Computeranwendungen in der Chemie: Visualisierung chemischer Reaktionen und Generierung von QSAR−Modellen

Vom Fachbereich Chemie und Chemietechnik
der Universität Paderborn genehmigte
Dissertation
zur Erlangung des Grades
eines Doktors der Naturwissenschaften
−Dr. rer. nat.−

von Diplom−Chemiker

Ingo Brunberg
aus Wickede (Ruhr)

Paderborn 2001

Die vorliegende Arbeit wurde im Fach für Organische Chemie der Universität Paderborn im Zeitraum von Juni 1997 bis Juli 2001 unter der Leitung von Herrn Prof. Dr. G. Fels angefertigt.

1. Referent:	Prof. Dr. G. Fels
2. Referent:	Prof. Dr. H.–S. Kitzerow
Eingereicht am:	12.07.2001
Tag der mündlichen Prüfung:	19.10.2001

Herrn Prof. Dr. G. Fels danke ich für die interessante Themenstellung sowie sein Interesse am Fortgang der Arbeit und seine Unterstützung durch viele anregende Diskussionen.

Herrn Prof. Dr. H.−S. Kitzerow danke ich für die bereitwillige Übernahme des Korreferats.

Weiterhin danke ich allen Mitarbeitern des Fachs Organische Chemie für die stets angenehme Arbeitsatmosphäre. Mein besonderer Dank gilt aber den Mitarbeitern aus dem Arbeitskreis Prof. Dr. G. Fels für die stete Diskussionsbereitschaft im Hinblick auf fachliche Fragen.

Inhaltsverzeichnis

1 Einleitung

Der Einsatz von Computern in der Chemie hat in den letzten Jahren stetig zugenommen und sich in vielen Teilbereichen fest etabliert. Zu den wichtigsten Anwendungsgebieten zählt das Molecular Modeling, welches durch den Einsatz von Quantenmechanik oder klassischer Mechanik (Kraftfeldern) Berechnungen an Einzelmolekülen oder Molekülaggregaten ermöglicht. So werden am Rechner heute routinemäßig Größen wie Atomladungen, Dipolmomente, Lipophilie, Siedepunkte, chemische Verschiebungen, Kopplungskonstanten, Enthalpien, Entropien usw. bestimmt. Dazu gehören auch Geometrieoptimierungen, die Modellierung dynamischer Vorgänge (chemische Reaktionen, Konformationsänderungen), das rechnerbasierte Wirkstoffdesign und nicht zuletzt die Visualisierung der dabei erhaltenen Ergebnisse. Wertvolle Hilfe leistet der Computer ebenfalls bei der Interpretation und Aufklärung von Spektren sowie der Recherche in Substanz– und Reaktionsdatenbanken.

Ganz nachhaltig hat letztendlich auch das Internet die Art des Arbeitens revolutioniert. Es ermöglicht Wissenschaftlern aus aller Welt, auf direktem Wege miteinander zu kommunizieren sowie Daten und Ergebnisse auszutauschen. Über das Internet hat jedermann den sofortigen Zugriff auf riesige Wissensbestände. Literaturbeschaffung, die früher den Gang in die Bibliothek und das manuelle Durchblättern ganzer Zeitschriftenbestände erforderte, läßt sich heute zumeist per Mausklick erledigen. Dabei bietet das elektronische Publizieren weit mehr Möglichkeiten als das gedruckte Medium. So können einzelne Aufsätze oder ganze Datenbanken nach Stichworten durchsucht werden. Gerade in der Chemie können über ein dreidimensionales Molekülmodell Informationen vermittelt werden, die auch ein aufwendiges Stereobild nicht ausreichend wiederzugeben vermag.

Bei all den Möglichkeiten, die die moderne Computerchemie heute bietet, ist es doch nicht ihre Aufgabe, den klassischen Chemiker überflüssig zu machen, sondern seine Arbeit zu ergänzen und ihm Mittel an die Hand zu geben, mit dem er die Effizienz seiner Arbeit steigern kann. So kann sich etwa der Synthesechemiker Vorschläge für Synthesewege aus Reaktionsdatenbanken oder einem Expertensystem generieren lassen. Der analytische Chemiker, der sich beispielsweise mit der Aufklärung der Struktur von Naturstoffen befaßt, weiß Programme zu schätzen, die es ihm erlauben, Spektren zu simulieren und diese mit den experimentell gewonnenen zu vergleichen. In der Arzneistofforschung können Vorhersagen der Wirkungen neuer potentieller Kandidaten als Filter dem aufwendigen Stadium der Synthesen und klinischen Tests vorgeschaltet werden.

2 Zielsetzung

Das Ziel des ersten Teils dieser Arbeit war die Entwicklung eines Systems zur Animation chemischer Reaktionen. Eine lediglich als Film (z. B. Avi, MPEG, Quicktime) ablaufende Animation besitzt eine Reihe von Nachteilen. So kann der Betrachter immer nur die vorgegebene Ansicht verfolgen, wodurch ihm die Möglichkeit, sich einen realistischen räumlichen Eindruck von dem Gesehenen zu verschaffen, erheblich erschwert wird. Außerdem führen die mangelnden Interaktionsmöglichkeiten dieses Ansatzes bei ihm schnell zu einem nachlassenden Interesse. Daher stellte sich die Aufgabe, ausgehend von quantenmechanisch berechneten Reaktionsverläufen, eine möglichst repräsentative Auswahl von einzelnen Strukturen auf einem Reaktionspfad in Form einer dreidimensional räumlichen Darstellung zu einer Animationssequenz in Ball&Stick−Darstellung zu vereinen. Insbesondere sollte Interaktivität, in Form einer freien Drehbarkeit der Szenenansicht mit Hilfe der Maus, gewährleistet sein. Die zweite wichtige Aufgabe bestand darin, eine Animationssequenz um die Möglichkeit zur dreidimensionalen Darstellung physikalischer Eigenschaften des betreffenden Reaktionssytems in Form von optional farblich kodierten Isoflächen zu ergänzen. Diese Isoflächen (meistens von Orbitalen oder Elektronendichten) sind z. B. aus Abbildungen in Lehrbüchern der organische Chemie bekannt und können dem Betrachter eine große Hilfe zum Verständnis und zur Interpretation des dargestellten Reaktionsmechanismus sein.
Um einer möglichst großen Zahl von Computernutzern den Zugang zu den Visualisierungen zu gewähren, stand schließlich die Internetintegration des Projektes im Vordergrund.

Gegenstand des zweiten Teils dieser Arbeit war die Entwicklung eines modularen, leicht zu erweiternden Programms zur Aufstellung von quantitativen Struktur−Wirkungsbeziehungen (QSAR: Quantitative Structure Activity Relationships). QSAR−Methoden werden heute in der Arzneimittelforschung erfolgreich eingesetzt, um biologische Aktivitäten experimentell noch nicht getesteter Substanzen vorherzusagen und Hinweise zur Weiterentwicklung bereits bekannter Wirkstoffe zu erhalten.
Bei der Programmentwicklung sollten bereits bekannte, aber auch neue Algorithmen implementiert werden, die beim praktischen Einsatz in einer neuartigen und flexiblen Weise miteinander kombiniert werden können. Schließlich sollten die im erstellten Programm eingebauten Verfahren anhand eines schon mit anderen Mitteln untersuchten Beispieldatensatzes evaluiert werden.

3 Die Programmiersprache Java

Die wichtigsten der in dieser Arbeit entwickelten Programme wurden in Java implementiert. Daher sollen hier kurz die Gründe für die Wahl dieser Programmiersprache und einige ihrer wichtigsten Eigenschaften besprochen werden.

Java wurde ursprünglich unter dem Namen Oak zur Steuerung intelligenter Elektrogeräte im Haushalt, wie z. B. Set−Top Boxen für Fernseher, entwickelt. Als sich im Jahr 1993 abzeichnete, daß sich kein ausreichender Markt für die anvisierten Anwendungen entwickeln würde, und etwa zur gleichen Zeit die Popularität des Internet explosionsartig zunahm, entschloß man sich bei Sun Microsystems, Oak bzw. Java für den Einsatz als Internet−Programmiersprache weiter zu fördern. Eine Reihe von Eigenschaften prädestinieren Java für diesen Zweck.

- Java ist portabel. Der Quellcode eines Java−Programms wird nicht direkt in Maschinencode übersetzt, sondern mit einem Compiler in sogenannten Bytecode vorübersetzt. Der Bytecode selber wird von einer virtuellen Java Maschine interpretiert. Auf diese Weise kann ein kompiliertes Java−Programm auf jeder Plattform ausgeführt werden, für die solch eine virtuelle Maschine verfügbar ist.
- Java ist robust und sicher. Beim Kompilieren werden Java−Programme einer strengen Prüfung unterzogen, und zur Laufzeit auftretende Fehler werden über einen Ausnahmemechanismus abgefangen. Die virtuelle Maschine, die den Java−Bytecode interpretiert, kann dem Programm den Zugriff auf bestimmte Systemressourcen verweigern. So wird z. B. von Internetbrowsern standardmäßig verhindert, daß ein in einer HTML−Seite eingebettetes Java−Programm Zugriff auf die Festplatte des Client−Rechners erhält oder Netzwerkverbindungen zu anderen Rechnern, als demjenigen, von dem es stammt, aufbauen kann. Diese Sicherheitsrestriktionen können nur dadurch umgangen werden, daß der Benutzer dem Programm die gewünschten Rechte explizit einräumt. Anders als z. B. C oder C++ kennt Java keine Zeiger (englisch: Pointer), die einen unkontrollierten Zugriff auf beliebige Speicherbereiche ermöglichen und bei fehlerhaftem Einsatz im günstigsten Fall zu Programmabstürzen führen können.
- Java ist objektorientiert. In der Terminologie der objektorientierten Programmierung sind Objekte Instanzen von Klassen. Das sind abgeschlossene Programmblöcke, in denen objekt− und klassenspezifische Variablen und Methoden miteinander vereint sind. Ein Objekt kommuniziert mit anderen Objekten immer über eine öffentliche Schnittstelle; die Methoden und Variablen, die das Objekt intern zur Erfüllung seiner Aufgaben verwendet, können mit dem Schlüsselwort

private gekennzeichnet und damit völlig vor unbefugtem Zugriff geschützt werden. Dadurch wird die Entwicklung größerer Programme vereinfacht, da einzelne Module, in Form von Klassen, unabhängig voneinander geschrieben werden können, ohne die Einzelheiten der Implementierung der anderen Programmteile zu kennen. Ein weiterer Vorteil des objektorientierten Entwurfs ist, daß die einmal entwickelten Klassen leicht in anderen Programmen weiterverwendet werden können.

- Java ist multithreaded. Beim Ausführen eines Programms wird ein Prozeß gestartet, der seinen eigenen Speicherbereich zugewiesen bekommt. Zur Abarbeitung von Teilaufgaben kann dieser Prozeß einen oder mehrere Threads starten, die nebeneinander in demselben gemeinsamen Speicherbereich ablaufen. Ein wirklich gleichzeitiger Ablauf mehrerer Threads ist natürlich nur auf Mehrprozessormaschinen möglich, sonst wird die zur Verfügung stehende Prozessorzeit auf die einzelnen Threads aufgeteilt. Das größte Problem bei der Programmierung ist die Vermeidung von Konflikten, die dadurch entstehen können, daß zwei nebeneinander ablaufende Threads auf denselben Speicherbereich zugreifen. Java unterstützt mit der Klasse *java.lang.Thread* ein sehr einfach zu benutzendes Modell zur Erzeugung von Threads und stellt mit dem Schlüsselwort *synchronized* eine Möglichkeit zur Konfliktvermeidung bereit.

4 Visualisierung chemischer Reaktionen

In diesem Teil werden die Arbeiten beschrieben, die als Beitrag zum BMBF−Projekt *newChem* geleistet und schließlich zum Reaktionsvisualisierungssystem *CAVOC* weitergeführt wurden.

4.1 Modellierung chemischer Reaktionen

Die grundlegende Voraussetzung für die dreidimensionale computergrafische Darstellung einer chemischen Reaktion ist das Vorliegen einer den Reaktionsablauf beschreibenden Serie von 3D−Strukturen. In der Regel wird man diese Strukturen durch eine quantenmechanische Berechnung des Reaktionspfades gewinnen, da Intermediate einer chemischen Reaktion experimentell, wenn überhaupt, nur mit erheblichem technischen Aufwand zugänglich sind.
Bei der Wahl der Rechenmethode hat man die zu erreichende Genauigkeit gegen den erforderlichen Zeitaufwand abzuwägen. Im Molecular Modeling werden im wesentlichen drei Klassen von Verfahren eingesetzt. Auf der einen Seite steht die Molekülmechanik, die über empirisch parametrisierte Kraftfelder mit geringem Rechenaufwand sehr gute Geometrievorhersagen für Moleküle machen kann. Der Erfolg bei deren Einsatz hängt jedoch entscheidend davon ab, ob der Parametersatz des benutzten Kraftfeldes auf das jeweilige Problem abgestimmt ist. Während für die meisten organisch chemischen Stoffklassen adäquate Kraftfelder zur Beschreibung von stabilen Grundzustandskonformationen existieren, gibt es zur Zeit noch kein allgemein einsetzbares Kraftfeld für die Optimierung von Übergangszuständen. Demgegenüber stehen die quantenchemischen Verfahren, von denen die ab initio−Methoden ganz ohne empirisch bestimmte Parameter auskommen und insofern am besten zur Modellierung von Übergangszuständen und Reaktionen geeignet erscheinen. Sie erforden jedoch, je nach Größe des verwendeten Basissatzes und Art und Anzahl der Atome im betrachteten System, enorme Rechenzeiten. Aus diesem Grund wurden die im Rahmen dieser Arbeit behandelten Reaktionsmechanismen alle mit semiempirischen Verfahren berechnet, unter denen die auf der NDDO−Näherung (neglect of diatomic differential overlap) basierenden Methoden MNDO (modified neglect of diatomic overlap [1]), AM1 (Austin method 1 [3]) und PM3 (Parameterization method 3 [2]) die für Geometrieoptimierungen und Energieberechnungen wichtigste Gruppe bilden. Zwar sind auch die genannten semiempirischen Verfahren für die Beschreibung stabiler Grundzustandskonformationen parametrisiert, doch sind die erzielbaren Ergebnisse für nichtquantitative Zwecke, wie eben zur Visualisierung, in den meisten Fällen durchaus ausreichend.

Ist man bereit, eine mathematisch korrekte quantenmechanische Behandlung noch weiter in den Hintergrund zu rücken, können Molekülgeometrien zur Simulation von Reaktionsmechanismen für Demonstrations– bzw. Lehrzwecke alternativ auch mit halbautomatischen Näherungsverfahren generiert werden. Oft ist es nämlich, zumindest ohne den Einsatz spezieller, aufwendiger Rechenmethoden, nicht möglich, bestimmte Reaktionen, z. B. solche, die erst durch Solvenseinflüsse energetisch favorisiert werden, mit den zur Zeit gängigen Programmen zu berechnen. In diesen Fällen kann der einzig gangbare Weg zur Erzeugung einer Reaktionsanimation darin bestehen, bestimmte geometrische Parameter, z.B. einen Abstand zwischen zwei Atomen oder einen Bindungswinkel, über den Reaktionsverlauf hinweg vorzugeben und nur die restlichen Parameter zu optimieren. Oder man führt, wenn alles andere versagt, einfach eine Interpolation zwischen den Atomkoordinaten der Edukte und Produkte aus.

4.1.1 Berechnung und Verifikation der Übergangszustände

Der erste, wichtigste und aufwendigste Schritt bei einer Reaktionsberechnung ist das Auffinden aller an der Reaktion beteiligten Übergangszustände. Bevor man eine solche Berechnung angeht, hat man in der Regel eine Vorstellung vom zugrunde liegenden Reaktionsmechanismus und dadurch auch von den ungefähren Geometrien der involvierten Übergangszustände.
Die Berechnung von Übergangszuständen ist nicht trivial. Sie entsprechen Sattelpunkten auf der Energiehyperfläche (Abbildung 1) und sind dadurch mit den Methoden der numerischen Mathematik viel schwieriger zu optimieren, als etwa die günstigsten Edukt– und Produktkonformationen, welche echte Minima darstellen.

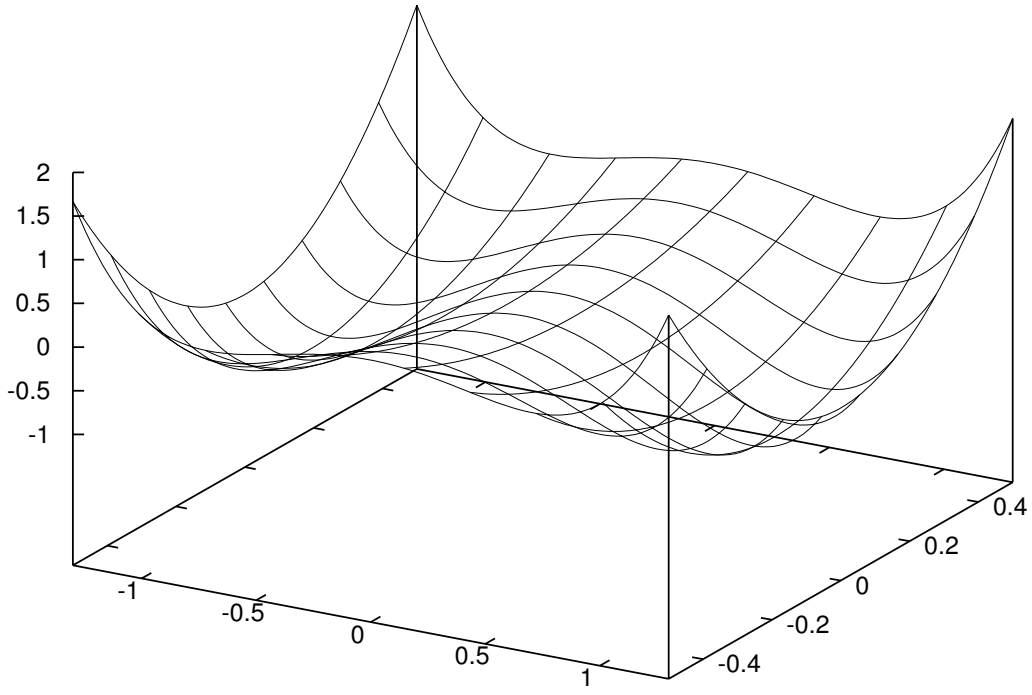

Abbildung 1: Die Funktion $f(x, y) = x^4 + 4x^2y^2 - 2x^2 + 2y^2$ hat an der Stelle (0, 0) einen Sattelpunkt und Minima bei (−1, 0) und (1, 0).

Charakterisiert ist ein Übergangszustand durch genau eine negative Kraftkonstante (ein negativer Eigenwert der Hesse−Matrix), die einer imaginären Schwingungsfrequenz entspricht, welche der Reaktionskoordinate zuzuordnen ist.

Die numerische Berechnung eines Sattelpunktes führt normalerweise nur dann zum Erfolg, wenn der Startpunkt bereits sehr nahe am Sattelpunkt liegt, in einem Gebiet, wo die Hesse−Matrix nur einen negativen Eigenwert besitzt. Um zu den erforderlichen guten Näherungen für die Übergangszustandsgeometrien zu kommen, gibt es verschiedene Möglichkeiten. Die einfachste besteht darin, daß man einen bekannten Übergangszustand einer eng verwandten Reaktion, z. B. einer mit leicht unterschiedlichen Substituenten an einem abseits vom Reaktionszentrum gelegenen Molekülteil, als Muster nimmt.

Eine andere Methode besteht darin, sich eine Näherung für den betreffenden Übergangszustand dadurch zu konstruieren, daß man eine für die betrachtete Elementarreaktion charakteristische Koordinate (z.B. einen Atomabstand bei einer Bindungsbildung bzw. einem Bindungsbruch) in

festen Schritten variiert und nach jedem Schritt eine normale Optimierung unter Festlegung dieser Koordinate durchführt. Die Struktur mit der höchsten Energie entlang dieser Reaktionskoordinate stellt dann die Näherung für den Übergangszustand dar. Führt dieses Verfahren nicht zum Ziel, können auch mehrere Koordinaten gleichzeitig variiert werden. Eine solche Gittersuche ist jedoch sehr zeitaufwendig, da sie u. U. eine große Zahl von Optimierungen erfordert. Neben diesen Möglichkeiten gibt es noch halbautomatische Verfahren, die eine Konstruktion der Ausgangsstruktur für eine Übergangszustandsoptimierung erlauben. Dazu zählen z.B. die in Spartan [5] implementierte „Linear Synchronous Transit"−Methode und die weiterentwickelte, in Gaussian 94/98 [4] eingebaute „Synchronous Transit−Guided Quasi−Newton"−Methode. Diese Verfahren erfordern die Angabe zweier Strukturen, je einer auf der Produkt− und der Eduktseite, sowie die Zuordnung der korrespondierenden Atome. Zwischen diesen beiden Strukturen wird dann durch eine Art Interpolation die angenäherte Struktur des Übergangszustandes konstruiert.

4.1.2 Berechnung der Reaktionspfade

Bei der Berechnung der Reaktionspfade werden Sätze von Strukturen generiert, die später zur Visualisierung der Reaktionsabläufe dienen können. Dies dient aber auch zur endgültigen Verifikation eines gefundenen Übergangszustandes, da die Untersuchung der imaginären Schwingungsfrequenz allein noch kein ausreichendes Kriterium dafür ist, daß dieser genau die betrachtete Reaktion beschreibt. Das am häufigsten eingesetzte Verfahren ist die IRC−Rechnung (IRC ≙ Intrinsische Reaktionskoordinate). Unter einem IRC versteht man einen idealisierten Reaktionspfad, der mit unendlich kleiner Schrittweite immer auf dem energetisch günstigsten (steilsten) Weg von einem Sattelpunkt auf der Energiehyperfläche zu einem den Edukten bzw. Produkten entsprechenden Minimum führt [9]. Bei einer real ablaufenden Reaktion ist dagegen beim Überqueren eines Übergangszustandes immer ein gewisser Energieüberschuß vorhanden, der dazu führt, daß das System auf seinem Weg zum Minimum um den jeweils energetisch günstigsten Zustand oszilliert (DRC ≙ Dynamische Reaktionskoordinate). Drei verwandte Algorithmen zur IRC−Bestimmung finden sich in [8], [7] und [6].

Vom Übergangszustand ausgehend werden zwei IRC−Rechnungen ausgeführt, eine entlang der durch die imaginäre Frequenz vorgegebenen Richtung, eine entgegengesetzt dieser Richtung. Um zu kontrollieren, ob die Rechnungen zu den günstigsten Konformationen der Edukte, Produkte bzw. einer Zwischenstufe geführt haben, werden die letzten der bei den IRC−Rechnungen generierten Strukturen mit denen der optimierten Zielstrukturen verglichen. Ergeben sich dabei

größere Abweichungen, müssen entweder weitere IRC−Rechnungen angeschlossen, oder sogar neue Übergangszustände für die erforderlichen Konformationsänderungen identifiziert werden. Dies muß solange wiederholt werden, bis der gesamte Pfad berechnet ist.

4.2 Computergrafische Grundlagen – Isoflächen

Zur Repräsentation einer dreidimensionalen Szene im Computer wird in der Regel ein Arrangement von geometrischen Primitiven, wie Kugeln, Zylindern, Kegeln und Quadern, die möglicherweise noch mit Texturen versehen werden, verwendet. Werden kompliziertere Formen benötigt, so lassen sich diese aus einfachen Polygonflächen aufbauen, womit auch die zur 3D−Programmierung benutzten Grafikbibliotheken die erwähnten geometrischen Primitive approximieren. Als besonders geeignet haben sich hierbei die einfachsten Polygone, die Dreiecke, herausgestellt. Mit ihnen kann jede denkbare Flächenform beliebig genau und ohne Lücken angenähert werden. Außerdem sind die Routinen zum Umgang mit Dreiecken in moderner Computerhardware bereits integriert (3D−Beschleuniger Grafikkarten), so daß aus deren Verwendung ein enormer Geschwindigkeitsvorteil resultiert.

Ein realistischer dreidimensionaler Eindruck wird erst durch die Kombination der Geometrien mit mindestens einer Lichtquelle und einem Reflexionsmodell erreicht. Zur Berechnung des Aussehens (Farbeindruck, Schattierung) eines Körpers in einer 3D−Szene benötigt das Reflexionsmodell neben den Eckpunkten (Vertices) der Dreiecke noch weitere Parameter, wie z. B. Farbwerte und Normalenvektoren der einzelnen Vertices. Die Normalen dienen zur Berechnung der Ein− und Ausfallswinkel auftreffender Lichtstrahlen und können dabei helfen, die Übergänge (Kanten) zwischen benachbarten Dreiecken optisch zu glätten.

Das grafische Element, auf dem das besondere Augenmerk dieser Arbeit liegt, ist die Isofläche. Mit ihrer Hilfe soll ein Einblick in die Abläufe bei chemischen Reaktion auf molekularer Ebene gewährt werden. Mathematisch gesehen stellt sie die Menge aller Raumpunkte dar, an denen die ihr zugrunde liegende Funktion einen ganz bestimmten Wert, den Isowert, besitzt. Der meistverwendete Algorithmus zur Erzeugung von Isoflächen eines auf einem regelmäßigen Gitter gegebenen Datensatzes einer Skalarfunktion von drei Koordinaten ist der Marching Cubes−Algorithmus von Lorenson und Cline [10]. Prinzipiell läßt er sich wie folgt beschreiben: Systematisch werden alle Würfelelemente des gesamten Gitters abgearbeitet. Dabei wird bestimmt, an welchen Ecken eines Würfels die Funktionswerte größer, kleiner oder gleich dem gegebenen Isowert sind. Wenn ein Würfel Ecken mit Werten größer oder gleich und Werten kleiner oder

gleich dem Isowert besitzt, so wird dieser Würfel von der Oberfläche durchschnitten. Dabei sind insgesamt $2^8 = 256$ verschiedene Fälle möglich. Durch Berücksichtigung von Symmetriebeziehungen lassen sich diese auf nur 14 unterschiedliche Fälle zurückführen. Diese sind in Abbildung 2 gezeigt.

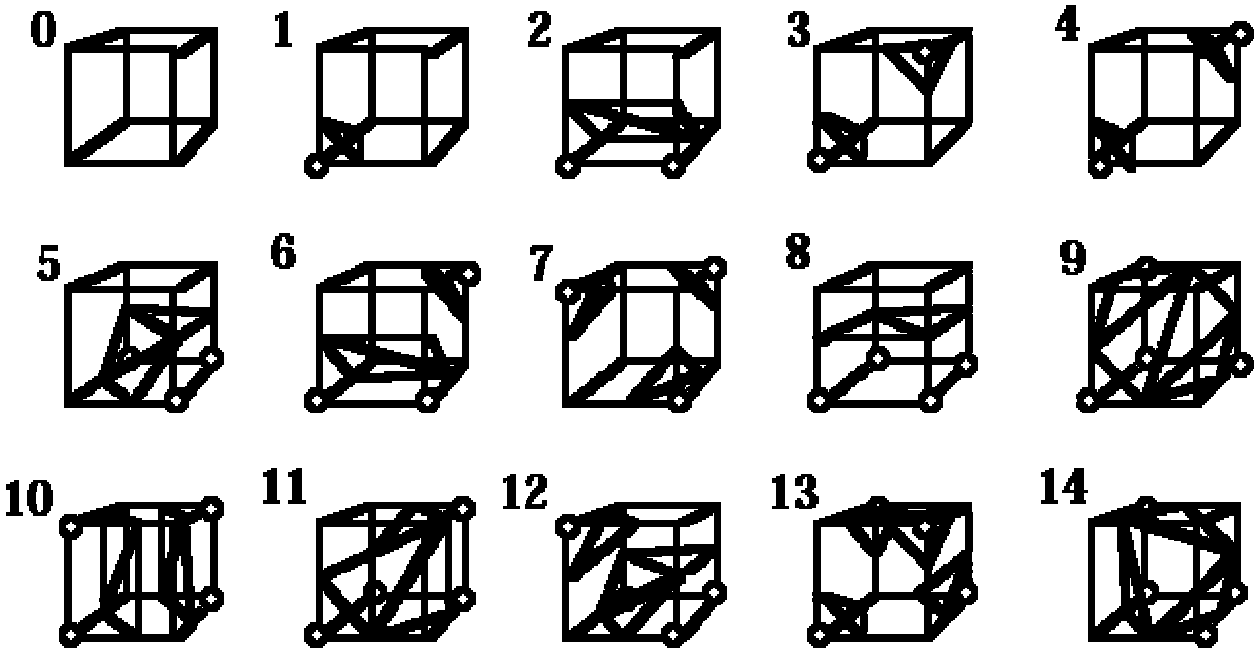

Abbildung 2: Die 14 verschiedenen Fälle im Marching Cubes−Algorithmus

Die genaue Position der Schnittpunkte auf den Würfelkanten wird durch lineare Interpolation ermittelt. Optional können anschließend die Normalenvektoren zu den so gefundenen Oberflächenpunkten berechnet werden.

Der Marching Cubes−Algorithmus liefert Isoflächen in Form von sogenannten „Indexed Trianglesets“, einer hinsichtlich des Speicherbedarfs sehr effizienten Form der Darstellung von 3D−Körpern. Dabei werden die Koordinaten der einzelnen Vertices und die Konnektivitäten, die angeben, wie die Vertices zu Dreiecken verknüpft sind, abgespeichert. Der Erzeugung der grafischen Rohdaten (Vertices, Normalen und Konnektivitäten) von Isoflächen wird in dieser Arbeit besondere Bedeutung beigemessen.

4.3 Generierung der grafischen Rohdaten für das *newChem*−Projekt

Das *newChem*−Projekt wurde unter dem Motto ins Leben gerufen, neue Wege bei der

Visualisierung organisch chemischer Reaktionen zu beschreiten. Die dahinter stehende Idee war, die üblichen Animationen in Ball&Stick–Darstellung mit 3D–Oberflächen aussagekräftiger physikalischer Eigenschaften zu versehen, wie sie für Einzelmoleküle durch Programme wie z. B. Spartan bekannt geworden sind. Dadurch sollte ein tieferer Einblick in die bei einer Reaktion auf molekularer Ebene ablaufenden Geschehnisse gewährt werden. Als am aussagekräftigsten für chemische Reaktionen wurden folgende Oberflächen bewertet und folglich im Projekt aufgenommen:

- eine Isoflächendarstellung des energetisch höchstliegenden besetzten Orbitals (HOMO)
- eine Isofächendarstellung des energetisch tiefstliegenden unbesetzten Orbitals (LUMO)
- eine Isoflächendarstellung der Elektronendichte, bei der jeder Oberflächenpunkt, mit Hilfe einer Regenbogenfarbskala zur Kodierung, je nach Größe des dort vorliegenden elektrostatischen Potentials eingefärbt ist (Elpot)

Die Arbeit am Projekt *newChem* wurde im wesentlichen von drei Seiten geleistet. An erster Stelle stand die Arbeitsgruppe von Prof. Dr. H. Zipse (LMU München), die den Großteil der visualisierten Reaktionen berechnet hat. Für die Realisierung des unter dem Betriebssystem Microsoft Windows laufenden Programms war die Firma *interactive Systems* aus Marburg verantwortlich. Die dazwischen liegende Aufgabe der Erzeugung von einfach in das Programm einzubindenden 3D–Daten und Energiediagrammen wird als Gegenstand dieser Arbeit hier beschrieben. Das Vorgehen gliedert sich im einzelnen in folgende Schritte:

1. Aufspaltung der IRC–Dateien
2. Selektion geeigneter Einzelstrukturen
3. Durchführung von Koordinatentransformationen (optional)
4. Berechnung und Aufbereitung der Isoflächen
5. Erstellung der Energiediagramme

4.3.1 Aufspaltung der IRC–Dateien

Ein für das weitere Vorgehen notwendiger, technischer Schritt ist die Erzeugung einzelner Dateien für jede der bei den IRC–Rechnungen generierten intermediären Strukturen. Dabei wird die richtige Sortierreihenfolge der Einzelstrukturen durch eine geeignete Numerierung innerhalb der

Dateinamen gewährleistet. Ein einfaches Beispiel soll das Vorgehen verdeutlichen: Bei einer Reaktion mit nur einem Übergangszustand erhält man zwei IRC−Dateien. Diejenige der beiden, die den Weg zum Edukt beschreibt, enthalte z. B. 50 Strukturen. Dann wird die erste Struktur daraus in einer Datei mit dem Namen „0050“ abgespeichert, die zweite in einer Datei mit dem Namen „0049“ usw. bis „0001“. Der Übergangszustand bekommt dann die „0051“ zugewiesen. Die IRC−Datei, die den Weg zum Produkt beschreibt, enthalte in diesem Beispiel 40 Strukturen. Dann bekommt die erste davon die Bezeichnung „0052“, die zweite „0053“ usw. bis zur letzten mit der Bezeichnung „0091“. Als Ergebnis dieser Prozedur hat man also einen Satz von Dateien, die mit Programmen wie „Babel“ [11] unkompliziert in andere Formate konvertiert werden können, und so der Visualisierung in einer Reihe von Modeling−Programmen zugänglich sind.
Zur Bewältigung dieser Aufgabe wurde das mit einer grafischen Benutzeroberfläche ausgestattete Tcl/Tk−Skript „irctool“ entwickelt, womit sich die einzelnen Abschnitte einer berechneten Reaktion unter Angabe der Verarbeitungsrichtung einlesen lassen. Unterstützt werden die folgenden Formate:

- IRC−Ausgaben sowie Grid−Rechnungen von Gaussian 94 und Gaussian 98
- Gaussian „.log“−Dateien einzelner Optimierungen
- IRC−Ausgaben von Gamess [12]
- IRC−Ausgaben von Mopac 93 [13]
- Mopac „.out“−Dateien einzelner Optimierungen

4.3.2 Selektion geeigneter Einzelstrukturen

Die Anzahl der bei den IRC−Rechnungen zu einer Reaktion generierten, einzelnen Strukturen kann mit den eingesetzten Programmen und verwendeten Schrittweitenparametern von ca. 50 bis weit über 1000 reichen. Da die Weiterverarbeitung so vieler Strukturen insbesondere bei der Umsetzung in 3D−Computergrafik und der Erzeugung der für die Isoflächen benötigten Volumendaten eine Datenmenge erzeugen würde, die mit heutigen Rechnerkapazitäten nicht gut zu bewältigen wäre und überdies nicht in vertretbarer Zeit über das Internet gesendet werden könnte, muß eine Vorauswahl von Einzelstrukturen getroffen werden. Dabei sollten zwei Kriterien erfüllt werden:

- Der chemische Informationsgehalt einer sich aus der Auswahl ergebenden grafischen Animation sollte möglichst hoch sein. Das bedeutet, daß an den besonders interessanten Stellen im

Reaktionsablauf, insbesondere den Übergangszuständen, mehr, dichter beieinanderliegende Strukturen ausgewählt werden als im Bereich der Minima, wo die Änderungen von einer Struktur zur nächsten in energetischer Hinsicht normalerweise nur gering sind.

- Es sollten keine zu großen und auffälligen Sprünge zwischen den einzelnen Frames auftreten, die den Eindruck einer flüssigen Animation stören. Das bedeutet, daß sich die Geometrien zweier aufeinanderfolgender, ausgewählter Strukturen nicht zu sehr unterscheiden dürfen.

Um den dargelegten Anforderungen zufriedenstellend gerecht zu werden, wurde die Selektion bei den bis jetzt bearbeiteten Reaktionen manuell vorgenommen. Dabei hat sich empirisch gezeigt, daß eine Anzahl von 30 ausgewählten Strukturen den besten Kompromiß zwischen benötigtem Speicherplatz und Detailtreue der Animationen ergibt.
Es wurde so vorgegangen, daß die aus der Aufspaltung resultierenden, im Mopac−, Gamess− oder Gaussian−Format vorliegenden Dateien mit „Babel“ in andere Formate (xyz, Spartan) konvertiert wurden, um mit Hilfe von Animationen unter Berücksichtigung aller pro Reaktion generierten Strukturen in Programmen wie Spartan oder „XMakemol“ [14] visuell die geeignetsten zu bestimmen.

4.3.3 Durchführung von Koordinatentransformationen

Aus verschiedenen Gründen kann es wünschenswert oder sogar erforderlich sein, bestimmte Sequenzen von Strukturen einer Koordinatentransformation zu unterwerfen. So kann z. B. auf die zeitraubende Berechnung der zweiten Hälfte des Reaktionspfades bei Identitätsreaktionen verzichtet werden, d. h. bei Reaktionen mit gleichen Edukten und Produkten, wie z. B. der Cope−Umlagerung von 1,5−Hexadien, weil sie sich, je nach Symmetrie, aus der bereits berechneten ersten Hälfte durch Spiegelung oder Inversion konstruieren läßt.
Das wichtigste Einsatzgebiet von Koordinatentransformationen sind aber die Reaktionen mit mehreren Übergangszuständen. Zwei, von benachbarten Übergangszuständen ausgehende, zu einem gemeinsamen Zwischenprodukt führende IRC−Rechnungen stehen im allgemeinen nicht in der passenden geometrischen Beziehung zueinander. Aus der Differenz der räumlichen Orientierungen der von beiden IRC−Rechnungen gelieferten Strukturen für das Zwischenprodukt kann nach deren Übereinanderlegen (z. B. mit dem RMS−Kriterium) eine Transformationsmatrix bestimmt werden, die dann auf alle Strukturen auf der einen Seite vom Zwischenzustand angewendet werden muß.
Anwendung findet die Koordinatentransformation auch bei Problemen von eher „kosmetischem“

Charakter. So ist z. B. bei der Berechnung der S_N2−Reaktion zwischen Chlormethan und Chlorid das Chlorid−Ion in den Ursprung des Koordinatensystems gelegt worden, was bei der Visualisierung mit den Originalkoordinaten den unerwünschten Effekt hat, daß sich das Chlormethan von dem Chlorid−Ion wegzubewegen scheint, was dem chemischen Verständnis von Substrat und Reagens widerspricht. In diesem Fall ist es also sinnvoll, jede Struktur so zu verschieben, daß sich immer das C−Atom im Koordinatenursprung befindet.
Angeregt durch diese Problemstellungen wurde das Tcl−Skript „kdtrans.tcl“ entworfen, das die folgenden Funktionen zur Manipulation einer 3D−Struktur anbietet:

- Translation in x−, y− oder z−Richtung
- Inversion an einem beliebigen Punkt
- Spiegelung an einer durch Orts− und Normalenvektor gegebenen Ebene
- allgemeine Transformation, gegeben durch vier Raumpunkte mit linear unabhängigen Ortsvektoren und deren Bildpunkte im neuen Koordinatensystem

Das Skript unterstützt die 3D−Formate pdb, mol2, xyz und Gaussian (kartesisch).
Bei bestimmten Transformationen, wie z. B. Spiegelungen oder Inversionen, kann es zu einer Vertauschung von Atomen kommen, so daß die einheitliche Numerierung unter den Frames nicht mehr gegeben ist. Dann müssen die Atome von Hand in die richtige Reihenfolge gebracht werden, was am einfachsten durch eine skriptgesteuerte Zeilenvertauschung im xyz−Format zu realisieren ist.

4.3.4 Berechnung und Aufbereitung der Isoflächen

Die Elpot−Flächen wurden im wesentlichen mit den Programmen Sybyl/Molcad [16] (für die eigentlichen Elektronendichteoberflächen) und Gaussian 94/98 (für die elektrostatischen Potentiale) erzeugt. Für die Generierung der Isoflächen der Orbitale wurden Gaussian 94/98 sowie eine aus dem Internet bezogene Marching Cubes−Implementierung („isovis_pw“ [15]) eingesetzt.
Als Ausgangsbasis müssen die ausgewählten Strukturen im Sybyl/mol2−Format (für Molcad) und im Gaussian−Format (für Gaussian 94/98) vorliegen, was durch eine Konvertierung mit „Babel“ erreicht wird. Die weiteren, über ein Skript automatisierten Schritte werden im folgenden beschrieben.
Zuerst wird mit dem Molcad−Modul eine Elektronendichteoberfläche berechnet. Als Parameter

werden der Name einer Sybyl/mol2−Datei und eine ganze Zahl im Bereich von 1 bis 9 als Qualitätsmaß übergeben. Bei der Wahl dieses Qualitätsmaßes muß ein Kompromiß zwischen der Auflösung (Anzahl der Vertices) der errechneten Oberfläche und deren Speicherbedarf gefunden werden. Für das *newChem*−Programm hat sich einzig ein Wert von 3 als akzeptabel herausgestellt, da die mit einem Wert von 2 erzeugten Oberflächen schon deutlich kantig aussehen, während die Datenmengen bei Werten von 4 und größer einfach die Kapazitäten der meisten Heimcomputer überschreiten. Als Isowert wird die Voreinstellung von 0,003 gewählt. Die von Molcad erzeugte Ausgabedatei enthält in einem Binärformat u. a. die Koordinaten der Oberflächenvertices, deren Normalenvektoren sowie die Triangulierung. Diese Daten werden mit einem kleinen, in C geschriebenen Programm ausgelesen und in einem eigenen ASCII−Format abgelegt.
An die für die Gaussian−Rechnung vorbereitete Eingabedatei werden die Koordinaten der Vertices der Molcad−Oberfläche, die Bezeichnungen der zu berechnenden Orbitale (HOMO, LUMO) sowie der Name der zu erzeugenden Cube−Datei mit den Orbitaldaten angehängt. Damit wird anschließend die Gaussian Single Point−Rechnung durchgeführt. Die dabei verwendeten Schlüsselwörter und deren Bedeutung sind im folgenden aufgelistet:

AM1	Damit wird festgelegt, daß die Berechnung nach der semiempirischen AM1−Methode vorgenommen werden soll.
NoSymm	Dieses Schlüsselwort veranlaßt Gaussian, die Molekülsymmetrie zu ignorieren und die vorgegebene räumliche Orientierung beizubehalten.
test	Damit wird die Ausgabe nicht relevanter Archivierungseinträge unterdrückt.
cube=(35,orbitals)	Bei Angabe dieses Schlüsselwortes erzeugt Gaussian eine oder mehrere Cube−Dateien für die später in der Eingabedatei aufgeführten Orbitale. Zusätzlich beinhaltet diese Spezifikation die Anweisung, eine mittlere Gitterdimension (Auflösung) von 35 zu benutzen. Gaussian legt die tatsächlichen Dimensionen für jede der drei Raumachsen nach der Geometrie des Moleküls fest.
prop=(potential,read)	Diese Angabe veranlaßt Gaussian, an den in der Eingabedatei aufgeführten Punkten die Werte des elektrostatischen Potentials zu berechnen.

Die von Molcad berechneten Elektronendichteoberflächen werden nicht quantenchemisch, sondern näherungsweise über empirische Funktionen und Parameter [18] berechnet. Die elektrostatischen

Potentiale werden von Gaussian bei Angabe von „prop=potential“ auf Grundlage des PRISM–Algorithmus [17] erzeugt.

Zur Isoflächenerzeugung aus den Cube–Dateien für die Orbitale wird die Marching Cubes–Implementierung „isovis_pw“ eingesetzt, die neben den Vertexkoordinaten und Konnektivitäten bei Einsatz des Kommandozeilenschalters „–norm 1“ auch die Vertexnormalen einer triangulierten Oberfläche liefert. Das erste Problem beim Einsatz dieses Programms ist die Konvertierung des Formats der Gaussian Cube–Dateien in ein von „isovis_pw“ unterstütztes Eingabeformat. Aus den zur Verfügung stehenden wurde das HDF–Format ausgewählt, weil es für Volumendatensätze sehr gut geeignet ist und außerdem in vielen anderen Visualisierungsapplikationen unterstützt wird. Bei der HDF–Distribution wird ein Hilfsprogramm mit dem Namen „fp2hdf“ mitgeliefert, welches das HDF–Format aus einem einfachen ASCII–Format erzeugen kann. Zur Schließung der Lücke wurde ein kleines C–Programm („cube2fp“) entworfen, das Cube–Dateien in das von „fp2hdf“ verwertbare ASCII–Format überführt. Auf den mit Hilfe von „cube2fp“ und „fp2hdf“ erzeugten HDF–Dateien für die HOMO– und die LUMO–Daten werden mit „isovis_pw“ je zwei Rechnungen, einmal mit positivem Isowert (für die Orbitallappen der Wellenfunktion mit positivem Vorzeichen) und einmal mit negativem Isowert (für die negative Phase), durchgeführt. Die zu den negativen und die zu den positiven Isowerten generierten Flächen werden bei der 3D–Darstellung später unterschiedlich eingefärbt.

In *newChem* wurde für alle Orbitale ein Isowert von ±0,032 angesetzt, weil dieser die Voreinstellung in dem Molecular Modeling–Programm Spartan ist und somit einfache Vergleiche zwischen beiden Programmen ermöglicht. Darüber hinaus liefert dieser Wert Bilder, die den Abbildungen aus Lehrbüchern der organischen Chemie ähneln, also einen hohen Wiedererkennungswert besitzen. Leider verwendet „isovis_pw“ die in den HDF–Dateien vorhandenen Gitterabmessungen nicht, sondern setzt einen Gitterabstand (Kantenlänge eines Würfels) von 1,0 in allen drei Raumrichtungen an und zentriert das Gitter um (0, 0, 0). Daher kommt abschließend noch ein selbst entworfenes Tcl–Skript zum Einsatz, das die Koordinaten nach der folgenden Formel wieder auf die ursprünglichen Dimensionen zurückrechnet.

$$X_{neu} = X_{alt} * h + X_0 \quad ; \text{für } X = \text{x, y, z}$$

Dabei entspricht der Skalierungsfaktor h einfach dem Gitterabstand in der Gaussian Cube–Datei. Der Offset X_0 setzt sich aus der halben Kantenlänge des gesamten Gitters und dem Wert der Koordinate des Gitterursprungs in der entsprechenden Dimension zusammen.

4.3.5 Erstellung der Energiediagramme

Sowohl die Diagramme mit dem Energieprofil des Reaktionsablaufs (Abbildung 3) als auch die Orbitalenergiediagramme (Abbildung 4), die zur Beurteilung einer eventuellen Orbitalkreuzung vorgesehen sind, werden mit dem selbst erstellten Programm „irctool“ generiert.

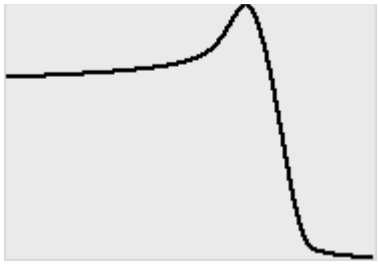

Abbildung 3: Energieprofildiagramm für die Diels–Alder–Reaktion zwischen Ethen und Butadien

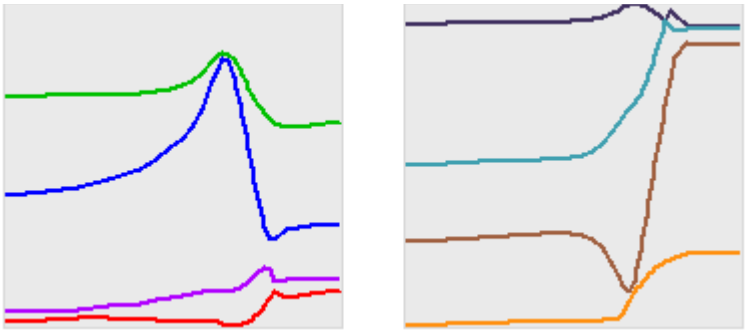

Abbildung 4: Orbitalenergiediagramme für die Diels–Alder–Reaktion zwischen Ethen und Butadien links: HOMO–3 – HOMO, rechts: LUMO – LUMO+3

In die Energieprofildiagramme gehen die Bildungswärmen aller Strukturen aus den IRC–Pfaden ein. Diese werden im Verlauf einer IRC–Rechnung von jedem der eingesetzten Rechenprogramme ausgegeben. Die Energiewerte werden schon beim Einlesen der IRC–Dateien in „irctool“ (Abschnitt 4.3.1) extrahiert und den Strukturen zugeordnet. Anschließend kann man sich die Energiewerte als Liste in der richtigen Reihenfolge zur Verwendung in einem anderen Programm ausgeben lassen oder das Energiediagramm direkt abrufen. Dabei wird die Kurve über eine kubische Spline–Interpolation erzeugt, um sie auch bei wenigen Stützpunkten glatt erscheinen zu lassen.

Die Orbitalenergiediagramme werden nur mit Hilfe der ausgewählten Strukturen konstruiert, da die Eigenwerte erst bei den Gaussian Single Point–Rechnungen bestimmt und ausgegeben werden, aber noch nicht bei den IRC–Rechnungen. Zur Bewahrung der Übersichtlichkeit werden jeweils nur die Kurven für die vier höchsten besetzten (HOMO, HOMO–1, HOMO–2, HOMO–3) und die

vier tiefsten unbesetzten (LUMO, LUMO+1, LUMO+2, LUMO+3) Orbitale erzeugt und in zwei getrennten Diagrammen im gleichen Maßstab dargestellt. Durch die Beschränkung auf insgesamt acht Kurven kann es natürlich vorkommen, daß sich einige Unregelmäßigkeiten in der 3D–Orbitaldarstellung anhand der Diagramme nicht erklären lassen, wenn energetisch tiefer oder höher liegende Orbitale an Kreuzungen beteiligt sind. Doch ist zumindest für die meistbeachteten Grenzorbitale HOMO und LUMO die Wahrscheinlichkeit gering, daß weitere, sich energetisch stärker von diesen unterscheidende Orbitale zur Erklärung von Kreuzungen herangezogen werden müssen. Mit „irctool“ werden Diagramme erzeugt, in denen die acht Kuven durch unterschiedliche Farben leicht auseinandergehalten werden können. Es hat sich hier gezeigt, daß es günstig ist, die Stützpunkte durch einfache Linienzüge zu verbinden und auf eine Spline–Interpolation zu verzichten. Die Gründe dafür liegen zum einen in der Verwendung von nur wenigen, normalerweise lediglich 30, Stützpunkten und andererseits an der Tatsache, daß der Energieverlauf einzelner Orbitalniveaus gerade im Bereich von Kreuzungspunkten unregelmäßig und nicht stetig differenzierbar ist. Daher kann es bei der Interpolation mit kubischen Splines zu starken Schwingungen im Kurvenverlauf und damit einer verfälschten Wiedergabe der berechneten Energiesituation kommen.

4.4 Das Reaktionsvisualisierungssystem *CAVOC*

Das Endprodukt des *newChem*–Projektes ist ein auf CD geliefertes Programm mit einer Reihe von Einschränkungen. So ist es nur auf einer einzigen Rechnerplattform, nämlich Microsoft Windows, lauffähig. Die mitgelieferten Reaktionsanimationen sind fest eingebunden, ein Hinzufügen neuer Datensätze ist ohne eine Überarbeitung des Programms nicht möglich. Zudem ist die Liste der angebotenen Oberflächen mit HOMO, LUMO und Elpot (Elektronendichte mit farblich kodiertem elektrostatischem Potential) sehr kurz ausgefallen. Der Anwender hat bezüglich der Oberflächen keine Möglichkeit, mit anderen Farbkodierungen und Isowerten zu experimentieren, bei der Polygondarstellung zwischen geschlossen, Gitter und Punktwolke zu wählen oder die Transparenz zu verändern. Außerdem können mehrere Oberflächen zu einer Reaktion nicht gleichzeitig im selben Fenster angezeigt werden. Zusammenfassend läßt sich daher feststellen, daß in *newChem* wichtige Funktionen zur Beeinflussung der grafischen Darstellung fehlen, die dem Benutzer die Möglichkeit geben, die angebotenen Reaktionen unter eigenen Gesichtspunkten zu untersuchen.
Aufbauend auf den Erfahrungen aus dem *newChem*–Projekt wurde mit dem primären Ziel, die Reaktionsanimationen im Internet zu präsentieren, das Programmsystem *CAVOC* (Computer Aided

Visualization of Chemical Reactions) entwickelt, das nicht nur alle im 3D−Grafikmodul von *newChem* vorhandenen Features bereitstellt, sondern darüber hinaus auch die dort vermißten, oben genannten Funktionen enthält. Es stellt jetzt wesentlich mehr Datensätze zur Isoflächen−Visualierung zur Verfügung als *newChem*. Angeboten werden zur Zeit die acht Orbitale von HOMO−3 bis LUMO+3, die Elektronendichte, das elektrostatische Potential und der Laplacian der Elektronendichte, der aus der Theorie von Atomen in Molekülen [19] bekannt ist. Bei Radikalreaktionen wird zusätzlich noch die Spindichte (Differenz der Elektronendichten, die nur von den α− bzw. nur von den β−Elektronen gebildet werden) angeboten, und es stehen dort insgesamt sechzehn Orbitale (HOMO−3 bis LUMO+3, jeweils α und β) zur Verfügung. Sämtliche dieser Eigenschaften lassen sich in *CAVOC* auch auf eine Isofläche einer anderen Größe abbilden.
Die erste Frage ist, mit welchen Mitteln ein solches System realisiert werden kann. Hier bieten sich vor allem zwei alternative Wege an:

1. Die 3D−Darstellung wird in VRML (Virtual Reality Modeling Language) realisiert, wobei die Szenenkontrolle von einem Java−Applet über eine standardisierte Schnittstelle, das EAI (External Authoring Interface), vorgenommen wird.
2. Benutzersteuerung und 3D−Darstellung werden von einem einzigen Java−Applet übernommen.

Die Entscheidung fiel zugunsten der zweiten Lösung aus, da sie wesentlich flexibler ist und zudem mit dem Java 3D−API ein leistungsfähiges Werkzeug zur Entwicklung von Programmen mit 3D−Grafik direkt in Java zur Verfügung steht. Eine Realisierung der ersten Variante ist kürzlich von Frank Oellien [20] entwickelt worden.
Die Architektur des im Rahmen der vorliegenden Arbeit entwickelten, auf Java 3D basierenden Programmsystems *CAVOC* ist in Abbildung 5 dargestellt.

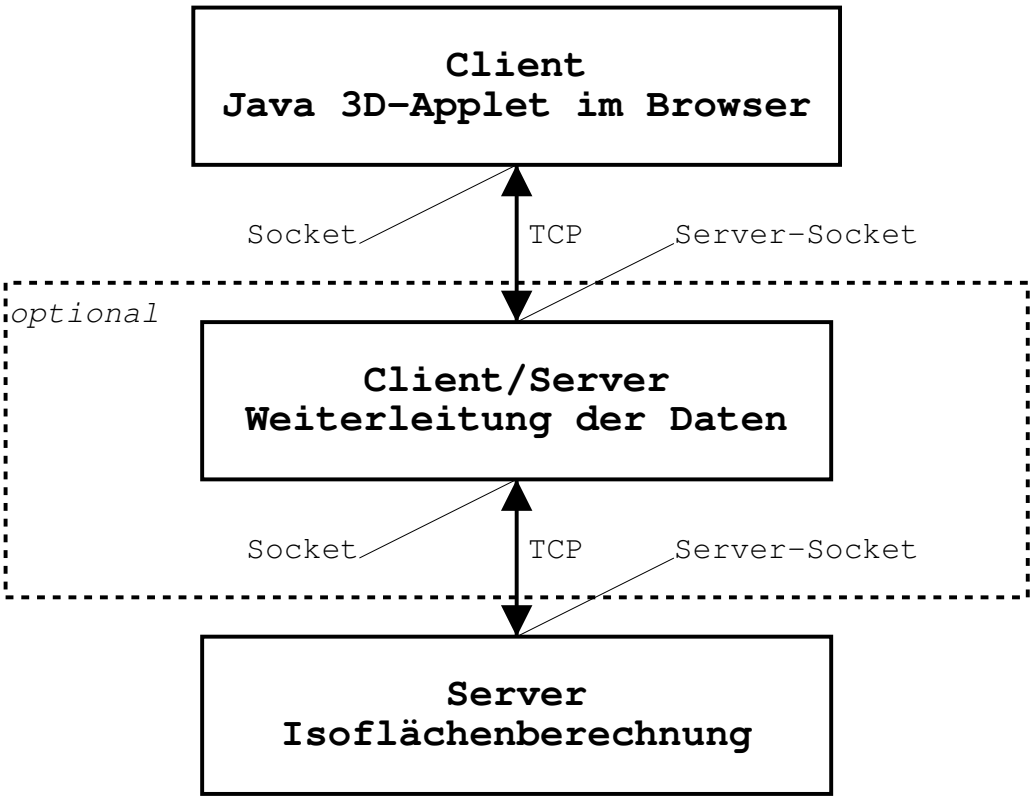

Abbildung 5: Architektur von CAVOC

Wie aus Abbildung 5 zu entnehmen ist, sind an einer Reaktionsvisualisierung mit Isoflächendarstellung drei (oder auch nur zwei) Programme beteiligt. Die primäre Komponente ist ein Java 3D–Applet das im Internet–Browser des Betrachters abläuft und für die Darstellung der Molekülmodelle und Oberflächen zuständig ist. Wenn vom Benutzer eine Isofläche angefordert wird, werden der Name der Reaktion, die Art der Isofläche sowie der Isowert an den die Berechnung ausführenden Server weitergeleitet, der wiederum nach erfolgreicher Berechnung die Oberfläche an das Applet zurücksendet. Falls der Server auf einem anderen Host läuft als der Web–Server, der das Applet bereit stellt, wird ein Hilfsserver zwischengeschaltet, über den der Datenverkehr umgeleitet wird.
Die einzelnen Komponenten dieses Systems werden in den folgenden Abschnitten beschrieben. Zunächst wird aber eine kurze Einführung in Java 3D gegeben.

4.4.1 Grundlagen des Java 3D–API

Java 3D ist eine Java Erweiterung, ein API zur Generierung von 3D–Grafikapplikationen. Da es noch nicht in den Java–Kern aufgenommen worden ist, muß es zusätzlich zu einer vorhandenen

Java−Umgebung installiert werden.

Von Java 3D werden drei verschiedene Ebenen zur Generierung dreidimensionaler Szenen unterstützt. Im „Immediate Mode“, dem flexibelsten Modus, können geometrische Objekte auf Kosten der Geschwindigkeit unter direkter Kontrolle des Programmierers grafisch dargestellt werden. Dagegen muß im „Retained Mode“ immer erst ein Scene Graph (s. u.) konstruiert werden, in dem die 3D−Szene genau definiert ist. Manipulationen an dieser Struktur sind später nur noch eingeschränkt möglich, so daß Java 3D daran intern eine Reihe von Optimierungen vornehmen kann, was eine höhere Darstellungsgeschwindigkeit ermöglicht. Der „Compiled−Retained Mode“ bietet schließlich die größte Performance. Im Unterschied zum „Retained Mode“ wird hier der Scene Graph auf Anweisung des Programmierers kompiliert, was bedeutet, daß Java 3D der Szenenaufbau aggressiv optimiert. Dabei kann dann auch die interne Darstellung in einer für den Programmierer unsichtbaren Weise verändert werden, so daß die Eingriffsmöglichkeiten in diesem Modus auf ein Minimum reduziert sind.

Der Szenenaufbau in Java 3D fußt auf einem objektorientierten Konzept, welches auch in den meisten anderen Beschreibungssprachen für 3D−Grafik, wie VRML oder Open Inventor, verwendet wird. In diesem Konzept werden die einzelnen Szenenelemente als Knoten in einem gerichteten azyklischen Graphen, dem sogenannten Scene Graph, dargestellt. Ein Scene Graph besitzt somit eine Baumstruktur, in der einzelne Elemente in hierarchischer Weise angeordnet sind. Bei den Knoten unterscheidet man zwischen Gruppenknoten und Blattknoten. Gruppenknoten können sowohl als Behälter für andere Elemente (Elternknoten) als auch selber als untergeordnete Einheiten (Kindknoten) fungieren. Dagegen können Blattknoten keine weiteren Kindknoten besitzen. Sie stellen vielmehr die sichtbaren Elemente und Aktionen einer Szene dar.

Das Scene Graph−Modell von Java 3D ist in der Abbildung 6 skizziert.

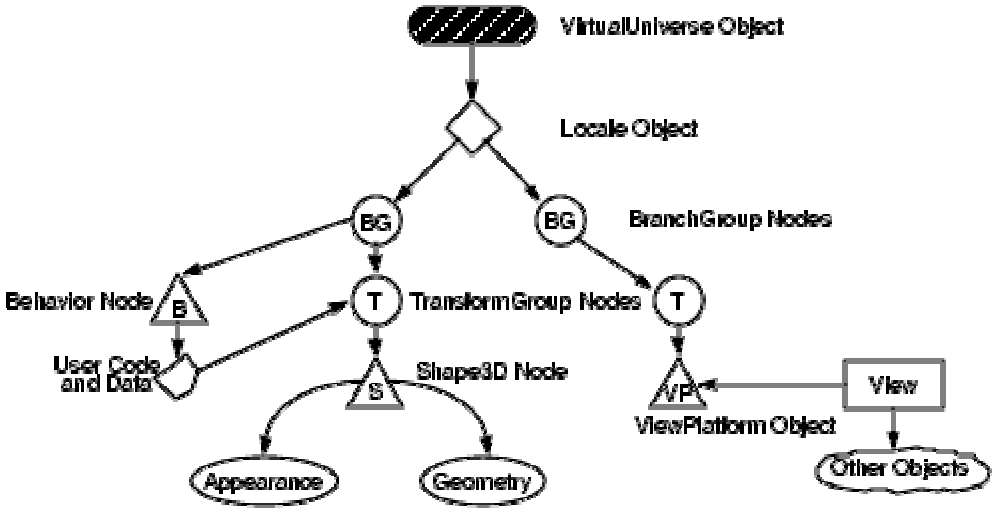

Abbildung 6: Java 3D Scene Graph Modell (aus der Java 3D API−Spezifikation)

An der Spitze eines Java 3D Scene Graph steht ein *VirtualUniverse*−Objekt, das eine Liste von *Locale*−Objekten beinhaltet. Ein *Locale*−Objekt repräsentiert einen Ort im Universum und dient gleichzeitig als Behälter für die einzelnen Subgraphen, an deren Spitzen die sogenannten *BranchGroup*−Knoten stehen. Ein *BranchGroup*−Knoten ist ein spezieller Gruppenknoten, der als einziges Java 3D−Objekt im „Retained Mode“ aus einem Scene Graph wieder entfernt werden kann, nachdem die Szene sichtbar gemacht worden ist. Ein anderer wichtiger Gruppenknoten ist der *TransformGroup*−Knoten, der die räumliche Lage, Orientierung und Skalierung seiner Kinder festlegt. Die sichtbaren Elemente einer Szene machen die *Shape3D*−Knoten aus, die sich aus den Komponenten *Appearance*, die Erscheinungsmerkmale wie Farbe und Lichtreflexion festlegt, und *Geometry*, die die geometrische Form angibt, zusammensetzen. Daneben gibt es noch die *Behavior*−Knoten, mit deren Hilfe erst „Leben“ in eine Szene kommt. Sie können beliebige programmierte Aktionen auslösen, indem sie auf benutzergesteurte oder vorher festgelegte Ereignisse reagieren.

Jeder Java 3D−Scene Graph benötigt darüber hinaus einen Teil, der das Betrachten der 3D−Inhalte ermöglicht, also sozusagen einen Blick auf die konstruierte Welt gewährt. Java 3D arbeitet hier mit einer sehr flexiblen, aber auch etwas komplexen Struktur, die sich aus mehreren Objekten zusammensetzt und in der rechten Hälfte von Abbildung 6 angedeutet ist.

4.4.2 Beschreibung des Java−Applets

Zunächst soll einmal der Scene Graph vorgestellt werden, der im Java 3D−Applet von *CAVOC* für eine Reaktionsanimation aufgebaut wird (Abbildung 7).

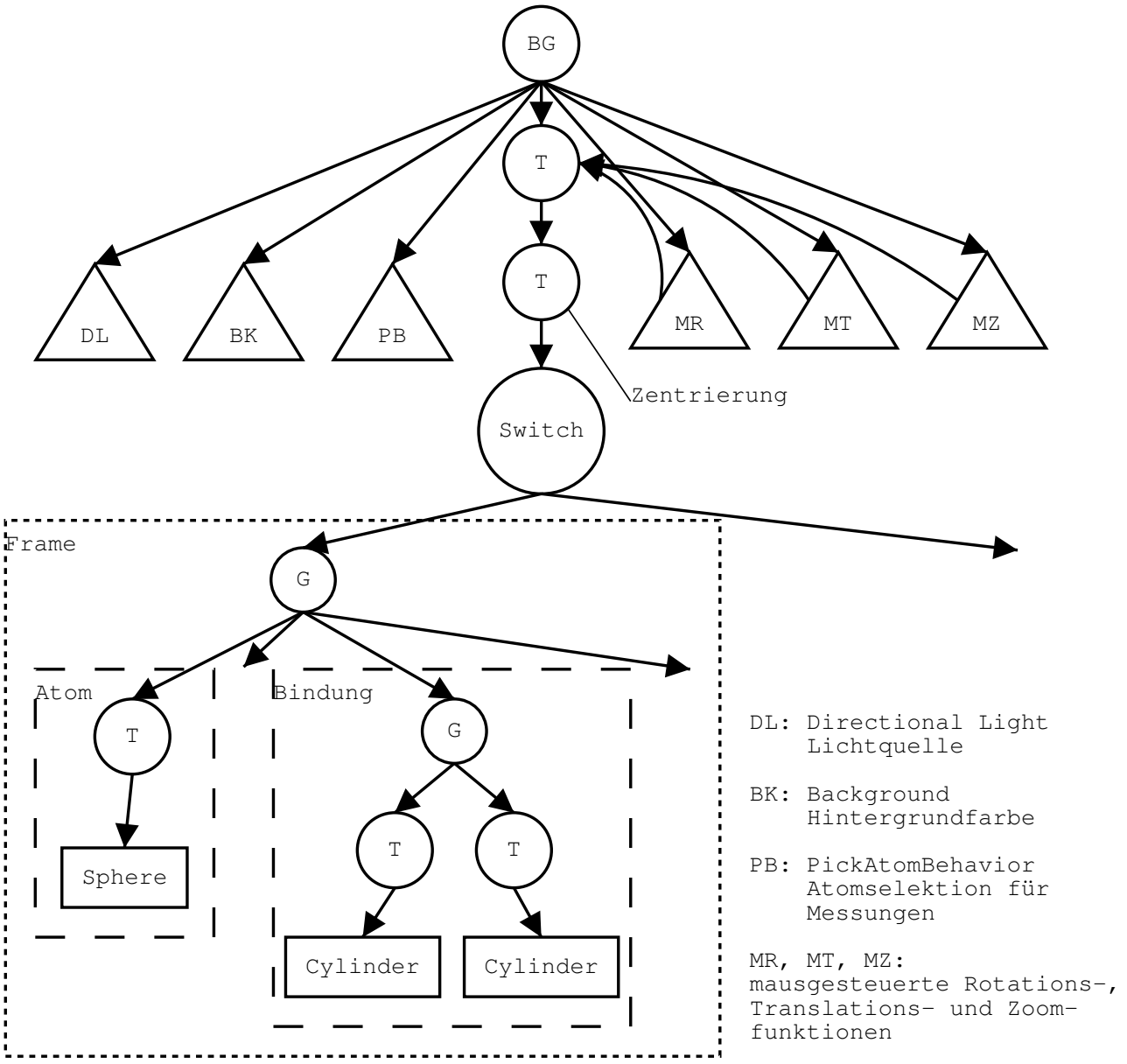

Abbildung 7: Der Scene Graph−Aufbau in CAVOC

Dargestellt ist nur der Zweig mit dem sichtbaren Szeneninhalt, der *View*−Zweig (in der rechten Hälfte von Abbildung 6 repräsentiert) fehlt aus Gründen der Übersichtlichkeit. Ebenfalls nicht abgebildet sind die Elemente, mit denen die Isoflächen und Atombeschriftungen realisiert werden und die erst auf Anforderung des Benutzers generiert und eingefügt werden. Sie werden später gesondert beschrieben.

Der *View*−Zweig samt *VirtualUniverse* und *Locale* wird in *CAVOC* mit einer Utility−Klasse aus Java 3D, der *SimpleUniverse*−Klasse, konstruiert. Kopf des Zweiges mit der eigentlichen 3D−Szene und Anknüpfungspunkt an das *Locale*−Objekt ist ein *BranchGroup*−Objekt. Dieses hat mehrere Kindknoten, die alle in Abbildung 7 gezeigt sind.

Für einen dreidimensionalen Eindruck wird unbedingt eine Lichtquelle benötigt. Zum Einsatz kommt hier ein Distanzlicht (Directional Light: DL), das aus der Richtung des Betrachters als paralleles Bündel auf die Szene fällt. Dieses Arrangement bietet einen durchaus realistischen und auch von anderen Programmen gewohnten Eindruck bei der Moleküldarstellung. Ein Einfügen weiterer Lichtquellen erübrigt sich von daher und hätte zudem einen negativen Einfluß auf die Darstellungsgeschwindigkeit.

Der *Background*−Knoten (BK) repräsentiert einen Hintergrund in der 3D−Szene. Er dient hier lediglich dazu, dessen Farbe einstellen zu können. Bei Fehlen dieses Knotens wäre der Hintergrund immer schwarz.

Der *PickAtomBehavior*−Knoten (PB), eine Subklasse von *javax.media.j3d.Behavior*, wurde zur Realisierung der Messungsfunktionalität entwickelt. Er registriert, wenn er aktiviert ist, das Anklicken von Atomen in der Szene und leitet dann die notwendigen Aktionen ein.

Die drei Knoten *MouseRotate* (MR), *MouseTranslate* (MT) und *MouseZoom* (MZ) aus den Java 3D Utility−Klassen ermöglichen das Rotieren, Translatieren und Zoomen der Molekülansicht mit der Maus. Sie wirken über den *TransformGroup*−Knoten ganz oben in der Hierarchie, indem sie dessen *Transform3D*−Objekt, welches letztlich die geometrische Orientierung bestimmt, verändern.

Der zweite *TransformGroup*−Knoten sorgt dafür, daß sich das Rotationszentrum für die Mausinteraktionen in dem „intuitiv richtigen“ Mittelpunkt befindet. Dazu werden beim Einladen einer Reaktion die Koordinaten aller Atome in allen Frames gemittelt. Aus den negierten Mittelwerten ergibt sich dann die notwendige Verschiebung, die durch diesen Knoten geleistet wird.

Der *Switch*−Knoten ist ein Gruppenknoten, mit dem gesteuert werden kann, welche seiner Kinder gezeichnet werden sollen und welche nicht. In Kombination mit einem *javax.swing.Timer*−Objekt, das in bestimmten Zeitintervallen eine Aktion auslöst, mit der die Frames über den *Switch*−Knoten nacheinander einzeln eingeblendet werden, entsteht die Reaktionsanimation.

Die einzelnen Frames werden jeweils unter eigenen Gruppenknoten (G) zusammengefaßt. Für die Ball&Stick−Darstellung werden die Atome durch Kugeln und die Bindungen durch Zylinder repräsentiert. Java 3D bietet auch hier Utility−Klassen an, mit denen diese Geometrien auf einfache Art erzeugt werden können. Diese Klassen heißen nicht überraschenderweise *Sphere* und *Cylinder*.

Sie befreien den Programmierer von der Arbeit, selbst die zur Darstellung der geometrischen Primitive notwendigen *Shape3D*–, *TransformGroup*– und *Appearance*–Knoten zusammenzustellen.

Für jedes Atom muß ein *Sphere*–Objekt erzeugt und mit Hilfe eines *TransformGroup*–Objekts, das einfach eine Translation zu den entsprechenden Koordinaten beschreibt, an der richtigen Stelle im Raum plaziert werden. Die Bindungen erfordern hingegen deutlich mehr Aufwand. Um den Anteilen der aneinander gebundenen Atome Rechnung zu tragen wird in *CAVOC*, wie in den meisten anderen Programmen zur Molekülbetrachtung, eine Bindung zweifarbig dargestellt, wobei jede Bindungshälfte die Farbe des mit ihr verbundenen Atoms zugewiesen bekommt. Dies erfordert im allgemeinen die Konstruktion einer Bindung aus zwei unterschiedlich eingefärbten Zylindern. Bei Bindungen zwischen gleichartigen Atomen ist es hingegen möglich und auch effizienter, nur einen Zylinder zu verwenden. Dementsprechend werden im Scene Graph entweder ein oder zwei *Cylinder*–Objekte über *TransformGroup*–Knoten unter einem Gruppenknoten zusammengefaßt. Aus einem solchen Subgraphen läßt sich eine abgeschlossene Einheit ableiten, die als Klasse *Bond* implementiert wurde. Zur Bestimmung der *Transform3D*–Objekte für die *TransformGroup*–Knoten müssen die anfangs auf der z–Achse liegenden Zylinder so rotiert werden, daß sie auf der Bindungsachse zu liegen kommen. Außerdem müssen sie auf die richtige Länge skaliert und an die passende Stelle auf der Bindungsachse verschoben werden.

In dem Scene Graph der Abbildung 7 sind noch nicht die Elemente für die Erzeugung von Atombeschriftungen und Isoflächen enthalten. Da diese als erweiterte Funktionalitäten erst auf Benutzeranforderung in den Scene Graph eingefügt werden, sind sie in den folgenden Abschnitten beschrieben.

4.4.2.1 Der Hauptbildschirm des Applets

In Abbildung 8 ist ein Snapshot des Hauptbildschirms des Java 3D basierten Applets zu sehen. In dessen Zentrum befindet sich das Fenster zur 3D–Darstellung der angewählten Reaktion, welches in Java 3D durch ein *Canvas3D*–Objekt realisiert ist. An allen anderen Stellen der grafischen Benutzeroberfläche kommen Swing–Objekte zum Einsatz, die die zur Zeit fortschrittlichsten GUI–Komponenten in Java darstellen und als Teil der JFC (Java Foundation Classes) seit der Version 1.2 zum Java–Kern gehören.

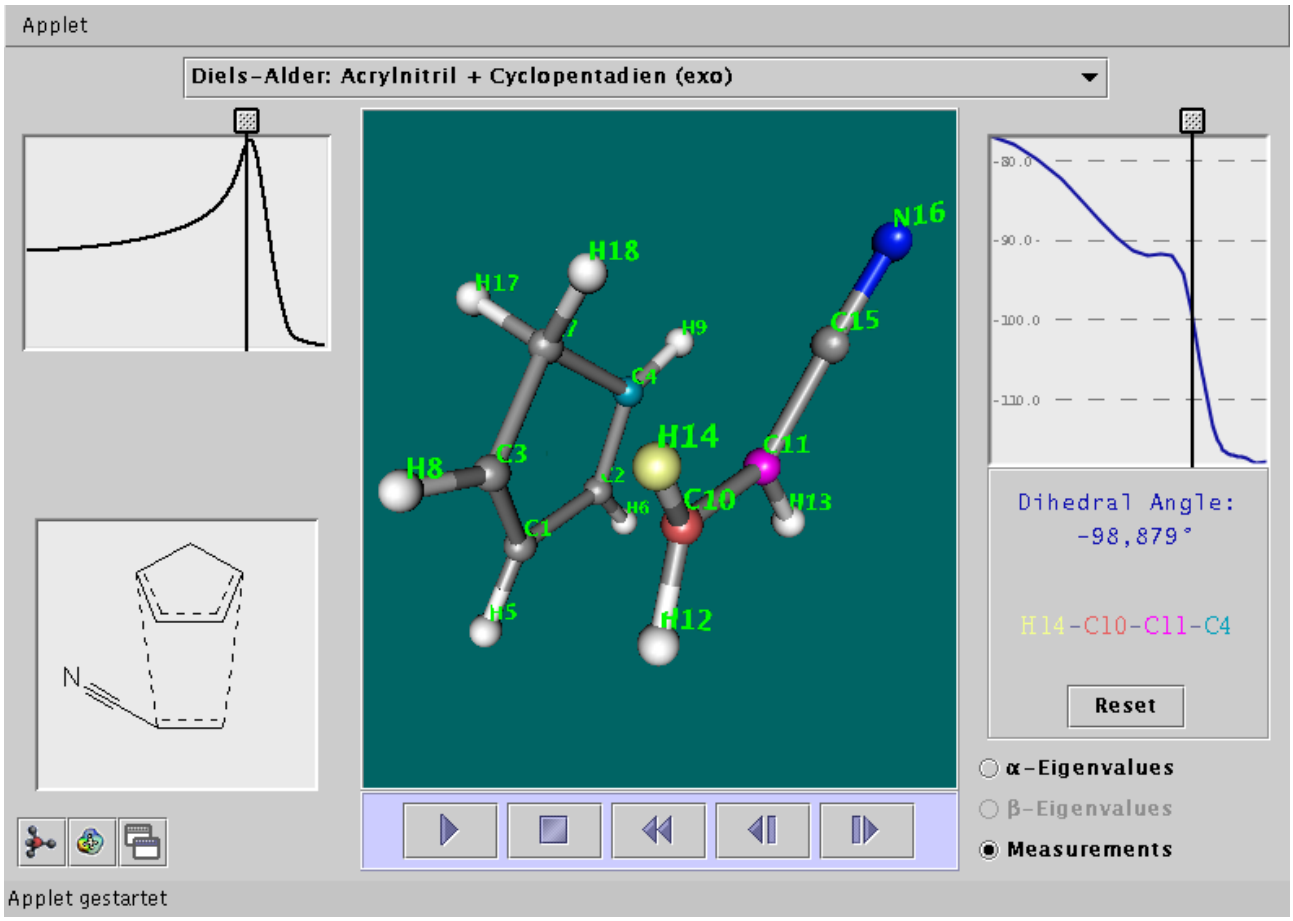

Abbildung 8: Der Hauptbildschirm des Java–Applets

Über dem 3D–Fenster ist eine Combobox angebracht, über die der Benutzer eine andere Reaktion anwählen kann. Beim Wechsel der Reaktion wird ein modaler Dialog angezeigt, um Benutzerinteraktionen mit dem Applet zu unterbinden. Währenddessen wird in einem eigenen Thread eine Verbindung zum Web–Server aufgebaut, um von diesem eine fertig vorbereitete zip–Datei zu laden, in der alle zu der angeforderten Reaktion gehörigen Atomkoordinaten und Diagramme enthalten sind. Nach dem Download wird der Zweig mit dem Inhalt der alten Szene aus dem Scene Graph durch einen neu aufgebauten Zweig ersetzt. Das dabei zum Einladen der 3D–Strukturen eingesetzte Dateiformat ist im Anhang A beschrieben. Schließlich werden alle Strichformel– und Energiediagramme ausgetauscht. Welche Reaktion beim Starten von *CAVOC* geladen wird, legt ein Applet–Parameter in der aufrufenden Internetseite fest. So ist es schon vor dem Programmstart möglich, den Benutzer über das Common Gateway Interface (CGI) die erste zu ladende Reaktion auswählen zu lassen. Eine Liste mit den zur Zeit in *CAVOC* abrufbaren Reaktionen findet sich im Anhang D.

Unter dem 3D–Fenster befindet sich eine Leiste mit den für eine Animation üblichen

Steuerelementen. Hier finden sich Buttons zum Abspielen, Anhalten, Zurücksetzen sowie zur Einzelschrittsteuerung der Reaktionsanimationen.
Rechts unten befindet sich eine Gruppe von Radiobuttons. Damit kann der Benutzer auswählen, ob rechts vom 3D–Fenster die Orbitalenergiediagramme dargestellt werden sollen oder ob er Messungen an den 3D–Geometrien durchführen möchte. In letzterem Fall wird der *PickAtomBehavior*–Knoten im Scene Graph aktiviert, und die Orbitalenergiediagramme werden durch zwei Fenster verdeckt, in denen die Meßergebnisse grafisch und numerisch angezeigt werden.
Auf der linken Seite befindet sich in der oberen Hälfte das zu der jeweils eingeladenen Reaktion gehörige Energiediagramm. Über diesem Diagramm, wie auch über den Orbitalenergie– bzw. Meßwertediagrammen, befindet sich ein Schieberegler, mit dessen Hilfe durch Ziehen mit der Maus sehr schnell die einzelnen Frames angesteuert werden können. Das ist z. B. dann nützlich, wenn direkt zu einem Übergangszustand oder einem Orbitalkreuzungspunkt gesprungen werden soll.
Unter dem Energiediagramm befindet sich ein Fenster, in dem parallel zur 3D– eine „2D“–Animation gezeigt wird. Hier werden einzelne Bilder einer Sequenz von Formelzeichnungen eingeblendet. Diese Formelzeichnungen müssen für jede in *CAVOC* aufgenommene Reaktion von Hand in einem geeigneten Programm (z. B. ISIS Draw [21]) erstellt und den einzelnen Frames der 3D–Darstellung zugeordnet werden. In der Regel setzt man hier etwa drei bis fünf Bilder ein, die die Edukte, den Bereich des Übergangszustandes und die Produkte repräsentieren. Die Einblendung der einzelnen Bilder ist technisch durch ein *CardLayout* realisiert, einen Layout Manager, der es erlaubt, zwischen verschiedenen Komponenten unter seiner Kontrolle hin und her zu schalten.
In der linken unteren Ecke des Hauptbildschirms ist eine Werkzeugleiste angebracht, die drei Buttons enthält. Der linke Button öffnet einen Dialog, in dem verschiedene Darstellungsoptionen eingestellt werden können, der mittlere präsentiert einen Dialog zum Umgang mit den Isoflächen. Durch Aktivierung des rechten Buttons wird ein neues 3D–Fenster erzeugt, das eine exakte Kopie des gerade im 3D–Fenster des Hauptbildschirms dargestellten Frames zeigt. In dem neuen Fenster kann wie gewohnt mit der Maus agiert werden, d. h. die Ansicht kann rotiert, translatiert und gezoomt werden. Diese Funktion ist besonders dann nützlich, wenn man Vergleiche zwischen verschieden Reaktionen oder Oberflächen anstellen möchte, da es damit möglich ist, mehrere verschiedene 3D–Szenarien auf dem Bildschirm nebeneinander zu betrachten.

4.4.2.2 Der Dialog zur Einstellung der Grafikoptionen

Die Möglichkeiten zur Beeinflussung der Darstellung werden im folgenden besprochen. Der zugehörige Dialog ist in Abbildung 9 gezeigt.

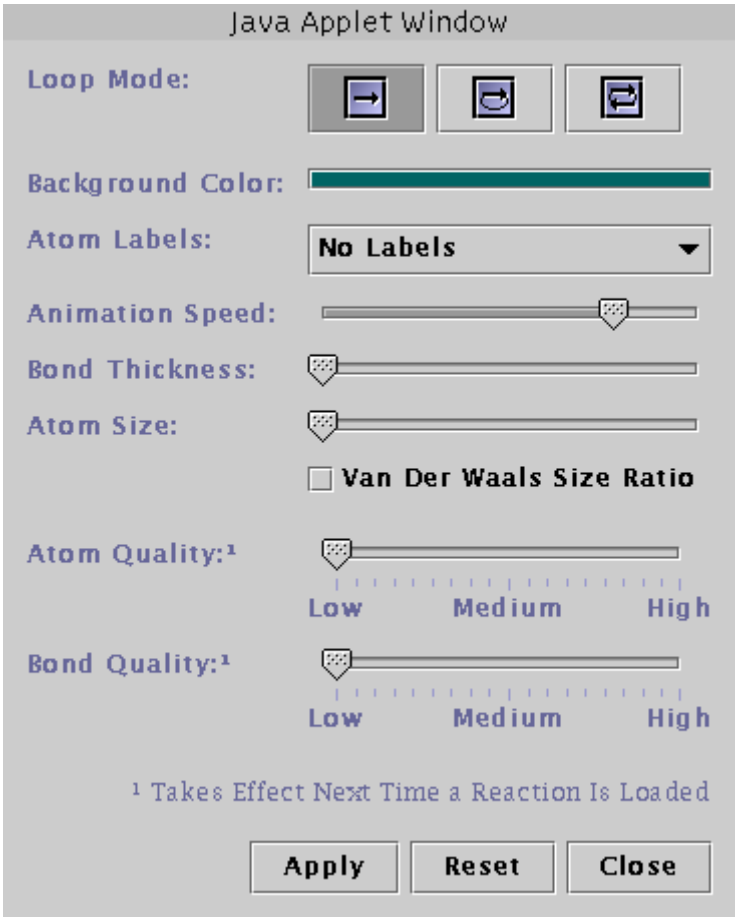

Abbildung 9: Dialogfenster zur Einstellung der nicht oberflächenbezogenen Darstellungsoptionen in CAVOC

Ganz oben befinden sich drei Buttons, mit denen der Ablauf der Animation gesteuert werden kann. Ist der linke Button aktiv, läuft die Animation, die durch Betätigen des Play–Buttons gestartet wird, bis zum letzten Frame durch und hält dann an. Bei der zweiten Option, die mit dem mittleren Button gewählt werden kann, wird die Animation in einer Endlosschleife nach Erreichen des letzten Frames beim ersten Frame wieder neu gestartet. Die dritte Möglichkeit (rechter Button) ist, die Reaktionsanimation bei Erreichen des letzten oder ersten Frames in einer Endlosschleife in der jeweils entgegengesetzten Richtung weiterlaufen zu lassen.

Der mit dem Label „Background Color“ versehene Button, dient zur Einstellung der Hintergrundfarbe im 3D–Fenster. Bei Betätigung dieses Buttons, wird ein Swing–Farbauswahldialog geöffnet, mit dem sich der Benutzer komfortabel eine ihm angenehme

Hintergrundfarbe aussuchen kann. Die hier eingestellte Farbe wird dem *Background*–Knoten der Java 3D–Szene und außerdem dem Button selbst zugewiesen.
Mit Hilfe der Combobox unter dem Titel „Atom Labels“ kann der Benutzer die Atome im 3D–Fenster mit Beschriftungen versehen. Diese können wahlweise aus den Nummern der Atome, deren Elementsymbolen oder beiden zusammen bestehen. Bestehende Label können natürlich auch wieder aus der Szene entfernt werden. In Abbildung 10 ist dargestellt, wie die Beschriftungen technisch innerhalb des Scene Graphs realisiert werden.

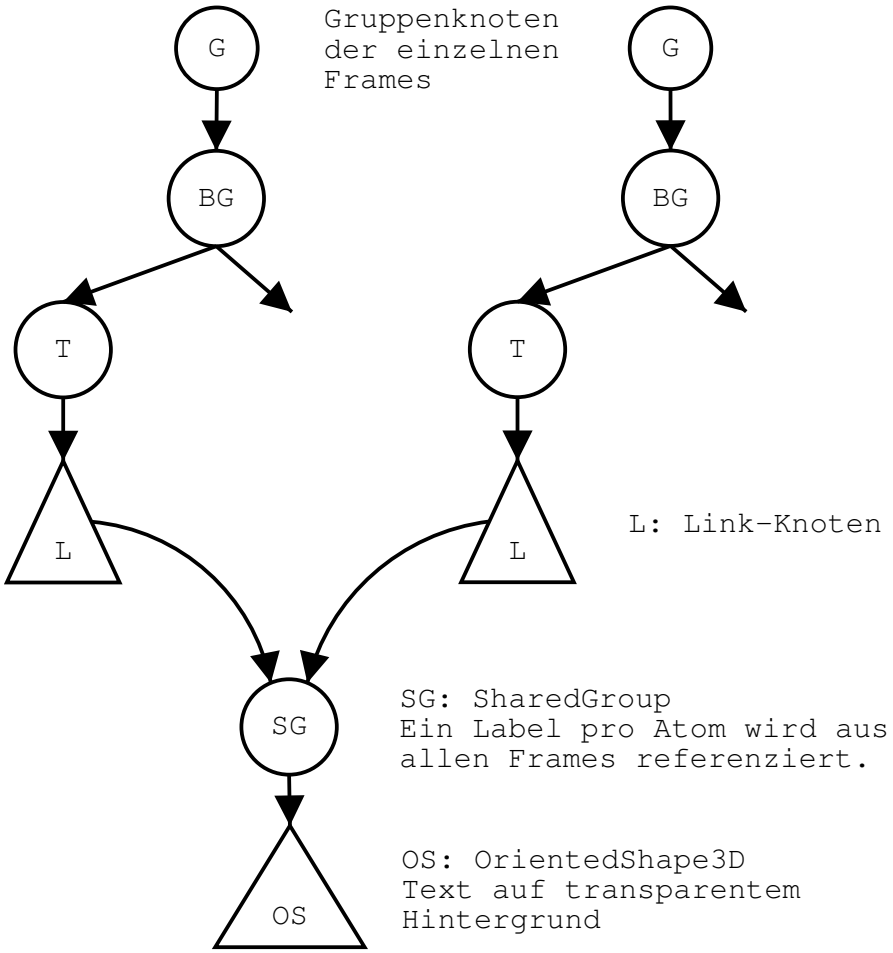

Abbildung 10: Realisierung von Atom–Beschriftungen im Scene Graph

Für jeden Frame wird ein eigenes *BranchGroup*–Objekt erzeugt, das die Label für den jeweiligen Frame in einem Zweig im Scene Graph zusammenfaßt. Jedes einzelne Label wird mit Hilfe eines *TransformGroup*–Knotens mit einem leichten Versatz zur Position des zugehörigen Atoms im Raum plaziert. Die Label selber werden aus *OrientedShape3D*–Knoten gebildet. Das sind Knoten, die ihre geometrischen Inhalte immer zum Blick des Betrachters hin ausrichten. In ihrer Funktion sind sie den *Billboard*–Knoten, die z. B. auch aus VRML bekannt sind, sehr ähnlich, übertreffen

diese in Java 3D aber deutlich an Geschwindigkeit. Sie bewirken, daß die Beschriftungen auch dann noch von vorne und links nach rechts lesbar bleiben, wenn der Benutzer die ganze Ansicht mit der Maus rotiert. Die Geometrien und das Aussehen der Label werden mit *Text2D*–Objekten erzeugt und den *OrientedShape3D*–Knoten zugewiesen. *Text2D* ist eine Utility–Klasse aus Java 3D, mit der sich zweidimensonal aussehender Text in eine 3D–Szene einfügen läßt. Gebildet wird ein *Text2D*–Objekt aus einem transparenten Rechteck, auf das der Text als Textur aufgetragen ist. Wie aus Abbildung 10 zu entnehmen ist, werden die *OrientedShape3D*–Knoten pro Atom nur einmal erzeugt und dann in allen Frames verwendet. Erreicht wird dies durch die Verwendung von *SharedGroup*–Knoten, die über *Link*–Knoten an beliebig vielen Stellen im Scene Graph referenziert werden können. Dieses Vorgehen bietet hier zwei Vorteile. Zum einen spart es Arbeitsspeicher, und zum anderen beschleunigt es die Erzeugung der Label um so mehr, je größer die Anzahl der Frames in der Animation ist. Andererseits muß dafür unbedingt die Forderung von Gleichheit der Anzahl, Art und Reihenfolge der Atome in allen Frames erfüllt sein, was bei Animationen chemischer Reaktionen aber kein Problem darstellt, da dabei ja grundsätzlich weder Atome hinzukommen oder verschwinden noch ineinander umgewandelt werden können.

Der Schieberegler, der mit dem Label „Animation Speed“ versehen ist, dient zur Einstellung der Abspielgeschwindigkeit. Damit kann die Pause zwischen zwei Frames über das die Animation steuernde j*avax.swing.Timer*–Objekt im Bereich von 2 bis 0,1 Sekunden eingestellt werden.

Die mit „Atom Size“ und „Bond Thickness“ beschrifteten Schieberegler dienen zur Einstellung der Größe der die Atome darstellenden Kugeln sowie der die Bindungen darstellenden Zylinder. Über den Checkbutton „Van Der Waals Size Ratio“ kann zudem festgelegt werden, ob alle Kugeln die gleiche Größe haben sollen, oder sich deren Größenverhältnis aus dem Verhältnis der van der Waals–Radien der repräsentierten Atomtypen ergeben soll. In dem Fall werden die Höhen der eine Bindung zwischen zwei ungleichen Atomen repräsentierenden Zylinder ebenfalls im Verhältnis der van der Waals–Radien skaliert. Realisiert werden die Größenänderungen im Scene Graph durch passende Änderungen der *Transform3D*–Objekte der *TransformGroup*–Knoten, die die *Sphere*– bzw. *Cylinder*–Objekte als Kindknoten haben.

Die unteren beiden Schieberegler mit den Labeln „Atom Quality“ und „Bond Quality“ dienen zur Einstellung der Auflösung der Kugeln und Zylinder, also der Qualität der grafischen Darstellung. Mit ihnen kann die Anzahl der Unterteilungen entlang der Achsen von Zylinder und Kugel im Bereich von 6 bis 24 geregelt werden, weil runde Formen in der Computergrafik eben nicht exakt darstellbar sind. Obwohl bei genauerem Hinsehen in der niedrigsten Auflösungsstufe z. B. leicht der sechseckige Querschnitt der Bindungen zu erkennen ist, ist dies zur Zeit die Voreinstellung, da

höhere Auflösungen zwangsläufig eine geringere Darstellungsgeschwindigkeit und größeren Speicherbedarf zur Folge haben. Eine Änderung dieser Einstellungen wird aber erst beim nächsten Laden einer anderen Reaktion berücksichtigt, da hierzu die Kugeln bzw. Zylinder aus dem Scene Graph entfernt, neue erzeugt und diese an den entsprechenden Stellen wieder eingefügt werden müßten. Im „Retained Mode“ von Java 3D wäre das dadurch zu realisieren, daß die Atome und Bindungen über *BranchGroup*−Knoten in den Scene Graph eingebunden werden. Ein solches Vorgehen wäre aber mit Einbußen in der Performance und einem höheren Speicherbedarf verbunden. Zudem gibt es in den Java 3D−Versionen vor 1.2.1 erhebliche Probleme mit der Speicherbereinigung, so daß dieser Ansatz nicht unbedingt ratsam erscheint.
Mit Ausnahme des Abspielmodus der Animation werden alle in diesem Dialogfenster gemachten Einstellungen erst dann aktiv, wenn der „Apply“−Button betätigt wird. Alle davor gemachten Änderungen können mit dem „Reset“−Button durch Zurücksetzen auf die zuvor eingestellten Werte wieder aufgehoben werden. Wird der „Close“−Button betätigt, werden noch nicht übernommene Änderungen ebenfalls verworfen.

4.4.2.3 Der Dialog zum Umgang mit den Molekülоberflächen

Abbildung 11 zeigt den Dialog, der zum Umgang mit den Molekülоberflächen dient.

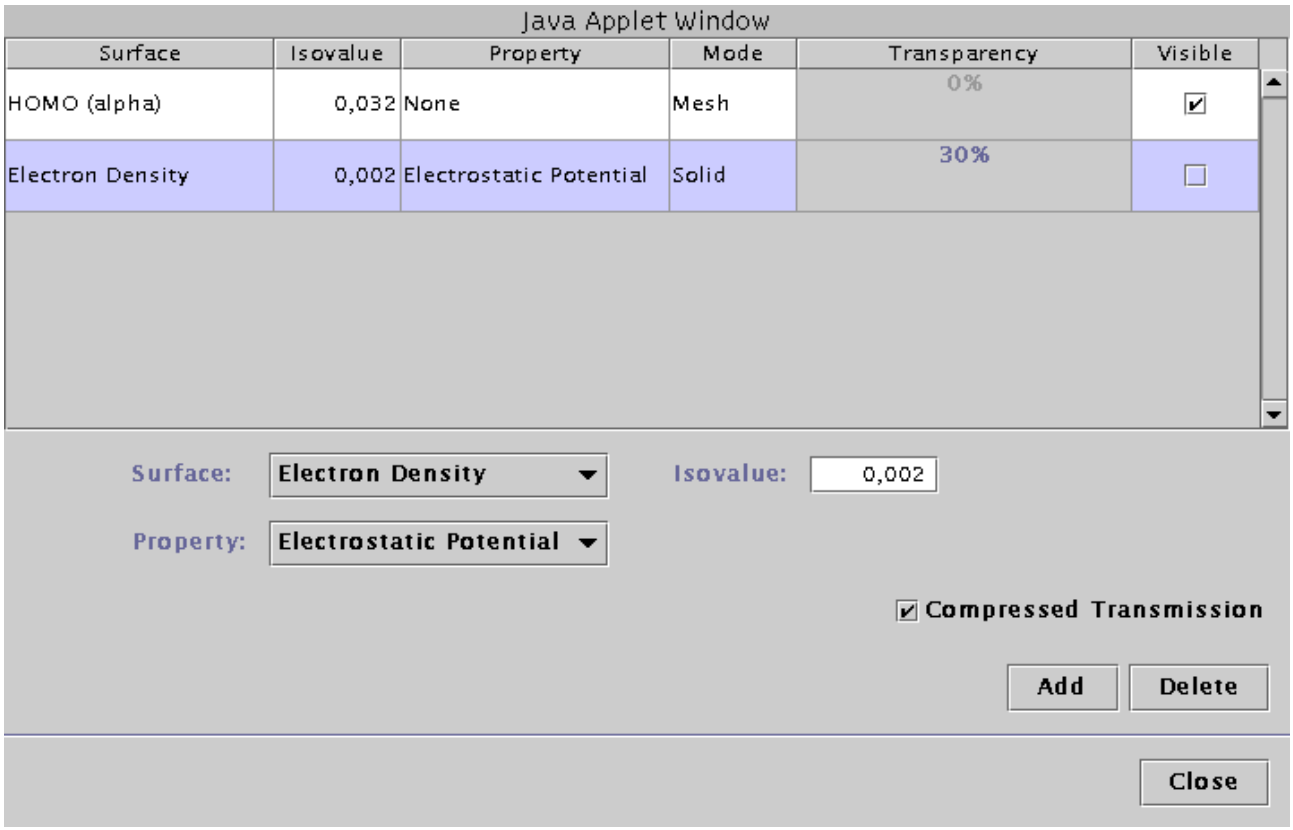

Abbildung 11: Dialogfenster zum Umgang mit den Molekülоberflächen

Im oberen Teil des Fensters befindet sich eine Tabelle mit sechs Spalten, in der jeder eingeladenen Oberfläche eine eigene Zeile zugewiesen wird. In der Spalte „Surface“ ist die Art der Isofläche eingetragen, in der Spalte „Isovalue“ ihr Isowert und in der Spalte „Property“ die auf ihr abgebildete Eigenschaft. Diese drei Parameter können über die zwei Comboboxen und das Textfeld eingestellt werden. Sie charakterisieren die Oberfläche, die beim Betätigen des „Add“–Buttons vom Server geladen wird.

Die Felder in den letzten drei Spalten der Tabelle sind editierbar. Beim Anklicken einer „Mode“–Zelle öffnet sich eine Combobox, mit der die Darstellungsart der Isofläche geändert werden kann. Als Optionen stehen hier „Solid“, „Mesh“ und „Dots“ zur Verfügung. Damit wird bestimmt, ob eine aus Dreiecken konstruierte, geschlossene Oberfläche, eine nur die Umrisse der Dreiecke wiedergebende Liniendarstellung oder lediglich eine Repräsentation der einzelnen Vertices als Punkte gezeigt wird. Die Umschaltung zwischen diesen Modi ist in Java 3D sehr einfach über ein *PolygonAttributes*–Objekt möglich, das dem mit der Isofläche assoziierten *Appearance*–Objekt zugewiesen wird.

In der Spalte „Transparency“ wird durch Anklicken einer Zelle ein Schieberegler aktiviert, mit dem die Transparenz der Oberfläche geregelt werden kann. Beim Einstellen wird zusätzlich der

Transparenzgrad in Prozent als Label über dem Schieberegler angezeigt. Ähnlich wie bei dem Polygon−Modus wird die Transparenz über ein *TransparencyAttributes*−Objekt gesteuert, das mit dem *Appearance*−Objekt der Oberfläche verknüpft werden kann.
Die Felder in der Spalte „Visible“ stellen einfache Checkboxen dar, mit denen die entsprechenden Oberflächen sichtbar oder unsichtbar gemacht werden können. Dies geschieht, indem die *BranchGroup*−Knoten, von denen die Zweige mit den Isoflächen ausgehen, in den Scene Graph eingefügt bzw. aus diesem entfernt werden.
Mit dem „Delete“−Button werden die Oberflächen, die den markierten Zeilen der Tabelle entsprechen, gegebenenfalls unsichtbar gemacht und anschließend aus dem Speicher entfernt. Um sie dann wieder betrachten zu können, müssen sie erneut vom Server geladen werden.
Schließlich findet sich in dem Dialogfenster noch eine Checkbox mit dem Titel „Compressed Transmission“. Damit ist es möglich, zwischen gzip−komprimierter und unkomprimierter Übertragung der Isoflächen auszuwählen. Die unkomprimierte Übertragung sollte aber normalerweise nicht sinnvoll sein, da die für den Transport der zusätzlichen Datenmenge über das Internet benötigte Zeit in keinem Fall die für die Dekompression benötigte Zeit unterschreiten dürfte.
Was geschieht, wenn mit dem „Add“−Button eine Oberfläche vom Server angefordert wird, sei noch kurz skizziert. Zuerst werden alle Komonenten der Benutzeroberfläche deaktiviert, die Konflikte während des Ladens der Oberfläche verursachen würden; das sind die „Add“− und „Delete“−Buttons dieses Dialogs sowie die Combobox zum Anwählen einer anderen Reaktion. Danach wird ein Thread gestartet, der eine Socket−Verbindung zum Server herstellt, über diese die Parameter zur Identifikation der gewünschten Oberfläche sendet und anschließend die Antwort des Servers empfängt. Wärenddessen wir ein Fortschrittsdialog eingeblendet, der anzeigt, wie viele der Isoflächen für die einzelnen Frames zum jeweiligen Zeitpunkt bereits empfangen worden sind. Ist die Berechnung der Oberfläche erfolgreich gewesen, so werden für jeden Frame ein *BranchGroup*−Knoten sowie ein oder zwei (bei Orbitalen) *Shape3D*−Knoten erzeugt, die der dreidimensionalen Darstellung der Isoflächen dienen. Flächen ohne Eigenschaftsprojektion wird hierbei ein fest im Programm eingestellter, je nach Oberflächentyp unterschiedlicher Farbwert zugewiesen. Abschließend wird die Socket−Verbindung geschlossen und die deaktivierten GUI−Komponenten werden wieder aktiviert.

4.4.2.4 Realisierung der Messungsfunktionalität

Messungen an den gezeigten Molekülstrukturen lassen sich in *CAVOC* sehr einfach vornehmen. Durch Anwählen des Radiobuttons „Measurements“ wird die Funktion aktiviert. Danach können bis zu vier Atome einfach dadurch selektiert werden, daß man sie mit der linken Maustaste anklickt. Dabei werden nacheinander die Stationen Atomabstand (bei zwei selektierten Atomen), Winkel (bei drei selektierten Atomen) und Diederwinkel (bei vier selektierten Atomen) durchlaufen.
Der Wertebereich der Winkel zwischen drei Atomen erstreckt sich von 0° bis 180°. Diederwinkel können Werte von −180° bis +180° annehmen. Dabei entscheidet sich die Wahl des Vorzeichens nach folgender Regel (Abbildung 12): Wenn man das Molekül so orientiert, daß die Atome B und C eine Achse bilden, die in der Blickrichtung liegt, ist der Diederwinkel positiv, wenn die Atome A und D auf dem kürzesten Weg durch eine Bewegung von Atom A um diese Achse im Uhrzeigersinn zur Deckung zu bringen sind. Muß Atom A dazu im Gegenuhrzeigersinn bewegt werden, ist der Diederwinkel negativ.

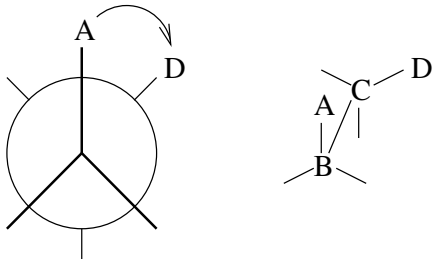

Abbildung 12: Definition des Diederwinkels

Die Anzeige der Meßwerte erfolgt sowohl grafisch als auch numerisch (Abbildung 8). Da bei der Animation einer chemischen Reaktion besonders die Veränderung einer geometrischen Größe über die Reaktionskoordinate interessant ist, kommt der grafischen Darstellung eine besondere Bedeutung zu. Bei der Selektion eines zweiten, dritten oder vierten Atoms werden für alle Frames der Animation die entsprechenden Werte berechnet und in einem Diagramm unter Verwendung des Java 2D−API (Bestandteil der Java Foundation Classes) gegen die Reaktionskoordinate aufgetragen. Damit der Nutzer bei häufigerem Umgang mit dem Programm leicht erkennen kann, welche Größe in dem Diagramm gerade dargestellt wird, werden die Graphen für Atomabstand, Winkel und Diederwinkel in unterschiedlichen Farben gezeichnet. Dieselben Farben werden auch für den Text der numerischen Anzeige verwendet, so daß der Zusammenhang mit der Grafik sofort ersichtlich ist. Zur Markierung der selektierten Atome werden ebenfalls verschiedene Farben eingesetzt, die sowohl den entsprechenden Kugeln in der 3D−Szene als auch den

Atombezeichnungen der Textanzeige zugewiesen werden (Abbildung 8). Auf diese Weise läßt sich leicht erkennen, welche Atome in welcher Reihenfolge angewählt worden sind.
Bei Betätigung des „Reset"−Buttons unterhalb der numerischen Meßwertanzeige werden alle Atome deselektiert und im 3D−Fenster wieder in der „normalen" Farbe dargestellt. Das Diagramm sowie die numerische Anzeige werden gelöscht und der Benutzer kann durch Anklicken weiterer Atome neue Messungen vornehmen.

4.4.3 Beschreibung des Servers zur Berechnung der Isoflächen

Ein entscheidendes Entwicklungsziel von *CAVOC* bestand darin, Volumendaten physikalischer Eigenschaften von Reaktionssystemen in Form von Isoflächen visualisieren zu können. Dieses Ziel kann mit einem Applet prinzipiell auf zwei verschiede Arten erreicht werden. Die erste Möglichkeit besteht darin, sämtliche Volumendaten in das Applet zu laden und clientseitig zu verarbeiten. Dazu muß im Applet eine Routine zur Isoflächenextraktion, z. B. nach dem Marching Cubes−Algorithmus, integriert sein. Dieser Ansatz bietet dort Vorteile, wo viele Isoflächen eines Datensatzes mit verschiedenen Isowerten erzeugt werden sollen, da die Volumendaten nur einmal über das Internet lokal in den Speicher des Client−Rechners geladen werden müssen, um dort für beliebig viele Berechnungen zur Verfügung zu stehen. Als Nachteil fällt bei diesem Vorgehen ins Gewicht, daß wegen deren Größe mit erheblichen Ladezeiten der Volumendatensätze zu rechnen ist.
Die zweite, hier zum Einsatz kommende Möglichkeit besteht darin, nicht die ganzen Volumendatensätze zu übertragen, sondern die Oberflächen auf einem Server zu berechnen und nur diese an das Applet zu übermitteln. Das hat den Vorteil, daß zur Isoflächenextraktion bereits vorhandene, auch in anderen Programmiersprachen als Java geschriebene, Programme eingesetzt werden können. Außerdem ist der Speicherbedarf einzelner Isoflächen in den meisten Fällen wesentlich kleiner als der der zugehörigen Volumendatensätze, was sich vor allem dann in der Menge der über das Internet zu übertragenden Daten positiv auswirkt, wenn nur zu einem oder zwei Isowerten Oberflächen aus einem Datensatz berechnet werden sollen.
Der für *CAVOC* entwickelte Grafikdaten−Server besteht im Kern aus einem Java−Programm, das auf einem bestimmten Port auf eingehende Verbindungen wartet und für jede Verbindung einen eigenen Thread startet, der die Oberflächenberechnung und −übertragung übernimmt. Um Rechenkapazität zu sparen und auch bei hoher Serverbelastung noch kurze Antwortzeiten zu erreichen, wurde außerdem eine Cache−Strategie entwickelt und implementiert. Zur eigentlichen

Berechnung der Isoflächen aus den Volumendatenätzen wird nicht mehr „isovis_pw“ eingesetzt, sondern der „Open Visualization Data Explorer“ (kurz: OpenDX) [23] [22], ein mächtiges Visualisierungsprogramm, welches von IBM entwickelt und bis vor kurzem auch noch kommerziell vermarktet wurde, jetzt aber unter einer Open Source–Lizenz steht. Der Wechsel wurde aus mehreren Gründen vollzogen. Zum einen ist OpenDX ein sehr bekanntes Programm, zum anderen bietet es weit mehr Möglichkeiten zur Datenmanipulation als nur die einfachen Isoflächenextraktionen, so daß sich daraus Optionen für eventuelle zukünftige Weiterentwicklungen von *CAVOC* ergeben. Der Hauptgrund aber ist eines der Features von OpenDX; zum Abbilden von Daten auf die Isoflächen eines anderen Datensatzes, wie z. B. bei der Elpot–Fläche in *newChem* (elektrostatisches Potential auf Elektronendichteoberfläche), kann nicht mehr das dort eingesetzte Verfahren benutzt werden, die Oberflächenpunkte in ein quantenchemisches Rechenprogramm wie Gaussian einzuspeisen und an diesen Stellen im Raum die Eigenschaften zu berechnen. Einerseits wäre dieses Verfahren für ein interaktives Programmsystem zu rechenaufwendig, andererseits ist es, zumindest in Gaussian, auf die Berechnung des elektrostatischen Potentials beschränkt. Nun sollen aber alle für eine Reaktion bereitgestellten Daten auf die Isoflächen eines anderen Datentyps bei beliebigen Isowerten abgebildet werden können. Dieses Ziel kann dadurch erreicht werden, daß die Eigenschaftswerte an den Oberflächenpunkten aus den Volumendaten des anderen Typs interpoliert werden. Genau diese Aufgabe kann OpenDX übernehmen. Darüber hinaus stellt OpenDX eine Funktion bereit, mit der die Eigenschaftswerte anhand einer einstellbaren Farbskala direkt in die entsprechenden Farbwerte umgerechnet werden können.
Die Funktionsweise des Servers soll mit dem folgenden Flußdiagramm (Abbildung 13) verdeutlicht werden.

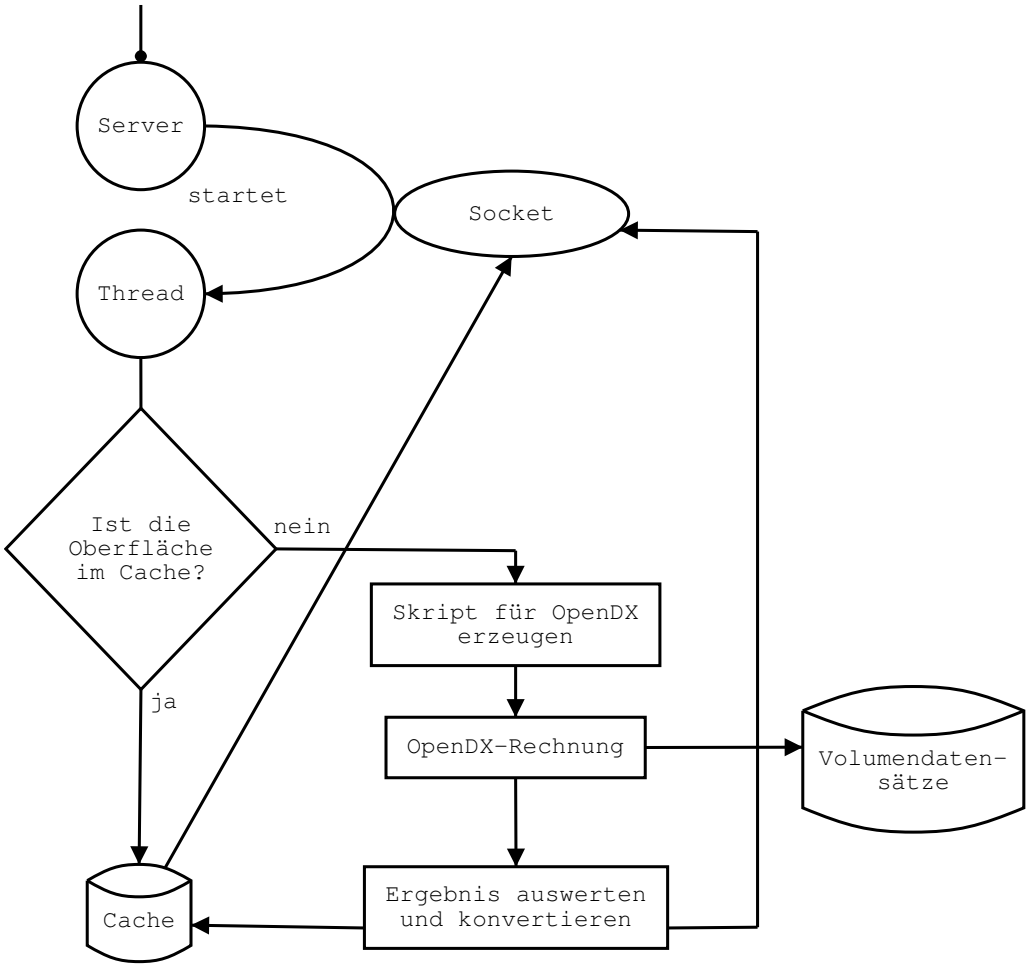

Abbildung 13: Funktionsweise des Servers zur Berechnung der Isoflächen

Beim Starten des Servers wird zuerst der Cache initialisiert. Das beim Aufruf angegebene Cache–Verzeichnis wird auf Dateien mit der Endung „.gz“ durchsucht. Die gefundenen Treffer werden in einem *CacheController*–Objekt registriert und deren Größen zur Überprüfung der maximalen Cache–Größe aufsummiert. Die selbst entwickelte Klasse *CacheController* dient zur Verwaltung des Oberflächen–Caches. Sie kontrolliert die Größe des Caches und darüber hinaus die Anzahl der gleichzeitig laufenden Rechnungen. Weiterhin speichert sie in einer Liste die Reihenfolge der Zugriffe auf die einzelnen Oberflächen.

Nach der Initialisierung wird ein *ServerSocket*–Objekt erzeugt, welches fortan auf einem im Programm fest eingestellten Port mit der *accept()*–Methode auf eingehende Verbindungen wartet. Bei jedem erfolgreichen Verbindungsaufbau wird ein neuer Thread gestartet, dem das von der *accept()*–Methode generierte *Socket*–Objekt als Parameter zur Identifikation der Verbindung übergeben wird. Danach setzt das Hauptprogramm, durch eine Endlosschleife gesteuert, sofort die Überwachung des Ports fort, so daß mehrere Anforderungen gleichzeitig bearbeitet werden können.

Jeder gestartete Thread übernimmt die alleinige Bedienung der ihm zugewiesenen Client–Verbindung. Dazu öffnet er zunächst die Kommunikationskanäle zum Lesen und Schreiben über den Socket. Danach werden die Paramameter eingelesen, die die zu liefernde Oberfläche eindeutig identifizieren, der Name der Reaktion, der Oberflächentyp, der Isowert sowie die einzukodierende Eigenschaft. Außerdem wird noch ein Wert vom Typ *boolean* eingelesen, der angibt, ob die fertige Oberfläche komprimiert oder unkomprimiert zurückgeschickt werden soll. Aus den Parametern wird der Name der zugehörigen Cache–Datei bestimmt, womit vom *CacheController*–Objekt ein *CacheInstance*–Objekt angefordert wird. Eine Instanz der selbst entwickelten Klasse *CacheInstance* wird vom *CacheController*–Objekt genau einmal für jede im Cache vorliegende Oberfläche erzeugt. Sie registriert, wie viele Clients gerade auf die ihr zugeordnete Oberfläche zugreifen und wie der Verfügbarkeitsstatus ist. Über das *CacheInstance*–Objekt wird abgefragt, ob die betreffende Oberfläche bereits im Cache vorliegt oder erst noch berechnet werden muß. Falls die Antwort positiv ausfällt, wird kontrolliert, ob die Oberfläche auch wirklich abrufbereit ist oder aber gerade erst von einem anderen, kurz zuvor gestarteten Thread berechnet wird. In letzterem Fall wird der Thread so lange unterbrochen, bis von dem die Oberfläche berechnenden Thread das Signal zum Weitermachen eintrifft. Danach wird die Cache–Datei gelesen und entweder roh oder in entpackter Form an den Client übermittelt.

Wenn der Thread hingegen nicht auf eine fertige Cache–Datei zurückgreifen kann, muß er selber die Berechnung der Oberfläche vornehmen. Dazu sucht er sich aus einer Datei, in der die Reaktionen aufgelistet sind, für die Volumendaten zur Verfügung stehen, die Dateinamen der zu verwendenden Volumendatensätze sowie eventuelle Phasenvertauschungsinformationen für die Orbitale heraus und erstellt damit ein Skript für die anschließende OpenDX–Rechnung (Beispielskript für eine farblich kodierte Fläche bei einem Isowert von 0,002 im Anhang C). Zuvor kontrolliert das *CacheController*–Objekt, ob nicht schon die maximale Anzahl zugelassener Rechnungen läuft. In dem Fall würde es den Thread erst einmal in eine Warteschleife schicken. Außerdem prüft es nach, ob die maximale Cache–Größe überschritten ist, und löscht gegebenenfalls vor dem Start der neuen Rechnung so viele Dateien aus dem Cache, bis dessen Größe unter den Maximalwert fällt. Dabei werden zuerst die Dateien gelöscht, auf die am längsten nicht mehr zugegriffen worden ist.

Mit den eingesetzten Skripts wird OpenDX angewiesen, als Ergebnis zwei Dateien zu produzieren. In der einen sind in binärer Form die Rohdaten enthalten, während in der anderen Datei die Informationen zu finden sind, wie die binäre Datei aufgebaut ist, d. h. wie viele Vertexkoordinaten, Verknüpfungen, Normalen, Farbwerte, usw. darin in welcher Reihenfolge zu finden sind. Bei

Orbitalen wird im Skript zusätzlich eine Rechnung mit dem negierten Isowert und die Ausgabe zweier weiterer Ergebnisdateien für die zweite Isofläche veranlaßt.
Als nächstes werden die Ausgabe−Dateien der OpenDX−Rechnung gelesen und verarbeitet, d. h. die benötigten Daten werden im Speicher abgelegt und anschließend in einem eigenen Format (Anhang B) in ein Byte−Array eingetragen. Bei komprimierter Übertragung wird das Byte−Array jetzt über ein *GZIPOutputStream*−Objekt gepackt. Anschließend erfolgt die Übertragung an den Client und parallel dazu in einem eigenen Thread das Rausschreiben der Cache−Datei. Falls eine unkomprimierte Übertragung gewünscht war, werden die Daten erst beim Schreiben der Cache−Datei gepackt. Wenn der Thread die Cache−Datei fertiggestellt hat, informiert er sowohl das *CacheController*−Objekt als auch das *CacheInstance*−Objekt, damit eventuell wartende Threads eine neue Berechnung starten bzw. die fertige Cache−Datei auslesen können.
Jeder Thread, der die Übertragung einer Isofläche beendet hat, gibt als letztes die von ihm belegten Systemressourcen frei, indem er die Kommunikationskanäle zum Client und den Socket schließt.

4.4.4 Beschreibung des Hilfsservers

Wie schon erwähnt, kann ein Java−Applet aus Sicherheitsgründen ohne besondere Maßnahmen keine Netzwerkverbindungen zu anderen Hosts als seinem Ursprungsrechner herstellen. Wenn es aber aus irgendwelchen Gründen nicht möglich oder sinnvoll ist, auf einem Computer neben dem Web−Server einen weiteren Ressourcen verbrauchenden Service, wie die Berechnung von Isoflächen, zu betreiben, sei es, weil nicht genug Festplattenkapazität zur Verfügung steht oder der Rechner einfach zu langsam ist, kann der die eigentliche Arbeit leistende Server auch auf einem anderen Host eingerichtet werden. In dem Fall reicht es aus, auf dem Host mit dem Web−Server einen Server laufen zu lassen, der lediglich als Schnittstelle zwischen dem Applet und dem eigentlichen Server fungiert und nicht viel mehr leisten muß, als die von einer Seite empfangenen Daten an die jeweils andere Seite weiterzuleiten. Genau diese Funktion erfüllt der für *CAVOC* entwickelte, in diesem Abschnitt beschriebene Hilfsserver.
Als einziges Kommandozeilenargument wird dem Programm der Name des Rechners, auf dem der eigentliche Server läuft, übergeben. Beim Starten registriert es sich auf einem fest eingestellten Port und wartet dort auf Verbindungen. Für jede eingehende Verbindung wird auch hier ein eigener Thread gestartet, der die weitere Kommunikation übernimmt, weil ansonsten immer nur ein Benutzer zur gleichen Zeit eine Oberfläche auf seinen Rechner laden könnte. Jeder so gestartete Thread gibt zu Protokollierungszwecken zuerst zusammen mit der Uhrzeit den Namen des

Rechners aus, von dem die Verbindung aufgebaut worden ist. Nach demselben Kommunikationsprotokoll wie bei einer direkten Client−Server−Verbindung werden danach die Parameter zur Abfrage der gewünschten Oberfläche vom Applet eingelesen und unmittelbar an den Server weitergeleitet. Dann wird einfach alles, was vom Server zurückgesendet wird, empfangen und an das Applet durchgereicht. Schließlich werden alle Kommunikationskanäle und die beiden Socket−Verbindungen geschlossen.

4.4.5 Hilfsprogramme

In diesem Abschnitt sollen kurz die Programme erwähnt werden, die entwickelt wurden, um die in *CAVOC* eingesetzten Dateiformate zu erzeugen.
Die Atomkoordinaten und Konnektivitäten für die Ball&Stick−Darstellung werden mit dem in Java gschriebenen Programm *PDBConv* aus den vorliegenden pdb−Dateien gewonnen. Dieses wird mit den Dateinamen der zu verarbeitenden Strukturen im pdb−Format aufgerufen und fragt dann nacheinander die Parameter für die Anfangsansicht im 3D−Fenster des Applets (Rotations− bzw. Euler−Winkel um x−, y− und z−Achse, Translationswerte in x−, y− und z−Richtung), die Gesamtzahl der berechneten Strukturen auf dem Reaktionspfad sowie den Namen der im *CAVOC*−Format (Anhang A) zu erzeugenden Datei ab.
Für die Generierung der die Volumendaten enthaltenden HDF−Dateien wurde mit Hilfe des DFSD−API von HDF 4 das C−Programm *cube2dfsd* geschrieben, welches die Datensätze einer Reihe von Gaussian Cube−Dateien in das HDF 4−Format konvertieren kann. Dieses Programm liest aus den zu einer Animationssequenz gehörenden Cube−Dateien die Volumendatensätze mit den Gittergeometrien und gibt eine HDF−Datei pro Reaktion (Animationssequenz) und Datentyp aus. Z. B. können aus 30 Cube−Dateien einer Reaktion, die die Daten für die Orbitale HOMO−3 bis LUMO+3 enthalten, mit einem Programmaufruf acht HDF−Dateien (eine pro Orbital) erzeugt werden, die jeweils 30 einzelne Datensätze enthalten.
Eine Voraussetzung für das korrekte Abbilden eines Datensatzes auf einen anderen als Oberflächeneigenschaft in OpenDX besteht darin, daß beide Datensätze die gleichen Gitterdimensionen besitzen. Daher wurde ebenfalls unter Zuhilfenahme des HDF−API das C−Programm *dimcheck* entwickelt, welches genutzt werden kann, um die Äquivalenz der Gitter in zwei verschiedenen HDF−Dateien zu überprüfen. Da Gaussian aber bei der Berechnung von Cube−Dateien zu einer gegebenen Struktur unabhängig von der Eigenschaft immer das gleiche Gitter verwendet, sollte der Einsatz von *dimcheck* nur dann interessant sein, wenn Datensätze anderer

Herkunft in *CAVOC* eingebaut werden.

4.4.6 Anwendungsbeispiele

In diesem Abschnitt soll anhand einiger bebilderter Beispiele gezeigt werden, wie *CAVOC* dem Chemiker dabei helfen kann, sich ein dreidimensionales Bild der Abläufe bei chemischen Reaktionen auf molekularer Ebene zu machen.

Anhand der radikalischen Addition des Methylradikals an Ethen läßt sich die Nützlichkeit einer Spindichteoberfläche veranschaulichen (Abbildung 14). Aus den Bildern von Eduktkomplex, Übergangszustand und Produktkomplex kann man ablesen, an welchen Atomen der Radikalcharakter in der jeweiligen Phase der Reaktion lokalisiert ist. Man erkennt auch sehr gut, daß das ungepaarte Elektron am Übergangszustand (ÜZ) über die beiden terminalen C−Atome delokalisiert ist.

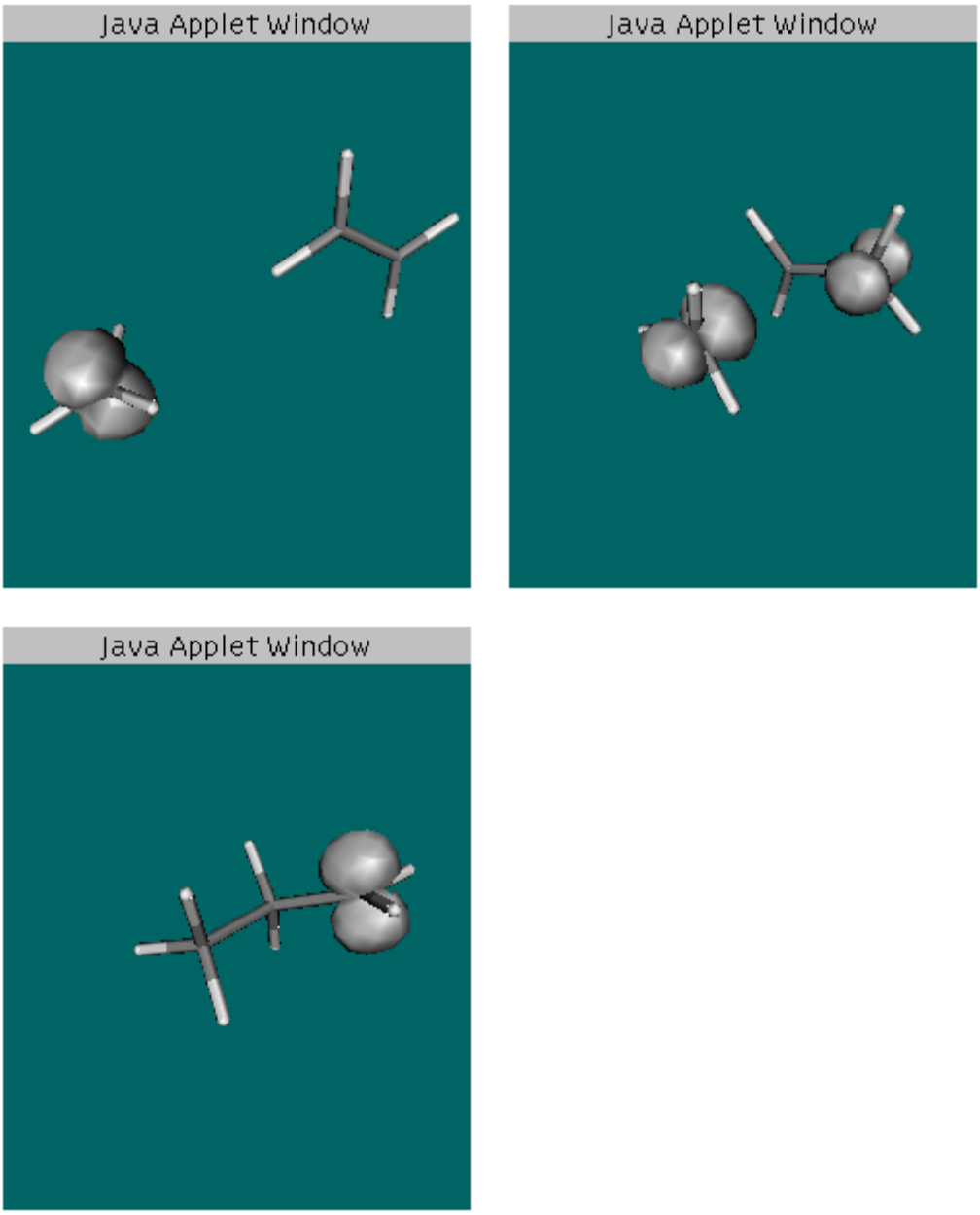

Abbildung 14: Spindichte bei der Reaktion zwischen einem Methylradikal und Ethen (Isowert: 0,002)
links oben: Eduktkomplex, rechts oben: ÜZ, unten: Produktkomplex

An dem sehr einfachen Beispiel der S_N2–Reaktion zwischen Methylchlorid und einem Chlorid–Anion läßt sich die Aussagekraft einer Elpot–Oberfläche (Abbildung 15) demonstrieren. Bei dem Eduktkomplex (entspricht dem Produktkomplex) erkennt man leicht die stark negative Ladungsdichte (rot) am freien Chlorid–Ion und das positive Zentrum (blau) am C–Atom. Am ÜZ verteilt sich die negative Ladung gleichmäßig auf die beiden Cl–Atome. Die gleichen Aussagen gewinnt man auch aus der Betrachtung der Isoflächen des elektrostatischen Potentials (Abbildung

16).

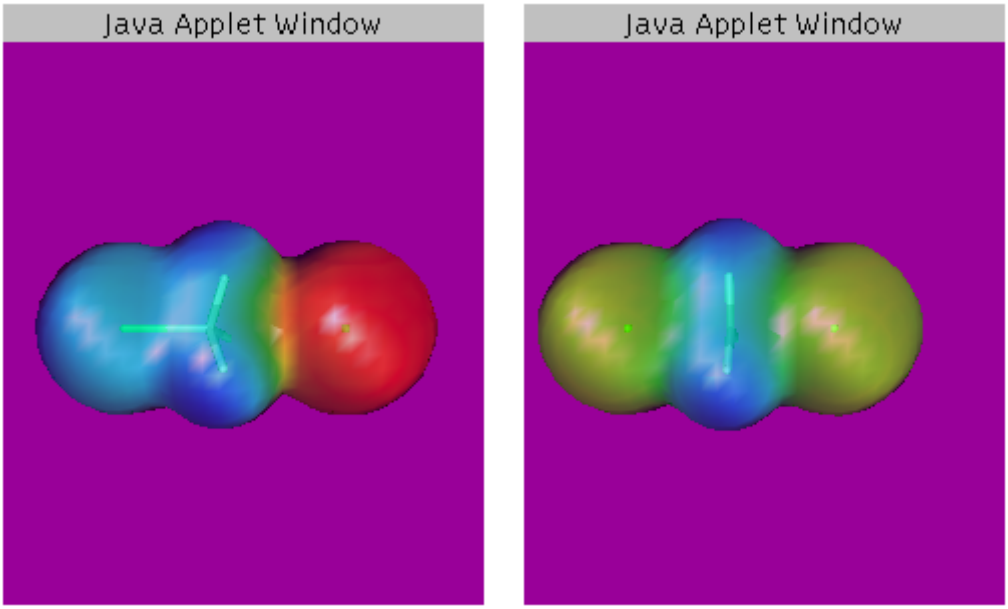

Abbildung 15: Elektronendichte mit farblich kodiertem elektrostatischen Potential bei der S_N2−Reaktion zwischen Methylchlorid und Chlorid (Isowert: 0,002)
links: Eduktkomplex, rechts: ÜZ

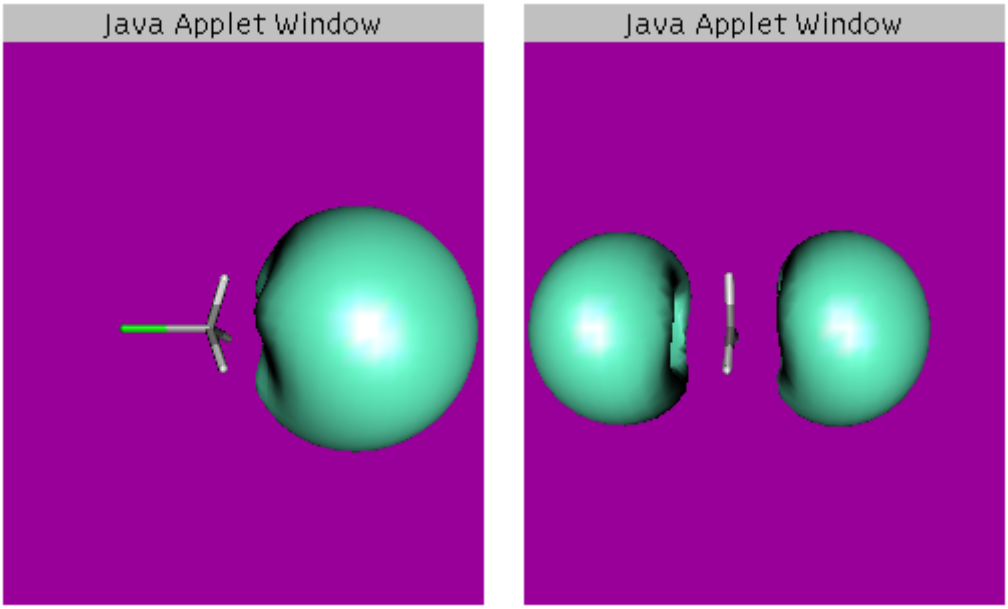

Abbildung 16: Elektrostatisches Potential bei der Reaktion zwischen Methylchlorid und Chlorid (Isowert: −0,2)
links: Eduktkomplex, rechts: ÜZ

Wie sich die Orbitalenergiediagramme zur Erklärung von auftretenden Sprüngen in der 3D–

Isoflächendarstellung einzelner Orbitale heranziehen lassen, sei am Beispiel der Diels–Alder–Reaktion zwischen Ethen und Butadien gezeigt. Beim Schritt von Frame 25 zu Frame 26 ändert sich die Gestalt der Oberfläche des LUMO+2 stark. Bei Betrachtung des Orbitalenergiediagramms (Abbildung 4) läßt sich vermuten, daß die Energie dieses Orbitals soweit ansteigt, bis es diejenige des LUMO+3 übertrifft. An dieser Stelle nimmt das LUMO+2 also die Identität des vorher mit LUMO+3 zu bezeichnenden Orbitals an, was anhand der Abbildung 17 auch sofort deutlich wird.

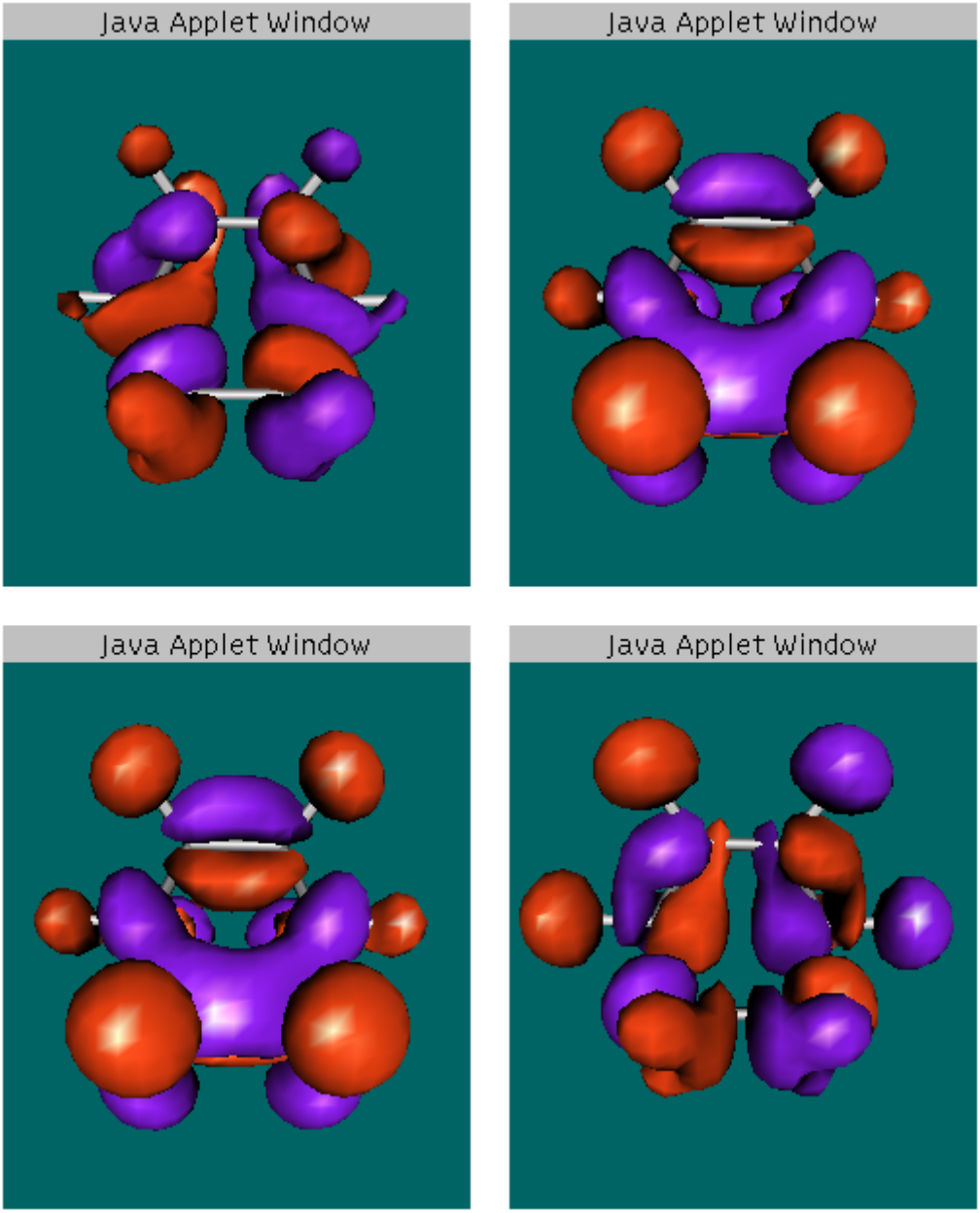

Abbildung 17: Veranschaulichung einer Orbitalkreuzung bei der Diels–Alder–Reaktion zwischen Butadien und Ethen (Isowerte: 0,032)
oben: zwei aufeinanderfolgende Frames (mit LUMO+2), unten: die gleichen Frames (jetzt mit LUMO+3)

5 Implementierung und Anwendung eines QSAR−Programms

In diesem Teil der Arbeit wird die Entwicklung des Programms *MultReg* zur Aufstellung möglichst optimaler QSAR/QSPR−Modelle vorgestellt. Anhand eines bereits mit anderen Methoden untersuchten Datensatzes wird die Leistungsfähigkeit des Programms untersucht und bewertet.
Die Entwicklung eines neuen Medikaments ist ein langwieriger und kostspieliger Prozeß. Ausgehend von einer Leitstruktur sind noch viele Schritte notwendig, um die gewünschte Wirkung zu optimieren und unerwünschte Nebenwirkungen zu minimieren. Geeignet scheinende Derivate müssen synthetisiert und biologisch evaluiert werden. Durch den Einsatz moderner Computerverfahren kann ein Großteil der potentiellen Wirkstoffkandidaten bereits am Rechner verworfen werden, was die Kosten für Synthesen und biologische Tests enorm zu senken hilft. Meistens wird versucht, mit quantitativen Struktur−Wirkungsbeziehungen (QSAR: Quantitative Structure Activity Relationships) einen mathematischen Zusammenhang zwischen chemischer Struktur und biologischer Wirkung herzustellen. Verallgemeinert man das Verfahren auf andere Zielgrößen, spricht man auch von QSPR (Quantitative Structure Property Relationships).
Der QSAR−Ansatz wurde 1963 von Hansch [29] angeregt durch die Hammet−Gleichung [28] eingeführt. Er basiert auf der Annahme, daß Unterschiede in der chemischen Struktur von Verbindungen für ihre unterschiedlichen biologischen Aktivitäten verantwortlich sind. Die drei wichtigsten Klassen von strukturabhängigen Eigenschaften, die die Affinität eines Wirkstoffs zu seinem Rezeptor und damit seine biologische Wirksamkeit bestimmen, sind die elektronischen, sterischen und hydrophoben Wechselwirkungen [27]. Mit Hilfe von Größen, wie Hammet−Konstanten, STERIMOL−Parametern [26] oder Hydrophobizitätskonstanten, die in diesem Zusammenhang auch als Deskriptoren bezeichnet werden, lassen sich solche Eigenschaften für jede Struktur quantitativ erfassen. Davon unabhängig kann aber jeder beliebige Parameter einer chemischen Verbindung, der als Zahlenwert ausgedrückt werden kann, als Deskriptor Verwendung finden. So sind z. B. die aus der Graphentheorie abgeleiteten topologischen Deskriptoren, die vor allem von Randic [25] und Kier und Hall [24] untersucht worden sind, heute weit verbreitet im Einsatz. Sie haben den Vorteil, ohne quantenmechanische oder andere aufwendige physikalische Rechnungen einfach aus 2D−Strukturen zugänglich zu sein.

5.1 Grundlagen: Mathematische Modelle

Das Ziel von QSAR−Untersuchungen ist es, zum Zweck einer möglichst genauen und robusten

Vorhersage der biologischen Aktivität (abhängige Variable: y) chemischer Verbindungen einen mathematisch funktionalen Zusammenhang mit strukturbeschreibenden Deskriptoren (unabhängige Variablen: x_1, x_2, x_3, ...) zu formulieren:

$$y = f(x_1, x_2, x_3, ...)$$

Dieses Problem läßt sich klassisch mit Hilfe einer Regressionsrechnung unter Minimierung der Summe der Fehlerquadrate lösen. Angewendet auf ein Problem mit mehreren unabhängigen Variablen und unter Einbeziehung nur linearer Terme führt das auf den Fall der multiplen linearen Regression (MLR). Dieser Ansatz birgt bei QSAR−Untersuchungen jedoch einige Probleme. Häufig besitzt der Trainingsdatensatz mehr unabhängige Variablen als Trainingsobjekte. Ein Extremfall ist z. B. die CoMFA−Methode (Comparative Molecular Field Analysis) [30], bei der leicht mehrere tausend Deskriptoren anfallen können. Dann ist das Problem unterbestimmt und die MLR nicht anwendbar. Ein zweites Problem ist, daß die Deskriptoren häufig nicht vollkommen unabhängig voneinander sind (Problem der Multikollinearität). So sind z. B. das Molekülvolumen und die Hydrophobizität miteinander korreliert. Das Multikollinearitätenproblem führt normalerweise zu einer Verschlechterung der Vorhersagekraft beim Aufstellen eines Regressionsmodells. Dies macht sich dadurch bemerkbar, daß zwar die Zielgrößen des Trainingsdatensatzes sehr gut vorhergesagt werden, man bei der Anwendung auf andere Testgrößen jedoch völlig falsche Ergebnisse erhält („Overfitting“).

Um bei der Modellgenerierung die externe Vorhersagekraft zu berücksichtigen, wird oft das Verfahren der Kreuzvalidierung eingesetzt. Hierbei werden systematisch oder zufällig ein oder mehrere Trainingsobjekte bei der Aufstellung eines Modells nicht berücksichtigt. Danach wird die Zielgröße für die herausgelassenen Objekte mit dem so aufgestellten Modell berechnet. Dieses Vorgehen wird dann mit anderen Trainingsobjekten aus demselben Datensatz wiederholt. Als quantitatives Maß für die Güte des gemachten Ansatzes erhält man schließlich den kreuzvalidierten Korrelationskoeffizienten q^2:

$$q^2 = 1 - \frac{\sum (y_{act,i} - y_{pred,i})^2}{\sum (y_{act,i} - \bar{y}_{act})^2}$$

$y_{act,i}$ gemessene Aktivität der Verbindung i

$y_{pred,i}$ vorhergesagte Aktivität der Verbindung i

$\bar{y}_{act}$ Mittelwert der gemessenen Aktivitäten

Für die möglichen Zahlenwerte von q^2 gilt folgende Relation:

$$1 \geqslant q^2 \geqslant -\infty$$

Die Güte eines aufgestellten Modells ist um so besser, je größer dessen q^2−Wert ist. Um es als vertrauenswürdig betrachten zu können, setzt man für q^2 allgemein eine Untergrenze von 0,4 bis 0,5 an [31].

Die am häufigsten eingesetzte Art der Kreuzvalidierung ist das Leave−one−out−Verfahren (LOO). Hierbei werden systematisch alle Trainingsobjekte jeweils einmal einzeln herausgelassen. Der Vorteil dieser Methode ist, daß sie bei einem mittleren Rechenaufwand jederzeit zu reproduzierende q^2−Werte liefert. Als Nachteil ist dem LOO−Verfahren anzukreiden, daß es gegen Gruppen von benachbarten Ausreißern empfindlich ist und so oftmals eine zu optimistische Einschätzung der Vorhersagekraft von QSAR−Modellen liefert. Prinzipiell besser geeignet sind daher die sogenannten Leave−several−out−Verfahren (LSO), bei denen größere Gruppen von Objekten aus einem Trainingsdatensatz herausgelassen werden. Allerdings wächst die Anzahl der zu evaluierenden Kombinationen und damit auch die Rechenzeit beim systematischen Herauslassen von Gruppen mit dem Umfang des Datensatzes und der Größe der Gruppen enorm stark an. Deswegen wird manchmal mit nur einer begrenzten Anzahl zufällig gewählter Gruppen gearbeitet, was jedoch den anderen Nachteil mit sich bringt, daß die so errechneten q^2−Werte dann nicht mehr reproduzierbar sind und von Lauf zu Lauf größeren Schwankungen unterliegen können.

Ein weiteres Problem bei der Modellgenerierung ist der häufig unterschiedlich große Wertebereich der verwendeten Deskriptoren. Wenn bei einem Datensatz ein solcher Fall vorliegt und eine Übergewichtung der Deskriptoren mit großen Zahlenwerten zu Lasten der Deskriptoren mit kleinen Zahlenwerten vermieden werden soll, bietet es sich an, die Deskriptoren zunächst zu skalieren. Eine für die meisten Fälle gut geeignete Skalierungsmethode ist die Autoskalierung. Damit werden die Werte x_{ij} eines Deskriptors j durch Abziehen des Mittelwerts $\bar{x}_j$ zentriert und durch Division durch die Standardabweichung σ_j auf eine Standardabweichung von eins skaliert.

$$x'_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_j} \quad ; \quad \sigma_j = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)}$$

Hierbei bezeichnet x'_{ij} den neuen autoskalierten Wert des Deskriptors *j* der Verbindung *i* und *n* die Anzahl der Verbindungen im Datensatz.

5.1.1 k Nearest Neighbours

Die Methode der k nächsten Nachbarn (KNN, k Nearest Neighbours) nach Fix und Hodges [32] ist ursprünglich eines der einfachsten unüberwachten Lernverfahren zur Klassifizierung von Testobjekten. Hierbei werden zu einem gegebenen Testobjekt die euklidischen Abstände aller Objekte aus dem Trainingsdatensatz bestimmt. Die Zuordnung erfolgt dann zu der Gruppe, aus der die Mehrheit der im Sinne des euklidischen Abstands k nächsten Nachbarn stammt, wobei k in der Regel eine kleine natürliche Zahl ist. Der euklidische Abstand d_{ij} zweier Punkte *i* und *j* im *N*−dimensionalen Raum ist definiert durch:

$$d_{ij} = \sqrt{\sum_{k=1}^{N} (x_{i,k} - x_{j,k})^2}$$

Die Festlegung des Parameters k stellt den entscheidenden Schritt bei der Anwendung der KNN−Methode dar. Bei zu kleinen Werten von k, etwa k = 1, besteht die Gefahr, daß die Klassifikation eines Testobjekts allein durch Ausreißer diktiert wird. Umgekehrt wird bei sehr großen Werte von k, im Extremfall k = Anzahl der Objekte im Trainingsdatensatz, die Zuordnung automatisch zu der Gruppe erfolgen, die am stärksten im Trainingsdatensatz vertreten ist.
Die Anwendung der KNN−Methode in QSAR−Anwendungen im Zusammenhang mit Variablenselektionsmethoden wurde zuerst von Tropsha und Zheng beschrieben [33]. Im Unterschied zur einfachen Klassifikation werden hier die Ergebnisse quantitativ ausgewertet. Das bedeutet, daß die Aktivität einer Verbindung als der Mittelwert der Aktivitäten ihrer k nächsten Nachbarn im Deskriptorraum vorhergesagt wird.

5.1.2 Partial Least Squares

Die Probleme, die die herkömmliche multiple lineare Regression aufwirft, sind bei der PLS−Methode (Partial Least Squares) nach Wold [34] weitgehend gelöst. Auch dabei handelt es sich um eine lineare Regressionsmethode. Allerdings treten an die Stelle der unabhängigen Variablen die sogenannten latenten Variablen (*LV*).

$$y = a_1 LV_1 + a_2 LV_2 + a_3 LV_3 + \ldots \quad ; a_k\text{: Regressionskoeffizienten}$$

Jede latente Variable LV_q (q = 1, 2, 3 ...) ergibt sich aus einer Linearkombination der unabhängigen Variablen:

$$LV_q = b_{q,1} x_1 + b_{q,2} x_2 + b_{q,3} x_3 + \ldots \quad ; b_{q,k}\text{: Wichtungsfaktoren}$$

Damit besitzt die PLS−Methode eine enge Verwandtschaft zu einer weiteren linearen Regressionsmethode, der Hauptkomponentenregression (PCR, Principal Components Regression), bei der andere Linearkombinationen der unabhängigen Variablen, nämlich die Hauptkomponenten, zum Einsatz kommen. Durch die Beschränkung auf nur die ersten wenigen latenten Variablen bzw. Hauptkomponenten wird bei der PLS−Methode wie bei der PCR die Dimensionalität des Problems reduziert. Die optimale Anzahl einzubeziehender latenter Variablen kann über Kreuzvalidierung bestimmt werden.

Die Hauptkomponenten bei der PCR werden anders als die latenten Variablen bei der PLS−Methode ausschließlich unter dem Ziel einer maximalen Varianzausschöpfung der unabhängigen Variablen konstruiert, Informationen über die abhängigen Variablen gehen in diesen Prozeß nicht ein. So werden bei einer PCR mit wenigen Hauptkomponenten zwar wirksam Multikollinearitäten beseitigt, doch besteht die Gefahr, daß die verwendeten Hauptkomponenten nur in geringer Beziehung zur Zielgröße stehen und die nicht eingehenden Hauptkomponenten höherer Ordnung die größere Vorhersagekraft besitzen. Mit dieser Eigenschaft steht die PCR folglich im Gegensatz zur MLR, wo die Korrelation mit den abhängigen Variablen das übergeordnete Ziel ist.

Die PLS−Methode stellt einen oftmals überlegenen Kompromiß zwischen diesen beiden Verfahren dar, weil bei der Konstruktion der latenten Variablen beide Kriterien, maximale Varianzausschöpfung und maximale Korrelation mit den abhängigen Variablen, gleichermaßen berücksichtigt werden. Wie in einer statistischen Arbeit gezeigt wurde, lassen sich alle drei Methoden in ein parametrisches Regressionsverfahren, genannt Continuum Regression [35], einbetten. Continuum Regression erlaubt, durch die Variation eines einzigen Parameters, von MLR über PLS bis PCR auch jeden weiteren Kompromiß zwischen den drei Methoden einzugehen.

Auf den genauen Algorithmus der PLS−Methode soll hier nicht weiter eingegangen werden. Er ist detailliert z. B. im Tutorial von Geladi [36] beschrieben.

5.2 Grundlagen: Optimierungsmethoden

Auch Verfahren wie PLS sind nicht völlig unempfindlich gegen Multikollinearitäten. Durch den Einsatz von Variablenselektionsmethoden kann daher bei QSAR−Anwendungen eine weitere Verbesserung der Vorhersagekraft von Modellen erreicht werden. Außerdem kann aus den Ergebnissen ein weiterer Nutzen gezogen werden. So gelingt es häufig, durch den Vergleich mehrerer guter Modelle aus den einfließenden Deskriptoren diejenigen mit der größten Signifikanz zu bestimmen, was wichtige Hinweise darauf gibt, welche Eigenschaften am Zielmolekül zu verändern sind, um eine Steigerung der biologischen Wirksamkeit zu erreichen. Ist z. B. eine Hydrophobizitätskonstante als relevant für die Effizienz eines Wirkstoffs identifiziert worden, heißt dies für die weitere Entwicklung, daß es sich lohnen kann, Modifikationen im Molekül vorzunehmen, welche die Hydrophobizität in der passenden Weise ändern.
Optimierungsalgorithmen werden eingesetzt, um das Modell mit dem besten q^2−Wert durch Eliminierung von Deskriptoren und gleichzeitiger Festlegung der anderen Optimierungsparameter (k bei KNN, die Anzahl der latenten Variablen bei PLS) zu bestimmen. Für diese Aufgabe eignen sich besonders stochastische Optimierungsmethoden.

5.2.1 Simulated Annealing

Simulated Annealing (SA) ist eine aus der statistischen Mechanik abgeleitete, stochastische Optimierungsmethode [37]. Die statistische Mechanik trifft folgende Aussage: Wenn sich ein System bei einer gegebenen Temperatur T im thermischen Gleichgewicht befindet, ist die Wahrscheinlichkeit $\pi_T(s)$, das System in einer bestimmten Konfiguration s vorzufinden, von der Energie E(s) dieser Konfiguration abhängig und folgt der Boltzmann−Verteilung:

$$\pi_T(s) = \frac{e^{\frac{-E(s)}{kT}}}{\sum_{w \in S} e^{\frac{-E(w)}{kT}}}$$

Dabei ist k die Boltzmann−Konstante und S die Menge aller möglichen Konfigurationen.
Das Verhalten eines Systems von Teilchen im thermischen Gleichgewicht bei einer Temperatur T kann man nach einer von Metropolis [38] entwickelten Methode simulieren. Man nehme an, zu einer Zeit t befinde sich das System in der Konfiguration q. Generiert man zufällig einen

Kandidaten *r* für die Konfiguration zur Zeit t + 1, so hängt die Wahrscheinlichkeit, ob die Konfiguration *r* angenommen wird, von der Energiedifferenz der Zustände *q* und *r* ab. Zur Entscheidung berechnet man das Verhältnis *p* der Wahrscheinlichkeiten, das System in Zustand *q* oder *r* zu finden:

$$p = \frac{\pi_T(r)}{\pi_T(q)} = \mathrm{e}^{\frac{-(E(r) - E(q))}{kT}}$$

Wenn $p > 1$ bzw. die Energie von *r* kleiner als die von *q* ist, wird *r* automatisch als die neue Konfiguration zur Zeit t + 1 akzeptiert. Wenn $p \leqslant 1$ ist, d. h. die Energie von *r* ist größer oder gleich derjenigen von *q*, wird *r* mit einer Wahrscheinlichkeit von *p* als die neue Konfiguration angenommen. Das bedeutet, daß auch Zustände einer höheren Energie angenommen werden können. Nur ist die Wahrscheinlichkeit dafür um so geringer, je größer der dabei entstehende Energiezuwachs und je niedriger die Temperatur ist.
Simulated Annealing simuliert den physikalischen Vorgang des Erstarrens einer Schmelze:

- In einer Schmelze sind die einzelnen Teilchen zufällig angeordnet. Dies entspricht einem zufällig generierten Lösungsvorschlag am Anfang des Optimierungsprozesses.
- Beim langsamen Abkühlen ordnen sich die Teilchen so an, daß im Feststoff der energieärmste Zustand erreicht wird. Dies entspricht der schrittweisen Verbesserung der Lösung bis zum Erreichen des Optimums.

Bei dieser Analogie kann der Zustand einer Schmelze mit einer möglichen Lösung beim Simulated Annealing identifiziert werden, während die Energie des Zustands dem Wert entspricht, den die zu optimierende Zielfunktion bei dieser Lösung annimmt.
Der Algorithmus des Simulated Annealing läßt sich wie folgt formulieren:

1. Erzeuge eine zufällig gewählte Anfangslösung und wähle eine geeignete Starttemperatur
2. Bewerte die Anfangslösung

Wiederhole, bis Abbruchbedingung erreicht ist

3. Setze $L_a = 0$ und $L = 0$

Wiederhole, bis $L_a = L_{a,max}$ oder $L = L_{max}$

4. Generiere eine neue Lösung durch eine zufällige Mutation der aktuellen Lösung
5. Bewerte die neue Lösung
6. Wenn das Metropolis−Kriterium erfüllt ist, ersetze die aktuelle durch die neue Lösung und inkrementiere L_a
7. Inkrementiere L

8. Verringere die Temperatur

Im obigen Algorithmus bezeichnet L die Anzahl der bei einer Temperatur vorgeschlagenen neuen Lösungen und L_a die Anzahl der akzeptierten neuen Lösungen. L_{max} und $L_{a,max}$ sind Parameter, mit denen die Zeit, die der Algorithmus bei einem Temperaturwert verweilt, beeinflußt werden kann.
Wie auch beim physikalischen Analogon ist der Erfolg beim Simulated Annealing entscheidend davon abhängig, wie der Abkühlungsvorgang erfolgt. Wird die Temperatur zu schnell gesenkt, bilden sich „Defekte“ aus, und das Optimum wird nicht erreicht. Umgekehrt kostet ein zu langsames Abkühlen unnötig viel Zeit. Der erste Schritt ist also die Wahl einer geeigneten Starttemperatur. Dabei kann so vorgegangen werden, daß man in aufsteigender Folge verschiedene Werte testet, bis eine Temperatur erreicht wird, bei der die Akzeptanz neuer Lösungen (das Verhältnis L_a / L) nahe 100% liegt.
Für den Temperaturverlauf wird zumeist ein exponentieller Ansatz gemacht:

$$\theta_k = \alpha * \theta_{k-1}$$

Der Parameter α, der den Abfall der Temperatur θ vom Schritt k−1 zum Schritt k bestimmt, liegt gewöhnlich im Bereich von 0,8 bis 0,95.
Der Algorithmus wird abgebrochen, wenn sich der Wert der Zielfunktion beim Durchlaufen einer bestimmten Anzahl von aufeinanderfolgenden Temperaturschritten nicht mehr verändert.

5.2.2 Genetische Algorithmen

Genetische Algorithmen (GAs) sind ebenfalls ein sehr leistungsfähiges und allgemein einsetzbares Werkzeug zur Lösung vieler Optimierungsprobleme, für die analytische Lösungen entweder nicht existieren oder zu aufwendig zu berechnen sind. Inspiriert durch den natürlichen Evolutionsprozeß wurden sie Ende der sechziger Jahre hauptsächlich von John Holland in den USA entwickelt. In seinem 1975 erschienenen Hauptwerk [39] zeigte er auf, wie durch eine einfache Kodierung eines Problems mit Hilfe einer Menge von binären Vektoren und mit aus der Natur adaptierten Operatoren wie Mutation, Rekombination und Selektion eine Verbesserung von Lösungsvorschlägen erzielt werden kann. Wie auch das Simulated Annealing sind GAs in der Lage, lokale Extrema zu überwinden und zum globalen Optimum zu konvergieren.
In Pseudocode läßt sich ein genetischer Algorithmus wie folgt formulieren:

0. Wähle eine geeignete Kodierung der Chromosomen
1. Initialisiere zufällig eine Ausgangspopulation von Chromosomen
2. Bewerte alle Individuen der Ausgangspopulation mit Bewertungs− oder Fitneßfunktion

Wiederhole, bis Abbruchbedingung erreicht ist

3. Inkrementiere den Generationszähler
4. Selektiere Paare (oder größere Subpopulationen) gemäß Heiratsschema
5. Erzeuge mittels Rekombination (Crossover) innerhalb der selektierten Subpopulationen Nachkommen der aktuellen Generation
6. Mutiere die Nachkommen
7. Bewerte die Nachkommen
8. Erzeuge eine neue Generation durch Austausch von Individuen der aktuellen Generation gegen die Nachkommen gemäß Ersetzungsschema

Die einzelnen Elemente des Algorithmus werden im folgenden näher beleuchtet.

5.2.2.1 Die Kodierung der Chromosomen

Das erste Problem, mit dem man beim Einsatz eines GA konfrontiert wird, ist, eine geeignete Repräsentation der unabhängigen Variablen zur Anwendung der genetischen Operatoren zu wählen.

Fast immer werden dazu bei den GAs binäre Vektoren (Bitstrings), also Folgen von Nullen und Einsen verwendet. Oft müssen jedoch noch einige Besonderheiten beachtet werden, um zu befriedigenden Ergebnissen zu gelangen. Betrachtet man z. B. die Kodierung natürlicher Zahlen als Bitstrings, so stellt man fest, daß deren gewöhnliche Binärdarstellung einige Nachteile mit sich bringt. Das liegt vor allem daran, daß die einzelnen Bitpositionen bei einer Binärzahl unterschiedliches Gewicht haben. Während z. B. eine Mutation bei den tieferen Bits nur eine kleine Änderung des Zahlenwertes zur Folge hat, wirkt sich eine Mutation bei den höheren Bits viel stärker aus. Aus diesem Grund sind für die Zahlendarstellung spezielle Kodierungstechniken entwickelt worden.
Bei der Anwendung von genetischen Algorithmen auf QSAR−Probleme, wie sie in dieser Arbeit besprochen werden, spielt die Frage der Kodierung allerdings keine besondere Rolle, da sich die Repräsentation als Bitstrings direkt aus der Aufgabenstellung ohne die Notwendigkeit irgendwelcher Transformationen ergibt. Der GA muß ja nur entscheiden, ob ein Deskriptor in das aufzustellende Model als unabhängige Variable eingehen soll oder nicht. Daher besteht die Kodierung einfach darin, jeden Deskriptor mit einem Bit zu identifizieren, das je nachdem, ob es gesetzt oder nicht gesetzt ist, die An− oder Abwesenheit des Deskriptors im Modell anzeigt.

5.2.2.2 Die Fitneß− und Bewertungsfunktionen

Die Bewertungsfunktion legt das eigentliche Optimierungskriterium und −ziel fest. Sie ist damit ein unmittelbares Maß für die Güte der von dem GA erzeugten Chromosomen.
Während die meisten Optimierungsalgorithmen nur mit der Ziel− bzw. Bewertungsfunktion arbeiten, ist es bei den genetischen Algorithmen üblich, eine weitere Funktion zu verwenden, mit der die Chance jedes Chromosoms bestimmt wird, sich an der Erzeugung von Nachkommen zu beteiligen. Diese Funktion wird Fitneßfunktion genannt. Sie muß immer problemspezifisch gewählt werden, so daß sich keine allgemeingültigen Kriterien für gute Fitneßfunktionen angeben lassen. Oft wird einfach die Bewertungsfunktion selber oder eine einfache, davon abgeleitete Transformation als Fitneßfunktion eingesetzt.
Eine wichtige Rolle spielt die Fitneßfunktion bei der Beschleunigung der Konvergenz des Optimierungsprozesses. Mit ihr kann z. B. gesteuert werden, daß überproportional Chromosomen mit einer guten bis sehr guten Bewertung zur Reproduktion und damit Bildung der nächsten Generation herangezogen werden. Abzuwägen ist dagegen immer die Gefahr, bei einer solchen Strategie alternatives Genmaterial zu früh zu unterdrücken und so die Suche in Bereichen lokaler

Optima einfrieren zu lassen. Wenn der Genpool nämlich zu stark eingeschränkt wird, indem zu viele gleiche oder ähnliche Chromosomen zur Reproduktion herangezogen werden, verlieren die Crossover an Bedeutung, und Fortschritte sind fast nur noch durch glückliche Mutationen zu erzielen.
Desweiteren kann die Fitneßfunktion auch dazu dienen, Nebenbedingungen in das Optimierungsproblem einzuarbeiten, ohne den eigentlichen Algorithmus anpassen zu müssen. Beispielsweise können so bestimmte verbotene Lösungen ausgeschlossen werden, indem ihnen praktisch keine Chance zur Reproduktion eingeräumt wird.

5.2.2.3 Crossover−Verfahren

Den Crossover− oder Rekombinationsverfahren kommt bei den GAs eine besondere Bedeutung zu. Sie stellen dort die wichtigsten Operatoren im Optimierungsprozeß dar. So ist es nicht verwunderlich, daß bis heute eine nahezu unüberschaubare Fülle an Crossover−Schemata entwickelt worden ist.
Das einfachste, immer noch häufig eingesetzte Schema ist das *one−point−crossover* (Abbildung 18). Dabei werden zwei Chromosomen an dem gleichen, zufällig gewählten, Punkt getrennt. Danach werden die Bruchstücke zu zwei neuen Chromosomen rekombiniert.

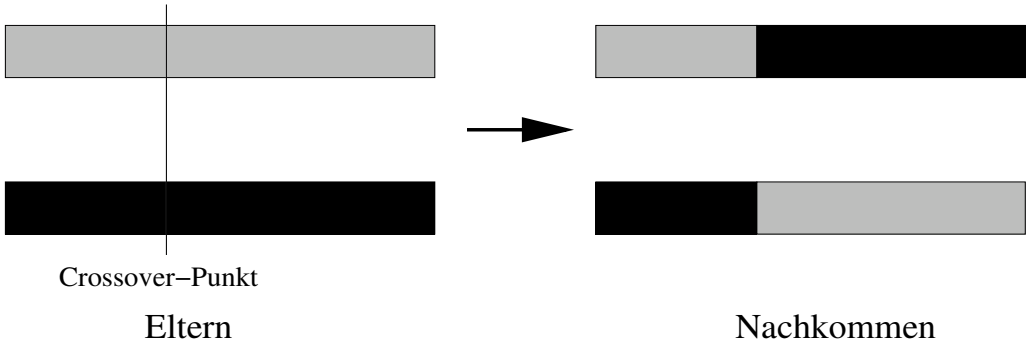

Abbildung 18: Beispiel für one−point−crossover

Ein für die meisten Probleme schon wesentlich effizienteres Crossover−Schema ist das *two−point−crossover* (Abbildung 19). Weil damit Bruchstücke zwischen zwei zufällig ausgewählten Punkten ausgetauscht werden, erreicht dieses Verfahren größere Veränderungen der Bitfolgen als das *one−point−crossover* und damit in der Regel eine schnellere Konvergenz des GA. Dieses Schema läßt sich für beliebig viele Crossover−Punkte erweitern. Allgemein spricht man dann von einem *n−point−crossover*, mit n = (1, 2, 3, 4, ...).

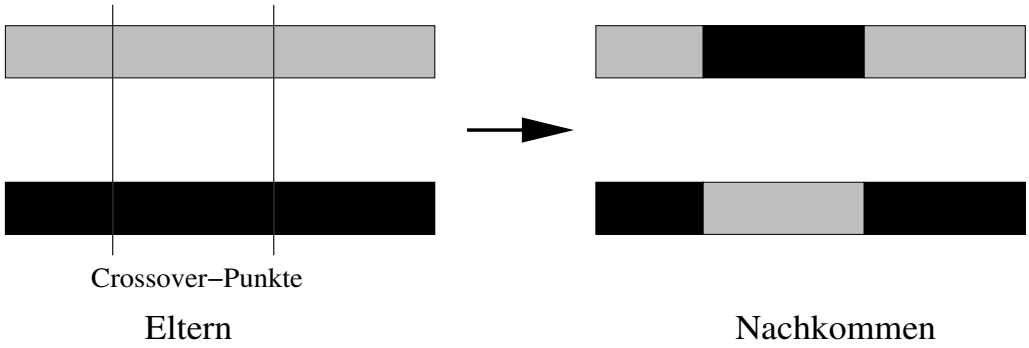

Abbildung 19: Beispiel für two−point−crossover

Noch weiter geht ein Schema, das die Crossover−Punkte allein mit einer zufällig bestimmten Schablone festlegt, das sogenannte *uniform−crossover* (Abbildung 20). Dabei ergeben sich im Mittel L / 2 Crossover−Punkte (L ≙ Chromosomenlänge). Dieses Verfahren bietet prinzipiell die Chance, in einem Schritt jeden theoretisch möglichen Austausch von Bitsequenzen zwischen zwei Chromosomen zu erreichen.

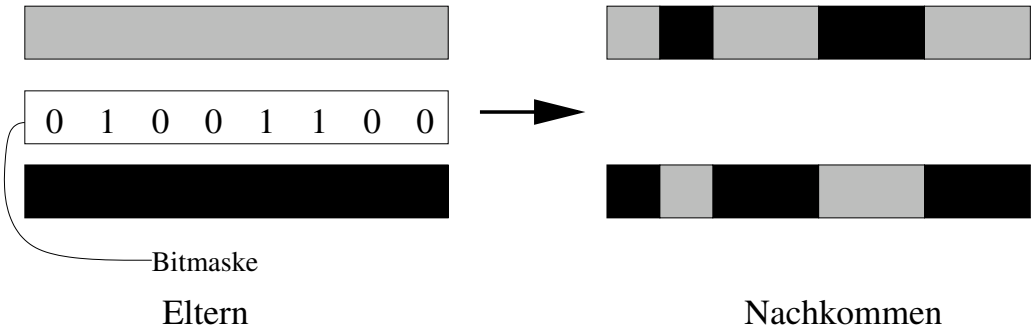

Abbildung 20: Beispiel für uniform−crossover

Die Vorteile von *uniform−crossover* und *n−point−crossover* mit großem n können natürlich auch ein Problem darstellen, weil dabei leicht viele vorteilhafte Bitsequenzen zerschlagen werden können. Insofern muß für jeden Anwendungsfall einzeln geprüft werden, ob nicht ein *two−point−crossover* schneller zum Ziel führt.

5.2.2.4 Mutationen

Die Mutationen spielen bei den GAs gegenüber den Crossover−Verfahren nur eine untergeordnete, allerdings keinesfalls unwichtige Rolle im Optimierungsprozeß. Sie dienen in erster Linie dazu, eine zu frühzeitige Konvergenz zu verhindern, indem sie für eine gewisse Inhomogenität in den

Populationen sorgen. Das ist erforderlich, weil der Selektionsdruck im Verlauf der Optimierung durch die bevorzugte Reproduktion hoch bewerteter Individuen zu immer homogeneren Populationen führt, in denen sich in den einzelnen Chromosomen mehr und mehr die gleichen Bitsequenzen finden lassen.

Auf der Ebene der Chromosomen bewirken die Mutationen die Invertierung einzelner Bits. Die meistverwendeten Verfahren unterscheiden sich lediglich hinsichtlich der Wahrscheinlichkeiten, mit denen die einzelnen Bitpositionen von einer solchen Veränderung betroffen sind. Je nach Art des vorliegenden Problems können für alle Bitpositionen entweder die gleichen oder aber individuell unterschiedliche Mutationswahrscheinlichkeiten gewählt werden. Entsprechend der untergeordneten Bedeutung der Mutationen bei den GAs wird normalerweise mit nur geringen Mutationsraten gearbeitet. In einigen Fällen kann es aber für die Konvergenzgeschwindigkeit durchaus vorteilhaft sein, mehr Mutationen zuzulassen und so einen weiteren aktiven Suchoperator im Einsatz zu haben. Als Problem ist dabei zu sehen, daß mit steigender Mutationsrate sehr viele vorteilhafte Bitsequenzen aus einer Population zerstört werden können.

5.2.2.5 Heiratsschemata

Das Heiratsschema legt fest, welche Chromosomen einer Population zur Rekombination und somit zur Erzeugung einer neuen Generation herangezogen werden. Die übliche Strategie besteht darin, einzelne Chromosomenpaare auszuwählen, aus denen wiederum neue Chromosomenpaare erzeugt werden. Das älteste und bedeutendste Schema, welches dies leistet, ist das *Roulette Wheel*−Verfahren. Damit werden die einzelnen Individuen proportional zu ihrer Fitneß zur Rekombination ausgewählt. Veranschaulicht man sich das Prinzip mit Hilfe eines Roulette−Rades (Abbildung 21), so bekommt jeder Kandidat auf dem Rad ein Segment zugewiesen, das in der Größe seinem Anteil an der Summe aller Fitneßwerte entspricht. So können bei diesem Schema durchaus auch Individuen mit geringer Fitneß zur Bildung einer neuen Generation herangezogen werden. Allerdings ist die Wahrscheinlichkeit dafür entsprechend gering.

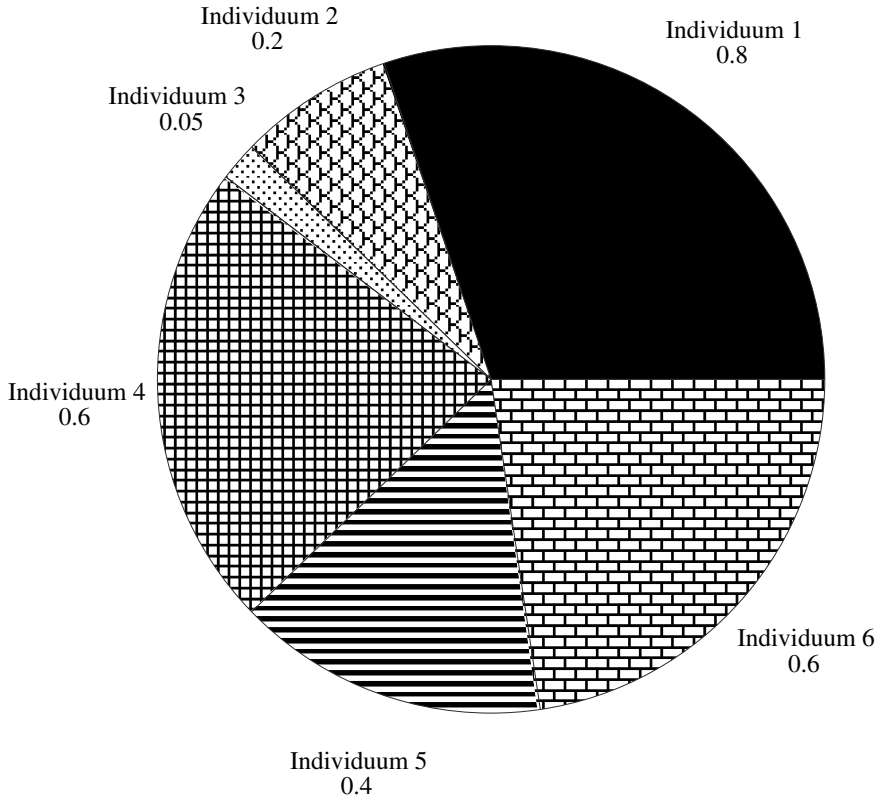

Abbildung 21: Veranschaulichung des Roulette Wheel−Verfahrens am Beispiel von sechs Individuen

5.2.2.6 Ersetzungsschemata

Während das Heiratsschema die Kandidaten für die Reproduktion auswählt, legt das Ersetzungsschema fest, wie die Individuen der Elterngeneration zugunsten der Nachkommen ausgetauscht werden. Die einfachste Form der Ersetzung ist das sogenannte *Generational Replacement*. Bei Anwendung dieses Schemas wird die Elterngeneration einfach vollständig durch die Generation der Nachkommen ersetzt. Der Nachteil ist, daß dabei hoch bewertete Chromosomen verloren gehen können und auch die durchschnittliche Fitneß der neuen Generation unter der der alten liegen kann. Andererseits kann so die Dominanz einiger weniger guter Elemente der Ausgangspopulation durchbrochen und eine zu frühzeitige Konvergenz verhindert werden.
Um die Nachteile des *Generational Replacements* zu umgehen und sicherzustellen, daß die am besten bewerteten Individuen in die Nachfolgegeneration übernommen werden, wurden verschiedene Ersetzungsschemata entwickelt, die mit dem Prinzip der Eliten arbeiten. Deren

Grundidee ist, daß eine bestimmte Anzahl der besten Individuen (Eliten) einer Generation entweder direkt oder mutiert in die Nachfolgegeneration übernommen werden.

5.3 Design des QSAR−Programms

Das hier entwickelte Programm *MultReg* wurde in der Programmiersprache Java geschrieben. Es bietet als Modellierungsverfahren KNN und PLS an. Beide können wahlweise sowohl mit Simulated Annealing als auch mit einem GA optimiert werden. Diese Kombinationsmöglichkeiten wurden beim objektorientierten Entwurf des Programms berücksichtigt. So wurde die abstrakte Klasse *Model* zur Repräsentation des QSAR−Modells entworfen. Als Subklassen wurden die Klassen *KNN* und *PLS* entwickelt, die eine konkrete Implementierung der beiden Algorithmen enthalten. Entsprechend wurde die abstrakte Basisklasse *VarSelector* als Grundlage der Optimierungsmethoden konzipiert. Die Subklassen zur Realisierung der Funktionen heißen hier *SA* und *GA*. Diagramme dieser Klassen in UML−Notation (UML: Unified Modeling Language) finden sich in Abbildung 22 bzw. Abbildung 23.

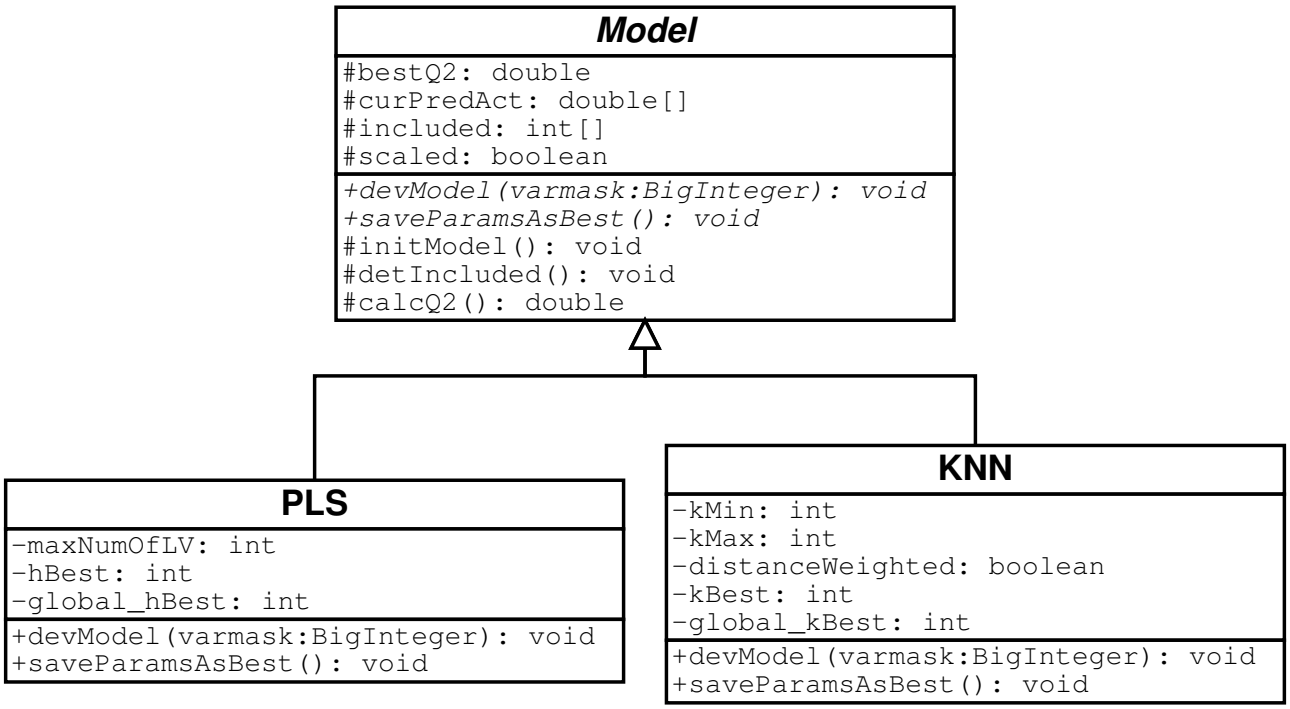

Abbildung 22: Klassendiagramm zur Implementierung der Modellverfahren

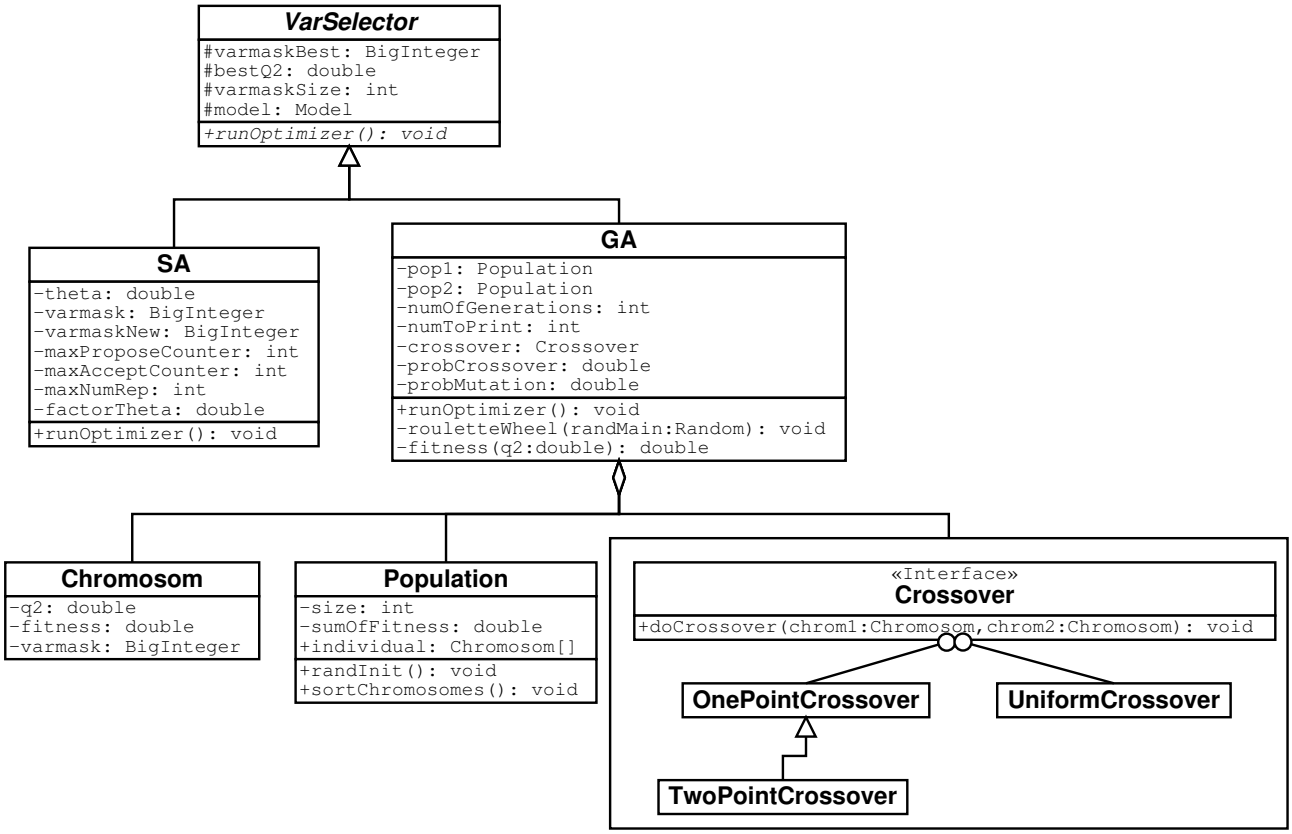

Abbildung 23: Klassendiagramm zur Implementierung der Variablenselektionsmethoden

MultReg erzeugt ein möglichst optimales QSAR−Modell durch Variablenselektion auf der Basis von KNN oder PLS. Als einziges Optimalitätskriterium wird der q^2−Wert herangezogen, der durch Kreuzvalidierung mit Hilfe der Leave−one−out−Methode bestimmt wird. Die Deskriptoren des Trainingsdatensatzes können zu Beginn einer Rechnung autoskaliert werden. Jede Variablenkombination wird intern durch ein *java.math.BigInteger*−Objekt repräsentiert. Die Entscheidung für diese Repräsentation fiel aufgrund der vielen nützlichen Methoden, die die Klasse *BigInteger* zur Verfügung stellt. So existieren z. B. Konstruktoren, um ein *BigInteger*−Objekt mit einer zufälligen Bitfolge zu initialisieren. Die Kontrolle, ob ein bestimmtes Bit gesetzt bzw. ein Deskriptor in der dargestellten Variablenkombination enthalten ist, kann mit der Methode *testBit()* erfolgen. Zum Invertieren einzelner Bits (Mutationen) kann die *flipBit()*−Methode eingesetzt werden.

Die oben dargestellten UML−Diagramme sollen einen Eindruck von der Programmstruktur vermitteln. Sie zeigen nur die wichtigsten Methoden und Instanzvariablen, die bei einem Programmlauf zur Aufstellung eines QSAR−Modells eine Rolle spielen. Anhand der Diagramme wird nun erläutert, welche Verfahren implementiert und welche Optionen vom Benutzer wählbar sind.

Die Methode *devModel()* steht im Zentrum der Modellgenerierung. Sie stellt zu einer ihr übergebenen Variablenkombination das entsprechende, optimale PLS− bzw. KNN−Modell auf und berechnet damit einen Satz von durch Kreuzvalidierung erhaltenen Vorhersagen, der in dem Feld *curPredAct* abgelegt wird. Damit läßt sich dann über die *calcQ2()*−Methode ein q^2−Wert berechnen. Das Feld *included* gibt während der Kreuzvalidierung an, welche Verbindungen in einem Durchgang herausgelassen werden. Es wird von der *detIncluded()*−Methode belegt, die zur Zeit nur die LOO−Strategie realisiert.

Die eingestellten Optionen bzw. Programmparameter werden von privaten Variablen in den einzelnen Klassen registriert. So zeigt die Variable *scaled* an, ob eine Autoskalierung der Deskriptoren vorgenommen werden soll. Beim PLS−Algorithmus kann der Benutzer die maximale Anzahl zu verwendender latenter Variablen (*maxNumOfLV*) festlegen, beim KNN−Verfahren die minimalen und maximalen Werte von k (*kMin* bzw. *kMax*).

In dem hier vorgestellten Programm wurde eine bisher noch nicht beschriebene, neue Variante der KNN−Methode implementiert. Die Idee dahinter ist, die Aktivität einer Verbindung nicht einfach als Mittelwert der Aktivitäten der k nächsten Nachbarn zu berechnen, sondern bei der Mittelwertbildung die Nachbarn gemäß ihrem euklidischen Abstand zur Zielverbindung zu wichten. So erhält ein Objekt bei einer Vorhersage um so mehr Einfluß, je näher es dem Zielobjekt benachbart ist. Ob diese neue oder die Originalvariante benutzt wird, steht in der Variable *distanceWeighted*.

Instanzen der Klassen *SA* oder *GA* steuern den Optimierungslauf. Mit der Variable *model* haben sie eine Zugriffsmöglichkeit (Referenz) auf das für die Modellgenerierung zuständige *KNN*− oder *PLS*−Objekt. Beim Simulated Annealing wird der Ablauf über mehrere Parameter festgelegt. Die maximalen Anzahlen erzeugter und akzeptierter neuer Vorschläge (*maxProposeCounter*, *maxAcceptCounter*) bestimmen, wie lange es dauern kann, bis zur nächsttieferen Temperatur gesprungen wird. Als Abbruchbedingung wird eine maximale Anzahl aufeinanderfolgender Temperaturschritte (*maxNumRep*), in denen die Lösung nicht verbessert werden konnte, verwendet. Außerdem kann der Benutzer den Faktor α (*factorTheta*) wählen, mit dem der aktuelle Temperaturwert beim Abkühlen multipliziert wird. Die im SA−Algorithmus erwähnte kleine Änderung zum Generieren eines neuen Lösungsvorschlags besteht in nichts anderem als dem Umkippen eines Bits in der Bitstringdarstellung der Deskriptorvariablenmaske und ist somit vergleichbar mit einer Punktmutation bei den GAs.

Der implementierte GA arbeitet mit keiner wirklichen Abbruchbedingung, sondern hält nach einer festgelegten Anzahl von Generationen (*numOfGenerations*) an. Weitere Parameter sind das

einzusetzende Rekombinationsverfahren (*crossover*) sowie die Crossover− und Mutationswahrscheinlichkeiten (*probCrossover* bzw. *probMutation*). Es stehen drei verschiedene Rekombinationsverfahren zur Auswahl, das one−point−crossover, das two−point−crossover und das uniform−crossover. Diese sind in eigenen Klassen innerhalb von *GA* definiert und implementieren das Interface *Crossover.* Wie man aus dem Klassendiagramm entnehmen kann, ist die Klasse *TwoPointCrossover* als Subklasse von *OnePointCrossover* realisiert. Entsprechend der gewählten Option wird eine der Klassen instanziiert und von der Variable *crossover* referenziert. Der Struktur eines GA wird noch durch zwei weitere, innerhalb von *GA* definierte, Klassen Rechnung getragen. Die Klasse *Chromosom* definiert ein Individuum über eine Variablenkombination (*varmask*), eine Bewertung (*q2*) und einen Fitneßwert (*fitness*). In der Klasse *Population* sind die Individuen einer Population im Feld *individual* zusammengefaßt. Die vom Benutzer wählbare Populationsgröße ist in der Variable *size* angegeben, die Summe der Fitneßwerte der enthaltenen Individuen wird nach erfolgter Evaluierung in der Variable *sumOfFitness* abgelegt. Zur zufälligen Initialisierung einer Population stellt die Klasse die Methode *randInit()* zur Verfügung und zur Sortierung ihrer Individuen in der Reihenfolge der Fitneßwerte die Methode *sortChromosomes()*. Die verwendete Fitneßfunktion (*fitness()*) bildet die Bewertungen (q^2−Werte) linear auf den Bereich [0, 1] ab:

$$fitness = \frac{q^2 - q^2_{min}}{1 - q^2_{min}} \quad ; \quad q^2_{min} \;\hat{=}\; \text{kleinster } q^2\text{−Wert der Population}$$

Als einziges Heiratsschema des GA ist das *Roulette Wheel*−Verfahren implementiert (Methode *rouletteWheel()*), als einziges Ersetzungsschema das *Generational Replacement*. Damit dennoch gesichert ist, daß die beste aller evaluierten Variablenkombinationen am Ende eines Optimierungslaufs auch dann noch verfügbar ist, wenn sie durch Crossover oder Mutationen wieder zerstört wird, hebt *MultReg* die beste bis zum jeweiligen Zeitpunkt gefundene Lösung gesondert auf. Das gleiche gilt auch für die SA−Implementierung, obwohl der oben dazu angegebene Algorithmus dies nicht vorsieht.

5.4 Die grafische Benutzeroberfläche für das QSAR−Programm

Da *MultReg* als kommandozeilenorientiertes Programm zum einfachen Einsatz im Batch−Betrieb konzipiert ist und somit keine grafische Benutzeroberfläche mitbringt, wurde eine solche in der

Programmiersprache Tcl/Tk entworfen. Dieses Programm (*multreg*) erleichtert nicht nur die Bedienung, da der Benutzer nicht alle Kommandozeilenschalter kennen muß, sondern stellt darüber hinaus auch noch eine Funktion zur Datenvorbehandlung zur Verfügung.
Der Hauptbildschirm (Abbildung 24) ist unter dem Gesichtspunkt der Gegenüberstellung von Modell und Optimierungsalgorithmus gestaltet.

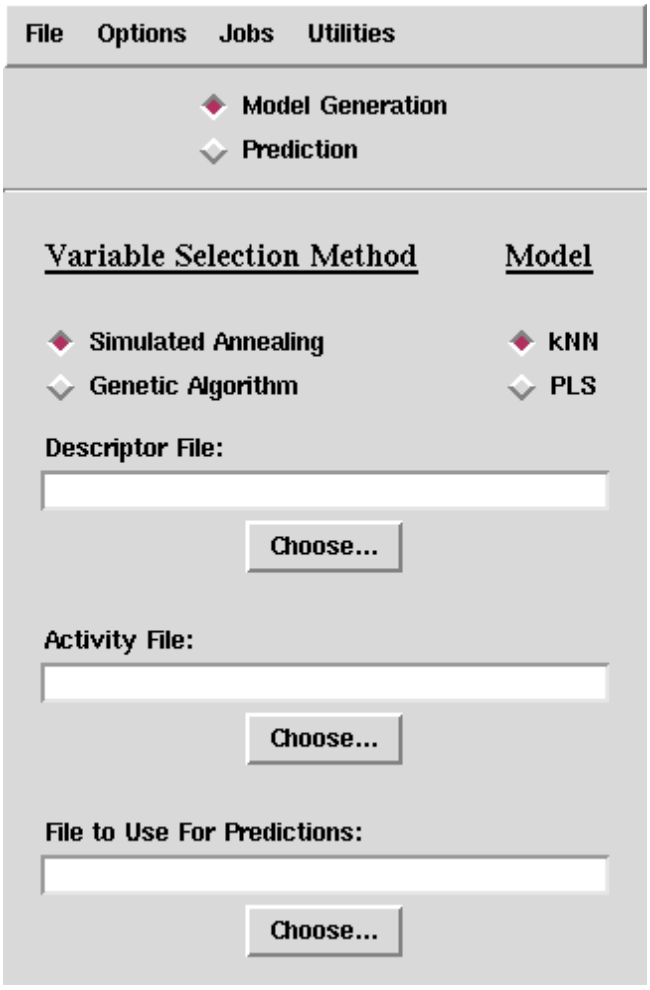

Abbildung 24: Der Hauptbildschirm der grafischen Benutzeroberfläche

Hier kann der Benutzer wählen, welche der Verfahren er einsetzen will.
Die Einstellung der einzelnen Optionen erfolgt in Dialogen, die über das „Options"−Menü zugänglich sind. Alle vorgenommenen Einstellungen können auch in einer Datei abgespeichert und bei Bedarf wieder eingeladen werden.

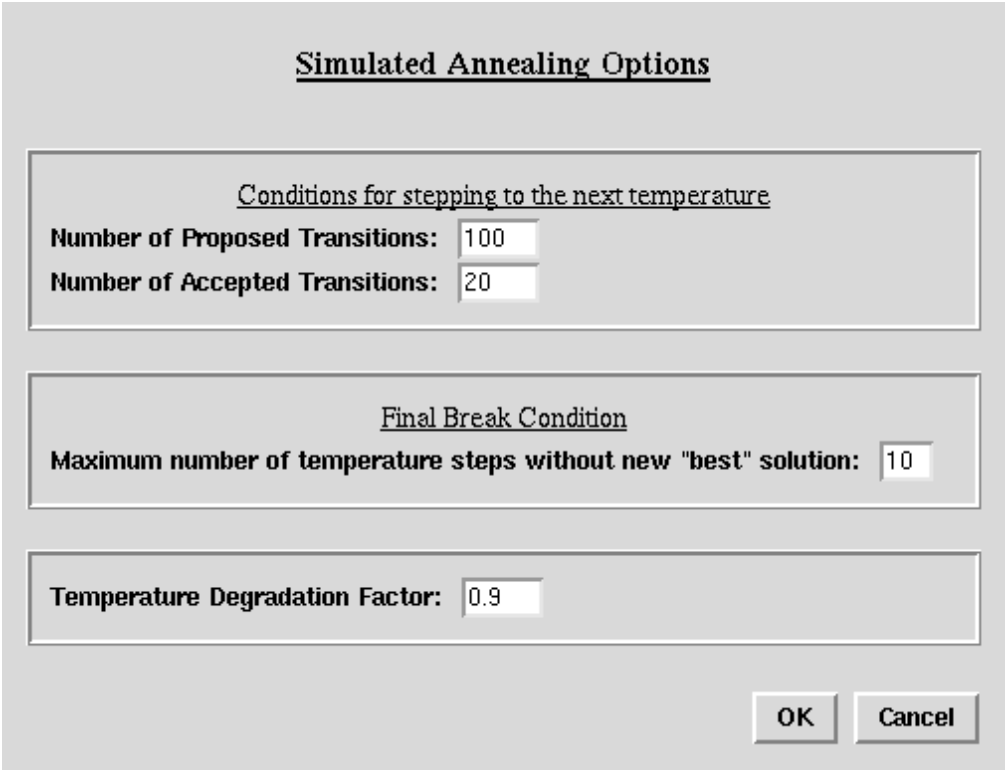

Abbildung 25: Dialog zur Einstellung der SA−Optionen

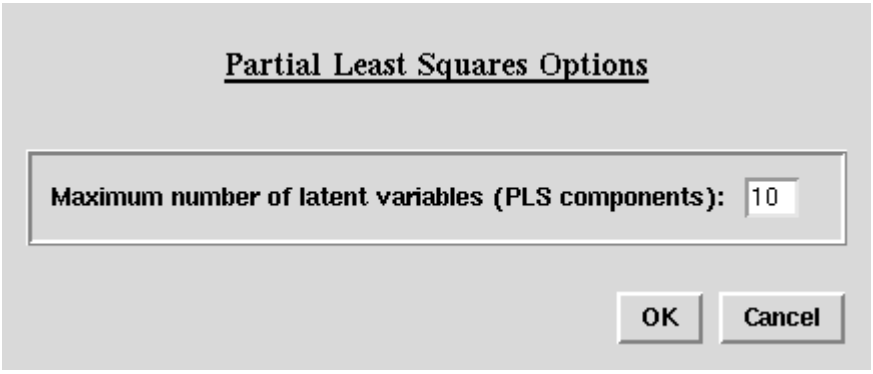

Abbildung 26: Dialog zur Einstellung der PLS−Optionen

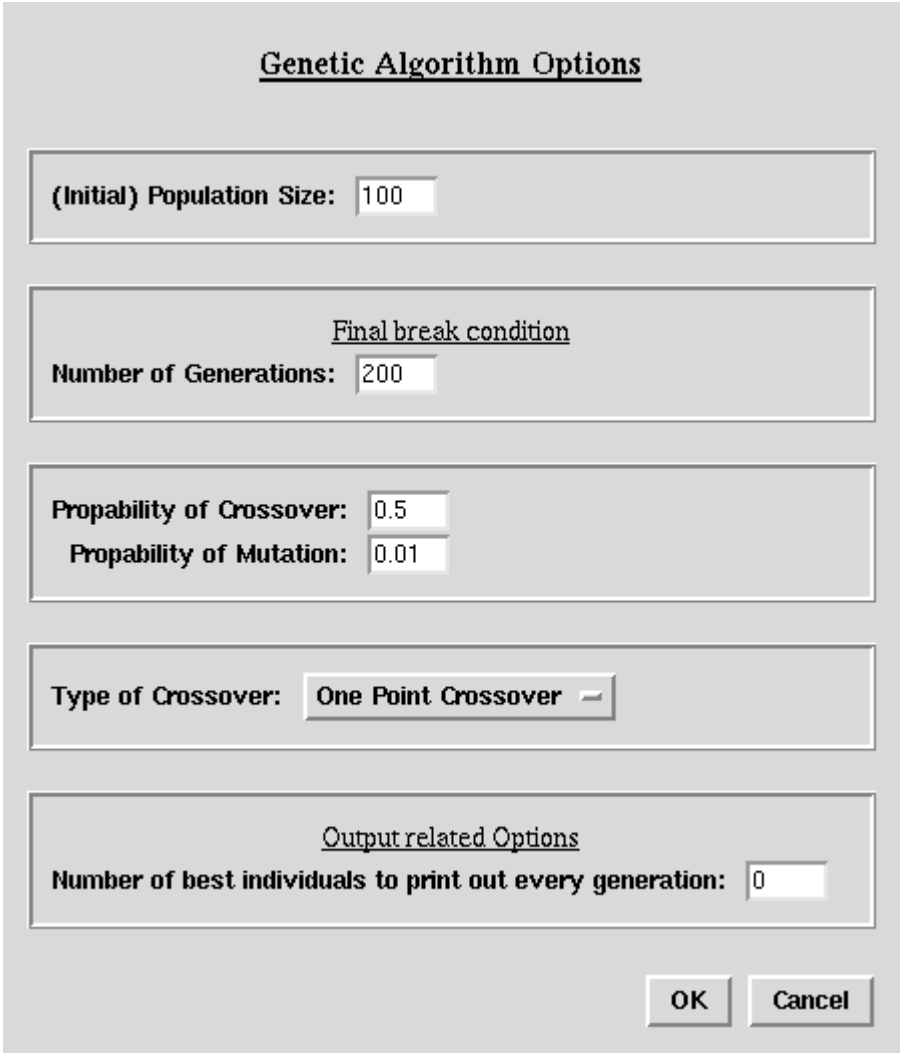

Abbildung 27: Dialog zur Einstellung der GA−Optionen

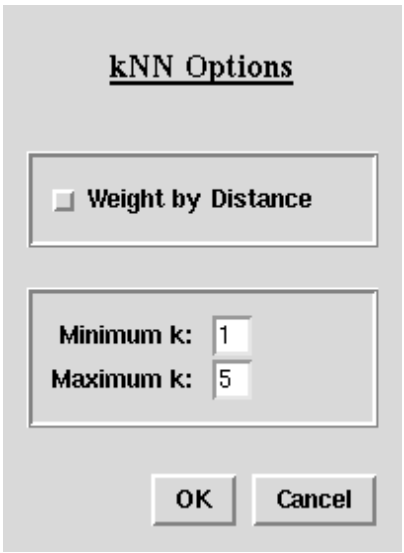

Abbildung 28: Dialog zur Einstellung der KNN−Optionen

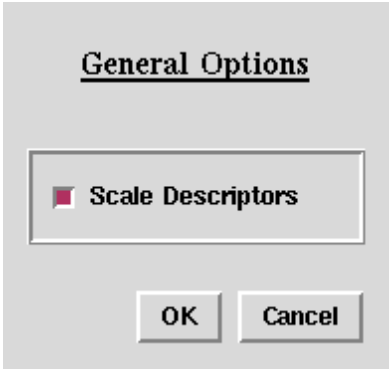

Abbildung 29: Dialog zur Einstellung allgemeiner Programmparameter

Über das „Jobs"−Menü kann eine Rechnung auf zwei verschiedene Arten gestartet werden. Wenn der Benutzer den Programmverlauf verfolgen möchte, wird ein Fenster geöffnet, in dem die Ausgaben von *MultReg* angezeigt werden. Bei nicht zufriedenstellenden Ergebnissen kann eine laufende Rechnung von hier aus abgebrochen werden. Am Ende eines erfolgreich durchgeführten Programmlaufs besteht die Möglichkeit, die im Fenster zu sehenden Ausgaben in einer Datei abzuspeichern.

Soll eine Rechnung im Hintergrund gestartet werden, wählt der Benutzer den Menüpunkt „Run in background..." an, worauf über eine Auswahlbox nach einer Datei gefragt wird, in der die Ausgaben von *MultReg* gesichert werden sollen. Die auf diese Weise initiierten Rechnungen können in einem Übersichtsfenster (Abbildung 30) angezeigt und auf Wunsch abgebrochen werden.

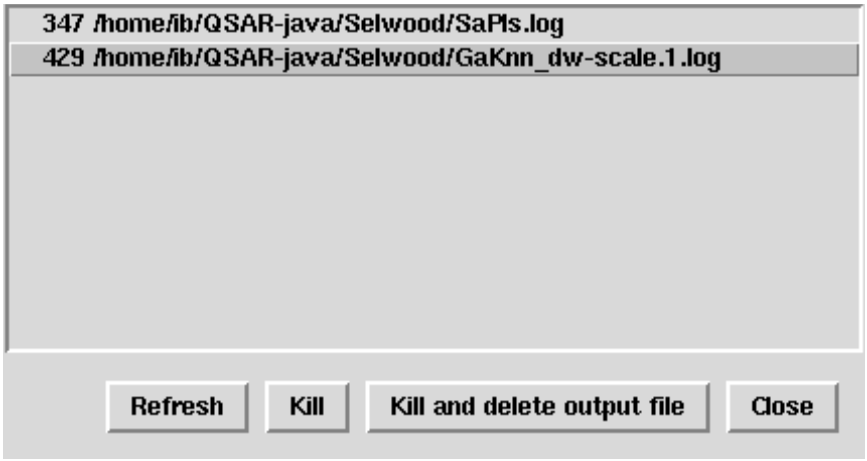

Abbildung 30: Übersichtsfenster zur Kontrolle der im Hintergrund laufenden Rechnungen

Das „Utilities"−Menü enthält zwei Unterpunkte. Der erste bietet die Möglichkeit, Datensätze aus dem Golpe/Simca−Format in das von *MultReg* verwendete Format zu konvertieren. Dieses besteht aus zwei Dateien, eine mit den Aktivitäten und eine mit den Deskriptoren.. In der Datei, die die

Aktivitäten enthält, steht in der ersten Zeile die Anzahl der Verbindungen. Danach folgt für jede Verbindung eine Zeile mit Name und Aktivität der Verbindung. Die Deskriptordatei beginnt ebenfalls mit der Anzahl der Verbindungen. In der zweiten Zeile stehen dann die Namen der Deskriptoren und in den weiteren Zeilen jeweils ein Verbindungsname gefolgt von den zugehörigen Deskriptorwerten. Hinter dem zweiten Unterpunkt verbirgt sich eine wichtige Option zur Datenvorbehandlung. Damit können aus einem Trainingsdatensatz Deskriptoren entfernt werden, die eine Varianz unter einem vom Benutzer zu bestimmenden Schwellenwert besitzen (Abbildung 31).

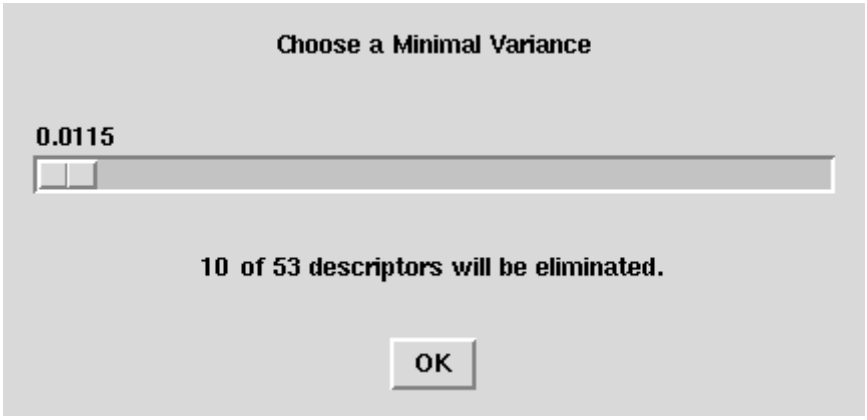

Abbildung 31: Dialogfenster zur Eliminierung von Deskriptoren mit geringer Varianz

Dies ist eine wichtige Möglichkeit, um bei sehr großen Datensätzen, wie sie z. B. bei der CoMFA−Methode erhalten werden, die Anzahl der Deskriptoren auf ein solches Maß zu reduzieren, daß die Rechenzeiten erträglich werden. Zumindest die Deskriptoren mit einer Varianz von null sollten so aus jedem Trainingsdatensatz eliminiert werden, da sie keinerlei relevante Informationen für das aufzustellende Modell beinhalten und außerdem bei der Rechnung stören.

5.5 Anwendung auf einen Beispieldatensatz

Um die Funktionsfähigkeit des erstellten Programms zu verifizieren, wurden einige Tests mit einem 1990 von Selwood veröffentlichten Datensatz [40] durchgeführt. Dieser Datensatz wurde ausgewählt, weil er bereits von mehreren Gruppen mit unterschiedlichen Methoden untersucht worden ist. Er beinhaltet 31 Verbindungen und 53 Deskriptoren und ist damit zu umfangreich für eine systematische Evaluierung aller möglichen Variablenkombinationen.
Es wurden Programmläufe für alle drei in *MultReg* implementierten Methoden, KNN, KNN mit

Abstandswichtung und PLS, jeweils ohne und mit Autoskalierung der Deskriptoren durchgeführt, so daß sechs verschiedene QSAR−Modelle aufgestellt werden konnten. Die Ergebnisse dieser Berechnungen sind in der folgenden Tabelle zusammengefaßt:

	q^2	**k bzw. Anzahl der latenten Variablen**	**Anzahl der Deskriptoren**
KNN ohne Abstandswichtung, unskaliert[1)]	0,250	3	20
KNN ohne Abstandswichtung, skaliert[2)]	0,719	1	24
KNN mit Abstandswichtung, unskaliert[3)]	0,299	10	21
KNN mit Abstandswichtung, skaliert[4)]	0,758	3	17
PLS unskaliert[5)]	0,844	13	16
PLS skaliert[6)]	0,849	13	16

1) $k_{min} = 1$, $k_{max} = 15$, Optimierungsmethode: GA

2) $k_{min} = 1$, $k_{max} = 10$, Optimierungsmethode: GA

3) $k_{min} = 1$, $k_{max} = 10$, Optimierungsmethode: GA

4) $k_{min} = 1$, $k_{max} = 10$, Optimierungsmethode: GA

5) maximale Anzahl latenter Variablen: 15, Optimierungsmethode: SA

6) maximale Anzahl latenter Variablen: 15, Optimierungsmethode: SA

Es muß betont werden, daß die aufgeführten Ergebnisse nicht das Optimum des Erreichbaren sein müssen, obwohl mehrere Läufe durchgeführt worden sind. Dazu ist die Zahl der möglichen Variablenkombinationen ($2^{53} - 1$ bei 53 Deskriptoren) einfach zu groß. Allerdings lassen sich mit einiger Sicherheit mehrere Trends erkennen. Am auffälligsten sind die niedrigen q^2−Werte der aufgestellten KNN−Modelle ohne Autoskalierung der Deskriptoren. Dieses Ergebnis ist darauf zurückzuführen, daß im Datensatz Deskriptoren mit Zahlenwerten ganz unterschiedlicher Größenordnung enthalten sind. Es ist leicht einzusehen, daß die KNN−Methode in solch einem Fall ungeeignet ist, da die euklidischen Abstände der Verbindungen und damit auch die Vorhersagen dann fast ausschließlich von den betragsmäßig größten Deskriptoren abhängen. Werden die Werte vorher skaliert, ergeben sich aber mit der KNN−Methode durchaus brauchbare Modelle. Im Gegensatz dazu ist mit der PLS−Methode bei diesem Datensatz kein eindeutiger Effekt der

Autoskalierung erkennbar.

Die erhaltenen PLS−Modelle liefern insgesamt wesentlich bessere q^2−Werte als das beste KNN−Modell. Wie wichtig die Variablenselektion ist wird deutlich, wenn man den q^2−Wert des PLS−Modells ohne Skalierung von 0,844 mit dem q^2−Wert des besten PLS−Modells bei Verwendung aller 53 Deskriptoren vergleicht. Dieser liegt nach Kubinyi [41] bei 0,279 (5 latente Variablen) und ist somit ein Indiz für die Empfindlichkeit der PLS−Methode gegenüber redundanten Deskriptoren.

Die Frage nach der Zweckmäßigkeit der Abstandswichtung bei der KNN−Methode kann nach den oben angeführten Ergebnissen bedingt positiv beantwortet werden. Die q^2−Werte sind jeweils etwas besser, als die ohne Abstandswichtung erhaltenen. Das ist plausibel, wenn man die Grundidee hinter der KNN−Methode betrachtet, daß ähnliche Verbindungen auch ähnliche Aktivitäten besitzen sollten. Erweitert man dieses Prinzip, so läßt sich sagen, daß die Aktivitäten um so ähnlicher sein sollten, je ähnlicher die Verbindungen sind. Die Abstandswichtung sollte sich um so positiver auf die Vorhersagen auswirken, je kleiner der Trainingsdatensatz ist. Bei umfangreichen Datensätzen hat die Verbindung, deren Aktivität vorhergesagt werden soll, in der Regel mehrere eng verwandte Nachbarn, so daß die einfache Mittelwertbildung eine sehr gute Abschätzung liefern sollte, sofern eine ausreichende Korrelation zwischen Deskriptoren und Aktivität besteht. Hingegen ist bei kleinen Datensätzen die Nachbarschaft der meisten Verbindung weniger dicht „besiedelt“, so daß eine genauere Betrachtung der Abstände unverzichtbar ist, wenn man vermeiden möchte, daß im Deskriptorraum weit entfernte nächste Nachbarn die Vorhersagen genauso stark beeinflussen wie sehr nahe Nachbarn. Das bedeutet, daß bei der Originalvariante die Berücksichtigung von mehr als nur ganz wenigen, im Extremfall nur einem, Nachbarn die Vorhersagen u. U. dramatisch verschlechtern kann, während bei der abstandsgewichteten Variante auch weiter entfernte Nachbarn einen angemessenen Beitrag zur Vorhersage liefern können. Aus dieser Betrachtung ergibt sich die Schlußfolgerung, daß bei der abstandsgewichteten Methode der Parameter k, bei dem die besten Vorhersagen erzielt werden, normalerweise größer als bei der Originalmethode sein wird. Dies wird durch die erhaltenen Ergebnisse auch bestätigt (unskaliert: 10 gegenüber 3, skaliert: 3 gegenüber 1). Zu beachten ist, daß sich bei k = 1 kein Unterschied zwischen den beiden Varianten ergibt.

Zur Veranschaulichung sind in den folgenden Diagrammen die mit den aufgestellten Modellen vorhergesagten −log EC_{50}−Werte gegen die experimentellen aufgetragen (EC_{50} ≙ effektive Konzentration, bei der sich eine gemessene Wirkung zu 50% eingestellt hat).

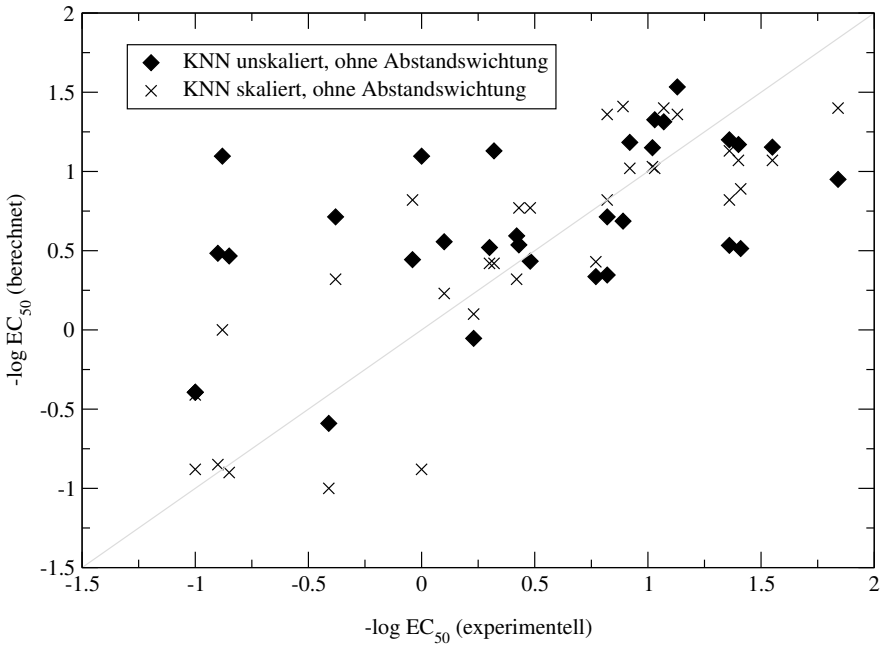

Abbildung 32: Ergebnisse aus dem KNN−Modell ohne Abstandswichtung

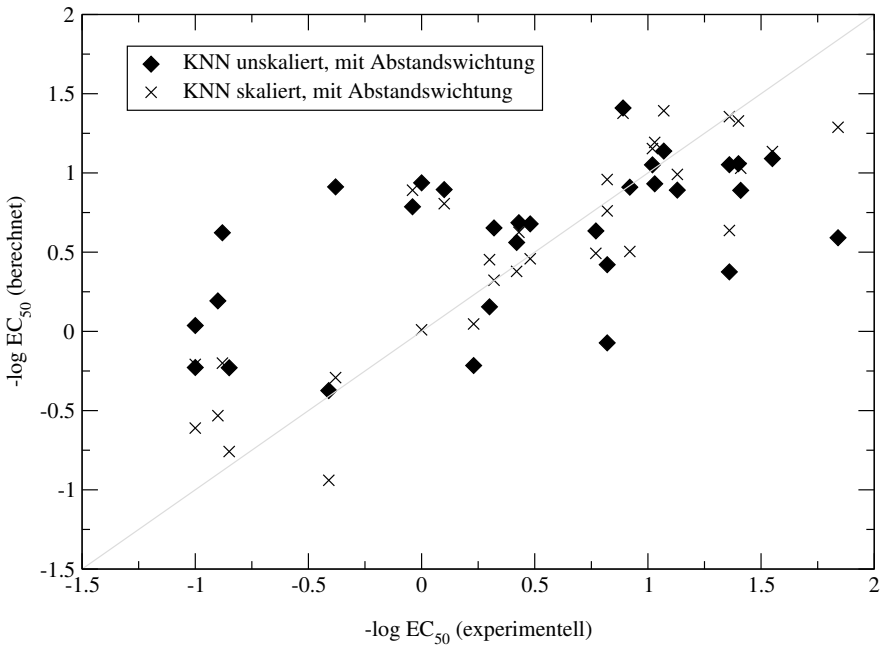

Abbildung 33: Ergebnisse aus dem KNN−Modell mit Abstandswichtung

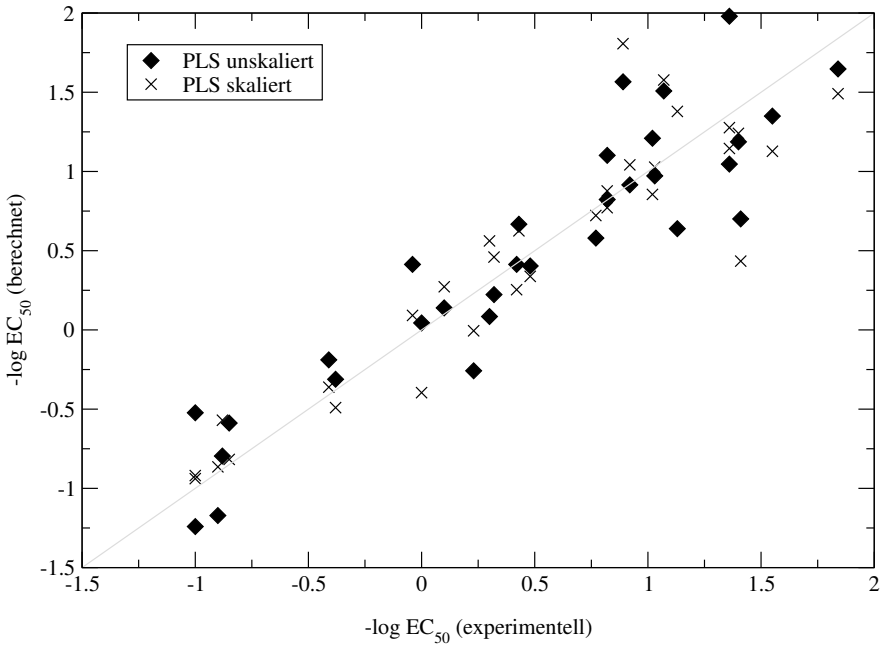

Abbildung 34: Ergebnisse aus dem PLS−Modell

Aus den Abständen zur 50°−Linie läßt sich die Güte der Vorhersagen ablesen. So schlagen sich z. B. die niedrigen q^2−Werte für die KNN−Modelle ohne Skalierung in einer breiten Streuung um diese Linie nieder.

6 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurden Verfahren und Computerprogramme zur Visualisierung chemischer Reaktionen und zur Aufstellung quantitativer Struktur–Wirkungsbeziehungen (QSAR) entwickelt.

Die Visualisierung chemischer Reaktionen in Form von Computeranimationen soll ein besseres Verständnis der Abläufe von Reaktionsmechanismen auf molekularer Ebene ermöglichen. Entwickelt wurde ein im Internet abrufbares Programmsystem (*CAVOC*), mit dem vorberechnete Reaktionen in einer dreidimensionalen Ball&Stick–Darstellung betrachtet werden können. Durch das Hinzufügen von Isoflächen physikalisch–chemischer Eigenschaften (Orbitale, Elektronen– und Spindichten, elektrostatische Potentiale) kann die Bedeutung dieser Größen für den betrachteten Reaktionsablauf beurteilt werden.
Obwohl die Liste der eingebauten Reaktionen schon sehr umfangreich ist, sollten zukünftig noch weitaus mehr Beispiele chemischer Grundreaktionen aufgenommen werden. Ein weiteres Ziel sollte die Öffnung des Visualisierungssytems für extern berechnete Reaktionen sein. Momentan läuft an der Universität Paderborn ein Projekt in Kooperation mit dem Centre for Molecular and Biomolecular Informatics in Nijmegen (Niederlande), das ein CGI–Interface zur interaktiven Berechnung nahezu beliebiger Reaktionen über das Internet bereitstellt. Zur Visualisierung der damit erhaltenen Ergebnisse soll eine Schnittstelle in *CAVOC* implementiert werden.

QSAR–Modelle (Quantitative Structure Activity Relationships) spielen im modernen Wirkstoffdesign eine bedeutende Rolle. Daher wurde ein modulares Programm zur Aufstellung solcher Struktur–Wirkungsbeziehungen entwickelt, das die Aktivitäten chemischer Strukturen mit sogenannten Deskriptoren korreliert. Als mathematische Modelle kommen PLS (Partial Least Squares) und die Methode der k nächsten Nachbarn (KNN) zum Einsatz. Die Optimierung der Modelle durch Variablenselektion in Hinblick auf eine möglichst hohe Vorhersagekraft (bestimmt durch das Leave–one–out–Verfahren) erfolgt mit Genetischen Algorithmen und Simulated Annealing.
Das entwickelte Programm konnte für einen Beispieldatensatz aus der Literatur zufriedenstellende Ergebnisse erzielen. Dabei wurde auch eine neue Variante der KNN–Methode untersucht, die leichte Verbesserungen gegenüber der Originalvariante erkennen läßt. Für ein abschließendes Urteil müssen jedoch noch weitere Tests mit anderen Datensätzen durchgeführt werden.

Literaturverzeichnis

[1] M. J. S. Dewar, W. Thiel "Ground States of Molecules, 38. The MNDO Method. Approximations and Parameters" *Journal of the American Chemical Society* **1977**, 99, 4899–4907

[2] J. J. P. Stewart "Optimization of Parameters for Semi–Empirical Methods I–Method" *Journal of Computational Chemistry* **1989**, 10, 209–220

[3] M. J. S. Dewar, E. G. Zoebisch, E. F. Healy "AM1: A New General Purpose Quantum Mechanical Molecular Model" *Journal of the American Chemical Society* **1985**, 107, 3902–3909

[4] Gaussian 98, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, O. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Menucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al–Laham, C. Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P. M. W. Gill, B. G. Johnson, W. Chen, M. W. Wong, J. L. Andres, M. Head–Gordon, E. S. Replogle, J. A. Pople, Gaussian, Inc., Pittsburgh PA (1998)

[5] Spartan, Wavefunction Inc., Von Karman Ave. Ste. 370, Irvine, CA 92612 USA

[6] C. Gonzalez, H. B. Schlegel "An Improved Algorithm for Reaction Path Following" *Journal of Chemical Physics* **1988**, 90, 2154–2161

[7] K. Ishida, K. Morokuma, A. Komornicki "The Intrinsic Reaction Coordiante. An Ab Initio Calculation for HNC –> HCN and H^- + CH4 –> CH4 + H^-" *Journal of Chemical Physics* **1977**, 66, 2153–2156

[8] K. Müller, L. D. Brown "Location of Saddle Points and Minimum Energy Paths by a Constrained Simplex Optimisation Procedure" *Theoretica Chimica Acta* **1979**, 53, 75–93

[9] Fukui "The Path of Chemical Reactions – The IRC Approach" *Accounts of Chemical Research* **1981**, 4, 57–64

[10] W. E. Lorenson, H. E. Cline "Marching Cubes: A High Resolution 3D Surface Construction Algorithm" *Computer Graphics* **1987**, 21, 163–169

[11] http://www.eyesopen.com/babel.html

[12] http://www.msg.ameslab.gov/GAMESS/GAMESS.html

[13] Mopac 93 R2, G. Ford, M. Orozco, M. B. Coolidge, D. Danovich, A. Klamt, V. I. Danilov, H. Kurtz, P. Korambath, F. Jensen, J. M. Simmie, J. A. Medrano, G. Purvis, Fujitsu Limited, Tokyo, Japan (1993)

[14] http://vegemite.chem.nott.ac.uk/~xmakemol

[15] ftp://ftp.ncsa.uiuc.edu/Visualization/Isovis/Unix/contrib/isovis_pw.tar.Z

[16] SYBYL 6.5, TRIPOS Inc., 1699 South Hanley Rd., St. Louis, Missouri, 63144, USA

[17] B. G. Johnson, P. M. W. Gill, J. A. Pople "Computing Molecular Electrostatic Potentials with the PRISM Algorithm" *Chem. Phys. Lett.* **1993**, 206, 239–246

[18] Sybyl Graphics Manual, Molcad Theory

[19] R. W. F. Bader, "Atoms in Molecules – A Quantum Theory", **1990**, Oxford University Press

[20] http://www9.informatik.uni–erlangen.de/Persons/Engel/orbitalanimation

[21] http://www.mdli.com

[22] http://www.opendx.org

[23] http://www.research.ibm.com/dx

[24] L. B. Kier, L. H. Hall, "Molecular Connectivity in Chemistry and Drug Research", **1976**, Academic Press

[25] M. Randic "On Characterization of Molecular Branching" *Journal of the American Chemical Society* **1975**, 97, 6609–6615

[26] A. Verloop, W. Hoogenstraaten, J. Tipker, "Drug Design", E. J. Ariens, Academic Press: New York (1976) Vol. VII: 165

[27] C. Hansch, A. Leo, "Exploring QSAR", Fundamentals and Applications in Chemistry and Biology, S. R. Heller, American Chemical Society: Washington DC (1995)

[28] P. L. Hammet "Some Relations Between Reaction Rates and Equilibrium Constants" *Chem. Rev.* **1935**, 17, 125–136

[29] C. Hansch, R. M. Muir, T. Fujita, P. P. Maloney, E. Geiger, M. Streich "The Correlation of Biological Activity of Plant Growth Regulators and Chloromycetin Derivatives with Hammet Constants and Partition Coefficients" *Journal of the American Chemical Society* **1963**, 85, 2817–2824

[30] R. D. Cramer, D. E. Patterson, J. D. Bunce "Comparative Molecular Field Analysis (CoMFA). 1. Effect of Shape on Binding of Steroids to Carrier Proteins" *Journal of the American Chemical Society* **1988**, 110, 5959–5967

[31] H.–J. Böhm, G. Klebe, H. Kubinyi, "Wirkstoffdesign", **1996**, Spektrum Akademischer

Verlag

[32] E. Fix, J. Hodges, "Discriminatory analysis, nonparametric discrimination: consistency properties" USAF School of Aviation Medicine Report no. 4 (1951), Randolph Field, Texas

[33] W. Zheng, A. Tropsha "Novel Variable Selection Quantitative Structure−Property Relationship Approach Based on the k−Nearest−Neighbor Principle" *J. Chem. Inf. Comput. Sci.* **2000**, 40, 185−194

[34] S. Wold, A. Ruhe, H. Wold, W. J. Dunn "The Collinearity Problem in Linear Regression. The Partial Least Squares (PLS) Approach to Generalized Inverses" *SIAM J. Sci. Stat. Comput.* **1984**, 5, 735−743

[35] M. Stone, R. J. Brooks "Continuum Regression: Cross−validated Sequentially Constructed Prediction Embracing Ordinary Least Squares, Partial Least Squares and Principal Components Regression" *Journal of the Royal Statistical Society* **1990**, 52B, 237−269

[36] P. Geladi, B. R. Kowalski "Partial Least−Squares Regression: A Tutorial" *Analytica Chimica Acta* **1986**, 185, 1−17

[37] S. Kirkpatrick, C. Gellat, M. Vecchi "Optimization by Simulated Annealing" *Science* **1983**, 220, 671−679

[38] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller "Equations of State Calculations by Fast Computing Machines" *Journal of Chemical Physics* **1953**, 21, 1087−1091

[39] J. H. Holland, "Adaptation in Natural and Artificial Systems", **1975**, MIT Press

[40] D. L. Selwood, D. J. Livingstone, J. C. W. Comley, A. B. O'Dowd, A. T. Hudson, P. Jackson, K. S. Jandu, V. S. Rose, J. N. Stables "Structure−Activity Relationships of Antimycin Analogues: A Multivariate Pattern Recognition Study" *J. Med. Chem.* **1990**, 33, 136−142

[41] H. Kubinyi "Variable Selection in QSAR Studies. I. An Evolutionary Algorithm" *Quantitative Structure−Activity Relationships* **1994**, 13, 285−294

Anhang

A Dateiformat für die Ball&Stick–Darstellung in *CAVOC*

B Dateiformat eines fertig berechneten Satzes von Isoflächen in *CAVOC*

```
short: Anzahl der Frames (Strukturen)
short: Anzahl der TriangleSets pro Frame (bei Orbitalen: 2)
byte: Eigenschaft einkodiert? (wenn nicht: 0)
long: Bitmaske, die die Phasenvertauschung kodiert (nur
      vorhanden bei 2 TriangleSets ohne einkodierte Eigenschaft)
short: Anzahl der Vertices
short: Anzahl der Indices
float: v1:X
float: v1:Y
float: v1:Z
float: v2:X
float: v2:Y
float: v2:Z
...
byte: v1:P r      ---
byte: v1:P g        |
byte: v1:P b        |
byte: v2:P r        | falls Eigenschaft einkodiert
byte: v2:P g        |
byte: v2:P b        |
...               ---
float: vn1:X
float: vn1:Y
float: vn1:Z
float: vn2:X
float: vn2:Y
float: vn2:Z
...
short[3]: vh, vk, vl
...
short: Anzahl der Vertices (nächster Frame oder TriangleSet)
...
```

C OpenDX–Beispielskript* zur Berechnung farblich kodierter Isoflächen

```
data1 = Import("surface.hdf");
data2 = Import("property.hdf");
multi = CollectMultiGrid();
macro main(iso, direction)–>(multi)
{
 member, index = ForEachMember(data1);
 isosurface = Isosurface(member, iso, NULL, NULL, 1, direction);
 d2 = Select(data2, index);
 mapped = Map(isosurface, d2, "positions", "data");
 multi, link = GetLocal(multi);
 multi = Append(multi, mapped);
 SetLocal(multi, link);
}
all = main(0.002, –1);
colored = AutoColor(all, 1, 1, 0, –0.6666);
output = Remove(colored, "data");
Export(output, "result", "dx msb ieee 2");
```

* Eine Erklärung der Befehle findet sich in der IBM Visualization Data Explorer User's Reference.

D Liste der in *CAVOC* abrufbaren Reaktionen

Diels–Alder–Reaktionen:	
Butadien + Ethen	
trans–Piperylen + Methylacrylat	oc, ot, mc, mt
Cyclopentadien + Methylacrylat	endo, exo
Acrylnitril + Cyclopentadien	endo, exo
Cyclopentadien + Acrolein	endo, exo
Cyclopentadien + Methylvinylketon	endo, exo
Cyclopentadien + Methacrylnitril	endo, exo
Cyclopentadien + (Z)–Crotononitril	endo, exo
Furan + Maleinsäureanhydrid	endo, exo
Furan + Acrylnitril	endo, exo
Furan + Methylacrylat	endo, exo
2,7,9–Decatrienal (intramolekular)	endo2–(E),7(E), exo2–(E),7(E)
S_N2–Reaktionen:	
Methylchlorid + Chlorid	
Borhydrid–Reduktionen:	
Formaldehyd + Tetrahydridoboranat	
1,3–dipolare Cycloadditionen:	
Knallsäure + Ethen	
Benzonitriloxid + Styrol	ortho, meta
Benzonitriloxid + Methylacrylat	ortho, meta
En–Reaktionen:	
Ethen + Propen	
syn–Eliminierungen:	
Ethylacetat	

Cope−Eliminierungen:
Dimethyl−ethyl−aminoxid

Sigmatrope Umlagerungen:
Cyclopentadien

Claisen−Umlagerungen:
Allylvinylether — Sessel−ÜZ, Boot−ÜZ

Cope−Umlagerungen:
1,5−Hexadien — Sessel−ÜZ, Boot−ÜZ

Hydroborierungen:
Ethen + Boran
1−Methylcyclopenten — Markownikow, anti−Markownikow
9−BBN

Epoxidierungen:
Peressigsäure + Ethen

Radikalische Additionen:
Methylradikal + Ethen

Radikalreaktionen:
Methylradikal + Ethan

Sulfonierungen:
Toluol — meta, ortho, para