Generation of Unique Chemical Libraries by Combinatorial Means

by

Łukasz Andrzej Weber

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2000

Author .....
Department of Electrical Engineering and Computer Science
February 7, 2000

Certified by.....
Bruce Tidor
Associate Professor
Thesis Supervisor

Accepted by...
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Generation of Unique Chemical Libraries by Combinatorial Means

by

Łukasz Andrzej Weber

Submitted to the Department of Electrical Engineering and Computer Science
on February 7, 2000, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The techniques used in drug discovery today range from similarity searches, where related compounds to past drugs are scanned, to combinatorial methods, where the search involves a diverse collection of chemicals. The majority of these searches and screenings are performed on a database or a library of chemicals. These databases, however, lack the breadth to encompass many of the viable compound families. Such searches for a lead compound often fail to find sufficient numbers of candidates for drug discovery. Combinatorial libraries, whether real or virtual, have the potential to encompass many new chemicals to create a broader chemical canvass.

The work described here investigates the issues, both theoretical and practical, of generating virtual, unique combinatorial libraries of molecules with 3D structural information. Software called "combine" has been written in C++, an object-oriented language, to implement a methodology for creating libraries of 3D chemical structures. The library is created from seed molecular fragments which are selectively joined in a breadth-first manner. The products of the join are screened for uniqueness by explicit matching done in $O(N * \lg(N))$ time, where N is the number of different molecules being compared. Facilities for rotamer sampling and memory management are also provided. Results are presented that illustrate the strengths and weaknesses of the technology. Suggestions for future extensions and applications are also discussed.

Thesis Supervisor: Bruce Tidor
Title: Associate Professor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of an effective drug is critical to patients with a debilitating disease and financially rewarding for the pharmaceutical company that develops the drug. Quite a few young biotechnology companies have based their success solely on the marketing of a single drug. Therefore, high interest exists on both the consumer's and producer's side of the pharmaceutical industry to discover lead compounds, or chemicals that have high potential of becoming new drugs. Each search for these compounds is shaped by how much is known about the target. The techniques used in the field range from similarity searches among related compounds to combinatorial methods where the search involves diverse pools of chemicals.

The majority of the present methods for drug discovery involves searches and screenings of a database or a library of chemicals. These libraries may be publicly accessible, but often they are the intellectual property of a drug company. Unfortunately, the present rule of thumb in the pharmaceutical industry holds that 10,000 compounds must be screened to find one that is a drug candidate, which is a very small ratio; corporate libraries generally range from 50,000 to 500,000 compounds. Understandably, the percentage of those drug candidates that will in fact become drugs is even lower. However, quite a number of drugs on the market today have been derived from screening corporate databases. SmithKline Beecham's discovery of a drug that binds the endothelin receptor is a prime example of such a search. Other examples include human immune-deficiency virus (HIV) protease inhibitors and

glycoprotein IIb/IIIa inhibitors [12]. Recent computational approaches make such discoveries faster and less expensive.

One way computers can help in screening large databases of molecules is by filtering chemicals for certain characteristics expected in a particular drug candidate. The particular properties that are used in the screen may vary from electrostatics to enumeration of pharmacophores, or arrangements and alignments of atoms that are thought to enable interaction between a drug and a target. However, to be successful, the screening techniques require structural information about the chemicals in the library. In many cases, corporate databases lack the breadth to encompass the necessary compound family so that a search for a lead compound will draw a blank. To supplement already available information, combinatorial libraries are made, since they may theoretically contain many new chemicals. Synthesis of such libraries is costly and time consuming, not to mention the fact that each laboratory screening depletes the library. Instead, a similar database can be created and searched by a computer to expedite the search and cut costs.

1.1 Current Methods in Computer-Aided Drug Discovery

A variety of computational methods exist to create or narrow a search for lead compounds. Some of these align pertient pharmacophores and then link them with some relatively inert chemical skeleton. This alignment of pharmacophores is either specified or designed from *a priori* knowledge of the drug target. Programs like MCSS, which stands for Multiple Copy Simultaneous Search [8], will take a protein's binding domain and select, as well as minimize, functional groups that best interact with that domain [22]. However, these chemical groups must still be linked into a single molecule, or ligand. HOOK will take entries from a database of molecular skeletons and try to link the pharmacophores into a ligand [14]. LUDI [4], GROW, and SPROUT [17] are alternative methods which generate such a ligand by attaching or

overlapping smaller pieces together to build a whole. Finally, a method like CCLD, or Computational Combinatorial Ligand Design, links the pharmacophores with linkers of 0–3 covalent bonds each, in a combinatorial fashion, rejecting molecules whose approximate binding free energy exceeds some threshold [7]. All of the above methods however, generate new molecules from detailed pharmacophore information, which may not be available for some of the drug targets under study.

Instead of generating molecules *de novo*, many algorithms are developed for high-throughput screening and three-dimensional database searching. These methods take advantage of either currently available structural databases of chemicals, or operate on newly synthesized combinatorial libraries. Different properties, deemed necessary for a particular lead compound, can be queried. Some of these characteristics may not be as directly related to structural information of the protein target, yet narrow the scope of the search significantly. Others however, may include specific queries, from MCSS [8] for example, that demand all candidates to contain a set of pharmacophores aligned in a prescribed manner. Examples of these methods fall into two major groups, the rigid and flexible search methods. Methods that involve either docking, like DOCK [15], or aligning single conformations of molecules, like RigFit [25] and CAVEAT, are known as rigid methods, since they did't, at least initially, take into consideration alternative conformations of a molecule. Flexible methods, like directed tweak [20], distance geometry or genetic algorithm, work with such molecular variants and may find candidates that a rigid search would reject. Albeit theoretically more powerful, flexible searches are still too slow to be widely used [13]. The results of these methods for screening will depend in a significant way on the size and breadth of the database that a query is performed on.

Ways of obtaining or creating chemical databases computationally include utilization of available crystallographic data and a combination of statistical and heuristic methods. The best-known crystallographic libraries are the Cambridge Structural Database (CSD) and the Protein Data Bank (PDB) [18]. In particular, the CSD is most useful to pharmaceutical companies, since its focus is on small molecules. These databases will contain structural information about its members, and can be

searched without further processing. Extensive libraries of two-dimensional, i.e. of atom connectivity, also exist in both public and private hands. Examples of public libraries include the Available Chemicals Directory, Spresi, and the Chemical Abstracts Database [13]. Additionally, each pharmaceutical company will have its own corporate database of proprietary chemicals. However, these databases often do not carry the structural information about their constituents, and thus further processing is required before any three-dimensional searching can be performed.

A large class of methods that perform just such a 2D to 3D conversion is based on tables of angles and bond lengths. These tables are determined either statistically from available data or heuristically. Most popular of these algorithms is CONCORD, which generates a single low-energy structure of a small molecule [18]. It is based on a rule-based algorithm and a pseudo-molecular mechanics approach. Other methods search through all the possible conformations, like MOLGEO, or use algorithms which aid in pruning the search, like COBRA and its A* algorithm. AIMB is another knowledge-based approach that takes the largest possible fragments from the CSD to assemble the conformers [13]. Most of the above methodologies depend on a formula or a prescription to generate the molecular structure, and are therefore limited to the information already available. However, commercially available software, prompted by the growing interest in combinatorial library generation, has been designed to generate chemical libraries combinatorially on a computer. Afferent Systems, Inc. offers one such system. The Afferent Structure software generates combinatorial libraries based on chemical synthesis reactions available in a chemical laboratory [1]. One advantage of this technique lies in the fact that a chemist can immediately synthesize any molecules generated. The software likewise retains all the benefits of combinatorial chemistry, which leads to synthesis of brand new molecules.

However, generation of chemical libraries combinatorially on a computer has many challenges. First and foremost is the combinatorial explosion problem shown in Figure 1-1. By repeatedly combining reactants with each other and allowing the products to complement the pool of reactants, the number of possible structures very quickly outpaces the storage capacity of any modern computer. Methods that use

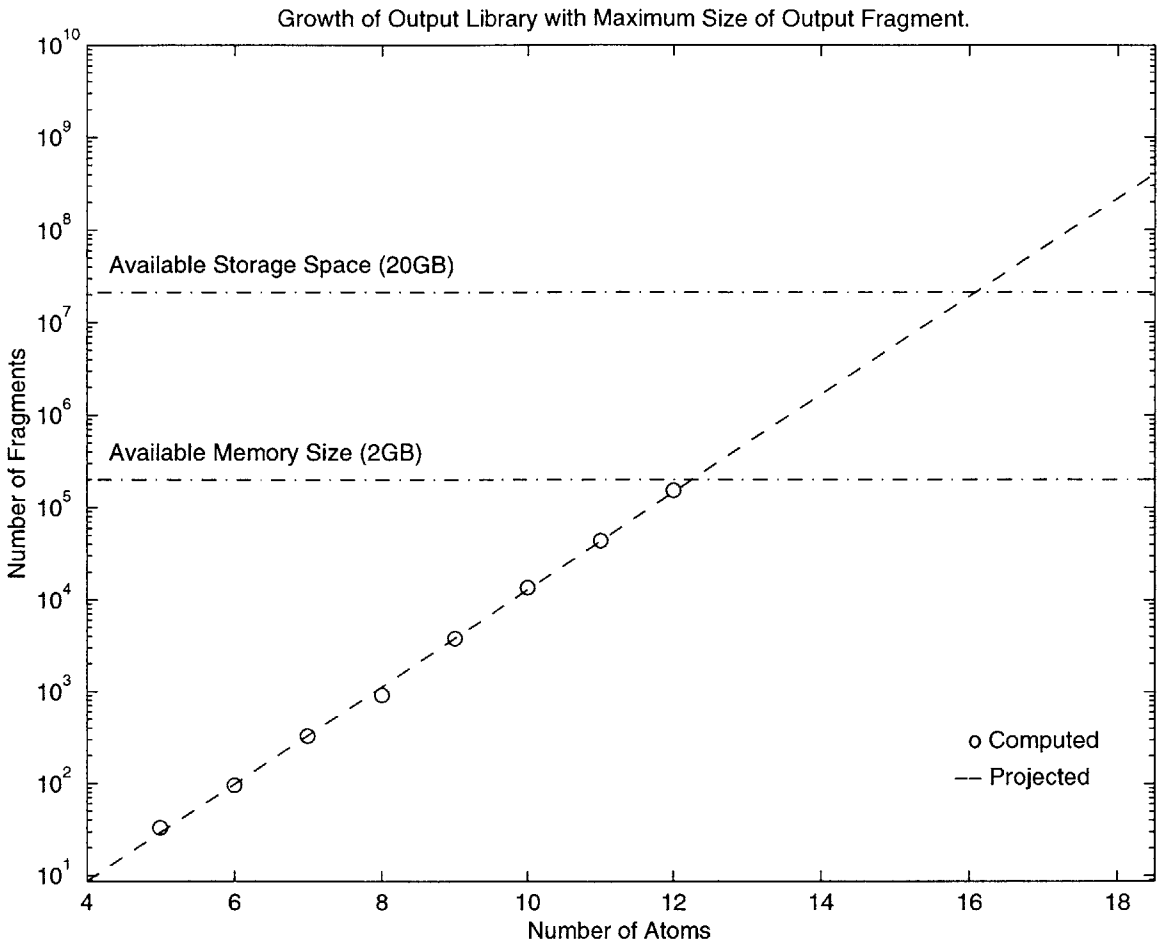

Figure 1-1: Computed and projected growth of output molecule library with increasing atom count ceiling. A discussion of this figure can be found in Chapter 3.

combinatorial assembly of molecules, like CCLD, use functions to limit the possible combinations and thus curb this explosive growth. Any algorithm that sets out to combinatorially assemble molecules must deal with the issues of library size and its efficacy. However, the prowess of combinatorial methods is widely recognized and used in both the industry and in scientific investigations alike, because the potential for finding new lead compounds in such libraries is unmatched by any other methods readily available today.

1.2 Virtual Libraries: Cheaper, Faster, Bigger

To build virtual, combinatorial libraries for drug design, a software package, called "combine", has been designed and implemented. This application combinatorially joins small molecular fragments from an initial seed library with each other in a breadth-first manner. The result is a library of three-dimensional chemical structures with bond lengths and angles as specified by parameter tables included with Quanta/CHARMM, a widely used molecular dynamics program from Molecular Simulations Inc. These parameters were developed to fit available experimental crystallographic data. The generation of molecules is relatively fast, since there is no minimization by molecular dynamics. The molecule joins are made to create valid chemical structures through direct geometrical alignments and tests, such as checks for steric clashes. By user-defined conformational sampling, all possible rotamers are generated and, by using a rejection filter, all unique molecules are kept. Although the program can join ringed molecules together, at this time *de novo* generation of ringed molecules is not integrated into the package. However, a promising methodology for creating rings has been developed and will be discussed. The resulting combinatorial library contains the three-dimensional coordinates and all necessary data to create CHARMM file formats, "PSF" and "CRD". This file format is well known in the field, with software available to translate such molecules to any other format. Moreover, this file format can be used directly to minimize the molecules and use them in further molecular investigations.

The details of the design and implementation are described in the following chapters. Specifically, Chapter 2 discusses the details of managing molecular information on a computer, all the steps involved in performing a join to create non-cyclic molecules and a method for creating ringed molecules within "combine" using a "thread and map" algorithm. This join operation is then leveraged to build libraries of molecules. Chapter 3 describes the methods of building such libraries, filtering molecules to reject duplicates and a memory management scheme that enables creation of libraries that make use of all available computer storage. Finally, Chapter 4

illustrates the use of "combine" to build libraries aimed at exploring a specific family of chemical compounds and to create a particular molecular structure. Chapter 4 also suggests other possible domains where "combine" can be essential. The appendix follows, describing the user interface to the software package "combine", which can be found in Appendix A.

Chapter 2

Generation of a Virtual Molecule

A search for a new drug begins and ends with a molecule. When generating molecules on the computer, i.e. without the appropriate wet lab space, certain considerations have to be made for handling, storing, and working with these structures, in essence, all anew. This chapter will talk about how that has been done with respect to the final goal, i.e. that of creating combinatorial molecular libraries for drug design.

2.1 Working with Molecules in Virtual Test Tubes

The most immediate concern while working with molecules on a computer is how to represent them and store them. Many ways are available to store the information necessary to represent a molecule. In general, the basic information needed is a list of atoms, their types and coordinates. The simplest molecular format is perhaps the "xyz" file format, which just lists the atoms and their coordinates in plain text. More restrictive formats may be created to protect intellectual property or to cater to certain applications. Some of these formats include CHARMM (Chemistry at HARvard Molecular Mechanics) "PSF" and "CRD", MDL Information Systems, Inc. "Isis" and "MOLfile", and those used by programs like AMBER, GRASP, and QUANTA. The format most widely used for protein visualization is the "PDB," or the Protein Data Bank, file format. An example of a "PDB" file for methane can be seen in Figure 2-4 on page 18. While for modeling, dynamic simulation and charge calculations,

```
PSF

       2 !NTITLE
* QUANTA-generated PSF for ch4
* Produced on Fri Dec 11 15:40:08 1998

       5 !NATOM
       1 CH4  1    CH4  C1     10   9.000000E-02   12.0110           0
       2 CH4  1    CH4  H2      3   5.000000E-02    1.0080           0
       3 CH4  1    CH4  H3      3   5.000000E-02    1.0080           0
       4 CH4  1    CH4  H4      3   5.000000E-02    1.0080           0
       5 CH4  1    CH4  H5      3   5.000000E-02    1.0080           0

       4 !NBONDS: bonds
       1       2       1       3       1       4       1       5

       6 !NTHETA: angles
       2       1       3       2       1       4       2       1       5
       3       1       4       3       1       5       4       1       5

       0 !NPHI: dihedrals


       0 !NIMPHI: impropers


       0 !NDON: donors


       0 !NACC: acceptors


       0 !NNB

       0       0       0       0       0

       1       0 !NGRP, NST2
       0       2       0
```

Figure 2-1: An example "PSF" file. This file was produced for methane.

the CHARMM and Gaussian file formats are quite popular. To deal with this wide array of formats, conversion utilities have been developed, so that most formats are interchangeable. One of the most popular of these utilities is Babel, written by Pat Walters and Matt Stahl [31]. "combine" uses the complementary CHARMM formats of "PSF", seen in Figure 2-1, and "CRD", seen in Figure 2-2 on the following page, for all of its input and output.

The foremost aspect of work with virtual molecules is the ability to visualize them in space. It is fairly difficult and pain-staking to be able to imagine a molecule simply from its atomic coordinates, especially if it is a protein containing a thousand or so atoms. Instead, researchers in the field use molecular viewers to see a three-

```
*
    5
    1     1 CH4  C1      0.00000  -0.02480   0.01820 CH4  1       0.00000
    2     1 CH4  H2      0.00000  -0.00890   1.10808 CH4  1       0.00000
    3     1 CH4  H3      0.00000  -1.05765  -0.33010 CH4  1       0.00000
    4     1 CH4  H4      0.88998   0.48367  -0.35259 CH4  1       0.00000
    5     1 CH4  H5     -0.88998   0.48367  -0.35259 CH4  1       0.00000
```

Figure 2-2: An example "CRD" file. This file was produced for methane.

dimensional rendering of the molecule. Figure 2-3 on the next page shows a graphic rendering of the CHARMM coordinate file for methane shown in Figure 2-2. There are many proprietary applications for protein and drug analysis available on the market that provide their own molecule viewers. Simpler and usually less expensive viewers can also be found in the freeware or shareware domains. When working on a Microsoft Windows platform, the free Internet browser plug-in Chime [9] from MDL Information Systems, Inc. was found to be very useful, especially for handling the display of multiple structures. More advanced visualization options, like boolean atom selection, as well as compatibility with a variety of computer and operating system platforms, can be found in RasMol [28] and VMD [19], both good, freely available molecule viewers. Molecule pictures like the one in Figure 2-3 on the next page and others in this report have been made by RasMol. All of these viewers support a majority of the popular molecular file formats that were mentioned above, if not natively then via Babel.

2.2 Simple Molecular Join

One of the most interesting aspects of chemistry is not necessarily the materials of chemistry, but what one can do with them, namely chemical reactions. In a chemical laboratory, a chemist is limited by nature as to which molecules can react together. When working with molecules in a virtual setting, any reaction is possible, whether the product is eventually synthesizeable in a real laboratory or not. The traditional approach to drug discovery, has been through actual chemical synthesis

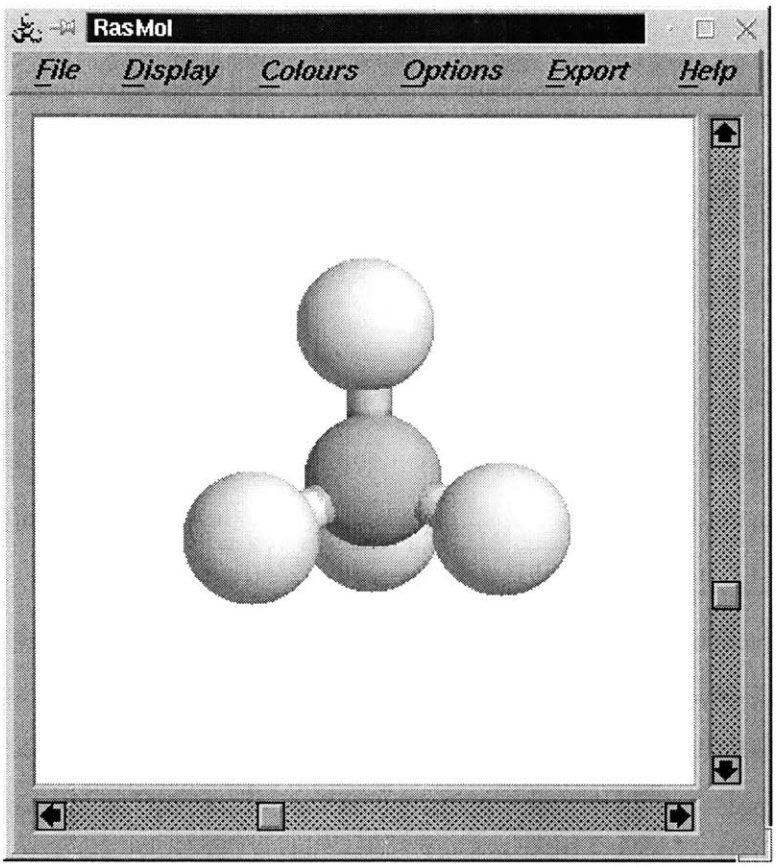

Figure 2-3: RasMol window showing the molecule methane rendered as Ball & Stick.

```
HEADER    PROTEIN
COMPND    ch4.pdb
AUTHOR    GENERATED BY BABEL 1.6
ATOM      1  C1   CH4     1       0.000  -0.025   0.018  1.00  0.00
ATOM      2  H2   CH4     1       0.000  -0.009   1.108  1.00  0.00
ATOM      3  H3   CH4     1       0.000  -1.058  -0.330  1.00  0.00
ATOM      4  H4   CH4     1       0.890   0.484  -0.353  1.00  0.00
ATOM      5  H5   CH4     1      -0.890   0.484  -0.353  1.00  0.00
CONECT    1    2    3    4    5
CONECT    2    1
CONECT    3    1
CONECT    4    1
CONECT    5    1
MASTER        0    0    0    0    0    0    0    0    5    0    5    0
END
```

Figure 2-4: An example "PDB" file. This file was generated by Babel from the methane "CRD" file.

of an assortment of molecules that would then be used for screening. Limitations of this approach have been discussed in Chapter 1. For theoretical purposes, it is more interesting to have a wealth of putative molecules that can be analyzed and selected as drug candidates. Some examples of such analysis can be found discussed in greater detail in Chapter 4. A hypothetical molecule can give an understanding of the target binding domain and the features that make a particular molecule a good drug candidate. Eventually, an attempt can be made to synthesize the hypothetical molecule, if not directly, then by finding analogous but synthesizeable molecules.

Given a wide array of possible reactions, the chemistry done by "combine" was chosen to be very simple. All reactions are joins made between two molecules, as shown in equation 2.1.

$$A + B \rightarrow C \tag{2.1}$$

It consists of stripping a certain atom, defined as the "contact," from both reactants and putting a bond in their place. Equation 2.2 illustrates this schematically.

$$A' - H + H - B' \Longrightarrow A' - B' \tag{2.2}$$

The atom picked as the "contact" in "combine" is presently taken to be a hydrogen, thus giving a wide array of bonding possibilities on most organic molecules. Alternatively, the exact atom type can be set as any of the many alpha-numerical choices provided in the CHARMM parameter file.

Since the reaction is taking place in 3-dimensional space, the whole "join" procedure involves steps done to make the product a "correct" 3-dimensional molecule. These steps include geometrical alignment of both reactants, removal of the "contact" atoms, formation of the bond between the reactants, and completing the necessary information to make the product a valid molecule. Essentially, from "PSF" and "CRD" files for the reactants, a valid set of files is generated for the product. Further processing is done to create rotamers, check for steric clashes, and generate rings.

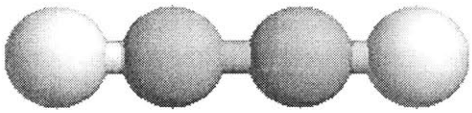

Figure 2-5: Ethyne: an example of a linear molecule.

2.2.1 Alignment

The first step to putting two molecules together is their alignment relative to each other in three-dimensional space. By translation and rotation, each of the reacting molecules is brought to the same geometric plane. Given two points in space, i.e. atoms, a molecule can be aligned to a geometrical axis. With a third point, that is non-collinear with the other two, the molecule can be set uniquely in a plane. To make the alignment complementary, the two molecules are aligned slightly differently. The molecules are set facing each other, one of them becoming the "donor" and the other the "receiver." Thus for both the "donor" and the "receiver," three unique atoms have to be picked for a planar alignment. The first two points are relatively straightforward to find since it is assumed that a molecule must have at least 2 atoms. Thus given a "contact," a host atom to that "contact" can be found. The third point may or may not exist, depending on whether the molecule in question is linear or not. In the former case, a linear molecule, as in Figure 2-5, is simply aligned along an axis, instead of the plane.

If methane, shown in Figure 2-3 on page 18, is taken as an example, then in the current implementation of "combine", one of the four hydrogens would be the "contact" and the carbon atom would be the host. Since methane is not a linear molecule, any of the other hydrogens can serve as the third point necessary for plane alignment. The three atoms and whether the molecule is a "donor" or a "receiver" determine the actual geometrical alignment that the molecule undergoes.

The "receiver" alignment consists of translation of the host atom to the origin and then rotation of the "contact" onto the positive x-axis. The choice of the origin or the x-axis for alignment is arbitrary, except that the choice must be consistent for correct alignment of both molecules. The choice of the origin had an additional

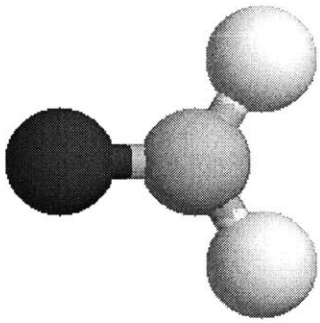

Figure 2-6: Formaldehyde: an example of a planar molecule.

advantage in that it made debugging easier. At this point, if the molecule is an ethyne, a linear molecule shown in Figure 2-5 on the preceding page, the alignment would be done. The categorization into linear, planar or three-dimensional molecules is based on the number of linkages, one, two or more than two, respectively, that originate from a host atom. In other words, in methane, the carbon atom has four linkages, each to one hydrogen, and therefore is a neither a linear nor planar molecule. Ethyne has only one linkage, therefore it's a linear molecule. Finally, formaldehyde, shown in Figure 2-6, has only two linkages, one, double bond, to an oxygen and one to to the second hydrogen. For molecules that have more than one linkage, or planar and three-dimensional molecules, a third atom is picked and aligned to the positive y-axis. In essence, the third atom must be linked to the host atom and cannot be the "contact" atom, which was already used for alignment.

The "donor" alignment complements that of the "receiver." Namely, the "donor" molecule is aligned to face the "receiver." This is done by translating the "contact" atom to the origin and aligning the host atom to the positive x-axis. When both molecules are placed in the same reference frame, the "contact" atoms and host atoms should now overlap on the x-axis. The "donor" molecule is also aligned to a plane, if it's not a linear, with it's third atom rotated to the the positive y-axis. It's from this position that the "donor" molecule, via the third atom, is rotated around the x-axis to make any rotational isomers that may be required. The making of rotational isomers is discussed in section 2.2.2. There's one last adjustment that needs to be made, before the molecules can joined, namely the bond distance between the host

atoms needs to be corrected. This is done by translating the aligned "donor" molecule along the x-axis. Since all aspects of the molecules that "combine" operates on are parametrized according to CHARMM parameters, this bond distance can simply be looked up. For a join between two methane molecules, the bond distances for a carbon–hydrogen bond and a carbon–carbon bond are 1.09 Å and 1.529 Å. Therefore the "donor" methane should be translated by 1.529 Å − 1.09 Å = 0.439 Å. After alignment of both the "receiver" and the "donor" molecules, the "contact" atoms can be removed and replaced with a single bond between the host atoms.

2.2.2 Sampling the Rotamer Space

Before the actual bonds can be made, the number of rotational isomers is determined and a list of conformational candidates is made. Rotational isomers exist due to the low potential energy around a single bond between two atoms. Two parts of the molecule are relatively free to rotate independently around such a bond. In general, a molecule will have a number of minimum energy states. These vary from molecule to molecule, mostly dependent on the symmetry of the groups on either side of the bond in question.

For example, in ethane as shown in Figure 2-7 on the following page, there are three minimum rotameric states. These are calculated from the *trans* position of a pair of hydrogens on two sides of the carbon–carbon bond. Since there are three hydrogens on one side of the bond, theoretically three ethane rotamers could be made. However, since the hydrogens cannot be differentiated, only one ethane rotamer is usually mentioned. The minimum energy conformation of ethane is the *trans* rotamer, where two hydrogens across the bond are at 180° with respect to each other. However, ethane can exist in other rotameric states as well, albeit at higher molecular energies. The highest energy conformation for ethane is the *cis*, or eclipsed, conformation. In the *cis* conformation, two hydrogens across the carbon–carbon bond are at 0° with respect to each other. Again, only one is usually given for the theoretical three. In any laboratory flask of ethane gas kept at STP (Standard Temperature and Pressure), there will be a distribution of rotameric states from *trans* to *cis* conformers, with

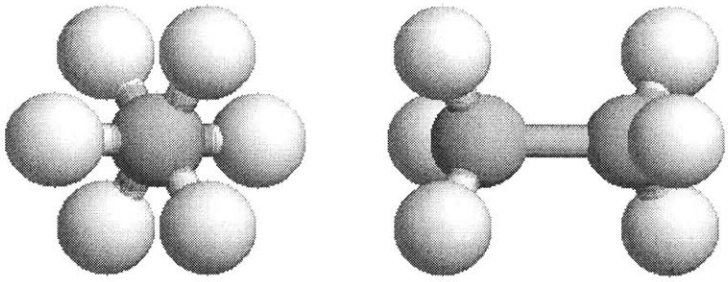

Figure 2-7: Two views of the *trans* ethane molecule. The molecule was produced by "combine" joining two methane molecules together at the minimum energy conformations.

all intermediate states. All these rotamers will exist with their numbers roughly exponential to their energies. Thus, most will be at or near the *trans* conformation and few or none at the *cis* conformation. This inherent "floppiness" of molecules is hard to model directly on a computer, instead rotameric states are sampled.

In "combine", the minimum energy conformations are taken from the parameter files. Arbitrary conformations can be generated from this information. A first starting point would be to take five samples for each minimum energy rotamer. Namely, two samples on each side of the minimum, each separated by a 5° rotation. For each of these rotamers, a symmetry is determined in order to remove the rotamers that are identical, as in the case of ethane described above. This rotamer enumeration is done around the bond between the "receiver" and the "donor" molecule. For each rotamer in the list of conformations, the "donor" molecule is copied, to retain the original alignment, and rotated by the third atom already picked to the appropriate rotamer angle. Subsequent joins are made with each of these "donor" copies, thus creating all the conformers for the join between the original two molecules. The significance of these join copies on library storage can be seen in Figure 3-1 on page 47.

2.2.3 Making Bonds

When the "receiver" and "donor" molecules face each other, after the geometric alignment described in the previous two sections, the "contact" and host atoms lie in the same line, the x-axis, at the right distance, a bond length, from each other. At this

point, the two "contact" atoms could be removed and a "bond" placed between the host atoms of the two molecules. However, to allow straightforward regeneration of the CHARMM molecular files, "PSF" and "CRD," "combine" maintains the molecular information contained in those files. To do so, careful arrangements have to be made to keep the bond, angle, dihedral, improper, donor and acceptor lists consistent. An example of some of these lists, only bond and angle lists are filled, can be seen in the methane "PSF" file shown in Figure 2-1 on page 16. As it turns out, keeping this information around is very useful in determining connectivity via simple look-up, instead of running a distance search algorithm. This is a gain in running speed with an approximate three-fold increase in memory storage per molecule. Therefore, when changing a molecule's atom list, adjustments need to be made to the connectivity lists as well.

A complete join operation involves removing bonds and atoms, renumbering the atoms, and adding a bond. First, in order to keep the atom numbering consecutive after atom removal, the "contact" atom is made last in the molecule atom list. This includes moving the atom from anywhere in the atom list and renumbering the other atoms, as well as all connectivity lists, to reflect this move. Next, the "contact" atom is stripped from the molecule, along with any linkages to it, or the bond between it and the host atom. The same is done to both the "receiver" and the "donor" molecule. The "donor" molecule's atoms are now transposed in count, so that the first atom of the "donor" molecule follows the last atom of the "receiver" molecule. The two atom lists are subsequently concatenated. Finally, the bond between the host atoms is appended to both the atom's bond lists to complete the join between the two molecules. The resulting ethane "CRD" file can be seen in Figure 2-8 on the next page. A join is almost complete at this point except for three bookkeeping details, namely charge conservation, dihedral generation and creation of a fragment name.

```
*(f1c2t0c2f1)
*
    8
    1     1  CH4 C1    -0.00000   0.00000   0.76450 CH4  1       0.00000
    2     1  CH4 H2     1.01963   0.00000   1.14979 CH4  1       0.00000
    3     1  CH4 H3    -0.50981  -0.88303   1.14979 CH4  1       0.00000
    4     1  CH4 H4    -0.50981   0.88303   1.14979 CH4  1       0.00000
    5     1  CH4 C5     0.00000  -0.00000  -0.76450 CH4  1       0.00000
    6     1  CH4 H6    -1.01963  -0.00000  -1.14979 CH4  1       0.00000
    7     1  CH4 H7     0.50981  -0.88303  -1.14979 CH4  1       0.00000
    8     1  CH4 H8     0.50981   0.88303  -1.14979 CH4  1       0.00000
```

Figure 2-8: The only ethane "CRD" file produced by "combine" when two methane molecules are joined.

2.2.4 Completing the Join

Calculation of Charge

Apart from the three-dimensional distribution of atomic volume, another key property of any molecule is its charge. Most molecules, like water, are neutral, i.e. of zero charge. All the known molecules have integer charge, since neither the proton nor the electron can be divided into a fraction. Thus, no molecule will have a charge of $+\frac{1}{2}$, for example. A molecule's charge is determined by a sum of the atomic charges resulting from a distribution of the electrons around a particular atom. When molecules are represented as points in space, the atomic charges are reduced to those points as well, usually resulting in non-integer "partial" charges. These partial charges in the CHARMM files can be seen in the seventh column of the "PSF" file for methane, seen in Figure 2-1 on page 16, and for ethane, seen in Figure 2-9 on page 28. There are many techniques to determine point charges, all of them are approximations of the actual electron cloud around an atom. Some of them are made from fitting to the results of explicit quantum calculations, done by programs like Gaussian98 [16] in conjunction with RESP [2], while the others are parameter fits to existing data. The latter are known as charge parameter sets, such as the AMBER [24], CHARMM [6], OPLS [21] and PARSE [29] charge sets.

The molecules that have been used with "combine" have been generated by Molecular Simulations Inc. Quanta/CHARMM program, which uses a similar charge set to

the AMBER set with some heuristic adjustments. Since "combine", in the process of joining two molecules, removes a "contact" atom from each molecule which may not have the same partial charge, a simple interpolation procedure was implemented to keep the overall charge of each molecule constant. Thus, since the "contact" atoms, in this case the hydrogens, are usually not atoms carrying much charge, the charge of the product molecule may be taken as the sum of the charges of the reactants. However, since an atom is taken away from each of the reactants, a fraction "contact" atom's charge is subtracted from all other atoms in the molecule such that the "contact" atom charge becomes equal to zero. At this point, removal of the "contact" atom will make no change in the molecule's charge, since all that charge is on atoms other than the "contact" atom. This very simple method has its flaws however. For example, in a methane molecule, the partial charges do not sum to zero. They should, however since methane is neutral. Thus, adding the charge of one of the hydrogens back to the other atoms will only make the overall charge of the molecule more positive. Here, the error in the partial charge approximation is thus being amplified. There are other simple alternatives, but the exact calculation of charge is left to more explicit methods. The ramifications of this charge approximation to drug design will be discussed in more detail in Chapter 4.

Dihedral Formation

When "combine" makes a join, much attention is paid to keep the information in the CHARMM files consistent for easy output. So far only adding and subtracting bonds was mentioned in the section 2.2.3. However, apart from the bond list, there are angle, dihedral, improper, donor and acceptor lists that need to be taken care of. The last three lists, namely improper, donor and acceptor lists are concatenated from the two reactant molecules, since making a single bond between host atoms to hydrogens will in general not change these lists. This leaves the angle and dihedral lists, which in fact may change during a join.

As two molecules are joined, each molecule is assumed to consist of at least two atoms. If such was the case, that only two atom molecules were joined together,

than adding in the bond between the host atoms would keep the information in the CHARMM files consistent. However, more interesting joins involve molecules much bigger than two atoms. In those cases, the angle lists and dihedral lists need to be kept updated as well. An angle is defined between three atoms, which around the join area would include the "contact," host and one other atom from either reactant. These angles are already defined and as long as the new host atom is substituted for the "contact" atom, the angle list will be consistent. There is thus no need to add or subtract angles.

On the other hand, a dihedral spans four atoms. The same substitution trick done with the angles only does part of the job here, since it only handles those cases where a "contact" atom is the terminal atom. To keep the enumeration of all dihedrals complete, new dihedrals must be added that include the two host atoms in the middle, with combinations of respective bonded atoms filling in the terminal positions of the dihedral. For a join between two methane molecules, the number of combinations of different atoms around the new bond is nine, as can be seen in ethane's dihedral list shown in Figure 2-9 on the following page.

Fragment Naming

After a join is performed by "combine", the molecule cannot be labeled as either of the reactants, since it is a new molecule and different from its substituents. A simple naming convention was developed to handle just such an ambiguity. An important feature of this convention is the ability to trace the history of the join, making the molecule unique and easy to regenerate. Such a molecule can be made directly given its name and its substituents. The naming convention calls for the first fragment's name to be listed first, then that fragment's "contact" atom. Next, the torsion, or the number of the rotamer, is concatenated. Lastly, the "contact" atom of the second fragment and the second fragment's name end the new name. These five fields are additionally delineated by unique characters. The resulting name can be used as a fragment name again, thus allowing recursive use of this convention.

In the example discussed before, where two methane molecules were joined to form

```
PSF

       2 !NTITLE
* QUANTA-generated PSF for ch4
* Produced on Fri Dec 11 15:40:08 1998

       8 !NATOM
       1 CH4  1    CH4  C1     10   1.025000E-01   12.0110           0
       2 CH4  1    CH4  H2      3   6.250000E-02    1.0080           0
       3 CH4  1    CH4  H3      3   6.250000E-02    1.0080           0
       4 CH4  1    CH4  H4      3   6.250000E-02    1.0080           0
       5 CH4  1    CH4  C5     10   1.025000E-01   12.0110           0
       6 CH4  1    CH4  H6      3   6.250000E-02    1.0080           0
       7 CH4  1    CH4  H7      3   6.250000E-02    1.0080           0
       8 CH4  1    CH4  H8      3   6.250000E-02    1.0080           0

       7 !NBONDS: bonds
       1       2       1       3       1       4       1       5
       5       6       5       7       5       8

      12 !NTHETA: angles
       2       1       3       2       1       4       2       1       5
       3       1       4       3       1       5       4       1       5
       1       5       6       1       5       7       1       5       8
       6       5       7       6       5       8       7       5       8

       9 !NPHI: dihedrals
       2       1       5       6       2       1       5       7
       2       1       5       8       3       1       5       6
       3       1       5       7       3       1       5       8
       4       1       5       6       4       1       5       7
       4       1       5       8

       0 !NIMPHI: impropers


       0 !NDON: donors


       0 !NACC: acceptors


       0 !NNB

       0       0       0       0       0       0       0       0

       1       0 !NGRP, NST2
       0       2       0
```

Figure 2-9: The ethane "PSF" file produced by "combine" when two methane molecules are joined.

ethane, the naming convention can be easily applied. If a methane molecule is called "f1," for the index of the fragment within a bigger library of molecules, then ethane made by removing the first hydrogen in each reactant would be named "(f1c2t0c2f1)". This is the name included in the ethane "CRD" file, as seen in Figure 2-8 on page 25. Within the name, "c2" means "contact" 2, which is the index of the first hydrogen in methane "CRD" file's atom list. "t0" stands for the first torsion, which, as discussed in section 2.2.2, is the only one made due to symmetry. Finally, the new name is pre- and postfixed with "(" and ")" characters. The choice of the delineating characters, namely "(", ")", "c" and "t", is arbitrary, and can be changed to avoid conflicts with special characters of a particular file system, for example.

At this point all mechanical aspects of a join have been described. Beginning with the alignment of the reactants, the removal of "contact" atoms and bond making, to balancing charge, making dihedrals and finally naming the new molecule, all these steps describe how "combine" takes the CHARMM files for the reactants and makes the correct "PSF" and "CRD" files for the product. In case of joining two methane molecules, the result of this join procedure, an ethane molecule, can be seen in Figure 2-7 on page 23.

2.2.5 Detection of Steric Clashes

Up to this point, any two molecules could be taken and by stripping two chosen "contact" atoms, put together to regenerate a correct CHARMM file. Will all molecules be chemically correct by this technique? As it turns out, many of the generated molecules will contain atoms that overlap each other. Overlap and atom crowding is usually referred to as steric crowding. Normally, real atoms will conform to relieve that crowding, possibly transferring the stress to other parts of the molecule. The only way to take such accommodation into account in virtual molecules is to perform a costly dynamic simulation. "combine" avoids this problem by sampling sufficient number of rotamers and allowing crowding up to a threshold. Determination of this threshold is key to finding many correct molecules and avoiding the false positives, those molecules that cannot exist.

The threshold in question is a point where two non-bonded atoms approach each other. This is usually quantified in the Lennard-Jones "6-12" relation, listed as equation 2.3.

$$E = \frac{A_{ij}}{r^6} - \frac{B_{ij}}{r^{12}} \tag{2.3}$$

where A and B are parameters for the types of atoms i and j. They are derived from the potential between two atoms of the same type. The above relation parameterizes the van der Waals interactions between atoms and produces a potential energy curve that has a minimum and two roots, one of which is at positive infinity. The same curve can be parametrized with e, the potential well depth at minimum energy, and either r* or σ. r* is defined as half of the internuclear distance r in equation 2.3, where the relation is taken between identical atoms at the minimum potential energy. The r* parameter is usually called the van der Waals atomic radius. On the other hand, σ is the distance where the potential energy equals zero, at the root of equation 2.3 not approaching infinity. The two are related by the following equation:

$$r^* = \sigma \sqrt[6]{2} \tag{2.4}$$

Both, r* and σ, are thus radii of an atom to which another atom can approach and where the resulting potential energy will be either at a minimum or zero, respectively. [24]

Initially, "combine" used the CHARMM parameter file's van der Waals radii to determine steric clashes, i.e. the threshold at which atoms in a molecule would be considered to overlap. However, this excluded some very common chemical compounds. Specifically, 1–5 interactions in those molecules were smaller than a sum of those atoms' van der Waal radii would allow. These interactions are called 1–5, since they occur in a five atom non-cyclic, bonded chain, the first and the fifth atom falling on the points of a molecular letter "U," so to speak. To better understand this phenomenon, 1–5 interactions in X-ray crystal protein structures were investigated to determine if there are better threshold radii.

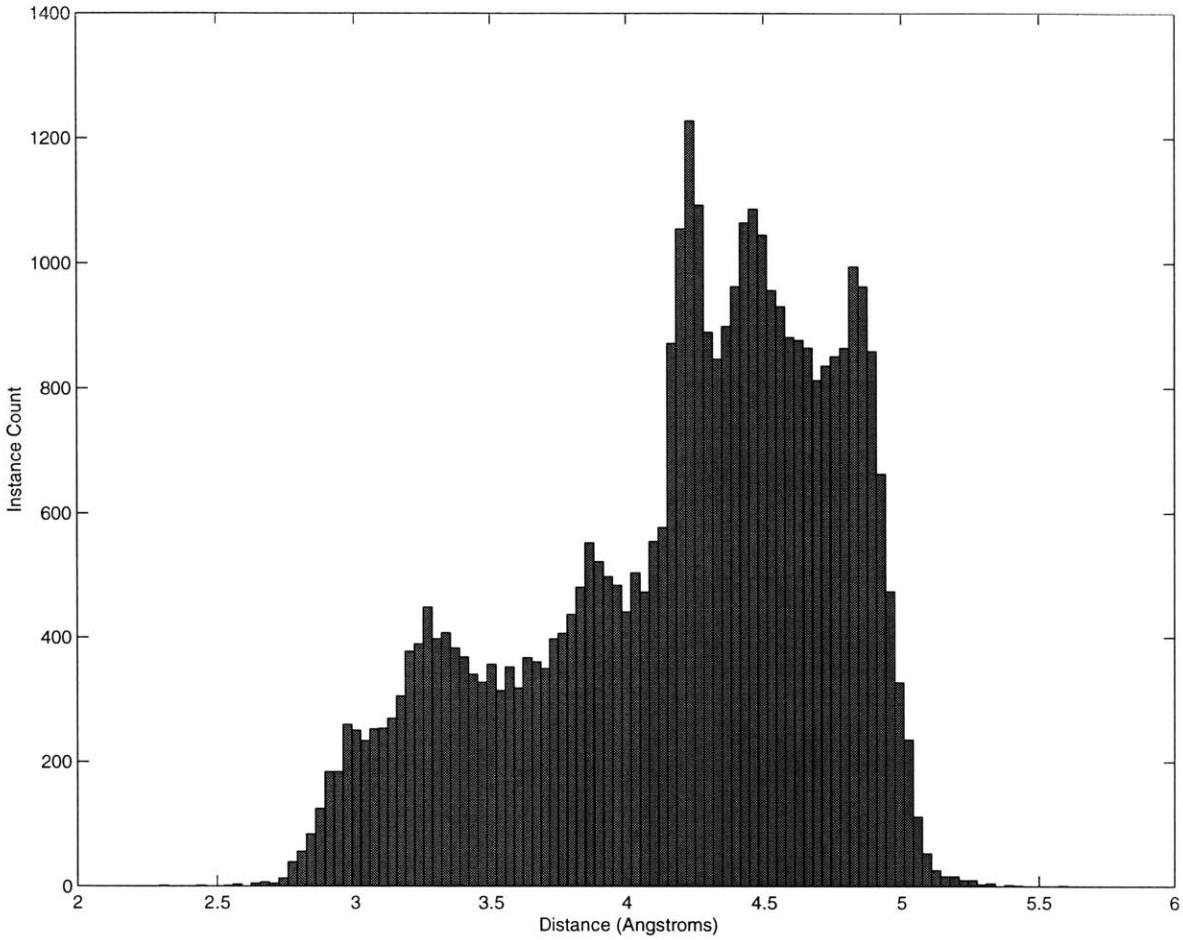

Figure 2-10: Histogram of 1–5 interactions from 20 X-ray structures from Table 2.1 on the following page.

All of the 1–5 interactions in the 20 structures listed in Table 2.1 on the next page, each with resolution of 2 Å or better, were measured. Pertinent interactions excluded rings and hydrogens. In total, 38,453 interactions were found. They are binned and plotted in Figure 2-10. It was found, by visual inspection, that most of the interactions below 3.1 Å involved either surface residues or residues interacting with heme, or iron, groups. These groups might be non-typical or have layer uncertainties.

When using the van der Waal radii for interactions between the most common 1–5 interactions in proteins, a significant portion of valid interactions were being neglected. Table 2.2 on page 33 shows the internuclear distances between these interactions in van der Waal radii and σ radii. Interactions above the 3.1 Å value were allowed.

Structure Name	PDB Id	Resolution (Å)	Residues(Chains)
Aminopeptidase	1AMP	1.8	291(1)
Carbonic Anhydrase I	1HCB	1.6	260(1)
Barnase	1A2P	1.5	330(3)
Calmodulin	1CLL	1.7	148(1)
Carboxypeptidase A	2CTB	1.5	307(1)
Gamma-Chymotrypsin	1AB9	1.6	246(4)
Concanavalin A	1JBC	1.2	237(1)
Cutinase	1CEX	1.0	214(1)
Cytochrome C	1CCR	1.5	112(1)
$Fe^{(II)}$ Superoxide Dismutase	1ISA	1.8	384(2)
Endonuclease V	2END	1.45	138(1)
Endostatin	1KOE	1.5	172(1)
Galectin-7	1BKZ	1.9	270(2)
Hemoglobin	1A3N	1.8	574(4)
Interleukin-6	1ALU	1.9	186(1)
Lectin	1LOE	1.9	466(4)
Lysin	1LIS	1.9	136(1)
Myoglobin	2MBW	1.5	154(1)
Oncomodulin	1RRO	1.3	108(1)
Phospholipase C	1AH7	1.5	245(1)
Protease I	1ARB	1.2	268(1)
Savinase	1SVN	1.5	269(1)
Troponin C	1TOP	1.78	162(1)
Trypsin	1DPO	1.59	223(1)

Table 2.1: X-ray crystal structures, their PDB identifiers, resolution and comments, used to survey 1–5 interactions in real proteins.

Thus, instead of using the van der Waal radius, or r*, "combine" presently uses the σ distance as the threshold to which 1–5 interactions are compared to. Those below this threshold are rejected and do not appear in output libraries.

2.3 Closing Rings via Templates

A single bond join can provide a large variety of molecules, however many pharmacologically active molecules contain distinct rings or ring structures. Unless these structures are provided for "combine", no new ringed structures will be made by the application. To generate rings, at least two simultaneous joins must be made. This is possible to do if very crude structures are needed, but there are no parameters

Interaction (i, j)	$r^*_i + r^*_j$ (Å)	$\sigma_i + \sigma_j$ (Å)
C, C	3.60	3.21
C, N	3.45	3.07
C, O	3.35	2.98

Table 2.2: Internuclear distances for the van der Waal interactions between most common atoms found in protein given in van der Waal radii and σ radii. Most 1–5 interactions were above 3.1 Å in the surveyed X-ray crystal structures.

that describe how a cyclopropane or a cyclobutane ring should look like. Thus, short of a dynamic simulation to minimize the molecular structure of a newly made ring, it is far from simple to create valid ring structures. Two approaches to generating rings were considered. Either "combine" would start with the basic ring structure and build from there or a larger molecule could be made into a ring by analogy. After weighing the advantages, the latter method was implemented and may be made part of "combine" in the near future.

Generation of a ringed structure from a pre-made ring seemed a winning approach at first, since any ring structures could *a priori* be minimized and then fed into "combine". This would require no new extensions to "combine", but would result in the user generating the wide tapestry of ring structures. This technique can be used when the target structure is known to contain rings of a certain sort, however if the libraries that "combine" is to create are to be truly combinatorial and thorough, the rings should be created from the molecules "combine" itself made. To assemble ring structures from molecules like the one in Figure 2-11 on the following page, a method involving choosing appropriate atoms and fitting them to a template ring was picked. To make a ring, a list of atoms must be chosen that would be "threaded" onto the ring. Next, these atoms are geometrically "mapped" onto a template ring.

2.3.1 Threading

Threading needs to be initialized with a non-contact, or seed, atom. This atom must also have a neighboring "contact" atom, which will later be removed to form the new ring. From the seed atom, a recursive search is started to find all enumerations of consecutive atoms in a bonded chain. None of these atoms can be "contacts."

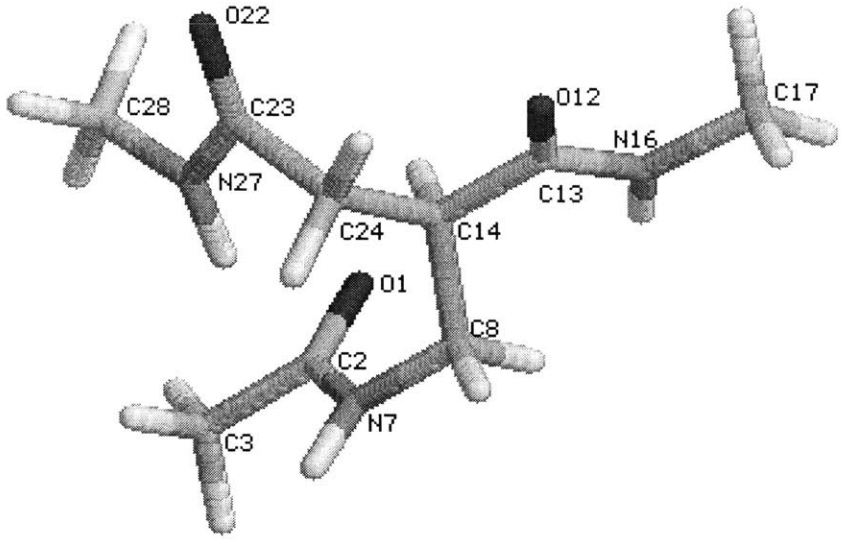

Figure 2-11: "Stick" rendered test molecule, originally generated by "combine", subjected to the "thread and map" method of making ringed structures. Only the non-hydrogen, or non-contact, atoms are labeled.

Furthermore, the terminal atoms must have at least one "contact" atom bonded to them. The enumerations must be at least three atoms in length, since no rings smaller than that can be formed. In general, the threading method assumes that the original molecule doesn't contain rings already, since doubly ringed systems impose further geometric constrains. However, extensions to this can be quickly implemented by tagging enumerations which contain ring atoms and then restricting what templates are applied to such an enumeration.

In the molecule shown in Figure 2-11, the threading algorithm produces a listing of 8 atomic enumerations of varying length. These enumerations are shown in Table 2.3 on the following page. The seed atom was a carbon atom, also the third atom in the molecule's atom list. Each enumeration in this list can now be mapped to a corresponding template ring.

2.3.2 Mapping

After a listing of all potential ring atoms of a test molecule is made via threading, these atoms must now be geometrically fit to a template ring. This is done by "mapping"

3 2 7 8 14 13 16 17
3 2 7 8 14 13 16
3 2 7 8 14 24 23 27 28
3 2 7 8 14 24 23 27
3 2 7 8 14 24
3 2 7 8 14
3 2 7 8
3 2 7

Table 2.3: Listing of all threading enumerations for the molecule shown in Figure 2-11 on the preceding page. The figure also contains atom labels along which the listed numbers here can be traced. The seed atom for this listing was the third atom in the molecule's atom list. Each of these listings can now be mapped onto corresponding template rings. Interesting features to note include the lengths of the enumerations, lack of "contact" atoms in any listing, and the presence of a "contact" atom (or hydrogen) at each terminal atom.

the atoms onto a minimized template ring of the same number of ring atoms. The template rings used initially were simple cyclic alkanes, however any rings can be used. The algorithm expects the template ring to have two "contact" atoms per each ring atom, thus all ring atoms are thought of as identical and the start of the ring wrapping can be arbitrary at this point. Given a non-cyclic test molecule and a template, mapping chooses the first atom in each molecule that either will be or is in the ring already, performs a join and aligns all atoms of the first molecule onto the template.

The starting atom for mapping is the first in the enumeration of putative ring atoms on the test molecule and the first ring atom in the template molecule's atom list. This works well if the template has either an equal or larger number of bonded non-ring atoms than the test molecule. From this point on, each atom in the test molecule will be aligned to a corresponding atom on the template ring atom. However, before any alignment is done, the non-cyclic molecule first undergoes a join to create the ring.

Just as with the simple joins made by "combine", and described in section 2.2, the ring creation process should create valid CHARMM format output files. To this end, before any geometrical alignment is undertaken, the first and last atom in the test molecule's ring atom enumeration are joined much in the same way as was described

previously. Namely, "contact" atoms are reshuffled, removed along with the bonds to their host atoms. Next, a new bond is made between the host atoms, angle and dihedral lists are merged and missing members to those lists added. At this point, by connectivity information alone, contained within the "PSF" file, the test molecule is now cyclic. The coordinates of the test molecule are still untouched however, and the molecule needs alignment before the ring can be viewed.

Given a starting atom for both, the test molecule and the template ring, mapping iterates over each template ring atom and placing the corresponding enumerated test atom at those coordinates. The side groups of the test atom are aligned along the bond vectors of the template ring atom. These bonds tie the template ring host atom to its two "contact" atoms. If the test atom has only one side group, the bond to that side group is aligned along a bisector of the two "contact" atoms of the template ring. It is assumed that such alignments make good approximations to how side groups would lie when attached to such a ring. Mapping is finished when all enumerated atoms, and their side groups, are aligned.

To illustrate what the mapping algorithm does, a test molecule shown in Figure 2-11 on page 34 was selected, from those made by "combine". The enumerations, chosen from the list in Table 2.3 on the preceding page, corresponding to a six and eight atom count rings were passed to the mapping function. Template cyclohexane and cyclooctane rings shown in Figure 2-12 on the next page were built and geometry optimized by Quanta/CHARMM. The mapping algorithm produced the molecule seen in Figure 2-13 on page 38 from the cyclohexane template and the molecule shown in Figure 2-15 on page 39 from the cyclooctane template. To understand how well the algorithm performed, these molecules were fed back into Quanta/CHARMM and geometrically optimized. The results for the cyclohexane template molecule can be seen in Figure 2-14 on page 38, while that for the cyclooctane template molecule in Figure 2-16 on page 39.

Although the structures produced by the mapping function look correct, they are just loose approximations to valid structures. This can be clearly seen when the new structures are compared to those that were geometry optimized. As expected, the

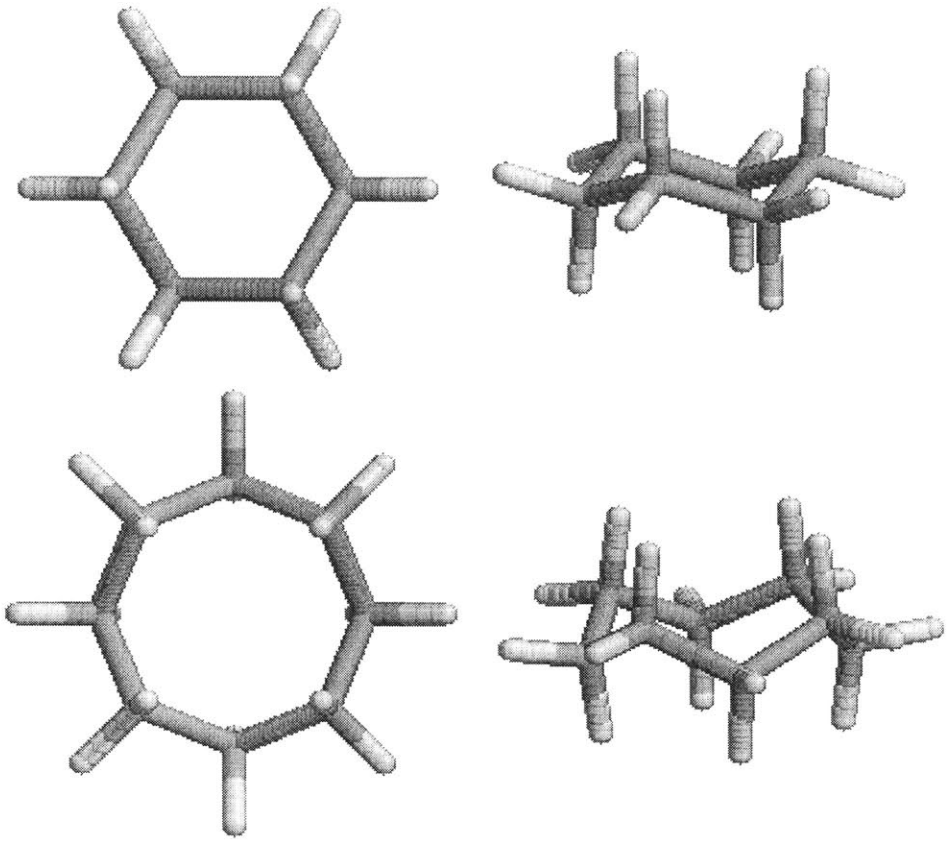

Figure 2-12: Examples of template ring molecules, cyclohexane, top, and cyclooctane, bottom. Above the ring view is on the left and side view is on the right. The test molecule shown in Figure 2-11 on page 34 was mapped onto these two molecules to produce either a cyclohexane ring, shown in Figure 2-13 on the following page, or a cyclooctane ring, shown in Figure 2-15 on page 39.

rings themselves relax differently given different constituent atoms of the rings than just carbons, as in the template rings. Furthermore, the side chains of the rings relax away from the ring by taking on different rotameric states. Although, cycloalkane rings seem to be poor templates for the diverse list of atoms in the test molecule, they prototype a structure that is more easily geometry optimized. The alternate alignments of the side chains fall into the minimum for that particular structure. It is possible that some other rotamer of the initial test molecule would satisfy this minimum conformation.

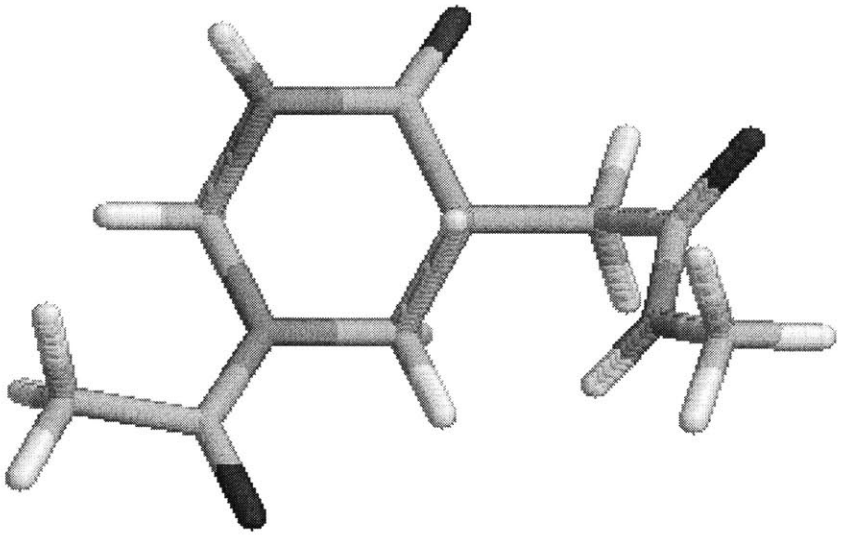

Figure 2-13: Molecule generated by the mapping method from a cyclohexane template.

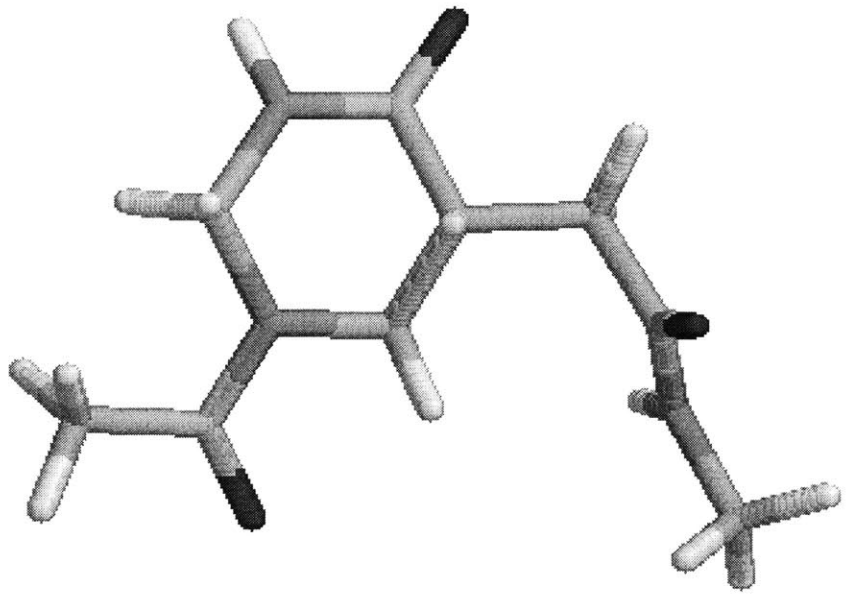

Figure 2-14: Quanta/CHARMM minimized molecule shown previously in Figure 2-13. Note the adjustments made in the ring and steric de-crowding in the ring side-groups.

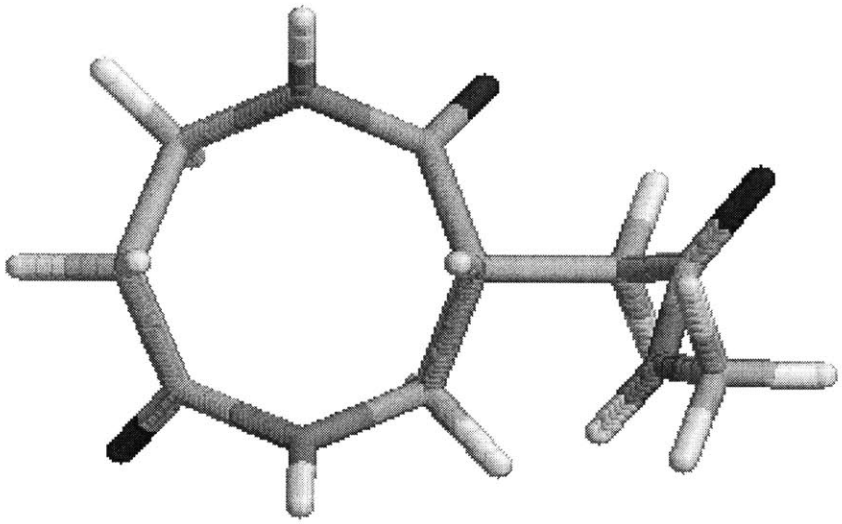

Figure 2-15: Molecule generated by the mapping method from a cyclooctane template.

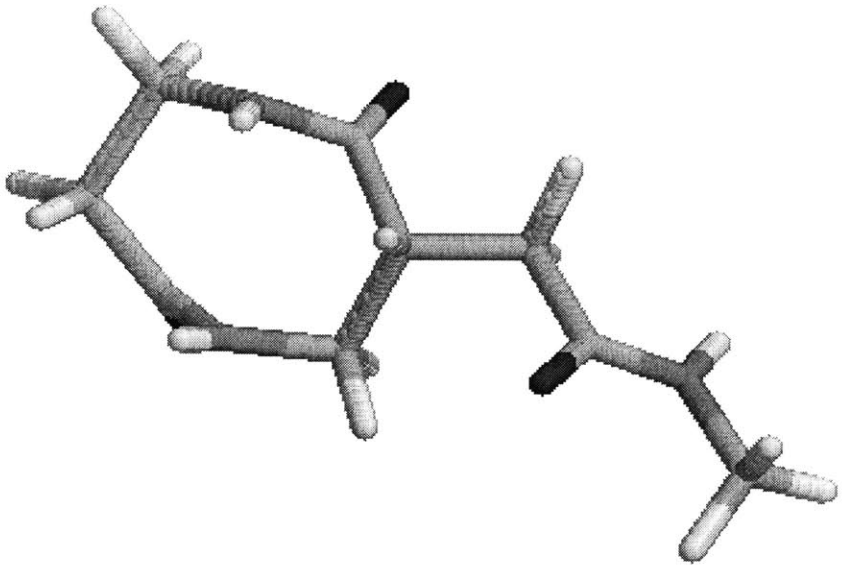

Figure 2-16: Quanta/CHARMM minimized molecule shown previously in Figure 2-15. Note the adjustments made in the ring and steric de-crowding in the ring side-groups.

Chapter 3

Making Molecular Libraries

Virtual combinatorial libraries are created by "combine" through repeated application of the join operation, described in Chapter 2. The largest library made by "combine" to date included 1.7 million valid theoretical chemical structures. Key elements of the library building process include rejection of duplicate molecules at runtime to reduce memory usage and computation as well as a memory management scheme to make efficient use of available storage.

3.1 Distinct Features of Molecule Library Building

The first step in generating a library via "combine" involves a selection of an initial, small, library of molecules. These molecules usually have only a few atoms, but cover the active chemical domains of interest. Such molecules could be nitrogen compounds such as ammonia, or carbohydrates, like methane or methanol. The exact choice of molecules is determined by the chemical space to be explored. Chapter 4 will describe examples of how "combine" can be used to generate such spaces. Molecules from the initial library are then used by "combine" as reactants to create new molecules by joining the initial fragments together. As new molecules are produced, they, and the initial molecules, become part of an expanding reactant library.

In the process of creating an output library, "combine" joins the molecules available in the reactant library in a breadth-first manner. The molecules are thus com-

bined with themselves in all possible combinations, numbers and "contact" sites. Thus, a methane molecule, which has four "contact" or hydrogen atoms, can be joined with itself in 16 combinations of "contact" atom pairs. Moreover, each of these joins can have a number of rotameric states. The products of these joins are then stored and used as reactants in the next round of joins. Quickly, the number of molecules generated grows, but many of the molecules produced in the example above would be identical. Methods involving symmetry and structure validity, as discussed in Chapter 2, are used to curb this growth. However, not all identical molecules, or isomers, can be eliminated by these simple methods.

To more thoroughly scan for identical molecules, an explicit search-to-reject filter has been implemented that eliminates most of the duplicate fragments from such libraries. The filter uses three stages to successively narrow the search space and to pairwise match possible duplicates by alignment of principal axes and a check of coordinate overlap. In particular, the filter uses cascaded red-black trees to reduce the general $O(N^2)$ problem to one which has a worst case order of growth of $O(N * \lg(N))$, where N is the total number of molecules. Using such a filter drastically reduces the size of the output library, however eventually the final library may exceed the memory capacity of the host computer. A memory management scheme akin to a most-recently-used cache that makes possible the use of all available storage has thus been implemented. This cache uses a doubly linked list to keep track of most recently used molecules. Once the list grows beyond a certain limit, molecules least used are archived to disk. Once archived, a molecule can be restored back to memory, if needed by the filter for example. The process of library generation terminates once any successive joins of molecules would produce a molecule whose atom count exceeds a set limit. At this point, any fragments not yet written to a file are archived, creating the final combinatorial molecular library.

3.2 Controlling Molecular Redundancy

To create an output library that contains only unique molecules, the joins that are generated by "combine" must be screened for duplicates. Since no clear locality of duplicate generation exists beyond those already mentioned in Chapter 2, the filter must compare a new molecule to all the molecules previously generated. If a match is found, the molecule is thrown out. Otherwise, it is kept as a unique molecule. No effective method has been determined of unambiguously verifying the similarity of two molecules without geometric alignment and an explicit match between atoms of the two fragments being compared, this search may be costly, running in a worst-case order of N^2, where N is the total number of fragments. Of the same order in the number of atoms is the overlap check, since each atom of one fragment must be compared with every atom of the other fragment. This latter computation is overshadowed by the immense number of fragments generated in relation to the maximum number of atoms contained in any output fragment. Therefore, a successful solution to filtering the output library should improve on the $O(N^2)$ worst case number of comparisons.

3.2.1 Divide and Conquer

Candidate approaches to improve this running time were hash tables and trees. Hash tables were considered due to their possible O(1) running time. This characteristic is ideal, but holds only under two assumptions. The first assumption relies on the fact that computation of a good hashing function is possible and relatively inexpensive. A hashing function determines the key, under which an entry is stored in the hash table. The key is calculated deterministically from some characteristics of the entry. The second assumption is that there are no collisions between entries. A collision occurs when, given two different entries, the hash function generates the same key. If a really bad hash function is chosen, the worst case running time becomes O(N), where N is the number of entries held in the table. Under such conditions, this approach reduces to the equivalent of searching a single long list of entries. Therefore, to gain an O(1) running time from a hash table approach, a good hash function must be found that

distributes entries uniformly. In order to effectively filter fragment libraries, this hash function should produce an even distribution of keys that persists over a large and dynamically growing range of output library sizes, which is not trivial.

Alternatively, a binary tree can be used to sort fragments. In this case, entries are inserted into the tree based on some criteria which determines whether a particular entry is less than or greater than another. If entries inserted into the binary tree have a good distribution, insert and search running times are O(h), where h is the height of the binary tree. The height of a binary tree, h, is lg(N), where N is the number of different entries in the tree and lg is the logarithm in base 2. Thus the running time is logarithmically slower than that of the hash table. Furthermore, the worst case running time of both insertion and sort can degrade to O(N), if the input sequence of entries is partly sorted.

Properties of Red-Black Trees

Albeit binary trees have a slower running time than hash tables and also contain dependence on the input entries, a small modification to the binary tree approach can guarantee O(lg(N)) running time for insert and search into the tree, regardless of the order of input entries. This modification transforms a binary tree into a so-called red-black tree, one of many schemes which keep the tree "balanced." In a red-black tree, or an RB tree for short, each of the nodes is assigned a color, red or black. The even dispersion of nodes in the tree is guaranteed by keeping the so-called "black height" of the tree constant. This ensures that no branch of the tree is more than twice as long as another. Constant tree height is maintained by operations called "rotations," whose exact description, as well as pseudocode for the insert and search functions can be found in the introductory text to algorithms written by Cormen, Leiserson, and Rivest [11].

Fragment Classifiers

When considered as part of the solution to the fragment filter problem, the RB tree gives an improved running time over a list search and, more importantly, guaran-

tees this performance over possibly dynamically varying sizes of the output libraries. However, to make the scheme complete, a discriminating function must be picked by which to sort the fragments in the tree. Two possible characteristics of a fragment were considered. First, since each fragment is made of a certain number and type of atoms, a string descriptor can be made from this information. The descriptor contains, in descending order of atom type, the numerical atom type equivalent and the number of atoms of this type found in the fragment. Second, the sum of inter-atomic distances can be used as a discerning characteristic. A distance matrix can be generated for a molecule, where each matrix entry represents the distance between two atoms of that molecule. In general, the resulting matrix is symmetric and the diagonal entries are equal to zero. Furthermore, this distance matrix is invariant to translation or rotation of the molecule. Taking the sum of the unique inter-atomic distances gives a good index to fragments of similar structure. Since, neither the descriptor nor the distance sum, summarize the same information directly in their generation, they can be used together for better partitioning of the output library.

3.2.2 Similarity Determination

Once the output library is partitioned by the fragment descriptor and the inter-atomic distance sum, a certain degree of similarity is assured within each "bin" of fragments. However, a diversity of molecules can still exist, but these are mostly isomers of each other. To guarantee a correct match between fragments, a new fragment is explicitly aligned and matched with only the fragments found in the corresponding bin.

Principal Axis Alignment

In order to align two molecules, each fragment's moment of inertia tensor is calculated around the molecule's center of mass. This 3x3-tensor matrix is diagonalized by finding the eigenvalues and eigenvectors that are the principal axes of each fragment [23]. Since the principal axes are invariant to translation, the coordinates of both molecules are moved so that the center of mass becomes the reference point for

all atom coordinates. Additionally, both the principal axes and, thus, the atoms of each molecule are aligned by rotation so that each principal axis coincides with one of the axes of the Cartesian coordinate system. Once both of the molecules have their principal axes aligned, they can be matched. This procedure is not sufficient for those molecules that have symmetry and whose principal axes are not unique.

Degeneracy and Fragment Matching

Plane, spherical and other types of symmetry, often occur in molecules. Such symmetry may cause two or all the principal axes to be equal. This causes degeneracy in the way molecules are aligned according to their principal axes. When comparing two symmetrical molecules, their respective principal axes can be flipped, or rotated by 180 degrees. This kind of a difference may result in a missed match for the filter. To account for this degeneracy, molecules are compared for 180 degree rotations of around each of the Cartesian axes. This is sufficient to correctly align molecules that have plane symmetry.

If, for example, the two molecules being compared are methanes, all three principal axes will have the same magnitude. In this case, the molecules may end up in an arbitrary alignment. To bring the two molecules to overlap, an explicit matching technique is used. The two most distant and non-collinear atoms from the fragment's center are selected. The farthest atom is then aligned to the positive x-axis, while the second farthest point is aligned to the positive y-axis. This alignment is repeated for all combinations of equivalently distant atoms found. Once aligned, the molecules' overlap is checked to determine similarity.

At this point, the two molecules should be aligned such that their structures and atom types overlap if the molecules are identical. A brute force approach is used as a straightforward way of matching up these fragments. Thus, if each atom of the first molecule is matched to an atom in the other molecule, the two molecules are the same. Specifically, a query atom's coordinates and type are used to search the target list of atoms for a corresponding match. Once a match between all the atoms' coordinates and atom types is found, the new molecule is thrown out, while the old

one is kept. The process is repeated for each molecule incoming to the output library.

3.3 Keeping Molecules in Mind: Memory Management

Since combinatorial generation of molecules shows an approximately exponential growth, as seen in Figure 3-1, ways of managing memory had to be implemented. In order to do so, two aspects of molecule generation were kept in mind. That already mentioned, or filtering out identical molecules, and putting made fragments onto media other than the computer's memory. Hard disks, as well as magnetic tape, is an example of this media which is often orders of magnitude bigger and cheaper than current computer memory. The migration of fragments at runtime between memory to a hard disk is handled by a technique similar to another very popular in computer engineering, namely caching.

3.3.1 Holding On To Most Recently Used Molecules

In general, as molecules are made, they are put into memory. Those that are kept permanently must be valid structures and must pass the rejection filter described in section 3.2. Once made, the molecule is not only stored, but it is used to screen against duplicates as new combinations of initial molecules are made. In fact, when filtering a new candidate molecule, all molecules previously made that match the descriptor and distance sum of the candidate, must be readily available. However, as the library generation process goes on, larger, more diverse molecules are made. This means that smaller molecules will no longer be needed for filtering purposes. Furthermore, certain descriptor and distance sum combinations may become exhausted such that no new molecules can be made to fall under them. Both the smaller molecule groups and specific classes of molecules can thus be archived to slower, but more spacious media.

The method that allows for the program to automatically detect which molecules

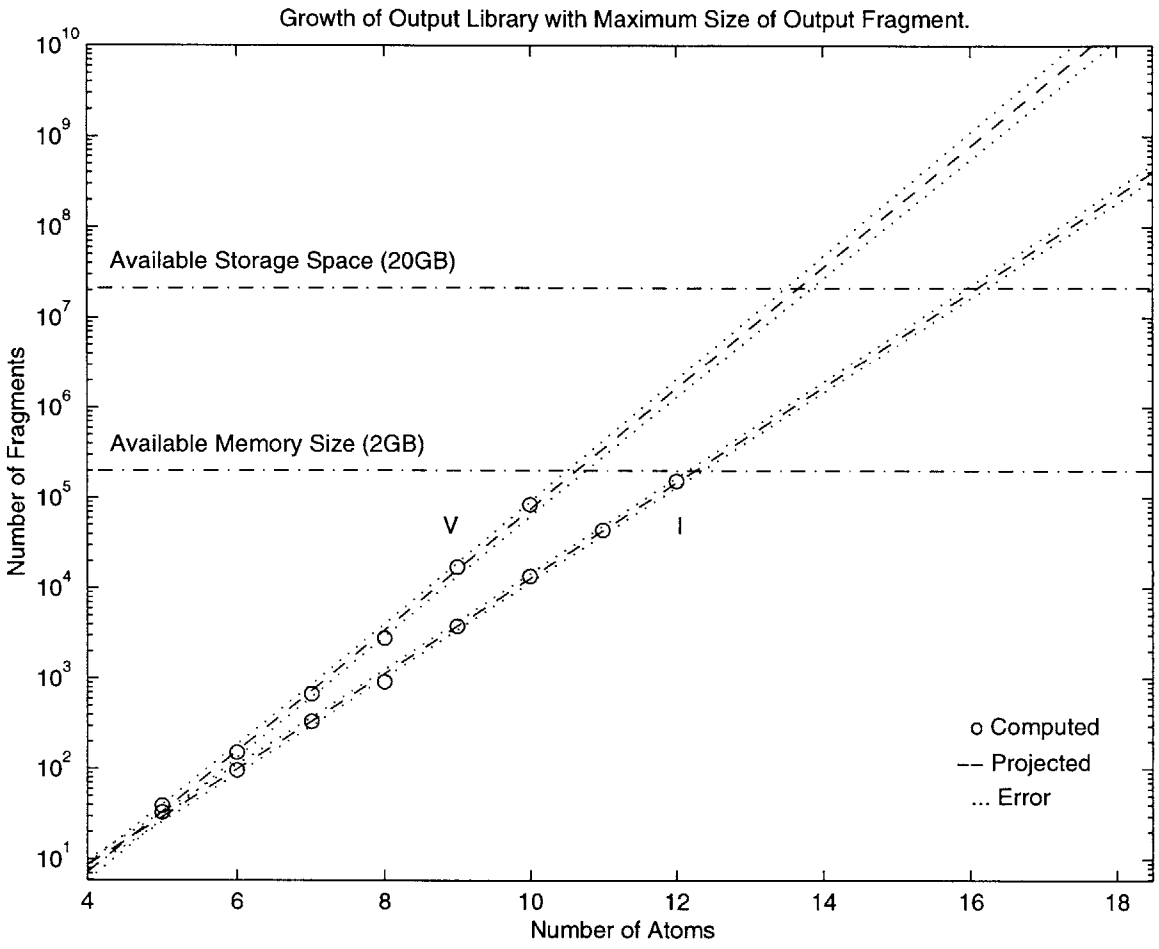

Figure 3-1: Computed and projected growth of output fragment library with increasing atom count limit. The curves labeled "I" are just the minimum energy conformers around each new bond. The curves labeled "V" are for a total of five conformations, one at and two on each side, 5° apart, for each one of the minimum energy conformer. The linear interpolation, as well as the upper and lower bounds for error, were calculated with MATLAB's *polyfit* and *polyval* functions [30] in the semi-log space.

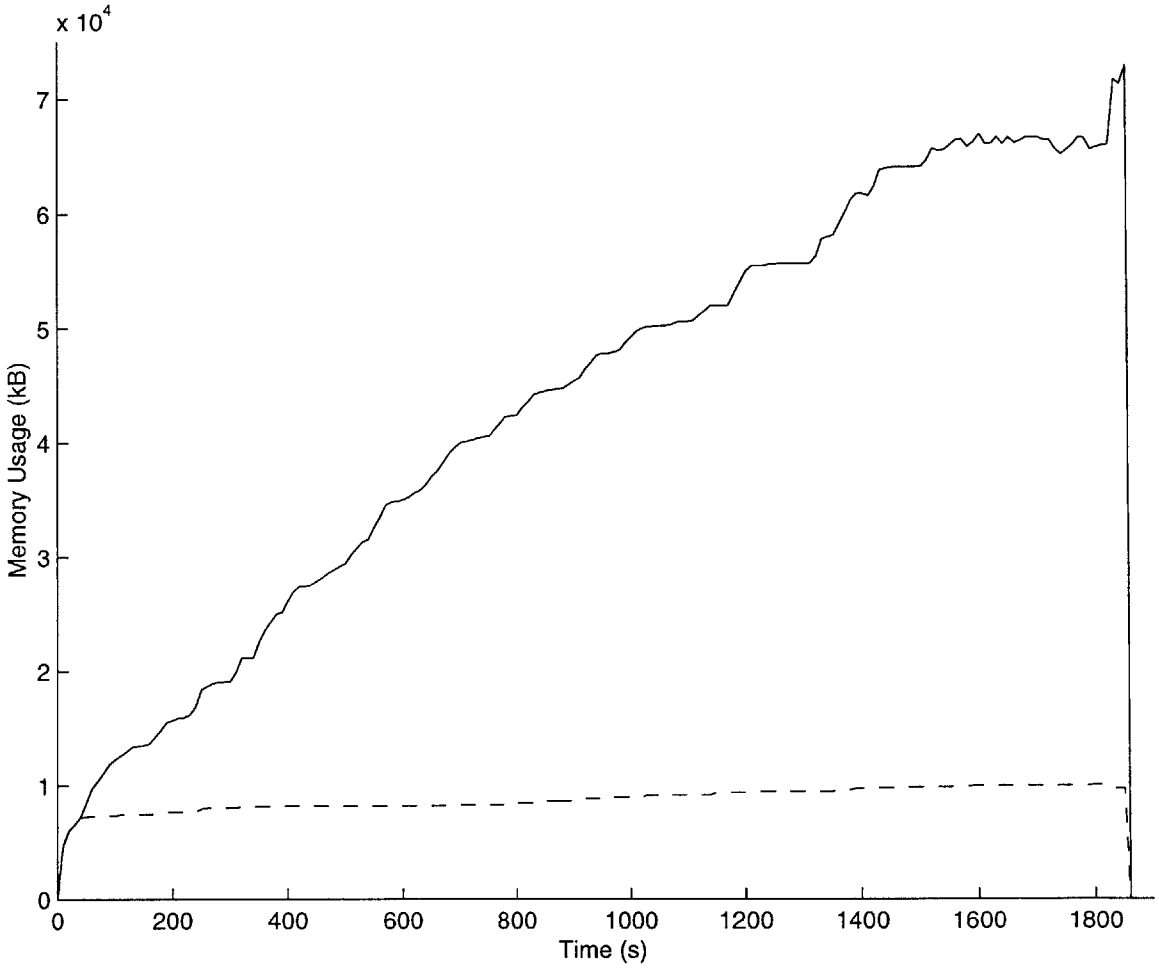

Figure 3-2: By using memory management, usage of space can be made almost a constant. Two runs of "combine" were made, one utilizing memory management, shown by a – line, and the other didn't, represented by a solid line. The program processed two molecular libraries containing about 25 molecules each, which resulted in an output library of 3750 molecules. Both processes ran for the same amount of time, down to a couple of seconds. Memory management was set to maximum fragment count of 200 and minimum of 100.

are not needed anymore is called MRU caching, where MRU stands for "Most Recently Used". The technique calls for a flag or counter to be toggled when an item in the cache, or fast memory, is used. As the memory fills up, the items which have not been marked or have a low count of usage are discarded from the cache, or in other words, written out to slower memory. Since a cache is usually pretty small, provisions are made to read the item back into the fast memory when it is needed the next time. One way of implementing this caching scheme is with a doubly linked list. In such a design, new items are added at the front of the list. Those items already in the list that are used are moved up to the front of the list. As the list grows too big, the date of the items from the tail of the list can be archived. The node itself cannot be discarded, since it needs to act as a trace of the archived data, in case that item's data is needed again.

In "combine", the data of interest are, of course, the molecules themselves. To implement the doubly linked list, a light-weight list node was designed to hold the fragment data. The doubly linked list is then built up from these nodes. Three pointers are provided for list management. The traditional head and tail pointers, plus an additional pointer to the last node in the list that contains unarchived data. Thus, as a molecule is made, a new node for it is also provided. The node is then inserted at the head of the list. The same node is used in the descriptor and distance sum rejection filter. When the molecule is needed for comparison purposes, it's node is removed from the list and re-inserted at the head of the list. This way molecules "most recently used" are near the head of the list. Both, the removal and insertion of the nodes takes constant time, since the list does not need to be traversed. When the list gets sufficiently big, molecules can be archived starting from the list's tail.

3.3.2 Archiving Library Molecules

In "combine", the point at which the data should be archived from the list of molecules is set to correspond to a large fraction of the available random-access memory, or RAM. When that amount of memory is used up, nodes with molecules, starting from the tail of the list, write the molecule to disk. To determine when this archiving of

molecules stops, the user can also set a minimum memory usage which, along with the upper limit, should be independently tuned to optimize performance. When the molecule gets written to disk, each node stores the address where its molecule was archived. Once the molecule is written, the last full node pointer is moved up towards the head of the list by one node. This assures that the node that the last full node pointer points to is truly one separating the nodes with and without molecules. The place on disk where the molecules are being written to is usually the output file. That means that all molecules must only be written to disk once. The fact that each node stores the address of its molecule guarantees that, if the molecule is needed again, it can be found and read back into memory. If such a recovery operation takes place, the node with its molecule is removed and re-inserted at the head of the list, just as if it was used. Thus, recovery is performed anytime the molecule is needed, but is not in the memory. Furthermore, recovery is abstracted, i.e. made invisibly, to the calling method.

Chapter 4

Applications of "combine"

Multiple extensions of the library building methodology can be implemented, once the details of the join operation are taken care of. Apart from the generation of molecules by combinatorial joins presented in Chapter 3, two libraries can be joined combinatorially with each other. Other methods may include sequential buildup of an output library by joining one fragment at a time to the growing library. At the present time, "combine" has two modes of operation. The first allows "combine" to simply generate extensive combinatorial libraries from small initial fragments. Each such library is made to contain up to and including all molecules of a specified atom count. The largest of these libraries contained 1.7 million molecules, a library of structural data with significant breadth compared to available alternatives. In the second mode, "combine" joins class libraries of molecules to build a larger target molecule. These class libraries may include groups that are similar in some way. A combination of such groups can create the target and many derivative molecules that are similar structurally or functionally.

4.1 Exploring Chemical Space

When searching for a particular drug, some of the properties of the target protein are usually known. These may include distinct features of the binding domain which suggest that certain pharmacophores, or chemical groups thought to determine drug-

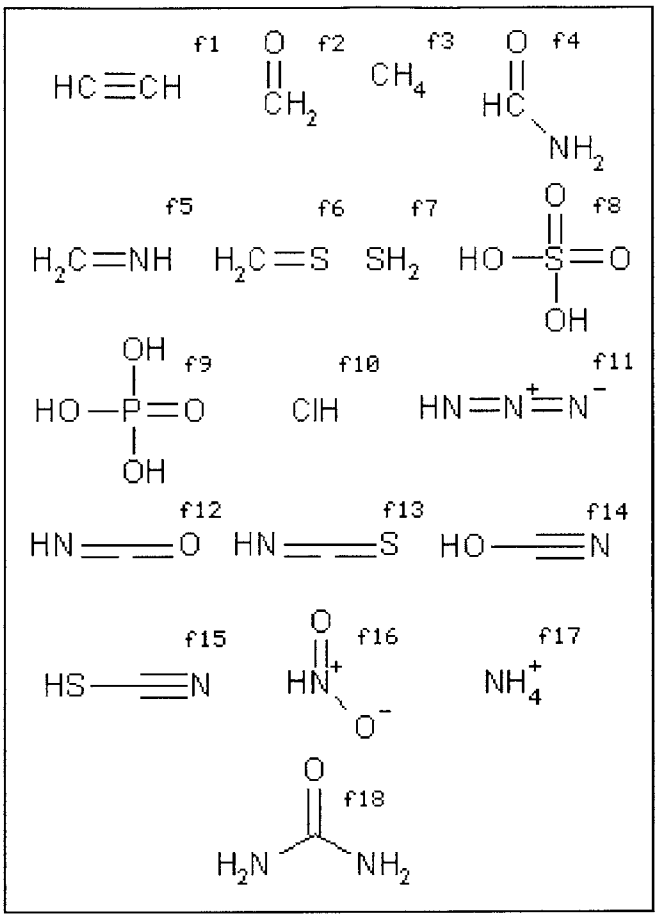

Figure 4-1: Initial fragment library used to explore a perticular chemical space. The labels "f1" through "f18" indicate names given to the fragments in the initial library. The fragments were generated by ChemNOTE, Molecular Simulations Inc. The figure was made by ISIS Draw, MDL Information Systems, Inc.

target interaction, should be part of the new drug. A collection of these groups, along with some more common structural linkers, like methane, can be compiled as an initial library for "combine". Subsequent combinatorial joining of these groups with each other will create a chemical space that encompasses molecules containing only the pertinent pharmacophores. This space can then be searched by any of the methods mentioned in Chapter 1.

As an example, an initial library illustrated in Figure 4-1 was generated by ChemNOTE, Molecular Simulations Inc., and represents a select group of chemical groups. "combine" joins two molecules from this library by selecting "contact" sites, which at this time were simply hydrogen atoms, on both fragments and replacing them with a

Atom Count	Single Conformer	Five Conformers
5	33	39
6	96	151
7	331	671
8	916	2790
9	3795	17049
10	13569	83978
11	44030	n.a.
12	152954	n.a.

Table 4.1: A list of output library sizes from combinatorial joins of fragments listed in Figure 4-1 on the page before. Sizes of both, single conformer and five conformers, for each minimum energy rotamer, libraries are listed.

bond by a process discussed in Chapter 2. Output libraries of increasing sizes were made that included both, the minimum energy conformers and five rotamers per each minimum energy conformer. These have been plotted in Figure 1-1 on page 12, Figure 3-1 on page 47 and listed here in Table 4.1. Selected fragments from the largest single conformer library are shown in Figure 4-2 on the following page.

4.2 Focused Library Generation: Making Histidine

Building a combinatorial library results in a large and diverse collection of fairly small molecules. For a particular drug design task, most of the combinatorial library's broad selection of molecules will not satisfy the target criteria. Once certain candidates are selected, however, a more thorough examination of each structure can be made. Specifically, the candidate's structure can be conserved, while the key substituents combinatorially enumerated. Additionally, higher rotamer sampling can be made when molecule generation is thus restrained. Such focused construction of molecules can aid in selecting a better fitting drug. Although the library generation techniques are approximations of real molecular structures, the fast generation of a vast number of them can give a good survey of sought-for drug candidates. The promising candidates can then undergo geometry optimization and charge calcula-

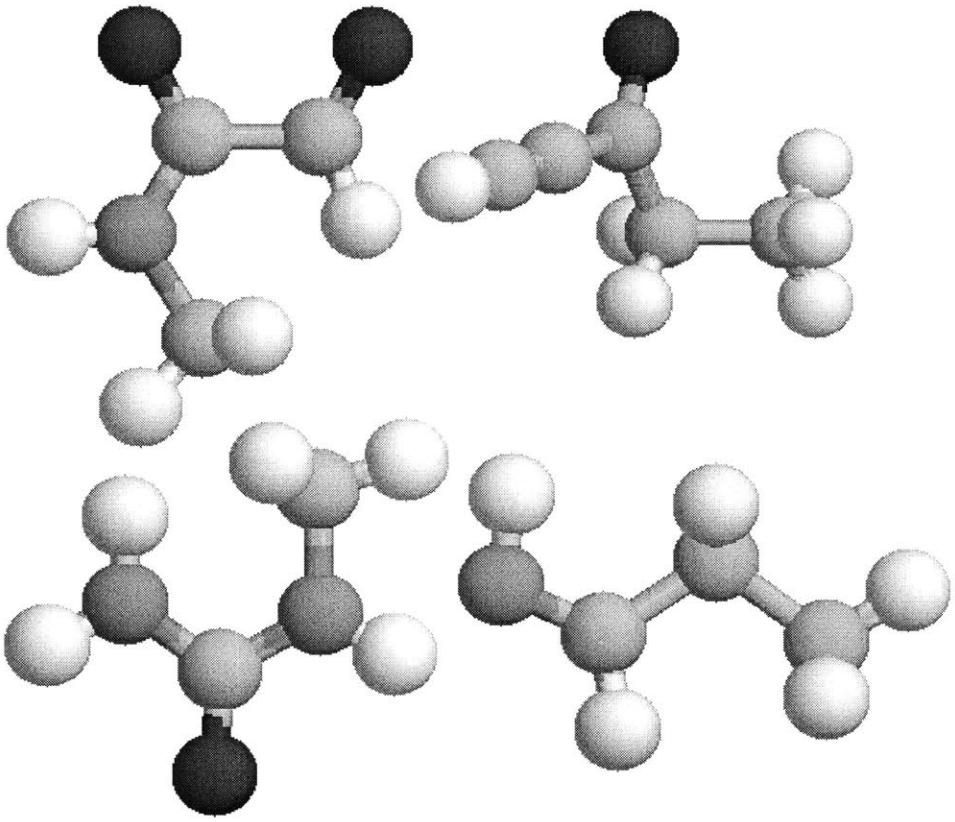

Figure 4-2: Sample output from a library made to explore a chemical space. The molecules from left to right, top to bottom, are (f2c3t0c8(f3c2t0c5f4)), (f3c2t0c8(f1c3t0c3(f2c3t0c2f3))), (f3c2t0c5f18) and (f3c2t0c2(f3c2t0c3f5)).

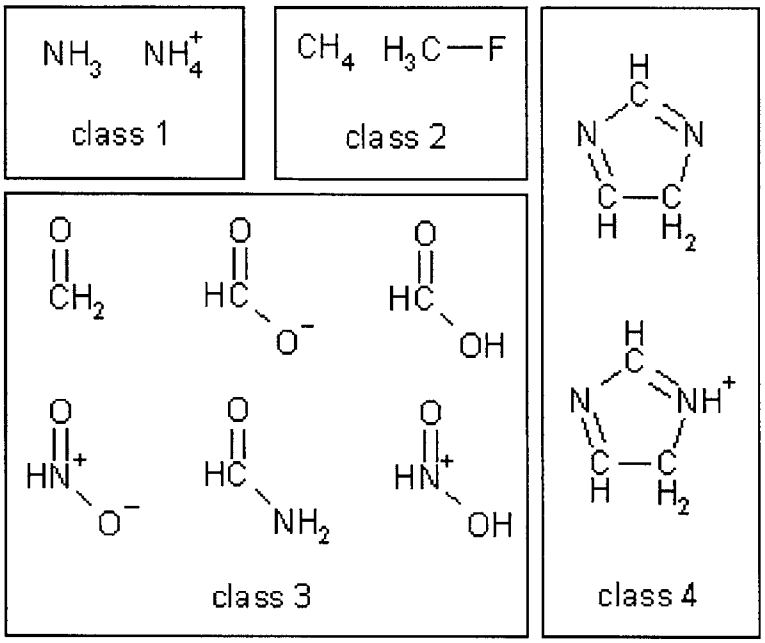

Figure 4-3: The molecules used in the generation of the amino acid histidine. The molecules were grouped into four classes and joined selectively to build up the target molecule.

tions by more explicit and computationally intensive methods.

To illustrate how "combine" can aid in the generation of a single molecule, four libraries of molecules were chosen from which the amino acid histidine was to be created. Each of these libraries contained a class of molecules which were similar in structure, but different in some other aspect, be that charge or constituent atoms. Each molecule within the class thus became interchangeable with another, so that diverse chemical derivatives of histidine could be made. "combine" was run a number of times to create intermediate libraries of structural elements that, when put together, would reconstitute histidine.

The process that allows construction of a 20 atom molecule, like histidine, started with small seed molecules. To build the histidine molecule the initial molecules selected in this case were ammonium, methane, formate and imidizol. These molecules are shown in Figure 4-4 on the next page. In the first round of class library joins, methane was joined with ammonium and imidizol. These second generation molecules are shown in Figure 4-5 on the following page. To create the protein backbone segment

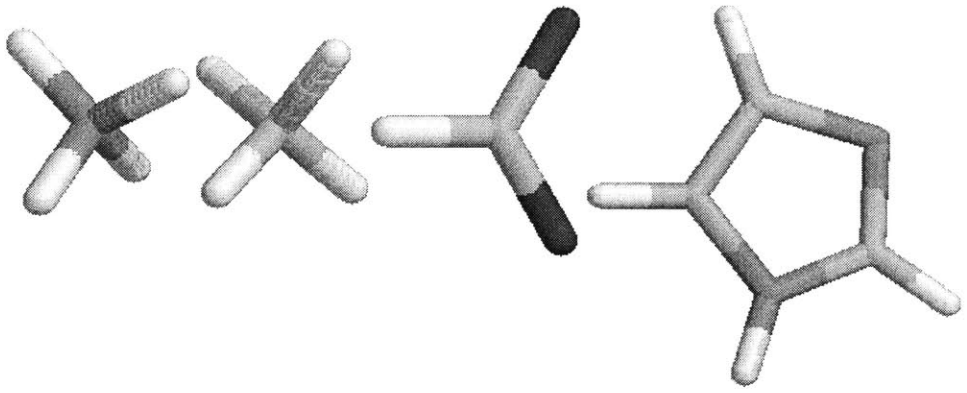

Figure 4-4: Initial molecules used for building histidine.

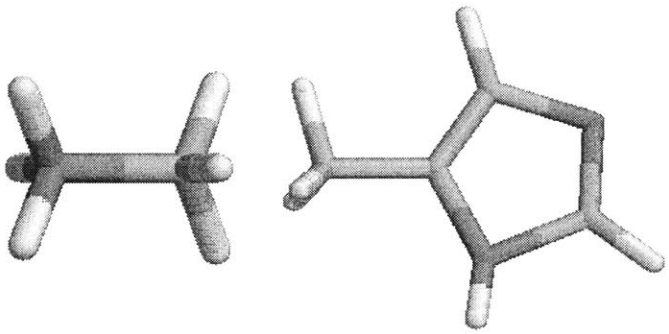

Figure 4-5: Second generation molecules.

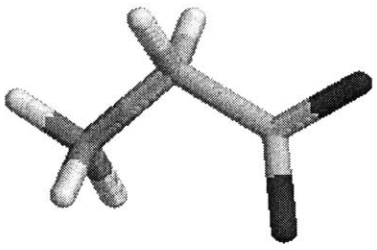

Figure 4-6: Third generation molecule.

of histidine, formate was joined with the ammonium and methane combination, or methylammonium. This created another amino acid, namely glycine. This molecule can be seen in Figure 4-6 on the page before. Finally, glycine was joined with the methane and imidizol combination, or methyl-imidizol, to create the amino acid histidine. Using just single minimum energy conformers, two confomers of histidine were made. These and the histidine produced by Quanta/CHARMM are shown in Figure 4-7 on the following page.

The histidines made by "combine" were not charge optimized at any point during the build-up. The partial charges existing on the molecules made by Quanta/CHARMM were taken from the parameter files. The joins made by "combine" used the charge balancing method described in section 2.2.4. Table 4.2 on page 59 shows the charges on each of the histidines illustrated in Figure 4-7 on the following page. Additionally, a quantum mechanical calculation was performed on the charges of the template histidine by Gaussian98 [16], a well known application in the field of computer-aided chemistry. There are many ways of approximating atomic partial charges, Table 4.2 on page 59 gives an idea of how much "combine"'s charge balancing technique throws these charges off.

When all the molecules in each class are allowed to join, the resulting library includes over 63,000 molecules. This library includes only the single minimum energy conformers, but contains molecules that were bonded on all "contacts" in the second and third generation joins. Among these were the molecules seen in Figure 4-7 on the following page. Other histidine derivatives made are shown in Figure 4-8 on page 60.

4.3 Current Limitations

Although "combine" can make large combinatorial libraries of correct and mostly unique chemical structures, there are a number of remaining limitations. "combine" works on computer hardware which at the present time will restrict it to produce on the order of 10 million fragments. Without increasing storage, the maximum library size can grow by a factor of 3, if the output library is compressed. Another

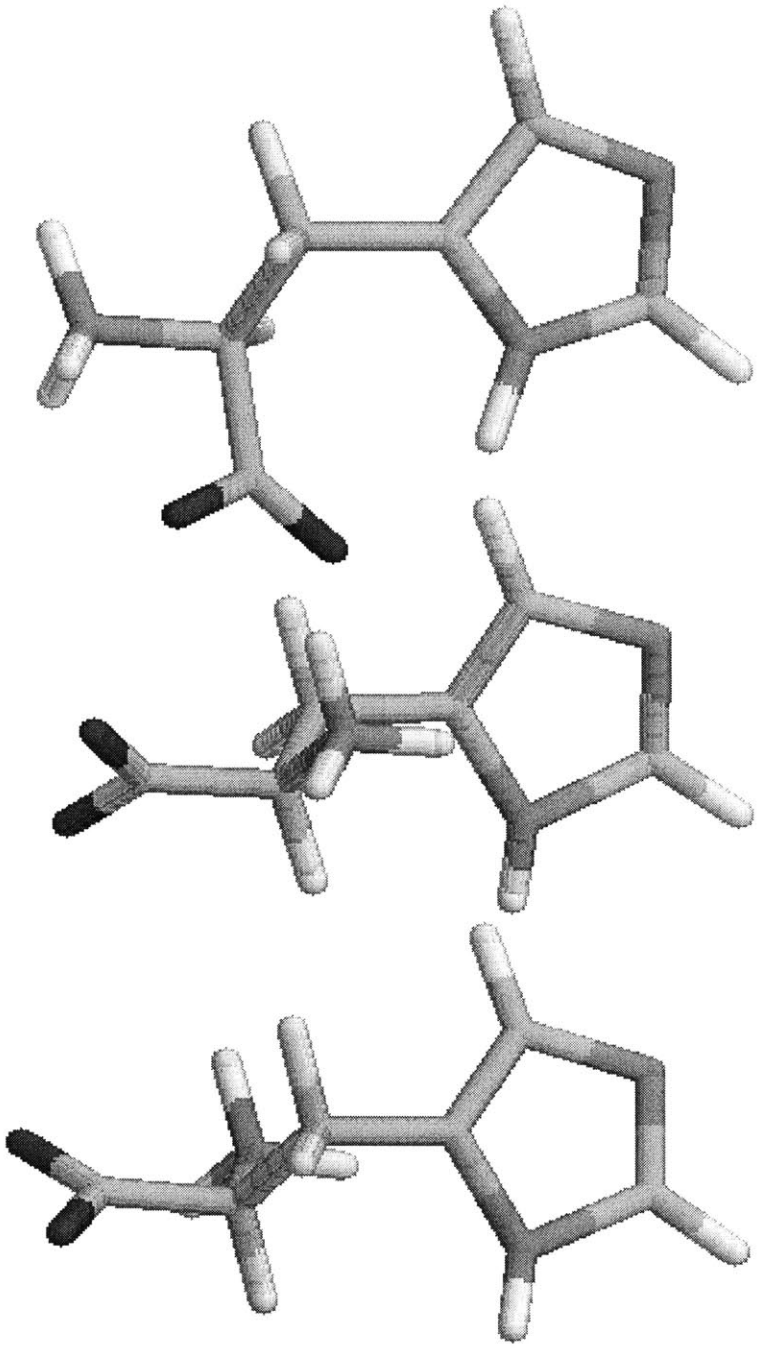

Figure 4-7: Histidines: the top molecule is the geometry optimized template made by Quanta/CHARMM, while the bottom two molecules were produced by "combine", using just minimum energy conformers.

Atom	Name	Type	$\mathbf{q}_T$	$\mathbf{q}_T^*$	$\mathbf{q}_N$
1	N	36	-0.30	-0.48	-0.22
2	H1	2	0.35	0.35	0.43
3	H2	2	0.35	0.34	0.43
4	H3	2	0.35	0.29	0.43
5	CA	10	0.22	0.20	0.12
6	CB	10	-0.08	-0.08	0.11
7	CG	21	0.12	-0.06	-0.18
8	ND1	34	-0.40	-0.31	-0.38
9	HD1	1	0.30	0.35	0.32
10	CD2	21	0.02	0.14	-0.18
11	NE2	34	-0.40	-0.62	-0.38
12	CE1	21	0.22	0.29	0.62
13	C	14	0.14	0.97	0.44
14	OCT1	43	-0.57	-0.80	-0.71
15	OCT2	43	-0.57	-0.82	-0.71
16	HA	3	0.30	-0.02	0.08
17	HB1	3	0.30	0.06	0.07
18	HB2	3	0.30	0.03	0.07
19	HD2	3	0.30	0.09	0.13
20	HE1	3	0.30	0.09	0.08

Table 4.2: Partial charges for the amino acid histidine. Of particular interest are atoms 6, 7, 10 and 16 whose charges change sign. Histidine charges, as generated by Quanta/CHARMM, are in the $\mathbf{q}_T$ column and by "combine", for the middle fragment of Figure 4-7 on the page before, in the $\mathbf{q}_N$ column. The $\mathbf{q}_T^*$ column shows charges for the Quanta/CHARMM fragment calculated by Gaussian98, a program that does quantum mechanical calculations of electron cloud distributions. Gaussian98 charges were calculated using the Hartree-Fock method, the 6-31+G* basis set and ChelpG population analysis [16]. "Type" column refers to the numerical atomic names as used by Quanta/CHARMM parameter files.

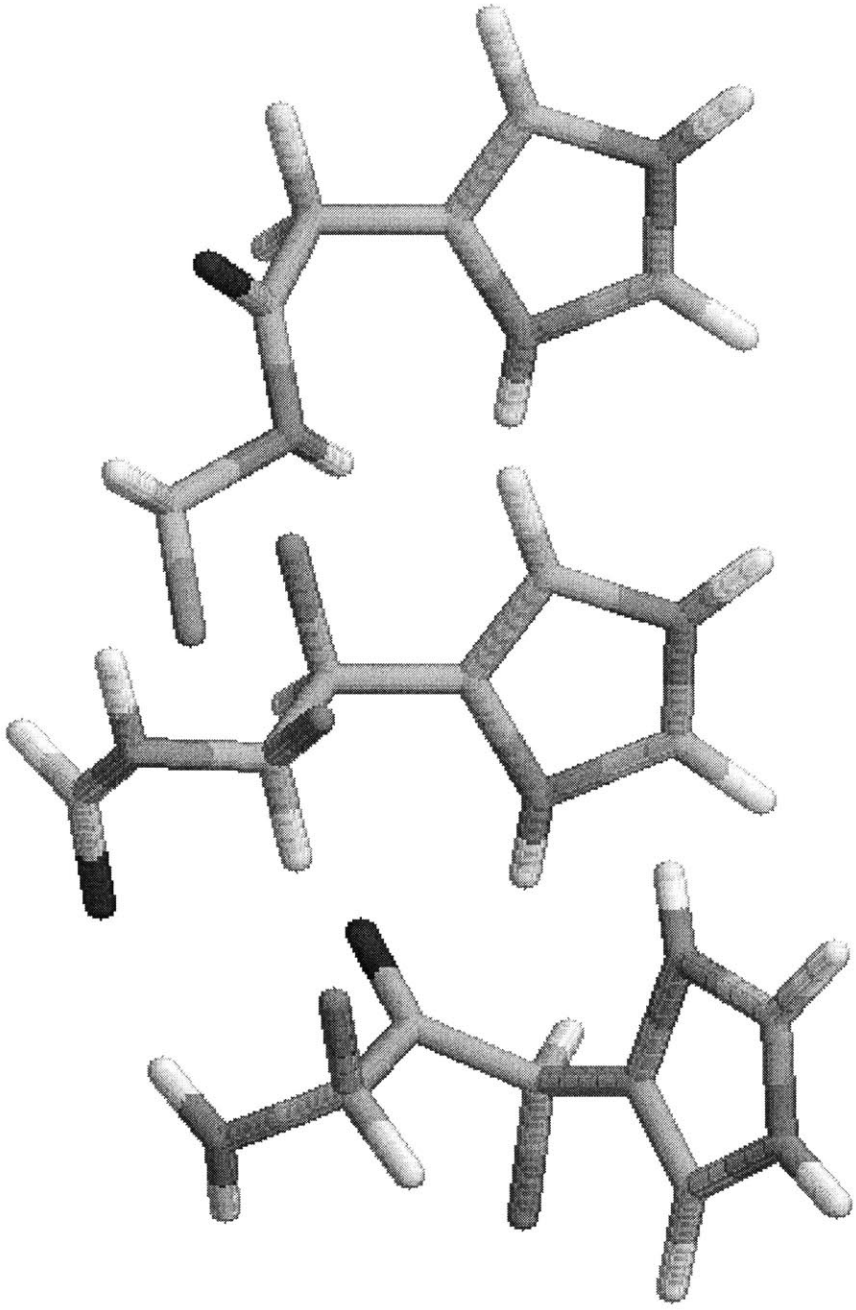

Figure 4-8: Sample molecules generated by "combine" from class libraries chosen to produce histidine derivatives.

way to restrict library growth is to limit the number of "contact" sites on the newly generated molecules. At this time, "combine" uses all "contacts" on second and further generation of molecules, whereas the initial fragments can have their "contact" sites explicitly specified. Thus, the hydrogen attached to the oxygen in a formic acid is a valid contact once another join has been made. Furhtermore, if one of the initial fragments is water, a water–water join is possible. To eliminate such combinations, a rule-based facility could be developed to manage how fragments are joined. Other alternatives, to restricting bonding sites, include designating custom atom types with proper parameters which would be appended to the Quanta/CHARMM parameter files.

The molecules produced by "combine", albeit structurally parametrized, are only approximations to minimized structures which are theoretically closer to real molecules. Quanta/CHARMM uses a fast parameter based method for optimizing the geometry of its structures. This is once again an approximation, but makes a better guess at the shapes molecules in response to molecular stresses. Gaussian98 can geometrically optimize structures according to the electron distributions around atoms. This is the most accurate, first-principles based calculation available to model molecules. It is also prohibitively expensive, with computation times for histidine optimization measured in days on currently available workstations.

When designing a drug for a specific target, a large factor of the molecule's efficacy lies in how well its charge distribution complements the electrostatic field of the binding domain. When "combine" joins two molecules, the atoms removed carry a certain charge which, by simple heuristic method described in section 2.2.4, is balanced so that the resulting charge of the product is equal to the sum of charges of the reactants. The current method alters every charge of a molecule, changing charges on domains near to and far from the location of the new bond. An alternative method would spread the surplus charge on only the host atoms of the bond. An explicit calculation can also be done by Gaussian98. Using simple basis sets, quantum mechanical computation of partial charges is relatively fast on small molecules. Which of these methods gives the best approximation to the actual atomic partial charges is

yet to be determined, but a combination of parameter-based geometry optimization and quantum mechanical calculation of charges could make the most promising drug candidates with much improved accuracy.

4.4 Future Applications

"combine" is a software package which can be extended easily to accommodate new methods and handle new molecule building domains. Two possible applications of "combine" were presented above, that of creating an extensive combinatorial library and of enumerating a diverse number of molecule derivatives from class libraries. In both of these cases, "combine" prototyped valid three-dimensional chemical structures that could undergo further optimization and investigation. This ability to build chemical structures could be extended to include only select parts of larger molecules. "combine" thus could combinatorially generate all possible varieties of a single side chain of a drug molecule or a protein.

Combinatorial chemistry is an approach that holds a large promise in the generation of new drug candidates and in the investigation of drug–protein interactions. These efforts require large libraries of unique fragments generated by combinatorial methods. A molecule building application, like "combine" which can prototype these libraries, can be essential to those studies and become one of the tools that are required to successfully design and develop new pharmaceuticals.

Appendix A

User Interfaces to "combine"

Contents

A.1 Operational Modes

```
Following is a list of possible run modes and tags they use: (* - optional)
Note: PARAMETERFILE, MASSFILE, by default, are assumed to be in local
      directory.

MODE = "SEQUENCE"
ARGS = "maximum number of atoms in fragment"
INPUTFILES, INPUTFORMAT, OUTPUTFILE, OUTPUTFORMAT, PARAMETERFILE*, MASSFILE*
FLAGS = "CONF REJECT ROT VDW WARN MEM"
    This mode sequentially joins fragments from an ascii/binary library input
    file.

MODE = "LIBRARY"
ARGS = "maximum number of atoms in fragment"
INPUTFILES, INPUTFORMAT, OUTPUTFILE, OUTPUTFORMAT, PARAMETERFILE*, MASSFILE*
FLAGS = "CONF REJECT ROT VDW WARN MEM"
    This mode joins fragments from an ascii/binary library input file.

MODE = "DISCRETE"
ARGS = "maximum number of atoms in fragment"
INPUTFILES, INPUTFORMAT, OUTPUTFILE, OUTPUTFORMAT, PARAMETERFILE*, MASSFILE*
FLAGS = "CONF REJECT ROT VDW WARN MEM"
    This mode joins fragments from discrete ascii/binary input files.

MODE = "CONVERT"
INPUTFILES, INPUTFORMAT, OUTPUTFILE, OUTPUTFORMAT
    This mode converts libraries from ascii to binary or vice-versa.

MODE = "DIGEST"
INPUTFILES, OUTPUTFILE
    This mode converts atom names into types in the parameter file. First input
    file is the mass file (eg. MASSES.RTF), the second is a parameter file
    (eg. PARM.PRM).

MODE = "STRING"
ARGS = "molecule string, i.e. (f1c2t0c2f1)"
INPUTFILES, INPUTFORMAT, OUTPUTFILE, OUTPUTFORMAT, PARAMETERFILE*, MASSFILE*
    This mode creates a single join from history specified,
    by the molecule string of the form (f1c2t0c2f1), from a library.

MODE = "JOIN"
ARGS = "<f1> <c1> <c2> <f2>"
INPUTFILES, INPUTFORMAT, OUTPUTFILE, OUTPUTFORMAT, PARAMETERFILE*, MASSFILE*
FLAGS = "CONF ROT VDW WARN PSF"
    This mode creates a single join between fragments f1 and f2,
    by using contacts c1 and c2, from a library. All these, f1, f2, c1, c2
    should be numbers, i.e. ARGS = "1 1 2 2".
    The ascii results are saved in <OUTPUTFILE#>.crd, where # stands for the
    torsion number (if 1, then it is not appended).
    The binary results are saved together in <OUTPUTFILE>.lib library.

MODE = "CLASS"
INPUTFILES, INPUTFORMAT, OUTPUTFILE, OUTPUTFORMAT, PARAMETERFILE*, MASSFILE*
FLAGS = "CONF REJECT ROT VDW WARN MEM"
    This mode joins libraries of fragments together, joining every fragment
    in one to every in the other. Produces an output library.

MODE = "SPILL"
INPUTFILES, INPUTFORMAT, OUTPUTFORMAT, PARAMETERFILE*
FLAGS = "PSF"
    This mode creates discrete files from a library. When PSF flag is
    specified, PSF file is NOT embedded in the ASCII CRD file. A legend file
    can be passed in as the PARAMETERFILE to recover original fragment
    names.

MODE = "GATHER"
INPUTFILES, INPUTFORMAT, OUTPUTFILE*, OUTPUTFORMAT
    This mode creates a library from listed ascii files.

MODE = "HELP"
    This mode prints the above list and a sample.ini file.
```

A.2 Command Line Usage

The "combine" application can be interfaced via the command line or a control file. The following is a template listing of the control line arguments. The layout of the control file follows.

```
Usage: main ["<file name>.ini" | options]
As default, main looks for "default.ini" file in current directory.
The command line is parsed if no .ini file is given.
Command line (CL) allowed switches are: (can be in any order)
 -md = mode:
       SEQUENCE, LIBRARY, DISCRETE, STRING, JOIN, CLASS - join modes
       SPILL, GATHER, CONVERT - fragment library modes
       DIGEST - parameter file conversion
       MODE_HELP - prints details on modes and needed parameters
       SAMPLE_INI - prints sample .ini file
 -if = input file(s) (multiple files => multiple CL switches)
 -it = input type (ASCII/BINARY)
 -of = output file (multiple files => multiple CL switches)
 -ot = output type (ASCII/BINARY)
 -pf = parameter file(s). Main file should be listed first.
 -mf = mass file. Location of MASSES.RTF file.
 -ag = argument(s) (multiple arguments => multiple CL switches)
 -fg = flag(s): (multiple flags => multiple CL switches)
CONF - enables checking for CONFormational symmetry.
REJECT - filters out redundant copies of molecules.
ROT - makes specified (in User.cc) ROTameric conformations.
VDW - enables checking for van der Waals clashes.
WARN - WARNings
MEM - enables MEmory Management, hard disk/memory paging.
PSF - embeds PSF file in discrete ASCII CRD files.
```

A.3 Sample Control File

The control file can be anywhere and of any name; by default the program will look for "default.ini" in the directory where "combine" was executed.

```
#
# This is a sample.ini file to be used with the combine application.
# All runtime options are set within this file.
# Created by Lukasz A. Weber, 2/11/99.
#

#
# Any lines that begin with a hash mark, i.e. '#', are treated as comments
# and thus ignored. The same with empty lines.
#

#
# First order of business: determine the functional mode of the application.
# Available choices are:
# SEQUENCE - sequential join from a library
# LIBRARY - combinatoric joins from a library
# DISCRETE - combinatoric joins from discrete files.
# CONVERT - library conversion between ascii/binary.
# DIGEST - pre-process the parameter file.
# STRING - create single join from molecule string, i.e. "(f1c2t0c2f1)".
# JOIN - join between 2 fragments w/ specified contacts all torsions are made.
# SPILL - creates discrete files from library.
# GATHER - creates a library from discrete files.
# HELP - prints help information and this sample.ini file.
#

MODE = "LIBRARY"

#
# Specify additional runtime arguments. This may get ignored by some modes.
# Varies with mode.
#

ARGS = "6"

#
# Specify input file(s).
# Multiple files should be separated by white space.
#

INPUTFILES = "fragment1.crd fragment2.crd /tmp/fragment3.crd"

#
# Select input file mode: ASCII or BINARY
#

INPUTFORMAT = "BINARY"

#
# Specify output file.
#

OUTPUTFILE = "out.lib"

#
# Select output file mode: ASCII or BINARY
#

OUTPUTFORMAT = "ASCII"

#
# Specify runtime flags. This varies and may get ignored with different modes.
# Options are: CONF REJECT ROT VDW WARN MEM PSF
# Multiple options should be separated by white space.
#

FLAGS = "CONF REJECT VDW"
```

```
#
# Define parameter file name. This must be a processed file. This field
# is used by modes that perform parameter look-ups only.
#

PARAMETERFILE = "allparm.prm"

#
# Define mass file name. This is the Quanta's Charmm MASSES.RTF file. This
# file should usually be found with the parameter file.
#

MASSFILE = "MASSES.RTF"

# End of file.
```

Bibliography

[1] Afferent Structure, 1999. http://www.afferent.com/products.html.

[2] C.I. Bayly, P. Cieplak, W.D. Cornell, and P.A. Kollman. A well-behaved electrostatic potential based method using charge restraints for determining atom-centered charges: The RESP model. *Journal of Phyical Chemistry*, 97:10269, 1993.

[3] T. L. Blundell. Structure-based drug design. *Nature*, 384(7):23–26, November 1996. Supplementary article.

[4] H. Bohm. On the use of LUDI to search the fine chemicals directory of ligands of proteins of known three-dimensional structure. *Journal of Computer-Aided Molecular Design*, 8:623–632, 1994.

[5] H. Bohm. Towards the automatic design of synthetically accessible protein ligands: Peptides, amides and peptidomimetics. *Journal of Computer-Aided Molecular Design*, 10:265–272, 1996.

[6] B. R. Brooks, R. E.Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4:187–217, 1983.

[7] A. Caflisch. Computational Combinatorial Ligand Design: Application to human alpha-thrombin. *Journal of Computer-Aided Molecular Design*, 10:372–396, 1996.

[8] A. Caflisch, A. Miranker, and M. Karplus. Multiple copy simultaneous search and construction of ligands in binding sites: application to inhibitors of HIV-1 aspartic proteinase. *Journal of Medicinal Chemistry*, 36:2142–2167, 1993.

[9] Chemscape Chime, 1999. For latest version, please see http://www.mdli.com.

[10] D. E. Clark, D. R. Westhead, R. A. Sykes, and C. W. Murray. Active-site-directed 3D database searching: Pharmacophore extraction and validation of hits. *Journal of Computer-Aided Molecular Design*, 10:397–416, 1996.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[12] J. M. Blaney D. C. Spellmeyer and E. Martin. *Computational Approaches to Chemical Libraries*, pages 165–194. Marcel Dekker, Inc., New York, NY, 1997.

[13] R. L. DesJarlais. *Generation and Use of Three-Dimensional Databases for Drug Discovery*, pages 73–104. Marcel Dekker, Inc., New York, NY, 1997.

[14] M. B. Eisen, D. C. Wiley, M. Karplus, and R. E. Hubbard. HOOK: A program for finding novel molecular architectures that satisfy the chemical and steric requirements of a macromolecule binding site. *Proteins: Structure, Function, and Genetics*, 19:199–221, 1994.

[15] T.J.A. Ewing and I.D. Kuntz. Critical evaluation of search algorithms for automated molecular docking and database screening. *Journal of Computational Chemistry*, 9(18):1175–1189, July 15 1997. Current version can be found at http://www.cmpharm.ucsf.edu/kuntz/dock4/dock4.html.

[16] 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. Mennucci, 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, and J. A. Pople. Gaussian 98 (Revision A.7). Gaussian, Inc., Pittsburgh PA, 1998. Official webpage available at www.gaussian.com.

[17] V. Gillet, A. P. Johnson, P. Mata, S. Sike, and P. Williams. SPROUT: A program for structure generation. *Journal of Computer-Aided Molecular Design*, 7:127–153, 1993.

[18] C. M. W. Ho and G. R. Marshall. DBMAKER: A set of programs to generate three-dimensional databases based upon user-specified criteria. *Journal of Computer-Aided Molecular Design*, 9:69, 1995. CONCORD was used in conjuction with DBMAKER.

[19] W. Humphrey, A. Dalke, and K. Schulten. VMD - Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14(1):33–38, 1996. For latest version, please see http://www.ks.uiuc.edu/Research/vmd/.

[20] T. Hurst. Flexible 3D searching – the directed tweak technique. *Journal of Chemical Information and Computer Sciences*, 1(34):190–196, January-February 1994.

[21] W. L. Jorgensen and J. Tirado-Rives. The OPLS potential functions for proteins. energy minimizations for crystals of cyclic peptides and Crambin. *Journal of American Chemical Society*, 110:1657–1666, 1988. OPLS stands for Optimized Potentials for Liquid Simulations.

[22] D. Joseph-McCarthy, A. A. Fedorov, and S. C. Almo. Comparison of experimental and computational functional group mapping of an RNase A structure: implications for computer-aided drug design. *Protein Engineering*, 9(9):773–780, 1996.

[23] J. H. Williams Jr. *Fundamentals of Applied Dynamics.* John Wiley & Sons, Inc., New York, 1996.

[24] P. Kollman, D. Case, K. Merz, D. Ferguson, T. Darden, and D. Pearlman. AMBER: Assisted model building with energy refinement. Home page on the WWW, 1999. For latest version, please see http://www.amber.ucsf.edu/amber/amber.html.

[25] C. Lemmen, C. Hiller, and T. Lengauer. RigFit: A new approach to superimposing ligand molecules. *Journal of Computer-Aided Molecular Design*, 5(12):491–502, September 1998. RIGFIT is part of the flexible superposition software FLEXS which can be accessed on the WWW at http://cartan.gmd.de/FlexS.

[26] R. A. Lewis and A. R. Leach. Current methods for site-directed structure generation. *Journal of Computer-Aided Molecular Design*, 8:467–475, 1994.

[27] A. Miranker and M. Karplus. An automated method for dynamic ligand design. *Proteins: Structure, Function, and Genetics*, 23:472–490, 1995.

[28] R. Sayle and H. J. Bernstein. RasMol molecular renderer. Downloaded from the WWW, 1999. For latest version, please see http://www.umass.edu/microbio/rasmol/.

[29] D. Sitkoff, K. A. Sharp, and B. Honig. Accurate calculation of hydration free energies using macroscopic solvent models. *Journal of Physical Chemistry*, 98:1978–1988, 1994.

[30] The MathWorks, Inc. MATLAB: Version 5.3.1.29215a (R11.1). Home page on the WWW., 1999. For product information visit www.mathworks.com.

[31] P. Walters and M. Stahl. Babel version 1.6. Downloaded from the WWW, 1996. For latest version, please write to babel@mercury.aichem.arizona.edu.