** definierte Konstanten ***/
14:
15: /*** definierte Variablentypen ***/
16:
17: /*** Funktionsprototypen ***/
18:      /* stehen in bkutil.h */
19:
20: /*** Funktionen ***/
21:
22:  int bk_isdigit(char character)
```

```
    23:   /* diese Funktion uebernimmt einen Char und gibt 1 zurueck, falls sein ASCII-Wert
zwischen 48 und 57 inclusive also zwischen '0' und '9' liegt. Somit muss ctype.h nicht
included werden. */
    24:   {
    25:   return ((character >= 48) && (character <= 57));
    26:   }/*( end function bk_isdigit) */
    27:
    28:
    29:
    30:  int bk_wrapped(char *stringvariable)
    31:   /* diese Funktion uebernimmt einen Zeiger (call by reference) auf einen String.
    32:   - sie kuerzt den String (keine Kopie) auf 6 Zeichen,
    33:   - prueft fuer jedes Zeichen, ob es eine Ziffer ist
    34:   - und wandelt gegebenfalls den String in ein int um.
    35:   - dieses integer gibt sie zurueck, oder "-1", falls die Umwandlung scheitert.*/
    36:   {
    37:   int oldstringlength, newstringlength, typfehler = 0, charposition = 0;
    38:
    39:   oldstringlength = strlen(stringvariable);
    40:
    41:   /* newstringlength bekommt den kleineren der Werte oldstringlength oder 6 zugewiesen
*/
    42:   newstringlength = (oldstringlength < 6) ? oldstringlength : 6;
    43:   while ( (charposition < newstringlength) && (!(typfehler)) )
    44:     {
    45:     typfehler = (!(bk_isdigit(stringvariable[charposition])));
    46:     charposition++;
    47:     } /* (end while) */
    48:   if (typfehler)
    49:     { return (-1); }
    50:    else
    51:      {
    52:      stringvariable[charposition]='\0';
    53:      return (atoi(stringvariable));
    54:      }
    55:   /* diese Zeile kann nicht erreicht werden (wg. if ... else ...) */
    56:   }/*( end function bk_wrapped) */
    57:
    58:
    59:
    60:  double abs_double (double signednumber)
    61:   /* übernimmt eine double-zahl, liefert ihren betrag zuruck (double)*/
    62:   {
    63:   if (signednumber >= 0) return (signednumber);
    64:   else return (-(signednumber));
    65:   }
    66:
    67:
    68:
    69:  char nonzero_double (double dummienumber)
    70:   /* übernimmt eine double-zahl, liefert true zuruck, wenn sie "nicht null" ist*/
    71:   {
    72:   if (abs_double(dummienumber) < 0.00001) return (FALSE);
    73:    else return (TRUE);
    74:   }
    75:
    76:
    77:
    78:  void checkmalloc (void *address)
    79:   /* übernimmt einen beliebigen Zeiger. Beendet das Programm, wenn er NULL ist*/
    80:   {
    81:   if (address != NULL) return;
    82:    else
    83:      {
    84:      fprintf(stderr, "\n\nError while allocating RAM\a\n");
    85:      exit(EXIT_FAILURE);
    86:      }
    87:   }
    88:
    89:
    90:
    91:  int num_fields(char* fullstring)
    92:   /* fullstring wird hier "by reference" uebergeben (pointer) */
    93:   /* da strtok den string zerlegt, wird zuerst in dummystring eine kopie angefertigt
*/
    94:   /* besser waere ein dynamisch allozierter dummystring mit malloc...*/
    95:   /* so ist die Funktion auf strings der Groesse <=256 Zeichen beschraenkt */
    96:   /* es muss string.h included werden */
```

```
   97:   {
   98:   char * pointertostring;
   99:   char dummystring[256];
  100:   int count = 0;
  101:
  102:   strcpy(dummystring,fullstring);
  103:   pointertostring = strtok(dummystring," ");
  104:   while (pointertostring != NULL)
  105:     {
  106:     count++;
  107:     pointertostring = strtok (NULL, " ");
  108:     }
  109:   return count;
  110:   }
  111:
  112:
  113:
  114: /* end file */
  115:
```

7.3.11 multidb_2_database.sh

```
    1: #!/bin/tcsh
    2: # multidb_2_database.sh
    3: # Arbeitstitel war "awkskript2"
    4: # Dieses Skript fuegt die .db-Files aller Cuffs zu einer grossen
    5: # Datenbank-Tabelle zusammen.
    6:
    7: if ( $#argv != "0" ) then
    8:   echo "usage $0 <no parameters>"
    9:   exit
   10:  endif
   11:
   12: setenv energylimit `pwd | awk 'BEGIN { FS="_" } {print $(NF-1)}'`
   13:
   14: echo "00" >! ./tempcounter
   15:
   16: foreach n (`ls -1 CR*.db | awk 'BEGIN { FS=" " } {print $1}' | awk 'BEGIN { FS="." }
{print $1}'`)
   17:   setenv cuffname `echo $n |  awk 'BEGIN { FS="_" } {print $1}'`
   18:   setenv energyname `echo $n.energy`
   19:   setenv energymin `head -1 ../$energyname`
   20:   setenv energymax `tail -1 ../$energyname`
   21:   setenv energyminforgrep `echo $energymin`
   22:   if (`echo $energymin` != `echo $energymin | awk 'BEGIN { FS="-" } {print $1}'`) then
   23:     setenv energyminforgrep `echo $energymin | awk 'BEGIN { FS="-" } {print $2}'`
   24:    endif
   25:   setenv energycutval `cat ./*.log | grep "Energy :" | awk -v energymin="$energymin" -
v energymax=" $energymax" 'BEGIN { FS="[,() ]"} {if ((NF==19) && ($6=="min") && ($10=="max")
&& ($8==energymin) && ($12==energymax)) print $18}'`
   26:   setenv totalcounter `cat ./tempcounter`
   27:   cat ./$n.db | awk -v count="$totalcounter" -v cuff="$cuffname" -v
energycut="$energylimit" -v energymin="$energymin" -v energymax="$energymax" -v
energycutval="$energycutval" 'BEGIN{s=count}{s=s+1}{print
cuff,energycut,energymin,energymax,energycutval,s,$0} END{print s > "./tempcounter" }'
   28:  end
   29:
   30: exit
   31:
```

7.3.12 dist_2_count.sh

```
    1: #!/bin/tcsh
    2: # dist_2_count.sh
    3: # Arbeitstitel war "awkskript3_wc_l"
    4: # Dieses Skript fuehrt eine Suche in der Datenbank durch. Es gibt die
    5: # Anzahl
 der Ankerpaare aus, die den Suchkriterien entsprechen.
    6:
    7: if ( $#argv != "2" ) then
    8:  echo "usage $0 <distance> <allowed deviation>"
```

```
     9:  exit
    10: endif
    11:
    12: setenv distance $1
    13: setenv deviation $2
    14:
    15: cat allcuffs_table_1k_22022005 | awk -v distance="$distance" -v deviation="$deviation"
'BEGIN { FS=" "} {if ((NF==14) && (distance>=($8-deviation)) && (distance<=($9+deviation)))
print $0}' | wc -l
    16:
    17: exit
    18:
```

7.3.13 dist_2_rotamerlist.sh

```
     1: #!/bin/tcsh
     2: # dist_2_rotamerlist.sh
     3: # Arbeitstitel war "ph_dist_2_mol2_v2"
     4: # Dieses Skript fuehrt eine Suche in der Datenbank durch.
     5: # Fuer jedes Ankerpaar, das den Suchkriterien entspricht, sucht es im
     6: # entsprechenden csv-File ein Rotamer günstiger Energie. Ausgabe ist eine Liste
     7: # von Zeilen, aus den csv-Files, die dann vom SPL-Skript rotamerlist_2_manymol2s
     8: # in mol2-files umgewandelt werden können.
     9: # Stand 25.04.2005, BKL
    10:
    11: if ( $#argv != "2" ) then
    12:  echo "usage $0 <distance> <allowed deviation>"
    13:  exit
    14: endif
    15:
    16: echo "" >! ./first_cufflist
    17:
    18: setenv distance $1
    19: setenv deviation $2
    20: setenv search "_search.csv"
    21:
    22: echo "# starting up ..."
    23: echo "# retrieving lines from ../allcuffs_table_1k_22022005"
    24:
    25: cat ../allcuffs_table_1k_22022005 | awk -v distance="$distance" -v
deviation="$deviation" 'BEGIN { FS=" "} {if ((NF==14) && (distance>=($8-deviation)) &&
(distance<=($9+deviation))) print $0}' >! ./first_cufflist
    26:
    27: echo "# evaluating lines"
    28:
    29: @ counter = 0
    30:
    31: foreach n("`cat ./first_cufflist`")
    32:   @ counter = $counter + 1
    33:   setenv cuff `echo $n | awk '{print $1}'`
    34:   setenv csvname `echo $cuff$search`
    35:   setenv linker `echo $cuff | awk 'BEGIN { FS="X"}{print$2}' | awk 'BEGIN {
FS="L"}{print$2}'`
    36:   setenv ecut `echo $n | awk '{print $5}'`
    37:   setenv distdim `echo $n | awk '{print $7}'`
    38:   setenv from_atom `echo $n | awk '{print $10}'`
    39:   setenv to_atom `echo $n | awk '{print $13}'`
    40:   @ distdimcol = $linker + $distdim + 2
    41:   @ energycol = $linker + 2
    42:
    43:   cat ../$csvname | \
    44:    awk -v distdimcol="$distdimcol" -v energycol="$energycol" -v energycut="$ecut" -v
distance="$distance" \
    45:    -v distdim="$distdim" -v from_atom="$from_atom" -v to_atom="$to_atom" \
    46:    'BEGIN {FS=","} {if ((NF>=distdimcol) && ($energycol<=energycut)) \
    47:    print (sqrt(($distdimcol-distance)*($distdimcol-distance))), distdim, from_atom,
to_atom, $0 }' | \
    48:    sort -nk1 | head -1 | awk -v cuff="$cuff" -v energycol="$energycol" '{print cuff,
(energycol-1), $0}'
    49:
    50: end
    51:
    52: echo "# done"
    53:
    54: exit
```

```
    55:
```

7.3.14 rotamerlist_2_manymol2s.spl

```
     1: ### rotamerlist_2_manymol2s.spl
     2: ### Arbeitstitel war : csv_list_to_mol2_files_v2.spl
     3: ### Skript das aus einer Liste von Rotameren (erzeugt von
     4: ### dist_2_rotamerlist.sh) mol2-files erzeugt.
     5: ### Stand 25.04.2005, BKL
     6:
     7: setvar phdist_file
%open(/usr/user12/bkroep/sybyl/mapleworkdir1/pharmacophoredist_2_mol2db/ph_dist_2_mol2_v2_outp
ut_with_15.135_0.2 "r")
     8:
     9: DATABASE OPEN /usr/user12/bkroep/sybyl/cuffs/databases/cuffs_CR0_CR8_L0_L4.mdb READ
    10:
    11: while %not( %eof( $phdist_file) )
    12:   setvar inputstring %read($phdist_file "`\'" "#" yes)
    13:   if %eq(%strlen($inputstring) 0)
    14:     BREAK
    15:    endif
    16:   setvar cuffname %arg(1 $inputstring)
    17:   setvar anzahl_winkel %arg(2 $inputstring)
    18:   setvar dist_diff %arg(3 $inputstring)
    19:   setvar dist_dim %arg(4 $inputstring)
    20:   setvar from_atom_id %arg(5 $inputstring)
    21:   setvar to_atom_id %arg(6 $inputstring)
    22:   setvar csv_line %set_unpack(%arg(7 $inputstring))
    23:   for i in %range(1 %math($anzahl_winkel))
    24:     setvar winkel[$i] %arg($i $csv_line)
    25:    endfor
    26:   echo
    27:
    28:   DATABASE GET $cuffname M1
    29:   setvar rotatablebonds %bonds(M1({BKL_ROTATABLE}) M)
    30:   for winkelnummer in %range(1 %math($anzahl_winkel))
    31:     if %eq($winkelnummer 1)
    32:       setvar modwinkelnummer 1
    33:      else
    34:        setvar modwinkelnummer %math($anzahl_winkel - $winkelnummer + 2)
    35:      endif
    36:     setvar pos2 %bond_info(%arg($modwinkelnummer $rotatablebonds) ORIGIN)
    37:     setvar pos3 %bond_info(%arg($modwinkelnummer $rotatablebonds) TARGET)
    38:     setvar pos2neighbours %atom_info($pos2 NEIGHBORS)
    39:     setvar goodneighbor 0
    40:     for neighbor in %atom_info($pos2 NEIGHBORS)
    41:       if %not(%STREQL(%atom_info($neighbor TYPE) "H"))
    42:         if      %not(%STREQL(%atom_info($neighbor ID) $pos3))
    43:           if %gt(%atom_info($neighbor ID) $goodneighbor)
    44:             setvar goodneighbor %atom_info($neighbor ID)
    45:           endif
    46:         endif
    47:       endif
    48:      endfor
    49:     setvar pos1 $goodneighbor
    50:     setvar goodneighbor 0
    51:     for neighbor in %atom_info($pos3 NEIGHBORS)
    52:       if %not(%STREQL(%atom_info($neighbor TYPE) "H"))
    53:         if      %not(%STREQL(%atom_info($neighbor ID) $pos2))
    54:           if %gt(%atom_info($neighbor ID) $goodneighbor)
    55:             setvar goodneighbor %atom_info($neighbor ID)
    56:           endif
    57:         endif
    58:       endif
    59:      endfor
    60:     setvar pos4 $goodneighbor
    61:     MODIFY TORSION $pos1 $pos2 $pos3 $pos4 $winkel[$winkelnummer]
    62:    endfor
    63:
    64: # dist bestimmen aus MEASURE DIST $from_atomid $to_atom_id
    65:   setvar dist %DISTANCE($from_atom_id $to_atom_id)
    66:
    67:   setvar name %cat($cuffname "_" $dist "_" $from_atom_id "_" $to_atom_id ".mol2")
    68:
    69:   mol2 out M1 $name
```

```
    70:
    71:   echo $name
    72:
    73:  endwhile
    74:
    75: echo %close($phdist_file)
    76:
```

7.3.15 ligand_2_ligandcenter.spl

```
     1: #!/bin/tcsh
     2: # ligand_2_ligandcenter.sh
     3: # Arbeitstitel war "ligandcenters_mol2_2_sph"
     4: # Dieses Skript liest einen Liganden fuer dock ein, der im
     5: # Format CRaXLbXCRc_distance_ID1_ID2.mol2 benannt sein muss.
     6: # Es erzeugt ein File namens CRaXLbXCRc_distance_ID1_ID2.sph,
     7: # mit den entsprechenden dock "ligand centers".
     8: # Stand 21.06.2005
     9:
    10: if ( $#argv != "1" ) then
    11:   echo " ERROR : give ONE parameter."
    12:   echo " usage $0 <ligand_mol2_file.mol2>"
    13:   exit
    14:  endif
    15:
    16: if (!(-e $1)) then
    17:   echo " ERROR : File  $1  does not exist."
    18:   exit
    19:  endif
    20:
    21: setenv ligandfile $1
    22: setenv startatomid `echo $ligandfile | awk 'BEGIN {FS="_"} {if (NF==4) print $3}'`
    23: setenv endatomid `echo $ligandfile | awk 'BEGIN {FS="_"} {if (NF==4) print $4}' | awk
'BEGIN {FS="."} {if (NF==2) print $1}'`
    24: setenv sphfilename `dirname $ligandfile`/`basename $ligandfile '.mol2'`.sph
    25: setenv startcoordinates `cat $ligandfile | sed -n '/@<TRIPOS>ATOM/,/@<TRIPOS>BOND/p' |
awk -v startatomid="$startatomid" 'BEGIN {FS=" "} {if (($1==startatomid) && (NF>=6))
printf("%4.5f %4.5f %4.5f",$3,$4,$5);}'`
    26: setenv endcoordinates `cat $ligandfile | sed -n '/@<TRIPOS>ATOM/,/@<TRIPOS>BOND/p' |
awk -v endatomid="$endatomid" 'BEGIN {FS=" "} {if (($1==endatomid) && (NF>=6)) printf("%4.5f
%4.5f %4.5f",$3,$4,$5);}'`
    27:
    28: echo "DOCK 3.5 receptor_spheres" >! $sphfilename
    29: echo "cluster     2   number of spheres in cluster     2" >> $sphfilename
    30:
    31: echo $startcoordinates | awk -v startatomid="$startatomid" -v endatomid="$endatomid"
'BEGIN {FS=" "} {printf("%5d %+5.5f  %+5.5f  %+5.5f  1.000  %5d 1 0 \n", startatomid,
$1,$2,$3, endatomid);}'  >> $sphfilename
    32: echo $endcoordinates | awk -v startatomid="$startatomid" -v endatomid="$endatomid"
'BEGIN {FS=" "} {printf("%5d %+5.5f  %+5.5f  %+5.5f  1.000  %5d 1 0 \n", endatomid, $1,$2,$3,
startatomid);}'  >> $sphfilename
    33:
    34: # echo "I5, 3F10.5, F8.3, I5, I2, I3"
    35:
    36: echo ""
    37: echo "$0 $argv "
    38: echo " INPUTFILE  : $ligandfile"
    39: echo " OUTPUTFILE : $sphfilename"
    40: echo ""
    41:
    42: exit
    43:
```

7.3.16 dock_single_ligand_to_B2B5.sh

```
     1: #!/bin/tcsh
     2: # dock_single_ligand_to_B2B5.sh
     3: # Dieses Skript liest einen Liganden fuer dock ein,
     4: # der im Format CRaXLbXCRc_distance_ID1_ID2.mol2 benannt
     5: # sein muss. Es ruft dann ligand_2_ligandcenter.sh auf,
     6: # stellt ein dock.in fuer diesen Liganden zusammen und
```

```
 7: # startet damit einen dock-Lauf.
 8: # Stand 22.06.2005
 9:
10: if ( $#argv != "1" ) then
11:   echo " ERROR : give ONE parameter."
12:   echo " usage $0 <ligand_mol2_file.mol2>"
13:   exit
14:  endif
15:
16: if (!(-e $1)) then
17:   echo " ERROR (dock_single) : File  $1  does not exist."
18:   exit
19:  endif
20:
21: setenv ligandfile $1
22:
23: ../tools/ligand_2_ligandcenter.sh $ligandfile
24:
25: setenv sphfilename `dirname $ligandfile`/`basename $ligandfile '.mol2'`.sph
26: setenv dockinfilename dock_`basename $ligandfile '.mol2'`.in
27: setenv energyfilename dock_`basename $ligandfile '.mol2'`_nrg.mol2
28: setenv docklogfilename dock_`basename $ligandfile '.mol2'`.log
29:
30: cp /usr/sgi2/bkroep/seccopy_for_dock/dock_B2B5/dock/reference/dock.in.template
$dockinfilename
31:
32: echo "ligand_atom_file \t $ligandfile" >> $dockinfilename
33: echo "ligand_center_file \t $sphfilename" >> $dockinfilename
34: echo "ligand_energy_file \t $energyfilename" >> $dockinfilename
35:
36: /usr/user/dock/dock -i $dockinfilename >! log/$docklogfilename
37:
38: exit
39:
```

7.3.17 dock_all_in_dir_to_B2B5.sh

```
 1: #!/bin/tcsh
 2: # dock_all_in_dir_to_B2B5
 3: # Arbeitstitel war "dock_all_in_dir_to_B2B5_v3"
 4: # Dieses Skript durchsucht das angegebenen Verzeichnis
 5: # nach mol2-Files. Fuer jedes gefundene mol2-file (ist
 6: # inzwischen nicht mehr beschraenkt, da keine wildcards
 7: # mehr verwendet werden) ruft es dock_single_ligand_to_B2B5
 8: # auf, fuehrt also dockings mit den Liganden durch.
 9: # Stand 02.09.2005
10:
11: if ( $#argv != "1" ) then
12:   echo "usage $0 <directory holding the ligands>"
13:   exit
14:  endif
15:
16: if (!(-e $1)) then
17:   echo " ERROR : File  $1  does not exist."
18:   exit
19:  endif
20:
21: foreach ligandfile (`ls -1 | grep "mol2"`)
22:   echo $ligandfile
23:   ../tools/dock_single_ligand_to_B2B5.sh $ligandfile
24:  end
25:
26: exit
27:
```

7.3.18 dock_struct_ligandsx_to_B2B5.sh

```
 1: #!/bin/tcsh
 2: # ../tools/dock_struct_ligandsx_to_B2B5
 3: # Dieses Skript fuehrt ein dateimaessig strukturiertes
 4: # Docking durch. Die Liganden-mol2-files, die sich in den
```

```
     5: # Unterverzeichnissen 1-b befinden, werden, verzeichnisweise
     6: # in das Hauptverzeichnis verschoben, dann dort gedockt und
     7: # anschliessend mitsamt der docking-Ergebnisse wieder in das
     8: # jeweilige Unterverzeichnis verschoben.
     9: # Stand : 14.07.2005
    10:
    11: foreach verzeichnis (1 2 3 4 5 6 7 8 9 a b)
    12:   echo ligands_$verzeichnis
    13:   mv ../struct/ligands_$verzeichnis/CR*.mol2 ../struct
    14:   ../tools/dock_all_in_dir_to_B2B5.sh ../struct/
    15:   mv ../struct/CR*.mol2 ../struct/ligands_$verzeichnis
    16:  end
    17:
    18: exit
    19:
```

7.3.19 analyze_dockings_in_dir.sh

```
     1: #!/bin/tcsh
     2: # analyse_dockings_in_dir.sh
     3: # Dieses Skript uebernimmt als Parameter den Namen
     4: # einer Directory, in der sich logfiles des Programmes
     5: # dock_all_in_dir_to_B2B5.sh befinden. Es erzeugt eine
     6: # Uebersicht ueber die dort vorhandenen dockings, indem
     7: # es zuerst alle auflistet, die keine gedockten
     8: # Orientierungen ergeben (nicht gedockt) haben. Im Anschluss
     9: # listet das Skript unter der Ueberschrift "SUCCESSFUL DOCKINGS"
    10: #  die dockings auf, die funktioniert haben,
    11: # aufsteigend nach Energie sortiert.
    12:
    13: if ( $#argv != "1" ) then
    14:   echo "usage $0 <directory holding the dock_logfiles>"
    15:   exit
    16:  endif
    17:
    18: if (!(-e $1)) then
    19:   echo " ERROR : File  $1  does not exist."
    20:   exit
    21:  endif
    22:
    23: echo "" >! ./temp_null_orientations
    24: echo "" >! ./temp_more_orientations
    25:
    26: foreach logfile (`ls -1 $1`)
    27:   setenv logfilename `echo $logfile`
    28:   setenv orientations `cat $1/$logfile | grep Orientations | awk 'BEGIN{FS=" "}{print
$NF}'`
    29:   setenv bestscore `cat $1/$logfile | grep "Best intermolecular energy score" | awk
'BEGIN{FS=" "}{print $NF}'`
    30:   if ( $orientations == "0" ) then
    31:     echo $logfilename, $orientations, $bestscore >> ./temp_null_orientations
    32:    else
    33:      echo $logfilename, $orientations, $bestscore >> ./temp_more_orientations
    34:    endif
    35:  end
    36:
    37: echo "NULL ORIENTATIONS FOR :"
    38: cat ./temp_null_orientations
    39:
    40: echo " "
    41:
    42: echo "SUCCESSFUL DOCKINGS :"
    43: cat ./temp_more_orientations | sort -nk3
    44:
    45: exit
    46:
```

7.3.20 B2B5T1_L01K00_15_41.col

```
     1: ###
     2: ###  build B2B5T1_L01K00_15_41, 12.08.2005, BKL
```

```
     3: ###
     4:
     5: MOL IN M5
/usr/sgi2/bkroep/seccopy_for_dock/dock_B2B5/top5/dock_CR3XL4XCR5_11.551_15_41_nrg.mol2
     6:
     7: MODIFY ATOM TYPE M5(42) C.3
     8:
     9: FILLVALENCE M5 H
    10:
    11: CHARGE "M5" COMPUTE GASTEIGER |
    12: CHARGE "M5" UNDISPLAY
    13:
    14: MOL2 OUT M5 B2B5T1_L01K00_15_41.mol2
    15:
```

7.3.21 B2B5T2_L00K07X_14_33.col

```
     1: ###
     2: ###  build B2B5T2_L00K07X_14_33, 01.09.2005, BKL
     3: ###
     4:
     5: MOL IN M5
/usr/sgi2/bkroep/seccopy_for_dock/dock_B2B5/top5/dock_CR3XL1XCR5_11.522_14_33_nrg.mol2
     6:
     7: MODIFY ATOM TYPE M5(13) C.3
     8:
     9: ADD ATOM M5(13) C.3
    10:
    11: ADD ATOM M5(36) N.4
    12:
    13: FILLVALENCE M5 H
    14:
    15: CHARGE "M5" COMPUTE GASTEIGER |
    16: CHARGE "M5" UNDISPLAY
    17:
    18: FOR i1 IN %RANGE(1 36)
    19:   setvar winkel1 %math($i1 * 10)
    20:
    21:   FOR i2 IN %RANGE(1 36)
    22:     setvar winkel2 %math($i2 * 10)
    23:
    24:     MODIFY TORSION M5(8) M5(7) M5(13) M5(36) $winkel1
    25:     MODIFY TORSION M5(7) M5(13) M5(36) M5(37) $winkel2
    26:
    27:     setvar moleculename %cat("B2B5T2_L00K07X" $winkel1 "X" $winkel2 "_14_33.mol2")
    28:
    29:     MOL2 OUT M5 $moleculename
    30:
    31:   ENDFOR
    32:
    33: ENDFOR
    34:
    35:
```

7.3.22 B2B5T2_LXXKXX_torsionscan.sh

```
     1: #!/bin/tcsh
     2: # B2B5T2_LXXKXX_torsionscan.sh
     3:
     4: if ( $#argv != "0" ) then
     5:   echo "usage $0"
     6:   exit
     7:  endif
     8:
     9: \rm -fr log\*
    10:
    11: if ( `hostname` != "sgi9.chemie.uni-hamburg.de") then
    12:   echo "script can only be run on the sgi9, because dock is installed there"
    13:   exit
    14:  endif
    15:
```

```
    16: if ( `pwd` != "/usr/sgi2/bkroep/seccopy_for_dock/dock_B2B5/top5_modified") then
    17:   echo "script can only be run from
/usr/sgi2/bkroep/seccopy_for_dock/dock_B2B5/top5_modified because relative pathes were used"
    18:   exit
    19:  endif
    20:
    21: foreach singlemod ( L00K01 L00K06 L00K07 L00K08 L00K09 L00K10 L00K11 L02K00 L04K00)
    22:   echo $singlemod started
    23:   setenv ligandname 'B2B5T2_'$singlemod'X'
    24:   echo  ... preparing files ...
    25:   foreach file_to_move (`ls -1 B2B5T2_torsionscan.prep | grep "B2B5T2_L" | grep "mol2"
| grep $singlemod` )
    26:     # echo $file_to_move
    27:     mv B2B5T2_torsionscan.prep/$file_to_move .
    28:    end
    29:   echo  ... docking ...
    30:   ../tools/dock_all_in_dir_to_B2B5.sh . >!
B2B5T2_torsionscan.results/01092005_torsionscan_$ligandname.log
    31:   echo  ... analyzing ...
    32:   ../tools/analyze_dockings_in_dir.sh log >!
B2B5T2_torsionscan.results/01092005_analyze_$ligandname
    33:   echo  ... cleaning up files ...
    34:   foreach docking_to_move (`ls -1 | grep "B2B5T2_L" | grep "sph" | grep $singlemod |
awk 'BEGIN{FS="."}{print $1}' `)
    35:     # echo $docking_to_move
    36:     mv $docking_to_move.mol2 B2B5T2_torsionscan.done/
    37:     mv dock_$docking_to_move.in B2B5T2_torsionscan.done/
    38:     mv $docking_to_move.sph B2B5T2_torsionscan.done/
    39:     mv dock_$docking_to_move'_nrg.mol2' B2B5T2_torsionscan.done/
    40:    end
    41:
    42:   mv log 'B2B5T2_torsionscan.done/'log_$singlemod
    43:   mkdir log
    44:
    45:   echo $ligandname done
    46:
    47:  end
    48:
    49: exit
    50:
```

7.3.23 B2B5T2_move_best_rotamers.sh

```
     1: #!/bin/tcsh
     2: # B2B5T2_move_best_rotamers.sh
     3: # Dieses Skript bearbeitet alle in der aktuellen directory
     4: # befindlichen log-Files von analyse_dockings_in_dir.sh .
     5: # In jedem dieser logfiles identifiziert dieses Skript den
     6: # Liganden mit der besten docking-Energie (steht dort
     7: # direkt unter "SUCCESSFUL DOCKINGS" und kopiert das
     8: # entsprechende Liganden-mol2-File eine Verzeichnisebene
     9: # hoeher zur weiteren Verwendung.
    10:
    11: if ( $#argv != "0" ) then
    12:   echo "usage $0"
    13:   exit
    14:  endif
    15:
    16: echo $0
    17: echo  .
    18: echo "die topscorer sind :"
    19: cat 01092005_analyze_B2B5T2_L*X | sed -n '/SUCCESSFUL DOCKINGS :/{n;p;}'
    20: echo .
    21:
    22: echo ... now copying the topscoreres ...
    23: foreach i (`cat 01092005_analyze_B2B5T2_L*X | sed -n '/SUCCESSFUL DOCKINGS :/{n;p;}' |
awk '{print $1}' | awk 'BEGIN{FS="."}{print $1}' | sed 's/dock_//' `)
    24:   setenv ligandfile `echo $i.mol2`
    25:   cp ../B2B5T2_torsionscan.done/$ligandfile ../
    26:  end
    27:
    28: exit
    29:
```

7.3.24 dock_B2B5T5_L04K01_random_seed_variation.sh

```
     1: #!/bin/tcsh
     2: # dock_B2B5T5_L04K01_random_seed_variation.sh
     3: # Dieses Skript fuehrt dockings durch.
     4: # Die Einstellungen aller dockings sind gleich, nur random_seed und
     5: # ligand_energy_file werden variiert.
     6: # Stand 18.09.2005, BKL
     7:
     8: if ( $#argv != "0" ) then
     9:   echo " ERROR : give NO parameter."
    10:   echo " usage : $0 "
    11:   exit
    12:  endif
    13:
    14: echo "Docking : dock_B2B5T5_L04K01_soft_02_xx with variation of random_seed"
    15: foreach x (01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30)
    16:   cp ./dock_B2B5T5_L04K01_soft_02.in dock_B2B5T5_L04K01_soft_02_$x.in
    17:   echo "random_seed \t $x" >> dock_B2B5T5_L04K01_soft_02_$x.in
    18:   echo "ligand_energy_file \t dock_B2B5T5_L04K01_soft_02_$x""_nrg.mol2" >>
dock_B2B5T5_L04K01_soft_02_$x.in
    19:   echo "Random Seed : $x"
    20:   /usr/user/dock/dock -i dock_B2B5T5_L04K01_soft_02_$x.in >!
log/dock_B2B5T5_L04K01_soft_02_$x.log
    21:   cat log/dock_B2B5T5_L04K01_soft_02_$x.log | grep "Best energy score"
    22:  end
    23:
    24: exit
    25:
```

7.4 Parametersätze

Die Parametersätze für das *constrained docking* (AAV 2) und das *unconstrained docking* (AAV 4) sind in Abschnitt 6.2 innerhalb der jeweiligen allgemeinen Arbeitsvorschrift abgedruckt.

Für die Erzeugung des *grid* in Abschnitt 3.3.3 und 3.5.1 wurden die folgenden Einstellungen verwendet:

Tabelle 27: Auflistung der Parameter, die zur Erzeugung des *grid* in Abschnitt 3.3.3 verwendet wurden.

compute_grids	yes	distance_dielectric	yes
grid_spacing	0.3	dielectric_factor	4
output_molecule	no	bump_filter	no
contact_score	no	receptor_file	
chemical_score	no	cd4_mol2_receptor.mol2	
energy_score	yes	box_file	
energy_cutoff_distance	10	box_around_receptor_spheres_for_B2B5.pdb	
atom_model	a	vdw_definition_file	
attractive_exponent	6	/usr/user/dock/vdw.defn	
repulsive_exponent	12	score_grid_prefix	grid

Tabelle 28: Auflistung der Parameter, die zur Erzeugung des *grid* in Abschnitt 3.5.1 verwendet wurden.

compute_grids	yes	distance_dielectric	yes
grid_spacing	0.3	dielectric_factor	4
output_molecule	no	bump_filter	no
contact_score	no	receptor_file	
chemical_score	no	cd4_mol2_receptor.mol2	
energy_score	yes	box_file	
energy_cutoff_distance	10	box_around_cd4_area_for_B2B5_20082005.pdb	
atom_model	a	vdw_definition_file	
attractive_exponent	6	/usr/user/dock/vdw.defn	
repulsive_exponent	12	score_grid_prefix	grid_20082005

7.5 Literatur

1. Kernighan B. W., Ritchie D. M., The C Programming Language, Englewood Cliffs: Prentice-Hall, **1988**.

2. Bernard C., Introduction a l'Etude de la Médicine Expérimentale, Paris: JB Bailliere, **1978**.

3. Houston J. G., Banks M., The chemical-biological interface: developments in automated and miniaturised screening technology, *Curr Opin Biotechnol,* **1997**, 8, 734-40.

4. Lahana R., How many leads from HTS?, *Drug Discov Today,* **1999**, 4, 447-8.

5. Hertzberg R. P., Pope A. J., High-throughput screening: new technology for the 21st century, *Curr Opin Chem Biol,* **2000**, 4, 445-51.

6. Keighley W. W., Wood T. P., Compound library management. An overview of an automated system, *Methods Mol Biol,* **2002**, 190, 129-52.

7. http://www.bayerresearchcenter.com/drug/organization.asp
   http://www.automationpartnership.com
   http://www.roche.ch/chemie_d-3.pdf.

8. Steinmeyer A., The hit-to-lead process at Schering AG: strategic aspects, *ChemMedChem,* **2006**, 1, 31-6.

9. Balkenhohl F., von dem Bussche-Huennefeld C., Lansky A., Zechel C., Combinatorial synthesis of small organic molecules., *Angew Chem Int Ed Engl,* **1996**, 35, 2288-337.

10. Terrett N. K., Gardner M., Gordon D. W., Kobylecki R. J., Steele J., Combinatorial synthesis - the design of compound libraries and their application to drug discovery, *Tetrahedron,* **1995**, 51, 8135-73.

11. Schneider G., Böhm H. J., Virtual screening and fast automated docking methods, *Drug Discov Today,* **2002**, 7, 64-70.

12. Leach A. R., Hann M. M., The in silico world of virtual libraries, *Drug Discov Today,* **2000**, 5, 326-36.

13. Teague S. J., Davis A. M., Leeson P. D., Oprea T., The Design of Leadlike Combinatorial Libraries, *Angew Chem Int Ed Engl,* **1999**, 38, 3743-8.

14. Dost F. H., Grundlagen der Pharmakokinetik, Stuttgart: Thieme, **1968**.

15. Ekins S., Systems-ADME/Tox: resources and network approaches, *J Pharmacol Toxicol Methods,* **2006**, 53, 38-66.

16. Lipinski C. A., Lombardo F., Dominy B. W., Feeney P. J., Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings, *Adv Drug Deliv Rev,* **2001**, 46, 3-26.

17. Ajay A., Walters W. P., Murcko M. A., Can we learn to distinguish between "drug-like" and "nondrug-like" molecules?, *J Med Chem,* **1998**, 41, 3314-24 .

18. Wang J., Ramnarayan K., Toward designing drug-like libraries: a novel computational approach for prediction of drug feasibility of compounds, *J Comb Chem,* **1999**, 1, 524-33.

19. Sadowski J., Kubinyi H., A scoring scheme for discriminating between drugs and nondrugs, *J Med Chem,* **1998**, 41, 3325-9.

20. Rassokhin D. N., Agrafiotis D. K., Kolmogorov-Smirnov statistic and its application in library design, *J Mol Graph Model,* **2000**, 18, 368-82.

21. Müller K. R., Ratsch G., Sonnenburg S., Mika S., Grimm M., Heinrich N., Classifying 'drug-likeness' with Kernel-based learning methods, *J Chem Inf Model,* **2005**, 45, 249-53.

22. Ajay A., Predicting drug-likeness: why and how?, *Curr Top Med Chem,* **2002**, 2, 1273-86.

23. Egan W. J., Walters W. P., Murcko M. A., Guiding molecules towards drug-likeness, *Curr Opin Drug Discov Devel,* **2002**, 5, 540-9.

24. Walters W. P., Murcko M. A., Prediction of 'drug-likeness', *Adv Drug Deliv Rev,* **2002**, 54, 255-71.

25. Bemis G. W., Murcko M. A., The properties of known drugs. 1. Molecular frameworks, *J Med Chem,* **1996**, 39, 2887-93.

26. http://www.mdl.com/products/knowledge/medicinal_chem/index.jsp.

27. Comprehensive Medicinal Chemistry (CMC-3D) Release 94.1, MDL Information Systems Inc., San Leandro, CA.

28. Hansch C., Sammes P. G., Taylor J. B., Comprehensive Medicinal Chemistry, Vol. 6, Oxford: Pergamon, **1990**.

29. Ajay A., Bemis G. W., Murcko M. A., Designing libraries with CNS activity, *J Med Chem,* **1999**, 42, 4942-51.

30. Gibbons A., Algorithmic Graph Theory, Cambridge: Cambridge University Press, **1988**.

31. Hansen P. J., Jurs P. C., Chemical Applications of Graph Theory. Part I. Fundamentals and topological indices, *J Chem Ed,* **1988**, 65, 574-80.

32. Bemis G. W., Murcko M. A., Properties of known drugs. 2. Side chains, *J Med Chem,* **1999**, 42, 5095-9.

33. Nilakantan R., Bauman N., Dixon J. S., Venkataraghavan R., Topological torsion: a new molecular descriptor for SAR applications. Comparison with other descriptors, *J Chem Inf Comput Sci,* **1987**, 27, 82-5.

34. Wülfken J., Entwicklung CD4 bindender Peptide als Inhibitoren der HIV-Infektion, Dissertation, Universität Hamburg, Hamburg, **2001**.

35. Böhm H. J., Klebe G., Kubinyi H., Wirkstoffdesign, Heidelberg - Berlin - Oxford: Spektrum Akademischer Verlag, **1996**.

36. Gante J., Peptidmimetica - massgeschneiderte Enzyminhibitoren, *Angew Chem,* **1994**, 106, 1780-802.

37. Loffet A., Peptides as drugs: is there a market?, *J Pept Sci,* **2002**, 8, 1-7.

38. Fuzeon, Packungsbeilage, ROCHE Pharmaceuticals, **2005**.

39. Giannis A., Kolter T., Peptidmimetica fuer Rezeptorliganden - Entdeckung, Entwicklung und medizinische Perspektiven, *Angew Chem,* **1993**, 105, 1303-26.

40. Ehrlich P., Über den jetzigen Stand der Chemotherapie, *Ber dtsch chem Ges,* **1909**, 42, 17-47.

41. Fersht A. R., Shi J. P., Knill-Jones J., Lowe D. M., Wilkinson A. J., Blow D. M., Brick P., Carter P., Waye M. M., Winter G., Hydrogen bonding and biological specificity analysed by protein engineering, *Nature,* **1985**, 314, 235-8.

42. Mayer M., Meyer B., Characterization of ligand binding by saturation transfer difference NMR spectroscopy, *Angew Chem Int Ed Engl,* **1999**, 38, 1784-8.

43. Meyer B., Peters T., NMR spectroscopy techniques for screening and identifying ligand binding to protein receptors, *Angew Chem Int Ed Engl,* **2003**, 42, 864-90.

44. Mayer M., STD-NMR Spektroskopie: Eine neue Methode zur Identifizierung und Charakterisierung von Ligand-Rezeptor-Interaktionen, Dissertation, Universität Hamburg, Hamburg, **2001**.

45. Mayer M., Meyer B., Group epitope mapping by saturation transfer difference NMR to identify segments of a ligand in direct contact with a protein receptor, *J Am Chem Soc,* **2001**, 123, 6108-17.

46. Jayalakshmi V., Krishna N. R., Complete relaxation and conformational exchange matrix (CORCEMA) analysis of intermolecular saturation transfer effects in reversibly forming ligand-receptor complexes, *J Magn Reson,* **2002**, 155, 106-18.

47. Klein J., Meinecke R., Mayer M., Meyer B., Detecting Binding Affinity to Immobilized Receptor Proteins in Compound Libraries by HR-MAS STD NMR, *J Am Chem Soc,* **1999**, 121, 5336-7.

48. Meinecke R., Meyer B., Determination of the binding specificity of an integral membrane protein by saturation transfer difference NMR: RGD peptide ligands binding to integrin alphaIIbbeta3, *J Med Chem,* **2001**, 44, 3059-65.

49. Claasen B., Axmann M., Meinecke R., Meyer B., Direct observation of ligand binding to membrane proteins in living cells by a saturation transfer double difference (STDD) NMR spectroscopy method shows a significantly higher affinity of integrin alpha(IIb)beta3 in native platelets than in liposomes, *J Am Chem Soc,* **2005**, 127, 916-9.

50. Bragg M. A., Bragg W. L., Reflection of X-rays by Crystals, *Proc Roy Soc London (A),* **1913**, 88, 428-39.

51. Luger P., Modern X-Ray Analysis in Single Crystals, Berlin - New York: Walter de Gruyter, **1980**.

52. McPherson A., Crystallization of Biological Macromolecules, New York: Cold Spring Harbor Laboratory Press, **1980**.

53. Blundell T. L., Jhoti H., Abell C., High-throughput crystallography for lead discovery in drug design, *Nat Rev Drug Discov,* **2002**, 1, 45-54.

54. Nienaber V. L., Richardson P. L., Klighofer V., Bouska J. J., Giranda V. L., Greer J., Discovering novel ligands for macromolecules using X-ray crystallographic screening, *Nat Biotechnol,* **2000**, 18, 1105-8.

55. Tusnady, G. E., Dosztanyi, Z., Simon, I., Transmembrane proteins in the Protein Data Bank: identification and classification. *Bioinformatics*, **2004**, 20, 2964-2972.

56. Bernstein F. C., Koetzle T. F., Williams G. J., Meyer E. F. Jr., Brice M. D., Rodgers J. R., Kennard O., Shimanouchi T., Tasumi M., The Protein Data Bank: a computer-based archival file for macromolecular structures, *J Mol Biol,* **1977**, 112, 535-42.

57. Internetseite der Protein Data Bank, http://www.rcsb.org/pdb.

58. Davis A. M., Teague S. J., Kleywegt G. J., Application and limitations of X-ray crystallographic data in structure-based ligand and drug design, *Angew Chem Int Ed Engl,* **2003**, 42, 2718-36.

59. Kleywegt G. J., Validation of protein crystal structures, *Acta Crystallogr D Biol Crystallogr,* **2000**, 56, 249-65.

60. Bränden C.-I., Jones T. A., Between objectivity and subjectivity, *Nature,* **1990**, 343, 687-9.

61. Brooijmans N., Kuntz I. D., Molecular recognition and docking algorithms, *Annu Rev Biophys Biomol Struct,* **2003**, 32, 335-73.

62. Solomon I., Relaxation processes in a system of two spins, *Phys Rev,* **1955**, 99, 559-65.

63. Ni F., Recent developments in transferred NOE methods, *Prog. Nucl. Magn. Reson. Spectrosc.,* **1994**, 26, 517-606.

64. Kaiser R., The nuclear Overhauser effect in the analysis of high-resolution nuclear magnetic resonance spectra, *J Chem Phys,* **1963**, 39, 2435-42 .

65. Anet F. A. L., Bourn A. J. R., Nuclear magnetic resonance spectral assignments from nuclear Overhauser effects, *J Am Chem Soc,* **1965**, 87, 5250-1.

66. Balaram P., Bothner-By A. A., Dadok J., Negative nuclear Overhauser effects as probes of macromolecular structure, *J Am Chem Soc,* **1972**, 94, 4015-7 .

67. Heitler W., London F., Wechselwirkung neutraler Atome und homöopolare Bindung nach der Quantenmechanik, *Z Phys A: Hadrons and Nuclei,* **1927**, 44, 455-72.

68. FIZ CHEMIE Berlin, http://www.chemgapedia.de.

69. Brooks B. R., Bruccoleri R. E., Olafson B. D., States D. J., Swaminathan S., Karplus M., CHARMM: a program for macromolecular energy, minimization, and dynamics calculations, *J Comput Chem,* **1983**, 4, 187-217.

70. http://brooks.scripps.edu/charmm_docs/charmm.htm.

71. Cornell W. D., Cieplak P., Bayly C. I., Gould I. R., Merz K. M. Jr., Ferguson D. M., Spellmeyer D. C., Fox T., Caldwell J. W., Kollman P. A., A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules, *J Am Chem Soc,* **1995**, 117, 5179-97.

72. http://amber.scripps.edu/.

73. Van Gunsteren W. F., Berendsen H. J. C., Molecular dynamics computer simulation. Method, application and perspectives in chemistry, *Angew. Chem.,* **1990**, 102, 1020-55.

74. http://www.igc.ethz.ch/gromos/.

75. Clark M., Cramer R. D. I., Van Opdenbosch N., Validation of the general purpose Tripos 5.2 force field, *J Comput Chem,* **1989**, 10, 982-1012.

76. Jorgensen W. L., Tirado-Rives J., The OPLS [optimized potentials for liquid simulations] potential functions for proteins, energy minimizations for crystals of cyclic peptides and crambin, *J Am Chem Soc,* **1988**, 110, 1657-66.

77. de Groot B. L., Grubmüller H., Water permeation across biological membranes: mechanism and dynamics of aquaporin-1 and GlpF, *Science,* **2001**, 294, 2353-7.

78. Joseph-McCarthy D., Computational approaches to structure-based ligand design, *Pharmacol Ther,* **1999**, 84, 179-91.

79. Kroemer R. T., Molecular modelling probes: docking and scoring, *Biochem Soc Trans,* **2003**, 31, 980-4.

80. Moult J., Murzin A., Zemla A., Tramontano A., Rost B., Hubbard T. J. P., Techniques for Protein Structure Prediction, *Proteins,* **2002**, Suppl. 5, 1-199.

81. Peitsch M. C., Schwede T., Diemand A., Guex N., Protein structure prediction by comparison: Homology-based modelling, In: Jiang T., Xu Y., Zhang M.Q., Current Topics in Computational Molecular Biology, Cambridge, MA: MIT Press, **2002**, 49–466.

82. Zimmer R., Lengauer T., Protein Structure Prediction, In: Lengauer T., Bioinformatics - From Genomes to Drugs, Weinheim: Wiley-VCH, **2001**, 37-313.

83. Willis R. C., Surveying the bind, *Modern Drug Discovery,* **2002**, 5, 28-30, 33-4.

84. Greco G., Novellino E., Martin Y. C., 3D-QSAR methods, *Designing Bioactive Molecules,* **1998**, 219-52.

85. Hansch C., Fujita T., ρ-σ-π Analysis; method for the correlation of biological activity and chemical structure, *J Am Chem Soc,* **1964**, 86, 1616-26.

86. Müller G., nD QSAR: a medicinal chemist's point of view, *Quant Struct-Act Relat,* **2002**, 21, 391-6.

87. Kubinyi H., From narcosis to hyperspace: the history of QSAR, *Quant Struct-Act Relat,* **2002**, 21, 348-56.

88. Cramer R. D. I., Patterson D. E., Bunce J. D., Comparative molecular field analysis (CoMFA). 1. Effect of shape on binding of steroids to carrier proteins, *J Am Chem Soc,* **1988**, 110, 5959-67.

89. Oprea T. I., Waller C. L., Theoretical and practical aspects of three-dimensional quantitative structure-activity relationships, *Rev Comput Chem,* **1997**, 11, 127-82.

90. Vedani A., Dobler M., 5D-QSAR: the key for simulating induced fit?, *J Med Chem,* **2002**, 45, 2139-49.

91. Free S. M. Jr., Wilson J. W., A mathematical contribution to structure-activitiy studies, *J Med Chem,* **1964**, 53, 395-9.

92. Schneider G., Fechner U., Computer-based de novo design of drug-like molecules, *Nat Rev Drug Discov,* **2005**, 4, 649-63.

93. Böhm H.-J., Schneider G., Virtual Screening for Bioactive Molecules, Weinheim: Wiley-VCH, **2000**.

94. Mohan V., Gibbs A. C., Cummings M. D., Jaeger E. P., DesJarlais R. L., Docking: successes and challenges, *Curr Pharm Des,* **2005**, 11, 323-33.

95. Krovat E. M., Steindl T., Langer T., Recent advances in docking and scoring, *Curr Comput-Aided Drug Des,* **2005**, 1, 93-102.

96. Walters W. P., Stahl M. T., Murcko M. A., Virtual screening - an overview, *Drug Discov Today,* **1998**, 3, 160-78.

97. Lyne P. D., Structure-based virtual screening: an overview, *Drug Discov Today,* **2002**, 7, 1047-55.

98. Taylor R. D., Jewsbury P. J., Essex J. W., A review of protein-small molecule docking methods, *J Comput Aided Mol Des,* **2002**, 16, 151-66.

99. Muegge I., Rarey M., Small molecule docking and scoring, *Rev Comput Chem,* **2001**, 17, 1-60.

100. Connolly M. L., Shape complementarity at the hemoglobin alpha 1 beta 1 subunit interface, *Biopolymers,* **1986**, 25, 1229-47.

101. Ajay A., Murcko M. A., Computational methods to predict binding free energy in ligand-receptor complexes, *J Med Chem,* **1995**, 38, 4953-67.

102. Koshland D. E. Jr., Application of a theory of enzyme specificity to protein synthesis, *Proc Natl Acad Sci U S A,* **1958**, 44, 98-105.

103. Teague S. J., Implications of protein flexibility for drug discovery, *Nat Rev Drug Discov,* **2003**, 2, 527-41.

104. Toledo-Sherman L. M., Chen D., High-throughput virtual screening for drug discovery in parallel, *Curr Opin Drug Discov Devel,* **2002**, 5, 414-21.

105. Erickson J. A., Jalaie M., Robertson D. H., Lewis R. A., Vieth M., Lessons in molecular recognition: the effects of ligand and protein flexibility on molecular docking accuracy, *J Med Chem,* **2004**, 47, 45-55.

106. Jorgensen W. L., Rusting of the lock and key model for protein-ligand binding, *Science,* **1991**, 254, 954-5.

107. Davis A. M., Teague S. J., Hydrogen bonding, hydrophobic interactions, and failure of the rigid receptor hypothesis, *Angew Chem Int Ed Engl,* **1999**, 38, 736-49.

108. Najmanovich R., Kuttner J., Sobolev V., Edelman M., Side-chain flexibility in proteins upon ligand binding, *Proteins,* **2000**, 39, 261-8.

109. Carlson H. A., McCammon J. A., Accommodating protein flexibility in computational drug design, *Mol Pharmacol,* **2000**, 57, 213-8.

110. Cavasotto C. N., Abagyan R. A., Protein flexibility in ligand docking and virtual screening to protein kinases, *J Mol Biol,* **2004**, 337, 209-25.

111. Claussen H., Buning C., Rarey M., Lengauer T., FlexE: efficient molecular docking considering protein structure variations, *J Mol Biol,* **2001**, 308, 377-95.

112. Nicklaus M. C., Wang S., Driscoll J. S., Milne G. W., Conformational changes of small molecules binding to proteins, *Bioorg Med Chem,* **1995**, 3, 411-28.

113. Kearsley S. K., Underwood D. J., Sheridan R. P., Miller M. D., Flexibases: a way to enhance the use of molecular docking methods, *J Comput Aided Mol Des,* **1994**, 8, 565-82.

114. Klebe G., Mietzner T., A fast and efficient method to generate biologically relevant conformations, *J Comput Aided Mol Des,* **1994**, 8, 583-606.

115. Rarey M., Kramer B., Lengauer T., Klebe G., A fast flexible docking method using an incremental construction algorithm, *J Mol Biol,* **1996**, 261, 470-89.

116. Rarey M., Kramer B., Lengauer T., Multiple automatic base selection: protein-ligand docking based on incremental construction without manual intervention, *J Comput Aided Mol Des,* **1997**, 11, 369-84.

117. Goodsell D. S., Olson A. J., Automated docking of substrates to proteins by simulated annealing, *Proteins,* **1990**, 8, 195-202.

118. Given J. A., Gilson M. K., A hierarchical method for generating low-energy conformers of a protein-ligand complex, *Proteins,* **1998**, 33, 475-95.

119. Trosset J. Y., Scheraga H. A., Reaching the global minimum in docking simulations: a Monte Carlo energy minimization approach using Bezier splines, *Proc Natl Acad Sci U S A,* **1998**, 95, 8011-5.

120. Trosset J.-Y., Scheraga H. A., PRODOCK: software package for protein modeling and docking, *J Comput Chem,* **1999**, 20, 412-27.

121. Trosset J.-Y., Scheraga H. A., Flexible docking simulations: scaled collective variable Monte Carlo minimization approach using Bezier splines, and comparison with a standard Monte Carlo algorithm, *J Comput Chem,* **1999**, 20, 244-52.

122. Jones G., Willett P., Glen R. C., Leach A. R., Taylor R., Development and validation of a genetic algorithm for flexible docking, *J Mol Biol,* **1997**, 267, 727-48.

123. Jones G., Willett P., Glen R. C., A genetic algorithm for flexible molecular overlay and pharmacophore elucidation, *J Comput Aided Mol Des,* **1995**, 9, 532-49.

124. Goldberg D. E., Genetic Algorithms in Search, Optimization and Machine Learning, Boston: Kluwer Academic Publishers, **1989**.

125. http://www.bio.vu.nl/nvtb/docking.html.

126. Ewing T. J., Makino S., Skillman A. G., Kuntz I. D., DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases, *J Comput Aided Mol Des,* **2001**, 15, 411-28.

127. Halgren T. A., Murphy R. B., Friesner R. A., Beard H. S., Frye L. L., Pollard W. T., Banks J. L., Glide: a new approach for rapid, accurate docking and scoring. 2. Enrichment factors in database screening, *J Med Chem,* **2004**, 47, 1750-9.

128. Friesner R. A., Banks J. L., Murphy R. B., Halgren T. A., Klicic J. J., Mainz D. T., Repasky M. P., Knoll E. H., Shelley M., Perry J. K., Shaw D. E., Francis P., Shenkin P. S., Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy, *J Med Chem,* **2004**, 47, 1739-49.

129. McMartin C., Bohacek R. S., QXP: powerful, rapid computer algorithms for structure-based drug design, *J Comput Aided Mol Des,* **1997**, 11, 333-44.

130. Abagyan R., Totrov M., Kuznetsov D., ICM - a new method for protein modeling and design: applications to docking and structure prediction from the distorted native conformation, *J Comput Chem,* **1994**, 15, 488-506.

131. Böhm H. J., The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure, *J Comput Aided Mol Des,* **1994**, 8, 243-56.

132. Brodmeier T., Pretsch E., Application of genetic algorithms in molecular modeling, *J Comput Chem,* **1994**, 15, 588-95.

133. Morris G. M., Goodsell D. S., Halliday R. S., Huey R., Hart W. E., Belew R. K., Olson A. J., Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function, *J Comput Chem,* **1998**, 19, 1639-62.

134. Perola E., Walters W. P., Charifson P. S., A detailed comparison of current docking and scoring methods on systems of pharmaceutical relevance, *Proteins,* **2004**, 56, 235-49.

135. Cheney D., Mueller L., Evaluation of strategies for molecular docking, *Abstracts of Papers, 226th ACS National Meeting, New York, NY, United States, September 7-11, 2003,* **2003**, COMP, 144.

136. Schulz-Gasch T., Stahl M., Binding site characteristics in structure-based virtual screening: evaluation of current docking tools, *J Mol Model (Online),* **2003**, 9, 47-57.

137. Chen H., Lyne P. D., Giordanetto F., Lovell T., Li J., On evaluating molecular-docking methods for pose prediction and enrichment factors, *J Chem Inf Model,* **2006**, 46, 401-15.

138. Schapira M., Abagyan R., Totrov M., Nuclear hormone receptor targeted virtual screening, *J Med Chem,* **2003**, 46, 3045-59.

139. Bursulaya B. D., Totrov M., Abagyan R., Brooks C. L. 3rd, Comparative study of several algorithms for flexible ligand docking, *J Comput Aided Mol Des,* **2003**, 17, 755-63.

140. Cummings M. D., DesJarlais R. L., Gibbs A. C., Mohan V., Jaeger E. P., Comparison of automated docking programs as virtual screening tools, *J Med Chem,* **2005**, 48, 962-76.

141. Charifson P. S., Corkery J. J., Murcko M. A., Walters W. P., Consensus scoring: A method for obtaining improved hit rates from docking databases of three-dimensional structures into proteins, *J Med Chem,* **1999**, 42, 5100-9.

142. Stahl M., Rarey M., Detailed analysis of scoring functions for virtual screening, *J Med Chem,* **2001**, 44, 1035-42.

143. Wang R., Lu Y., Wang S., Comparative evaluation of 11 scoring functions for molecular docking, *J Med Chem,* **2003**, 46, 2287-303.

144. Tame J. R., Scoring functions: a view from the bench, *J Comput Aided Mol Des,* **1999**, 13, 99-108.

145. Weisstein, E. W., Clique, http://mathworld.wolfram.com/clique.html.

146. Kuntz I. D., Blaney J. M., Oatley S. J., Langridge R., Ferrin T. E., A geometric approach to macromolecule-ligand interactions, *J Mol Biol,* **1982**, 161, 269-88.

147. http://dock.compbio.ucsf.edu.

148. Connolly M. L., The molecular surface package, *J Mol Graph,* **1993**, 11, 139-41.

149. Kontoyianni M., McClellan L. M., Sokol G. S., Evaluation of docking performance: comparative data on docking algorithms, *J Med Chem,* **2004**, 47, 558-65.

150. Kellenberger E., Rodrigo J., Muller P., Rognan D., Comparative evaluation of eight docking tools for docking and virtual screening accuracy, *Proteins,* **2004**, 57, 225-42.

151. DesJarlais R. L., Seibel G. L., Kuntz I. D., Furth P. S., Alvarez J. C., Ortiz de Montellano P. R., DeCamp D. L., Babe L. M., Craik C. S., Structure-based design of nonpeptide inhibitors specific for the human immunodeficiency virus 1 protease, *Proc Natl Acad Sci U S A,* **1990**, 87, 6644-8.

152. Vangrevelinghe E., Zimmermann K., Schoepfer J., Portmann R., Fabbro D., Furet P., Discovery of a potent and selective protein kinase CK2 inhibitor by high-throughput docking, *J Med Chem,* **2003**, 46, 2656-62.

153. Tondi D., Slomczynska U., Costi M. P., Watterson D. M., Ghelli S., Shoichet B. K., Structure-based discovery and in-parallel optimization of novel competitive inhibitors of thymidylate synthase, *Chem Biol,* **1999**, 6, 319-31.

154. Doman T. N., McGovern S. L., Witherbee B. J., Kasten T. P., Kurumbail R., Stallings W. C., Connolly D. T., Shoichet B. K., Molecular docking and high-throughput screening for novel inhibitors of protein tyrosine phosphatase-1B, *J Med Chem,* **2002**, 45, 2213-21.

155. Hopkins S. C., Vale R. D., Kuntz I. D., Inhibitors of kinesin activity from structure-based computer screening, *Biochemistry,* **2000**, 39, 2805-14.

156. Aronov A. M., Munagala N. R., Ortiz De Montellano P. R., Kuntz I. D., Wang C. C., Rational design of selective submicromolar inhibitors of Tritrichomonas foetus hypoxanthine-guanine-xanthine phosphoribosyltransferase, *Biochemistry,* **2000**, 39, 4684-91.

157. Kuntz I. D., Structure-based strategies for drug design and discovery, *Science,* **1992**, 257, 1078-82.

158. Shoichet B. K., Kuntz I. D., Protein docking and complementarity, *J Mol Biol,* **1991**, 221, 327-46.

159. Leach A. R., Kuntz I. D., Conformation analysis of flexible ligands in macromolecular receptor sites, *J Comput Chem,* **1992**, 13, 730-48.

160. Shoichet B. K., Leach A. R., Kuntz I. D., Ligand solvation in molecular docking, *Proteins,* **1999**, 34, 4-16.

161. Smellie A. S., Crippen G. M., Richards W. G., Fast drug-receptor mapping by site-directed distances: a novel method of predicting new pharmacological leads, *J Chem Inf Comput Sci,* **1991**, 31, 386-92.

162. Meng E. C., Shoichet B. K., Kuntz I. D., Automated docking with grid-based energy evaluation, *J Comput Chem,* **1992**, 13, 505-24.

163. DOCK Version 4.0 Reference Manual, University of California at San Francisco (UCSF), USA, **1998**.

164. Makino S., Kuntz I. D., Automated flexible ligand docking method and its application for database search, *J Comput Chem,* **1997**, 18, 1812-25.

165. Ewing T. J. A., Kuntz I. D., Critical evaluation of search algorithms for automated molecular docking and database screening, *J Comput Chem,* **1997**, 18, 1175-89.

166. Goodford P. J., A computational procedure for determining energetically favorable binding sites on biologically important macromolecules, *J Med Chem,* **1985**, 28, 849-57.

167. Boobbyer D. N., Goodford P. J., McWhinnie P. M., Wade R. C., New hydrogen-bond potentials for use in determining energetically favorable binding sites on molecules of known structure, *J Med Chem,* **1989**, 32, 1083-94.

168. Wu G., Robertson D. H., Brooks C. L. 3., Vieth M., Detailed analysis of grid-based molecular docking: A case study of CDOCKER-A CHARMm-based MD docking algorithm, *J Comput Chem,* **2003**, 24, 1549-62.

169. Shoichet B. K., Kuntz I. D., Matching chemistry and shape in molecular docking, *Protein Eng,* **1993**, 6, 723-32.

170. Knegtel R. M., Kuntz I. D., Oshiro C. M., Molecular docking to ensembles of protein structures, *J Mol Biol,* **1997**, 266, 424-40.

171. Ferro D. R., Hermans J., A different best rigid-body molecular fit routine, *Acta Crystallogr A Crystal Physics,* **1977**, A33, 345-7.

172. Kuhl F. S., Crippen G. M., Friesen D. K., A combinatorial algorithm for calculating ligand binding, *J Comput Chem,* **1984**, 5, 24-34.

173. Mortimer C. E., Chemie. Das Basiswissen der Chemie, Stuttgart: Georg Thieme Verlag, **2001**.

174. Schwab C. H., Konformative Flexibilität von Liganden im Wirkstoffdesign, Dissertation, Universität Erlangen, Erlangen, **2001**.

175. Klebe G., Toward a more efficient handling of conformational flexibility in computer-assisted modeling of drug molecules, *Perspect Drug Discovery Des,* **1995**, 3(De Novo Design), 85-105.

176. Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Teller E., Equation-of-state calculations by fast computing machines, *J Chem Phys,* **1953**, 21, 1087-92.

177. Rommel-Moehle K., Hofmann H.-J., Conformation dynamics in peptides: quantum chemical calculations and molecular dynamics simulations on N-acetylalanyl-N'-methylamide, *THEOCHEM,* **1993**, 104, 211-19.

178. Haile J. M., Molecular Dynamics Simulation: Elementary Methods, New York: Wiley-Interscience, **1997**.

179. Metropolis N., The Beginning of the Monte Carlo Method, *Los Alamos Science,* **1987**, 15, 125-30.

180. Metropolis N., Ulam S., The Monte Carlo Method, *J Amer Stat Assoc,* **1949**, 44, 335-41.

181. UNAIDS. Report on the global HIV/AIDS epidemic, **2006**.

182. UNAIDS. AIDS Epidemic Update December 2005, **2005**.

183. Meadows D. C., Gervay-Hague J., Current developments in HIV chemotherapy, *ChemMedChem,* **2006**, 1, 16-29.

184. Palani A., Tagat J. R., Discovery and Development of Small-Molecule Chemokine Coreceptor CCR5 Antagonists, *J Med Chem,* **2006**, 49, 2851-7.

185. Bartlett J. G., Moore R. D., Improving HIV therapy, *Sci Am,* **1998**, 279, 84-7, 89.

186. Ramratnam B., Bonhoeffer S., Binley J., Hurley A., Zhang L., Mittler J. E., Markowitz M., Moore J. P., Perelson A. S., Ho D. D., Rapid production and clearance of HIV-1 and hepatitis C virus assessed by large volume plasma apheresis, *Lancet,* **1999**, 354, 1782-5.

187. Kuritzkes D. R., Jacobson J., Powderly W. G., Godofsky E., DeJesus E., Haas F., Reimann K. A., Larson J. L., Yarbough P. O., Curt V., Shanahan W. R. Jr., Antiretroviral activity of the anti-CD4 monoclonal antibody TNX-355 in patients infected with HIV type 1, *J Infect Dis,* **2004**, 189, 286-91.

188. Kwong P. D., Wyatt R., Robinson J., Sweet R. W., Sodroski J., Hendrickson W. A., Structure of an HIV gp120 envelope glycoprotein in complex with the CD4 receptor and a neutralizing human antibody, *Nature,* **1998**, 393, 648-59.

189. Mascola J. R., Neutralization of HIV-1 Infection of Human Peripheral Blood Mononuclear Cells (PBMC): Antibody Dilution Method, In: Michael N. L., Kim J., Methods in HIV Research, Totowa: Humana Press Inc., **1999**, 09-19.

190. Tripos Bookshelf 7.0, Tripos Inc., 1699 South Hanley Rd., St. Louis, Missouri, 63144, USA, Tripos Inc., **2004**.

191. Roche Lexikon Medizin, 5. Auflage, München – Jena: Elsevier GmbH, Urban & Fischer Verlag, **2003**.

192. Vostrowsky O., Chemie der Naturstoffe Universität Erlangen, Erlangen, **2005**.

193. Somerville A. N., SciFinder Scholar, edited by Edward J. Walsh, *J Chem Educ,* **1998**, 75, 959, 975-6.

194. Scifinder Scholar, American Chemical Society, **2004**.

195. SYBYL 7.0, Tripos Inc., 1699 South Hanley Rd., St. Louis, Missouri, 63144, USA, Tripos Inc., **2004**.

196. Chang G., Guida W. C., Still W. C., An internal-coordinate Monte Carlo method for searching conformational space, *J Am Chem Soc,* **1989**, 111, 4379-86.

197. Mathematica 5.1, Wolfram Research, Champaign, USA, **2004**.

198. Maple 8, Maple Soft; Ontario, Canada, **2002**.

199. Pickett S. D., Mason J. S., McLay I. M., Diversity Profiling and Design Using 3D Pharmacophores: Pharmacophore-Derived Queries (PDQ), *J. Chem. Inf. Comput. Sci.,* **1996**, 36, 1214-23.

200. Jakes S. E., Willett P., Pharmacophoric pattern matching in files of 3-D chemical structures: selection of interatomic distance screens, *Journal of Molecular Graphics,* **1986**, 4, 12-20, 35.

201. Mason J. S., Morize I., Menard P. R., Cheney D. L., Hulme C., Labaudiniere R. F., New 4-point pharmacophore method for molecular similarity and diversity applications: overview of the method and applications, including a novel approach to the design of combinatorial libraries containing privileged substructures, *J Med Chem,* **1999**, 42, 3251-64.

202. SYBYL Mol2 File Format, Tripos Inc., **2001**.

203. Gasteiger J., Marsili M., Iterative partial equalization of orbital electronegativity: a rapid access to atomic charges, *Tetrahedron,* **1980**, 36, 3219-22.

204. Shoichet B. K., Kuntz I. D., Matching chemistry and shape in molecular docking, *Protein Eng,* **1993**, 6, 723-32.

205. Rashin A. A., Hydration phenomena, classical electrostatics, and the boundary element method, *J Phys Chem,* **1990**, 94, 1725-33.

206. DOCK, Version 4.0, Kuntz, I. D. (UCSF), **1997**.

7.6 Danksagungen

Bei meinen Eltern bedanke ich mich für die jahrelange Unterstützung, die mir Studium und Promotion erst ermöglicht haben.

Thomas Kühnemund danke ich für die jahrelange ausgezeichnete Zusammenarbeit innerhalb und außerhalb der Uni, sowie dafür, dass er mir bei dieser Arbeit stets mit gutem Rat zur Seite stand.

Bei Birgit Claasen bedanke ich mich für die schöne Zeit in Labor 20 und die vielfältige moralische Unterstützung.

Robin Job, Winrich Scherres, Jan Christoph Westermann und Thomas Kühnemund danke ich für das exzellente Korrekturlesen dieser Arbeit.

Christian Seeberger danke ich für die ausgezeichnete und geduldige Einführung in das Betriebssystem UNIX, und in die Administration der Server.

7.7 Lebenslauf

Persönliche Daten

Name	Boris Kröplien
Geburtsort/-datum	Hamburg, 03.11.1973
Kinder	eine Tochter, geboren am 24.04.2001

Ausbildung

Seit 03.2001 Dissertation bei Prof. Dr. B. Meyer. Thema: "*Fragmentbasierte Entwicklung neuer Leitstrukturen in silico durch Integration von 3D-Pharmakophorhypothesen*".

03.2000 - 12.2000 Diplomarbeit am Institut für Organische Chemie der Universität Hamburg, betreut durch Prof. Dr. B. Meyer. Thema: "*Synthese eines N-Typ-Glykopeptides analog der dritten extrazellulären Loop des humanen CCR5 und Bindungsstudien am GP120 des HIV-1*".

1996 - 2000 Hauptstudium der Chemie, Universität Hamburg,
Diplom: 21.12.2000 (Note: sehr gut)

1994 - 1996 Grundstudium der Chemie, Universität Hamburg,
Vordiplom: 21.10.1996 (Note: gut)

1993 - 1994 Grundwehrdienst, PzGrenBtl 72

1984 - 1993 GS Jahnschule, Hamburg
Abitur Juni 1993 (Note: sehr gut)

Wissenschaftliche Konferenzen

2003 1st Computational Chemistry Workshop, Bayer Healthcare, Wuppertal

2002 16th GDCh-CIC Chemoinformatics-Workshop, Kleinmachnow

2000 20th International Carbohydrate Symposium, Hamburg

Anstellungsverhältnisse

seit 2002 Datenbankentwicklung, Programmierung, Server-Administration, netXsite, Hamburg.

03.2001 - 03.2005 Wissenschaftlicher Mitarbeiter und UNIX-Systemadministrator, Institut für Organische Chemie, Universität Hamburg.

11.2001 - 12.2001 Lehrbeauftragter für das Grundpraktikum in Organischer und Anorganischer Chemie für Medizinstudenten, Medizinische Universität zu Lübeck.

03.2000 - 11.2000 Wissenschaftliche Hilfskraft, NMR-Abteilung OC, Universität Hamburg.

04.1997 - 04.1998 Wissenschaftliche Hilfskraft, GKSS-Forschungszentrum Geesthacht. Aufbau des Elbeinformationssystems für Forschung und Verwaltung ELBiS.

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und alle verwendeten Quellen und Hilfsmittel als solche gekennzeichnet habe.

Diese Arbeit ist zuvor in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde zur Erlangung des Doktorgrades vorgelegt worden.

Hamburg den 14.11 2006