Systematic Conformational Search with Constraint Satisfaction

by

Lisa Tucker-Kellogg

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

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2002

Author .....

Department of
Electrical Engineering and Computer Science
May 24, 2002

Certified by .....

Tomás Lozano-Pérez
Cecil H. Green Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....

Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Systematic Conformational Search with Constraint Satisfaction

by

Lisa Tucker-Kellogg

Submitted to the Department of
Electrical Engineering and Computer Science
on May 24, 2002, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Determining the conformations of biological molecules is a high scientific priority for biochemists and for the pharmaceutical industry. This thesis describes a systematic method for conformational search, an application of the method to determining the structure of the formyl-Met-Leu-Phe-OH (fMLF) peptide by solid-state NMR spectroscopy, and a separate project to determine the structure of a protein-DNA complex by X-ray crystallography.

The purpose of the systematic search method is to enumerate all conformations of a molecule (at a given level of torsion angle resolution) that satisfy a set of local geometric constraints. Constraints would typically come from NMR experiments, but applications such as docking or homology modelling could also give rise to similar constraints.

The molecule to be searched is partitioned into small subchains so that the set of possible conformations for the whole molecule may be constructed by merging the feasible conformations for the parts. However, instead of using a binary tree for straightforward divide-and-conquer, four innovations are introduced: **(1)** OMNIMERGE searches a subproblem for every possible subchain of the molecule. Searching every subchain provides the advantage that every possible merge is available; by choosing the most favorable merge for each subchain, the bottleneck subchain(s) and therefore the whole search may be completed more efficiently. **(2)** A cost function evaluates alternative divide-and-conquer trees, provided that a preliminary OMNIMERGE search of the molecule has been completed. Then dynamic programming determines the optimal partitioning or "merge-tree" for the molecule; this merge-tree can be used to improve the efficiency of future searches. **(3)** PROPAGATION shares information by enforcing arc consistency between the solution sets of overlapping subchains. By filtering the solution set of each subchain, infeasible conformations are discarded rapidly. **(4)** An A* function prioritizes each subchain based on estimated future costs. Subchains with sufficiently low priority can be skipped, which improves efficiency.

A common theme of these four ideas is to make good choices about how to break the large search problem into lower-dimensional subproblems. These novel algorithms were implemented and the effectiveness of each is demonstrated on a well-constrained peptide with 40 degrees of freedom.

Thesis Supervisor: Tomás Lozano-Pérez
Title: Cecil H. Green Professor of Computer Science and Engineering

Acknowledgments

I am grateful to Professors Tomás Lozano-Pérez, Bruce Tidor, and Carl Pabo for their leadership and to the people at all ranks in the AI Lab, Tidor Lab, and Pabo Lab for fostering my development as a scientist.

I would also like to thank Barb Moore Bryant, Matt Cordes, Sue Felshin, Greta Kaemmer, Kimberle Koile, Helen Mazarakis, Lena Nekludova, Mark Round, Ellen Spertus, Amanda Tucker, Greg Tucker-Kellogg, Özlem Uzuner, Kathleen Wage, Ed Wang, and Patrick Winston.

Chapter 2 describes work done jointly with Tomás Lozano-Pérez. Chapter 3 is co-authored with Chad Rienstra, Chris Jaroniec, Morten Hohwy, Bernd Reif, Mike McMahon, Bruce Tidor, Tomás Lozano-Pérez, and Bob Griffin. Chapter 5 is co-authored with Mark Rould, Kristen Chambers, Sarah Ades, Bob Sauer, and Carl Pabo.

Helen Mazarakis provided graphic design assistance that was was particularly important in the preparation of this document. The More Magic group and the system administrators in Pabo Lab and Tidor Lab provided support for the computationally intensive research that led to this thesis. Financial support was provided by the Cecil H. Green Chair Fund, the NIH (grant GM31471), and an NSF Fellowship.

Finally, I would like reiterate my thanks to Tomás Lozano-Pérez because of the sheer quantity of his help, both intellectual and practical, including candid opinions, creative ideas, technical expertise, patient encouragement, uninterrupted financing, and a corpus of running code.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding the biological function of proteins and other flexible molecules often requires investigating the possible *conformations* or three-dimensional arrangements of atoms in the molecule. The famous "protein folding problem" is, in essence, the problem of finding the conformation with lowest Gibbs free energy. The problem addressed in this thesis is enumerating conformations that satisfy a given set of geometric constraints. The most important occurrence of this problem is when NMR spectroscopy experiments yield constraints on the geometry of atoms in a molecule (interatomic distances and dihedral angles in a molecule), and one then needs to determine what conformations of the molecule are consistent with those constraints.

Chapter 2 describes a systematic method for finding conformations that satisfy a user's constraints. Chapter 3 is an application of that method to determining the structure of a small peptide called fMLF (or formyl-Met-Leu-Phe-OH), based on constraints from solid-state NMR spectroscopy. Chapter 4 is a set of novel algorithmic enhancements to the methods of Chapter 2. Chapter 5 describes the determination of a large structure (a protein molecule bound to a double-helix of DNA) using a contrasting method, X-ray crystallography.

1.1 Context and Motivation

Protein Structure

Proteins, some of the most important types of molecules for life, are made from simple building blocks called *amino acids* or, after they've been connected, amino acid *residues*. The residues in a protein are connected sequentially to form a flexible chain. The spatial arrangement of a protein in three dimensions is called the protein's *conformation*, or its *tertiary structure* or, to emphasize the atomic-level resolution of the arrangement, the *atomic structure*.

Each amino acid building block has one of twenty possible *sidechains* attached to an invariant set of backbone atoms. *Peptide bonds* connect the backbone atoms of adjacent residues in a protein, thus forming one *polypeptide backbone* through the entire length of the protein. Because the various sidechains along the backbone have different chemical properties, they attract or repel one another, creating a complicated and non-uniform set of forces. The protein (both backbone and sidechains) bends and *folds* in response to the forces and often converges to one energetic minimum.

The sequence of amino acids, or primary structure, of a protein is widely believed to dictate the protein's tertiary structure, which determines the chemical activity and biological function. Other molecules besides proteins have similar structural freedoms and often have similar unique minimum conformations, but proteins are the canonical example.

Structure Determination

Atomic structures are too small to observe directly, such as with a microscope. Structure determination is experimentally difficult, but it is a high scientific priority for biologists, biochemists, and the pharmaceutical industry. Experimental methods for structure determination must somehow align or amplify signals from many copies of the same molecule, such as through crystallizing the sample so that all copies have precisely the same orientation (as for X-ray diffraction in Chapter 5), or subjecting the tiny magnetic dipoles in the nuclei of the atoms to an enormous magnetic field (as in NMR spectroscopy in Chapter 3). The raw data from such experiments can, with extensive and sophisticated processing, eventually provide sufficient information to deduce the structure of a molecule at atomic resolution. X-ray diffraction yields a map of electron density and NMR spectroscopy typ-

ically provides constraints on distances between particular pairs of atoms and constraints on dihedral angles for rotatable bonds. Some molecules are better suited to investigation by one method or the other, and both methods have become extremely popular.

With the growing number of scientists performing these experiments and with increasing automation of the structure determination process, software test suites [38, 72, 39, 55] have been developed to provide simple but automatic measures of the quality of an atomic structure. For X-ray crystallography, an analytical expression called the free-R [7] has been adopted to help reduce unintentional overfitting of atomic structures to the observed diffraction data. (The structure in Chapter 5 is noteworthy in part for the rigor with which the free-R was applied, relative to competing studies at that time.) It is conceivable that some atomic structures determined by NMR spectroscopy could also involve some overinterpretation of the available experimental data [14]. That is, there may be NMR constraint sets that permit a wider range of conformations than were originally identified. In any case, the ability to determine the full range of conformations consistent with a set of constraints would certainly aid in error analysis and confidence measures of structures determined by NMR.

A protein has hundreds or even thousands of degrees of freedom, and its conformation space (like configuration space in traditional artificial intelligence) is exponential in the number of degrees of freedom. Moreover, small increments in one variable can cause extreme and nonlinear changes in properties of interest (such as whether the constraints are satisfied). Absolute completeness in searching such a space may prove as elusive as protein folding itself. The primary goal of this thesis is to provide methods that guarantee some minimum interval of coverage, even though absolute completeness on the continuous space is never guaranteed.

The number of degrees of freedom that can be searched by our methods is not yet enough for application to typical protein structures. Therefore, as we make progress towards the eventual goal of systematically searching large structures, we address the important problem of searching smaller molecules, such as peptides.

The constraints on the fMLF peptide, determined experimentally by Rienstra *et al.*, include multiple distinct alternative ranges for many of the dihedral angles in the peptide. For example, the ϕ dihedral angle of methionine (the M in fMLF) was determined by experiments to be between -152.5° and -145.5° **or** between -92.5° and -85.5° **or** between -8.5°

and -3.5°. Systematic methods are particularly well-suited to considering disjoint alternative ranges for torsion angle constraints, and we applied our methodology to the problem of determining what conformations satisfy the fMLF constraints. By using a systematic method, we were also able to provide a guarantee that all conformations consistent with the constraints are either represented, within the margin of the search interval, or else are able to escape detection by virtue of being narrower than the search interval. By using a search interval of only 5° (for all but the two most unconstrained torsions in the peptide, which had a search interval of 30°), we provide an exceedingly high degree of confidence that there is no other region of conformation space that might satisfy the constraints but that somehow has been overlooked.

> The accuracy of structures determined from NMR data depends even in principle on several factors in addition to the quality of the data. The calculation is not a simple mathematical transformation, but involves a search of the protein conformational space to find all conformations compatible with the given set of experimental constraints. The adequacy of the search depends on the experimenter.
>
> Zhao and Jardetzky, 1994
> from *J.Mol.Biol.*, 239: 601–607.

1.2 Conformational Search

Conformational search algorithms, and systematic conformational searches in particular, are used for many applications including docking [20, 52, 77, 53], homology modelling [46, 15, 17], the pharmacophore problem [67, 69, 63], interpretation of crystallographic

electron density [76, 60], and determining stereospecific NMR assignments [26, 68, 24], as well as for interpreting NMR constraints [3, 66, 12, 29] for outright conformational analysis of important molecules [78], and for validating heuristic search and molecular design methodologies [47].

Some of the differences among conformational search algorithms include how they represent the molecular structures, what definition of success they use, and what constraints they place on the search. Some methods represent each conformation using Cartesian coordinates for the atoms. The most popular representation, "rigid geometry," assumes that the lengths and angles of covalent bonds are held fixed at their equilibrium values. The only remaining degrees of freedom, dihedral rotations about single bonds, can then be represented as a vector of torsion angles.

Most conformational search algorithms have an engine that generates many different starting structures from the breadth of the conformation space, and then another module that evaluates each conformation to determine whether it is acceptable. Search methods can then be categorized according to whether the engine for generating conformations is systematic or stochastic. A systematic engine (sometimes also called *exhaustive*) generates conformations deterministically, usually covering the conformation space at predefined intervals and in a predefined order. A stochastic algorithm generates starting conformations using some type of randomness. A simple stochastic algorithm might randomize the parameters of the conformation (the torsion angles or Cartesian coordinates of atoms) directly. A more popular approach uses chemically-inspired "move sets" to create a Markov random walk through conformation space, tweaking structures in ways that attempt to bias the sampling away from trivially unacceptable (self-colliding) conformations. Many researchers adjust the transition probabilities in a Markov walk to achieve importance sampling. The Metropolis algorithm samples states with probabilities proportional to their Boltzmann weights. The famous simulated annealing algorithm is simply a combination of the Metropolis algorithm with a slowly decreasing temperature parameter. "Distance geometry" algorithms [12], developed specifically for the NMR structure determination problem, search stochastically for conformations that can be "embedded" in matrices of interatomic distances.

Constrained Conformational Search

The constrained conformational search problem takes a primary structure (such as a protein sequence) and some constraints on the 3–dimensional conformation of the molecule.

The goal is to enumerate the conformations of the molecule that satisfy the constraints; that is, to output a list of atomic structures that "cover" the space of feasible conformations. (Let us leave "coverage" [65, 10] undefined at present.) The constraints are typically upper and lower bounds on interatomic distances, but they can also constrain torsional angles or even combinations of torsional angles. Whereas some formulations of the conformational search problem (sometimes called *restrained* search) involve ranking conformations according to preferences such as relative energies, the *constrained* version of the problem—the problem addressed in this thesis—defines conformations as either acceptable or unacceptable, with no preferences or relative order among the acceptable conformations.

Systematic Approaches

There is an extensive literature on systematic approaches to the constrained conformational search problem. For reviews, see the book by Andrew Leach [43], the article by Martin Saunders *et al.* [62], or other sources [34, 6, 4, 16, 61].

One important commonality among systematic approaches is that the user chooses the size of the interval(s) to use when generating starting structures. In other words, the coverage is an independent variable. In contrast, the independent variables for a stochastic search might be the number of trials, the length of time, or even the probability of coverage, but never the absolute minimum coverage.

In contexts where a guarantee is not required, stochastic searches can create good conformational hypotheses very quickly, even for large or underconstrained problems. Systematic methods are often too slow for large or underconstrained problems because of the number of satisfying solutions to be enumerated.

The systematic approach has been used often for small molecules of great interest such as drugs. Many of the earliest systematic methods were originally developed for rings, and rings remain a popular application for systematic methods [30, 25, 66, 36, 56, 75]. The systematic approach has also been used for searches where the backbone is fixed and only the sidechain conformations must be searched [41, 40, 44].

The goal of the thesis is not to compare systematic with stochastic methods, nor to argue for the superiority of systematic methods. Both systematic and stochastic methods are useful and important. Stochastic methods are currently used far more widely than systematic methods, and the vast majority of conformational search questions that are of interest to chemists and biologists tend to be cast automatically as problems for stochastic search. By extending the state of the art in systematic search, we hope both to tempt and to allow these scientists to cast their questions as problems for systematic search.

1.3 Overview

This thesis consists of a method for systematic conformational search and two structure determination projects. The conformational search method is the centerpiece and it includes several novel ideas, with demonstrations of how each innovation contributes to algorithmic efficiency. The structure of the fMLF peptide is determined using the systematic search method. The second structure, a protein-DNA complex determined using traditional X-ray crystallography, is methodologically separate.

In the body of the thesis, our description of the systematic search method starts with elementary search algorithms and then builds up our method as a series of modifications or new levels beyond the elementary algorithms: the voxel model, divide-and-conquer, OMNI-MERGE, merge-tree optimization, propagation, and prioritizing subchains with an A* cost function.

Voxel Model

The voxel model, proposed by Tomás Lozano-Pérez, divides conformation space into small hypercubes called voxels, and the goal of the search is recast in terms of continuous voxels, instead of discrete points. We attempt to enumerate all voxels that contain any satisfying region of conformation space, instead of enumerating regularly spaced points that satisfy the constraints.

One advantage of this model is that it allows the issue of completeness to be addressed at the level of local search, not just as a global problem. However, the primary motivation for using the voxel model is that it can provide a systematic guarantee of how finely the space has been examined (namely at the width of the voxels), while also allowing the

use of non-systematic methods at the local level. Within a voxel, we can evaluate points using randomness, heuristics, or any search method with good empirical performance. Our method uses minimization to evaluate and search each voxel.

Divide-and-Conquer

The second improvement uses divide-and-conquer to decompose the search of the full molecule into many small searches. Other groups have also applied the idea of divide-and-conquer to conformational searches (sometimes called "buildup procedures" or "combine operations").

We divide the molecule chain into smaller and smaller subchains, enumerate the solutions for the smallest subchains, and then combine the solutions for small subchains to create solutions for larger subchains, until solutions have been enumerated for the whole molecule. A "merge-tree" directs how the chain is divided into subchains, and we show that the choice of merge-tree can be critically important to the efficiency of a divide-and-conquer search. The novel portion of this work is a comparison of the empirical performance of divide-and-conquer with alternatives. In addition, we clarify the distinction between the size of a subchain and the size of a subproblem.

OmniMerge

We propose an algorithm that searches all possible subchains instead of just the particular subchains specified in one merge-tree. Quite counterintuitively, the run time of OMNI-MERGE is sometimes better than divide-and-conquer. If the subproblems defined by a particular merge-tree do not exploit the available constraints very well, it may be preferable to solve all subproblems (including the ones that don't exploit the constraints very well) rather than to solve only the default subproblems.

The key insight is that during most divide-and-conquer searches, most of the run time is consumed by one merge operation, usually the final merge of the right and left halves of the molecule. Because OMNIMERGE searches all subchains, it has many alternative left and right fragments to choose from when confronting the final bottleneck step. For a molecule with 10 residues, there will be 9 choices: one residue on the left and nine on the right, two on the left and eight on the right, etc. Although the time required to search every possible fragment can be substantial, is can easily be less than the time to perform the final merge

using a poor default choice of fragments. A crucial concept throughout this research is to improve the speed of searching a chain by making a good choice of which right and left fragments to combine.

Merge-Tree Optimization

We develop cost functions to evaluate alternative merge-trees, based not just on minimizing the size of the *subchains* but the size of the *subproblems*. Instead of assuming that the length of a subchain reflects the complexity of searching it, we consider the number of solutions for a chosen subchain as well as the number of solutions for the subsequent subchains implied by the first choice of subchain. The cost functions we develop in the merge-tree optimization section require knowing how many satisfying voxels there are for each subchain. Once the information is available, the cost function is amenable to optimization by dynamic programming and an optimal merge-tree can be computed.

Searching a molecule with OMNIMERGE provides the required information, the number of satisfying voxels for each subchain. Several other circumstances may also necessitate searching the same molecule more than once, but in general one can always search a molecule once at low resolution, compute the optimal merge-tree for low resolution searches of the molecule, and then use that same merge-tree for a higher resolution search.

Propagation

When we search every subchain of the molecule, many of the subchains will overlap each other, which is a form of redundancy. Any time we have two subchains that share at least one residue, we would like to reuse some of the information about what torsion angles are allowed or disallowed for those residues. We discovered a way to formalize this desire with the concept of arc consistency (or constraint propagation), a common artificial intelligence technique for general constraint satisfaction problems. However, the name "constraint propagation" may be misleading because instead of applying the technique directly to the constraints on conformations (the distance constraints and dihedral constraints provided by NMR experiments), we create new, higher-level constraints requiring that overlapping subchains must have solution sets that are compatible with each other. To be more precise, each conformation of one subchain must be compatible with at least one conformation of the overlapping subchain, or else it is illegal. We use a simple propagation algorithm to enforce

arc consistency between the solution sets of overlapping subchains. This reduces the number of candidate conformations for each subchain and results in a substantial improvement to run time.

Ordering Subchains

The final and most complex innovation we present is the idea of choosing the order in which to search the subchains. If the cost of searching a subchain will be high, it is postponed until all better alternatives have been searched. If a subchain is postponed until after the whole molecule chain has been searched, that is the same as skipping it entirely.

The flexibility of skipping arbitrary subchains opens up an exponential breadth of possible "merge-strategies." Merge-trees are some of the possible merge-strategies, but another merge strategy might begin evaluating one merge-tree and then skip to another if the first is disappointing.

When choosing which subchain to search next, we evaluate each according to an A* cost function that reflects both the cost of searching the subchain and an optimistic estimate of the potential contribution of the subchain to the search of the whole molecule.

Commonality

A common thread running through many of our ideas is investing computational resources in order to identify which subproblems will be cheapest to solve. The extra computation involved, whether searching small subproblems or calculating the costs of alternatives, can pay off dramatically if it allows the overall method to avoid searching a subproblem that is astronomically time consuming. Thus, the potential benefit of the computational investment may be enormous, whereas the worst-case cost from using these methods is limited to the time spent performing the extra calculations. In the final section of Chapter 4 we examine cases expected to have minimal benefit from the extra investment, but even in these cases, the combined package of improvements in our method still manages to provide more than enough benefit to compensate for the "overhead" time of using it.

Chapter 2

Systematic Conformational Search

To explain the search algorithm we have designed, we start with basic search methods and then explain the modules we have designed as improvements to the basic framework. The most important improvements in this chapter are using minimization to evaluate small regions of space, and using divide-and-conquer to build up the whole solution using solutions to smaller (lower-dimensional) subproblems. Chapter 4 will continue from there to explain further improvements that build upon divide-and-conquer.

2.1 Elementary Search

Suppose the molecule to be searched has N rotatable bonds as the only degrees of freedom (the "rigid geometry" model), and suppose one has chosen some resolution, say 120 degree increments in each dimension of torsion space, as constituting the desired level of detail for enumerating satisfying conformations.

Gridsearch

The simplest systematic search algorithm is gridsearch, a type of generate-and-test search. If a grid (a higher-dimensional cubic lattice) were drawn over the N-dimensional search space with, in this case, 120 degree spacing in each dimension between adjacent points, gridsearch would iterate over every gridpoint and evaluate the molecular conformation corresponding to each point; hence the name. The only noteworthy features of gridsearch are

simplicity, ease of implementation, and low overhead. In the pseudo-code below,[1] the subroutine that converts a torsion representation of a structure into an all-atom representation in Cartesian coordinates is called INSTANTIATECONFORMATION.

```
GRIDSEARCH
For θ1 = 120°, 240°, 360° {
    For θ2 = 120°, 240°, 360° {
            ⋮
        For θN = 120°, 240°, 360° {
            Structure S = INSTANTIATECONFORMATION([θ1, θ2, ..., θN]).
            Output S if it satisfies all the constraints.
        }
            ⋮
    }
}
```

Gridsearch is a form of British Museum Search, which is to say it simply generates and checks every possibility. The run time of GRIDSEARCH is always exponential in the number of torsions.

Treesearch

Backtracking is an elementary systematic algorithm for solving constraint satisfaction problems, and when applied to conformational search problems it is often called Treesearch [48]. Treesearch differs from gridsearch because it evaluates each partial vector of torsion angles each time the vector is extended or altered, instead of waiting until angles have been provided for all the torsions.[2] Provided that the torsions are ordered according to adjacency in the molecule, each partial vector of torsions, called a partial path in search terminology, corresponds to a unique conformation for a contiguous fragment of the molecule. (The fragment contains all atoms in the bond frames of the assigned torsions. Bond frames are

[1] For simplicity, we write the pseudo-code with N nested loops as if N were an absolute constant. Whether the algorithm is actually coded using explicit iteration or recursion does not change the substance of the algorithm.

[2] We draw no distinction [24] between the iterative and recursive forms of this algorithm.

explained in appendix A.) Treesearch evaluates each fragment it creates before extending the instantiation. A fragment can be evaluated by determining whether the constraints active on (or relevant to) that fragment are satisfied. (Active constraints are discussed in section 2.3.)

```
TREESEARCH
For θ1 = 120°, 240°, 360° {
   Structure s1 = INSTANTIATECONFORMATION([θ1]).
   If structure s1 satisfies all constraints active on it,
   Then For θ2 = 120°, 240°, 360° {
      Structure s2 = INSTANTIATECONFORMATION([θ1, θ2]).
      If structure s2 satisfies all constraints active on it,
         Then For θ3 = 120°, 240°, 360° {
                ⋮
                Then For θN = 120°, 240°, 360° {
                   Structure SN =
                      INSTANTIATECONFORMATION([θ1, θ2, θ3, ..., θN]).
                   Output S if it satisfies all the constraints.
                }
                ⋮
   } } }
```

The pseudo-code for Treesearch is like that for Gridsearch except that an evaluation is done after the assignment step of every loop, not just after the assignment in the innermost loop.

Comparison

The advantage of Treesearch over Gridsearch is apparent if you consider what happens when some fragment smaller than the whole molecule actually is found to violate some constraint. Whereas gridsearch would automatically evaluate all extensions of the bad start, treesearch *prunes* away that whole branch of the search tree and skips ahead to the next fragment. See Figures 2-1 and 2-2.

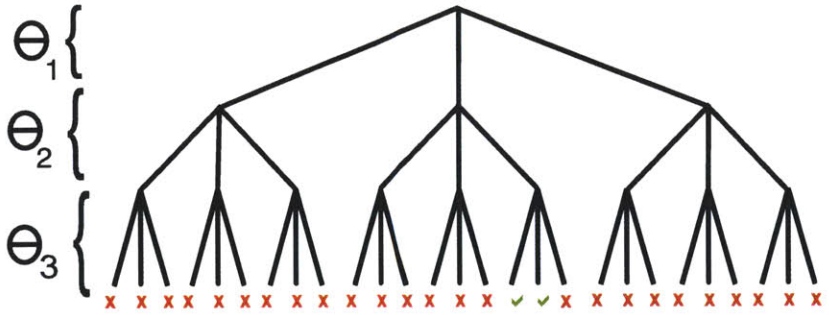

Figure 2-1: A depiction of how Gridsearch might sample three angles per bond on a molecule with three bonds. All branches are explored and evaluation only occurs after angles have been assigned to all bonds.

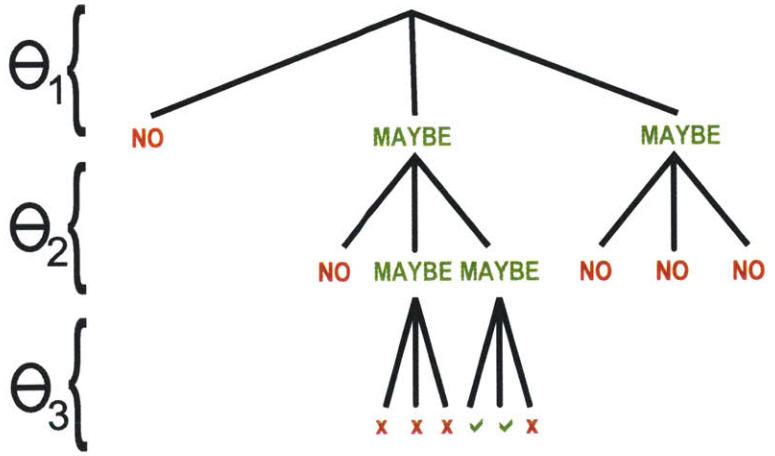

Figure 2-2: A depiction of Treesearch (or backtracking) showing the evaluation ("MAYBE" or "NO") for conformations of partial fragments of the molecule. The branches that appear in gridsearch (above) and not in treesearch are said to be "pruned."

When pruning occurs, the number of conformations pruned is exponential in the number of uninstantiated torsions. When few variables are instantiated, the probability of finding violations of an active constraint may diminish, but the reward from such a find is much larger.

The run time of TREESEARCH can be exponential, like GRIDSEARCH, but it can also be polynomial if constraints prune most of the branches. In the extreme case where all but one possibility is pruned for every torsion, the run time of TREESEARCH could even be linear in the number of torsions. See Figure 2-3.

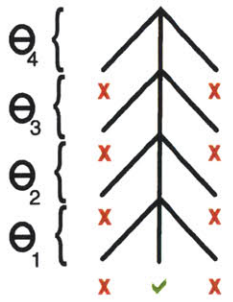

Figure 2-3: An ideal case for TREESEARCH.

One disadvantage of treesearch is that N evaluations of overhead must occur before the first whole conformation can be considered for output. Gridsearch in contrast has virtually no overhead. If no constraints are active on fragments smaller than the whole molecule,[3] then treesearch will perform worse than gridsearch. However, in circumstances where some constraints are active on smaller pieces of the molecule (which means fragments smaller than the whole can be disqualified and the search pruned), then treesearch will be a clear winner.

Completeness

Note that when treesearch excludes a high-dimensional slab of space based only on evaluating a lower-dimensional fragment of the molecule, the algorithm's completeness guarantee, such as it is, does not diminish. Higher-dimensional versions of the same gridpoints would

[3]One such circumstance might be if the trial conformation is constrained such that the intensities of its overall diffraction pattern must match a set of crystallographic intensity data.

have been excluded by the same constraints during a gridsearch.

The conformations that correspond to regularly spaced gridpoints are not necessarily representative of all nearby conformations. Satisfactory regions of conformation space that don't intersect the chosen gridpoints will be missed. While one might be tempted to think of all systematic search methods as guaranteeing complete coverage of the space, in fact these systematic methods only guarantee that some conformation in each region of space was considered.

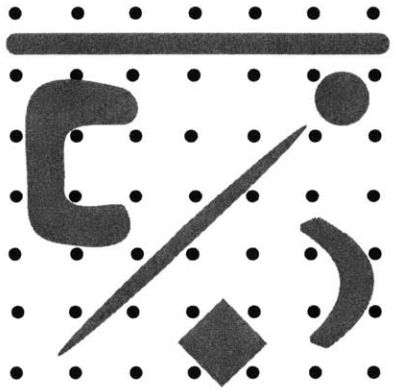

Figure 2-4: A two-dimensional example showing how a variety of shapes with non-trivial volumes can theoretically exist without intersecting any of the gridpoints.

Intuitively we understand that if a satisfactory region of conformation space exists and was not identified by the search, then that region must somehow be "narrow" enough to have avoided detection. Figure 2-4 shows a variety of large shapes that are "narrow" enough to avoid a 2-dimensional grid. The human instinct to consider such grids "thorough enough" to exclude significant volumes is what allows magicians to amaze audiences by placing an assistant in a box and then sticking swords through the box.

Rotamers

Because biologically interesting proteins are generally folded into low-energy conformations, algorithms like gridsearch or treesearch sometimes turn out to work very well for conformational search problems when their search intervals coincide with the patterns of low-energy conformations that we will call *rotamers* [58, 54].

Although single covalent bonds generally permit full rotation, there are certain angles of rotation about a bond that cause slight steric clashes between its neighboring atoms,

and some that avoid or minimize the clash. See Figure 2-5. Suppose two large atoms are

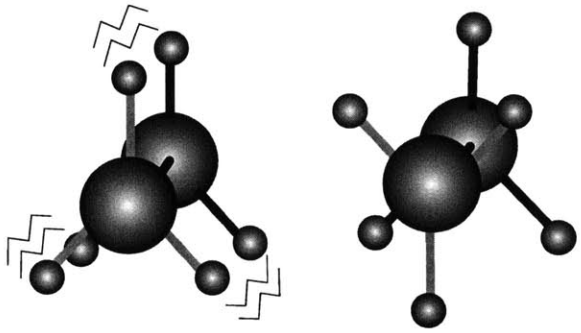

Figure 2-5: For a pair of bonded atoms with tetrahedral geometry, torsional angles of 120°, 240°, or 360° cause steric clashes, and torsional angles of 60°, 180°, and 300° do not.

connected by a covalent bond, and suppose each has tetrahedral geometry, with bonds to three other small atoms. Even though the three atoms on each side are pointing away from the central bond, torsional rotations that place the small atoms in closest proximity (in an "eclipsed position") cause steric clashes and are less favorable than other rotations.

These clashes, combined with the fact that energetically unfavorable conformations are improbable, cause the set of likely torsional angles to be divided into distinct ranges. For example, if steric clashes are minimized when a torsional angle is at 60, 180, or 300 degrees, then the most energetically favorable (and the most probable) conformations for that torsional rotation will have angles in the ranges of $60 \pm \delta$, $180 \pm \delta$, or $300 \pm \delta$ degrees, for some value of δ. With two adjacent rotatable bonds, the spread of likely conformations looks like a grid. See Figure 2-6.

Rotamers were originally defined as clusters of sidechain conformations that correlate with locally optimal torsion angles and that were found to occur frequently in X-ray crystal structures [58]. We will use the term rotamers more generally to refer to any likely or frequently-occurring neighborhood of conformations for any molecular fragment [49]. Some modern conformational search algorithms even use rotamers explicitly as their representation of conformation space [13, 44].

When our algorithm searches for conformations that satisfy constraints, we will not generally require the conformations to be in rotamers, but we would like the option of searching rotamers first or of calibrating the resolution of our search to match regular intervals of rotamers. This corresponds to the "succeed first" approach to variable-ordering in generalized constraint satisfaction problems. For example, the TREESEARCH algorithm

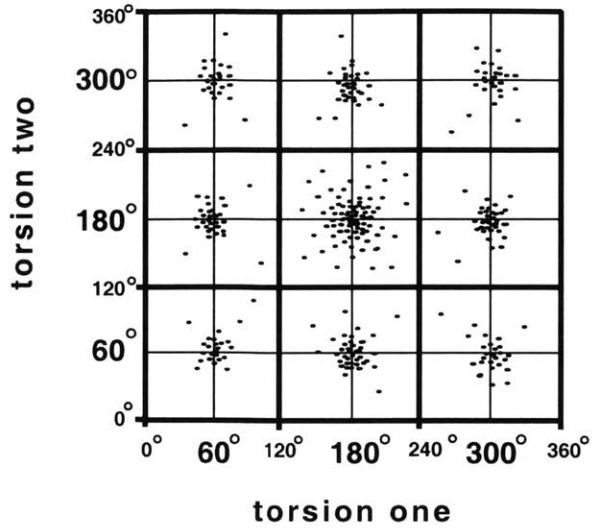

Figure 2-6: A hypothetical scatter plot of angles for two dihedral angles with tetrahedral geometry.

using 120° resolution should instead use the following assignments in place of the original values:

For $\theta_i = 60°, 180°, 300°$

2.2 Evaluating Voxels with Minimization

Conformation space is continuous, even though we discretize it for our representations. Consider again a grid drawn over the N-dimensional search space, with a given level of resolution, but instead of evaluating the conformations that happen to correspond to the grid points, consider evaluating the little multi-dimensional volumes of conformation space, the *voxels* defined by the grid. (The term voxel refers to a unit of higher-dimensional volume by analogy to a pixel as the unit of a two-dimensional picture.) Instead of asking whether each gridpoint conformation satisfies the constraints, we ask whether *there exists any* conformation in the voxel that satisfies the constraints. The voxel model is a previously unpublished idea of Tomás Lozano-Pérez. This thesis is the first application, test, and demonstration of performance for the voxel model. Because the voxel model is such a simple, readily-implemented idea, it may prove to be the most popular aspect of our systematic search methodology.

A one-dimensional voxel is a *range* of torsions, such as $[30, 60]$, and a d–dimensional voxel

consists of d independent, simultaneous ranges for d adjacent torsions. A set of torsions are adjacent if they are connected in the original molecule by a path of bonds that does not include any other rotatable bond (that is, if they form a connected component of the bond tree described in appendix A).

There is no general, perfectly accurate way to determine whether a satisfying conformation exists in a given volume. However, we can certainly do better than evaluating one point per volume.

We will often use a satisfying point in a voxel as an easy and useful representation for the whole voxel. Rather than enumerating satisfying conformations that happen to fall at regularly-spaced intervals, the goal of the search problem will now be to output one satisfying conformation from each voxel that contains a feasible region. We now formulate VOXELIZED TREESEARCH, the treesearch algorithm in terms of voxels instead of points. Whenever convenient, we will try to center our voxels on rotamers.

```
VOXELIZED TREESEARCH
For Range θ1 = [0–120°], [120–240°], [240–360°] {
   If voxel (Range θ1) has a satisfying conformation,
   Then For Range θ2 = [0–120°], [120–240°], [240–360°] {
      If voxel (Range θ1, Range θ2) has a satisfying conformation
      Then For Range θ3 = [0–120°], [120–240°], [240–360°] {
         ⋮
               Then For Range θN = [0–120°], [120–240°], [240–360°] {
                  If voxel (Range θ1, Range θ2, ..., Range θN)
                  has a satisfying conformation,
                  Then output that conformation.
               }
         ⋮
   }
}
```

Minimization as search

Gradient descent and other types of local optimization can be powerful search tools if local optima don't provide too great an obstacle to exploration. Motivated by the hope that our voxels might be sufficiently small to permit minimization to traverse much of the voxel before getting stuck, we chose local minimization (as opposed to explicitly global methods) for finding satisfying conformations within a voxel. Minimization of potential energy functions and other non-energetic objective functions has long been a popular tool for all types of conformational search and conformational analysis, usually alone but occasionally in conjunction with explicitly systematic search methods [45, 43].

Constraints on the conformation of a molecule can be trivially converted to an objective function by summing the squared violations of the constraints. Assigning proper relative weighting to disparate types and units of violation is less obvious. Extrapolating from a case that involved experimental data, we chose weighting coefficients proportional to the inverses of the standard deviations of the experimental measurements that gave rise to the constraints. If an additional standard deviation in the data for the distance constraints would be 0.1 Å on average, and 2° on average for the torsion data, then a violation of 0.1 Å in the distance constraints should receive the same absolute penalty as a 2° violation of the torsion constraints.

Note that the objective function created here has nothing to do with potential energy or with ranking the desirability of conformations. The constraints determine whether or not a conformation is acceptable, not how good it is relative to other good conformations. We use minimization of a constraint violation function as a heuristic for searching within a voxel for satisfying conformations. The ensemble of conformations output by our algorithm would still have to undergo energy minimization before they could be screened according to energetic criteria.

The minimizer always starts from some initial point in the voxel, perhaps the same point that gridsearch would have chosen, and then searches within the voxel for points that minimize the objective function—that is, for conformations with smaller violations of the constraints. The minimization halts when the violation function is sufficiently close to zero (success) or when a maximum number of iterations has been reached (failure).

For the actual minimizer, we tried a quasi-Newton minimization method [59] and we

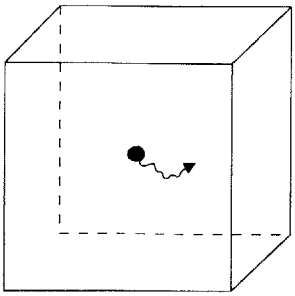

Figure 2-7: The search within a voxel is guided by minimization of constraint violations.

added additional penalties in the objective function to prohibit solutions outside the voxel being searched, but we found that a more effective and efficient method in practice is FSQP [42] (Feasible Sequential Quadratic Programming), a suite of constraint-based nonlinear programming methods that never considers points outside the restricted domain of the voxel. Note that for our use of this optimization software, the constraints on the molecule (such as the limits on interatomic distances) had already been incorporated into the objective function, and what the FSQP saw as "constraints" were merely the voxel boundaries. In either case, the user would specify a starting point in the voxel, a maximum limit on the number of steps or *iterations* the minimizer can spend searching for a solution, and possibly also parameters for multiple passes to reset the minimizer to different locations in the voxel. With either method, the result of the minimization attempt(s) would become the answer to whether a satisfying conformation exists within the voxel.

Minimization within Treesearch

If voxels are defined (and iterated over) by a treesearch algorithm, the first voxels defined will have only one dimension, then two dimensions, and so on. Treesearch defines a $(d+1)$–dimensional voxel only if it is an extension of a d–dimensional voxel that was already found to contain a feasible region. When the minimizer begins evaluating the $(d+1)$–dimensional voxel, we can initialize its starting conformation such that the first d torsions have the same solution found previously. Then only the newest, $d+1$ torsion must be chosen arbitrarily.

Note that this reuse of previous information is not relevant for the basic version of treesearch because without minimization, the midpoints (or gridpoints or rotamer-centers or whatever points represent the voxels) are the only conformations evaluated. Without

minimization, the same conformation is chosen for evaluation whether one chooses the midpoint of a voxel and then extends its dimension, or if one extends the voxel and then chooses its midpoint.

When running treesearch with minimization, we choose in many cases to perform two or more passes of minimization, if the first pass fails. The first pass uses the d-dimensional solution found at the previous level of the search, initializing only the $(d+1)$-st dimension arbitrarily to the center of the range. The second pass uses the midpoint of the voxel, which is the middle of the range for all $(d+1)$ dimensions. When we do perform more than two passes, the additional starting points are chosen at random from within the voxel.

Multi-resolution Search

Another strategy we considered for evaluating whether a voxel contains satisfying conformations, assuming that the voxel has already failed to yield any satisfying conformations during some preliminary, low-resolution search, is to search within the voxel systematically at a higher level of resolution.

For example, if a user is interested in knowing how many 120° rotamers satisfy a set of constraints, voxels would be defined with 120° resolution. Some voxels might have a large obvious region of satisfying conformations that could be found just by evaluating the midpoint of the voxel or by performing a minimization-search starting from the midpoint of the voxel. Other voxels that fail at this level of search might nonetheless contain small or obscure feasible regions, and for these case we would like to search more closely. For example we could begin a gridsearch at 30 degree resolution within the voxel and quit the gridsearch as soon as one feasible solution in the voxel is found.

Performing gridsearch as a second-pass filter on voxels that fail a first-pass is very different from performing gridsearch on the entire space. Not only will we avoid performing gridsearch on voxels that have "obvious" satisfying conformations, but we will avoid performing gridsearch on d-dimensional voxels unless they are extensions of $(d-1)$-dimensional voxels that are already known to contain some feasible region.

We will use the term "multi-resolution" search to mean evaluating voxels using minimization followed by gridsearch.

Results

To verify that our minimization method for evaluating voxels is a better way to evaluate voxels than other options (such as systematic sampling or random sampling), we constructed two simple test problems and searched them using treesearch with different methods of voxel evaluation. The first problem uses a set of distance constraints that are easily satisfied and we measure how many satisfying conformations are found by each method. The other problem uses an extensive set of distance constraints—only satisfied by a single narrow region of conformation space—and measures how extensive the search must be in order to locate that satisfying region.

For both problems, we started with a tetrapeptide of polyalanine folded so its ϕ and ψ angles were all $-57°$. (The WHATIF [72] software uses ϕ and ψ angles of $-57°$ as the canonical backbone for alpha-helical conformations, so we will refer to this peptide as helical despite its short length.)

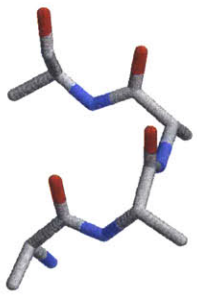

Figure 2-8: The structure of a helical tetrapeptide of polyalanine.

No hydrogens were modeled, and all the peptide bonds were rigidly fixed at 180 degrees, leaving two rotatable bonds per residue. Since the phi rotation of the first residue does not have enough heavy atoms at the N-terminus to define it uniquely, the molecule has only 7 degrees of freedom in total. The run time results throughout this thesis are rounded off from hundredths of seconds to whole seconds or to three significant digits, whichever precision is greater.

We took the set of all interatomic distances between 2.5 and 6.0 angstroms in the helical structure as the available pool of interatomic distances to constrain. Given the 20 atoms in the tetrapeptide, there are 124 pairs in this set. For the first problem, we used 10 percent of

the available distances (selected at random) and constrained those pairs to be within plus or minus 0.5 Å of the values found in the helical structure. For the second, "tight" problem, we used all of the available distances and constrained them to within plus or minus 0.1 Å of the helical values.

The first problem (with many possible solutions) requires solutions at 120° resolution and uses van der Waals radii set to 90% of the half sigma values[4] from the CNS library `parallhdg.pro` [8, 19]. For the first problem, the constraints (distance constraints and hard-sphere van der Waals constraints) are considered to be satisfied if the sum of the squared constraint violations is ≤ 0.005 Å^2. The tighter problem uses 40° resolution, 85% of half sigma and a threshold of ≤ 0.0005 Å^2 for the sum of the squared violations of the constraints.

For all runs, the voxels are defined by treesearch according to resolution and without regard to how the voxels will be evaluated. Since treesearch would define 1-dimensional and 2-dimensional voxels and evaluate them before defining higher-dimensional voxels, an unsatisfactory region of space would always have the opportunity to be excluded at low dimension ("pruned") before being searched at higher resolution.

VOXELIZED TREESEARCH was applied to both problems, with various voxel-evaluation methods as subroutines to decide whether a voxel contained any satisfying conformations. (See Figures 2-9 and 2-11.)

- **Random sampling.** We tried random sampling for both problems. Random sampling repeatedly selects conformations (uniformly at random) from inside the voxel, with the maximum number of samples determined by some cap. For the first problem we tried not only absolute caps but also caps that depend exponentially on the number of dimensions of the voxel (that is, the cap is proportional to the volume being searched).

- **Systematic sampling.** We tried some form of systematic sampling for each problem. For the first problem, we tried multi-resolution search with one pass of minimization followed by evaluation of regularly spaced points (gridsearch) in the voxel. The tightly constrained problem turned out to be easy enough that a single run of minimization

[4] Sigma is the distance at which two atoms of the same type have zero van der Waals interaction energy. Any closer and they would start to repel each other.

Method for Evaluating Voxels	Run Time (seconds)	Conformations Found
Evaluate 1 random point per voxel	0.09	1
Evaluate 10 random points per voxel	0.16	1
Evaluate 100 random points per voxel	1.61	5
Evaluate 1,000 random points per voxel	22.6	13
Evaluate 10,000 random points per voxel	265	36
Evaluate 100,000 random points per voxel	2,761	81
Evaluate 1,000,000 random points per voxel	27,536	114
Evaluate 10,000,000 random points per voxel	268,199	128
Minimize once, then gridsearch with 60° resolution	3.78	103
Minimize once, then gridsearch with 51.4° resolution	29.0	119
Minimize once, then gridsearch with 45° resolution	28.3	108
Minimize once, then gridsearch with 40° resolution	26.4	109
Minimize once, then gridsearch with 36° resolution	193	127
Minimize once, then gridsearch with 32.7° resolution	189	119
Minimize once, then gridsearch with 30° resolution	183	114
Minimize once, then gridsearch with 27.7° resolution	900	134
Minimize once, then gridsearch with 25.7° resolution	917	132
Minimize once, then gridsearch with 24° resolution	827	107
Minimize once, then gridsearch with 20° resolution	3184	131
Minimize once, then gridsearch with 17.1° resolution	9518	137
Minimize once, then gridsearch with 12° resolution	114,441	146
Minimize once, then gridsearch with 10° resolution	400,202	147
Minimize 1 time per voxel. (up to 5 steps)	0.64	49
Minimize 1 time per voxel. (up to 10 steps)	1.11	85
Minimize 1 time per voxel. (up to 20 steps)	1.75	101
Minimize 1 time per voxel. (up to 50 steps)	2.13	103
Minimize 2 times per voxel. (up to 50 steps)	3.92	134
Minimize 5 times per voxel. (up to 50 steps)	8.55	157
Minimize 10 times per voxel.	15.6	162
Minimize 100 times per voxel.	139	167
Minimize 1,000 times per voxel.	1,354	171
Minimize 10,000 times per voxel.	13,382	171
Minimize 100,000 times per voxel.	133,556	171

Figure 2-9: Treesearch on a tetrapeptide of polyalanine, with loose constraints, and a variety of methods for evaluating whether there are any satisfying conformations within each voxel. Because the search of a voxel exits at the first success, it is possible for a random search with a higher bound on the number of steps (such as 11^{dim}) to finish sooner than a search with a lower bound.

could find the answer, so for the tight problem we tried gridsearch within each voxel but without any minimization.

- **Minimization.** Finally, minimization was used in both problems, with a variety of caps on the number of steps and passes.

For voxel-evaluation methods that sample many points (whether systematically or at random) the sampling is halted when the first satisfying conformation in the voxel is found. However, for those voxel-evaluation methods to conclude that no satisfying conformations exist, they must have tried and rejected all the candidate points for that voxel.

In the first problem, where many voxels contain satisfying conformations, minimization dramatically outperformed random sampling in the number of conformations it finds for comparable amounts of time (see Figure 2-10). In contrast to the smoothly improving

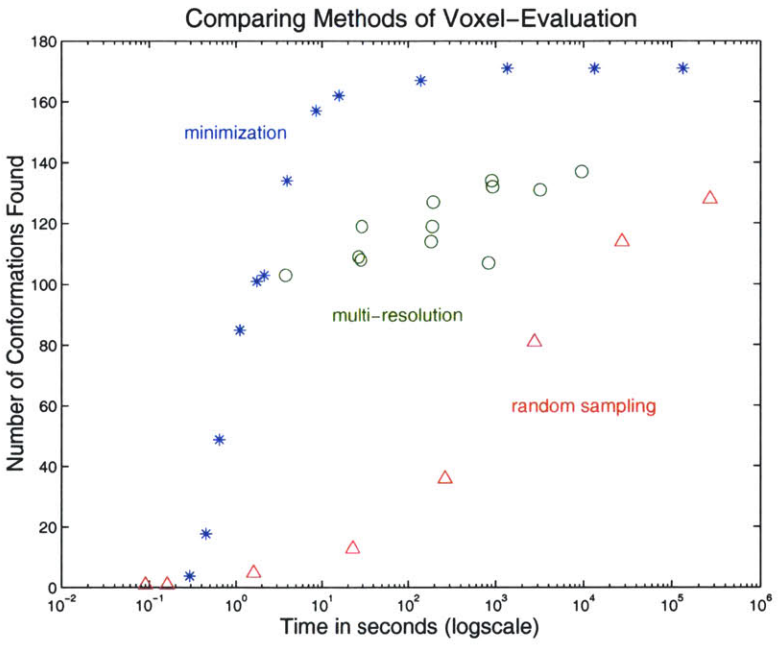

Figure 2-10: A plot of the data from Figure 2-9 with stars for runs using minimization, circles for runs using the multi-resolution method, and triangles for runs using random sampling.

completeness with time for the minimization and random sampling methods, the multi-resolution method performed erratically. We hypothesize the performance plot is dispersed between the minimization curve and the random sampling curve because multi-resolution search is a hybrid method, using aspects of the minimization search and aspects of the sam-

pling search; however, further investigation of this phenomenon remains for future research.

In the tighter problem, where only a small region satisfies all the constraints, minimization found the one satisfying voxel easily, whereas the other methods had to sample so many points that they consumed several orders of magnitude more time than minimization. (All

Method for Evaluating Voxels	Run Time (seconds)	Conformations Found
Evaluate 1 million random points per voxel	FAILED	0
Evaluate 2 million random points per voxel	6,186	1
Evaluate 3 million random points per voxel	FAILED	0
Evaluate 4 million random points per voxel	11,962	1
Evaluate 9^{dim} random points per voxel	FAILED	0
Evaluate 10^{dim} random points per voxel	17,894	1
Evaluate 11^{dim} random points per voxel	16,636	1
Evaluate at gridpoints with 4.0° spacing	FAILED	0
Evaluate at gridpoints with 3.8° spacing	17,341	1
Evaluate at gridpoints with 3.6° spacing	31,470	1
Minimize once per voxel. (up to 5 steps)	FAILED	0
Minimize once per voxel. (up to 10 steps)	0.18	1
Minimize once per voxel. (up to 50 steps)	0.19	1
Minimize twice per voxel. (up to 50 steps)	0.29	1

Figure 2-11: Treesearch on a tetrapeptide of polyalanine, with tight constraints, and a variety of methods for evaluating whether there are any satisfying conformations within each voxel. The times for failed runs are not shown because failures can occur at any time during the search, sometimes quite early.

simulations reported in this thesis were performed on a Dell Precision 420 with 933-MHz speed processors, with a surplus of RAM memory, unless otherwise specified.)

Sampling without minimization (at random or at gridpoints) performed much worse than minimization, both when measuring the amount of repetition required to find a narrowly-defined region, and when measuring the completeness of finding a broad region. The problems we constructed here do not by any means represent the breadth of applications where conformational search might be used. However, these two cases were chosen for simplicity of design and evaluation and were not optimized to sabotage any particular method of voxel-evaluation. Multiple orders of magnitude in increased performance for all the cases we studied (four orders of magnitude in Figure 2-11 and later five orders of magnitude in

Figure 2-15) were sufficiently persuasive to us that we chose to perform minimization with voxels for all future conformational searches.

2.3 Divide-and-Conquer

Divide-and-conquer partitions a large problem into pieces, solves the pieces, and assembles the solutions of the pieces into a solution for the large problem. Divide-and-conquer is typically used recursively, with large pieces divided into medium pieces, which are divided into small pieces, etc. After explaining how to apply divide-and-conquer to the conformational search of macromolecules, we explain and demonstrate the improved efficiency of conformational searches that use divide-and-conquer.

Divide-and-conquer is powerful because it allows violating regions of conformation space to be removed from further consideration based on evaluating low-dimensional pieces of the overall problem. Just as treesearch is an improvement over gridsearch because it evaluates "prefix" fragments of the molecule instead of only evaluating conformations of the complete molecule, divide-and-conquer is an improvement over treesearch because (A) it evaluates individually every piece it adds, *before* adding it onto an existing conformation, (B) once a subproblem is solved, the answer is saved for use in future contexts rather than reinvented, and (C) it can define its subproblems so that their average size (voxel dimensionality) is smaller than for treesearch.

Previous Work

The divide-and-conquer idea has appeared in many forms throughout the history of conformational search. Divide-and-conquer has been applied to docking searches [77] as well. In the mid-1980's, Scheraga and colleagues developed a "buildup" procedure [57, 71, 22] which has been successfully applied to a variety of applications [45]. Their buildup procedure enumerates the low-energy structures of single residues, combines them and minimizes to create low-energy dipeptides, combines overlapping dipeptides and minimizes to create low-energy tripeptides, and so on [45]. This method does not address quite the same problem as ours because at each level it retains only a limited number of conformations with the best energies, not all the conformations with acceptable energies. Other authors [11, 32] have used "buildup" procedures very similar to that of Scheraga *et al.*

Gippert *et al.* [24, 23] published a set of programs for systematic search including a divide-and-conquer module for use on top of an *ab initio* search module. The *ab initio* method, DTAGS or "Distributed Torsion Angle Grid Search," is a cleverly optimized variant of gridsearch that enumerates feasible conformations for a small molecule or fragment, provided that all the rotatable bonds are in a linear chain. Although DTAGS itself cannot search branched molecules such as amino acids, this limitation is overcome by using DTAGS on each branch and then reattaching the pieces. The reattachment is done pairwise according to a tree and each "combine operation" (SPACEJOIN) pairs every possible conformation for one fragment with every possible conformation for the other fragment. This is the same idea we use, although with several differences in implementation. NEWMOL then converts then evaluates whether the combined fragments satisfy the constraints. Gippert *et al.* articulated the problem of choosing a good "binary tree" for divide-and-conquer, which is a topic we address at length in Chapter 4.

Countless other methods build databases of protein structure fragments and then build models for new proteins by piecing together compatible fragments from their databases [35, 33]. These "model building" [43] methods are a form of "divide-and-conquer" with only one level of division and reconnection.

Active Constraints

Whether evaluating a subproblem will remove conformations from further consideration depends on what constraints apply to that subproblem. A constraint is not *active* until all of its variables are instantiated; that is, until angle values are assigned to all the torsions necessary to define coordinates for the atoms being constrained. Henceforth, *the constraints on a subchain* will denote the set of constraints that are active when the torsions in only that subchain are instantiated, and a *satisfying* conformation [for a subchain] will be a conformation that satisfies all active constraints [on that subchain].

For $(x \leq y)$, let x–y denote the subchain extending from residue x to residue y, inclusive of both. An interatomic distance constraint between two different atoms of residue 6 is active on subchain 6–6 or on subchain 1–99, but not on subchain 3–5 because subchain 3–5 does not provide torsion angles (which uniquely determine atomic coordinates) for the bonds of residue 6.

This seems trivial until you consider that an interatomic distance constraint requiring

some atom in residue 1 to be within a few angstroms of some atom in residue 30 is active on subchain 1–30 but not on 1–29. A conformation of subchain 1–29 that places residue 1 dozens of angstroms away from residue 29 might satisfy all the active constraints, even though there is certainly no possible way to extend or build onto such a conformation so that the 30th residue would satisfy active constraints on the 1–30 subchain.

In cases as clear as this, the triangle inequality can use a tight constraint on the atoms in residues 1 and 30 in order to infer looser constraints on nearby atoms, such as between residues 1 and 29.

$$AB + BC \geq AC$$

$$\text{dist}(1,30)+\text{dist}(30,29) \geq \text{dist}(1,29)$$

$$[\text{dist}(1,30) \leq 3\text{Å}] \text{ AND } [\text{dist}(29,30) \leq X] \Rightarrow [\text{dist}(1,29) \leq X + 3\text{Å}]$$

Our algorithm provides a "triangle smoothing" option that automatically applies the triangle inequality to infer all possible additional constraints. Some other algorithms also apply the tetrangle inequality [18]. Constraints created in this way can be considered "redundant" or "trivial" because they include no novel information, even though they attach existing information to new parts of the molecule.

The looser, inferred constraint would be active on subchain 1–29 and would prohibit the endpoints of subchain 1–29 from being dozens of angstroms apart. However, there would still be some conformations of 1–29 that satisfy the inferred constraint but that cannot be extended in any way that would satisfy the constraint on 1–30.

The Combine Operation

The combine operation uses the satisfying conformations for two pieces of a chain to define candidate conformations for the whole chain. One can imagine how repeated use of combine operations could be used in a divide-and-conquer strategy to build up the solution to the whole problem based on solutions to smaller problems.

More formally, let S_1 be the set of satisfying conformations for subchain x–y and let S_2 be the set of satisfying voxels for subchain $(y+1)$–z, where $z \geq (y+1)$. Note that since a satisfying voxel is represented by a satisfying conformation from within that voxel, we may speak of satisfying voxels interchangeably with satisfying conformations.

Any conformation of x–z can be divided into left and right pieces corresponding to

residues x–y and residues $(y+1)$–z. If a conformation of x–z satisfies all the constraints active on x–z, then its left half must also satisfy all the constraints on x–y and its right half must satisfy all the constraints on $(y+1)$–z. However, not all conformations that satisfy the constraints on the pieces also satisfy the constraints on the whole.

Therefore the set of all conformations of x–z derived by combining any satisfying conformation for x–y with any satisfying conformation for $(y+1)$–z is a superset of the satisfying conformation for x–z. Call it the set of candidates. In order to determine which candidate conformations satisfy the constraints on x–z, each must be evaluated individually. However, the effort of checking each member of the candidate set is generally far less than if the constraints on x–y and y–z had not already been applied as filters.

Each candidate conformation corresponds to a unique voxel, and verifying whether each one satisfies the constraints is the familiar problem of determining whether a voxel contains any satisfying conformations. We continue to use minimization to evaluate all voxels. Note, however, that the volume of a voxel increases with dimension and the problem of evaluating it (determining whether it contains any satisfying region of space) becomes significantly more difficult for higher-dimensional voxels.

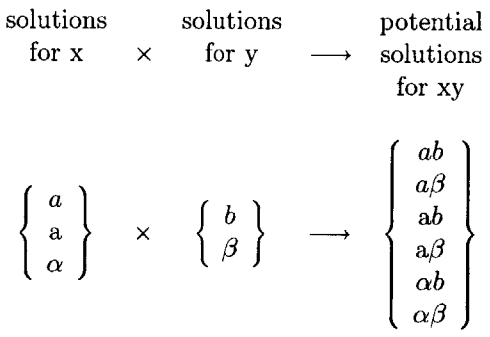

Figure 2-12: Combining 1D alternatives into all possible 2D combinations.

Defining the Combined Conformation

In our implementation, we always divide a molecule chain at a boundary between residues, and we define the resulting pieces such that both pieces include the inter-residue bond and the atoms defining that bond. Hence, when we combine satisfying conformations for left and right subchains to create conformations for a larger subchain, each bond from the larger subchain is first marked according to whether it appears in only the left subchain, in only

the right subchain, or in both. Then as the combine operation proceeds, going through each possible conformation for the left subchain and each possible conformation for the right, the combine routine creates a conformation for the large subchain one bond at a time: For a bond that appears in only one subchain, its angle (range of angles) can be taken directly from the angles of that child subchain. For a bond that appears in both subchains, both subchains must have compatible angles (the same range of angles), or else that pair of conformations cannot be combined and is rejected. Note that this procedure defines a particular conformation (a particular assignment of angles for each bond) as well as a voxel (a range of angles for each bond). When evaluating whether a combined voxel contains any satisfying conformations, we will have the option of using this combined conformation as a starting point for minimization, in addition to just the midpoint of the voxel.

In proteins, amino acid residues are connected by peptide bonds. Since peptide bonds are often fixed in a *trans* planar orientation (or restricted to a narrow range around a planar orientation), the task of reconnecting subchains at their inter-residue bonds is especially easy. An interesting direction for future work would be to permit our combine operation to operate on fragments with larger regions of overlap. Although modifications to the actual combine operation would be trivial, they would have profound implications for the issues discussed in chapter 4.

Merge-Trees

The repeated division of large problems into smaller problems can be depicted using a divide-and-conquer tree, which we will call a *merge-tree* for short. For the application to conformational search, each node of the tree corresponds to a subchain and searching that subchain is a subproblem of the overall search algorithm. The leaf nodes correspond to subchains that are small enough to search easily *ab initio*. (We search leaf subchains using voxelized treesearch with minimization.) Each internal node corresponds to a larger subchain such that the problem of searching that subchain can always be solved by using the "combine" operation on its left and right child subchains. The root of the merge-tree is always the whole molecule.

For treesearch trees, each level corresponds to one torsion, the branches at a level are the possible angle or angles ranges for a torsion, and a path from root to leaf defines a single conformation. For merge-trees (divide-and-conquer trees), each node is a subproblem, the

leaf nodes are the easiest subproblems, the root node is the whole search problem, and the connectivity of the tree defines a strategy for how to use the solutions to the leaf subproblems for solving the whole problem. Search trees are explored starting at the root progressing down but merge-trees are traversed starting at the leaves.

Assume that the molecule to be searched is a chain of residues. We often talk about amino acid residues in a protein but the whole molecule could also be a single-stranded nucleic acid chain. Multiple strands, docking, and quaternary structures are not currently possible. (Although the algorithmic changes necessary to implement the six degrees of freedom necessary for docking or for searching quaternary structure might be relatively small, the resulting complexity of the search problem is well beyond the scope of this work.)

Let us define a *legal* tree to be a binary tree such that (1) the leaves of the tree have a one-to-one correspondence with the residues of the molecule, (2) the left-to-right order of the leaves in the tree is the same as the order of the residues in the molecule, and (3) every internal node has two children. We now show that every legal tree is a merge-tree specifying a possible strategy for divide-and-conquer.

Define the label at each internal node as the list of residues corresponding to the leaves in its subtree. Because leaves are sequential from left to right in the tree, the residues for any internal node will always form a single contiguous subchain of the whole molecule. The root will be labelled with the subchain corresponding to the whole molecule. Because every internal node, I, has left and right children, the subchains (possibly consisting of just single residues) at the left and right child nodes correspond to left and right pieces of I's subchain, and I's subproblem can be solved using the combine operation on the solutions of its left and right child subproblem. Therefore, every legal merge-tree corresponds to a divide-and-conquer strategy for searching the whole molecule.

We assume that all merge-trees use single residues for leaves because single residues are already small enough to search quickly, and there is little potential benefit from dividing into smaller fragments than residues. Larger units, such as dipeptides, could trivially be substituted for single residues in the algorithm and throughout this discussion. In any case, the leaves of merge-trees would be explored using a basic method, such as TREESEARCH, no matter how the rest of the merge-tree is constructed.

Divide-and-conquer as Merge-Tree Traversal

We can now describe the whole divide-and-conquer algorithm. Create a default merge-tree according to the number of residues in the molecule and traverse the tree from bottom to top (and without loss of generality, from left to right) solving the subproblem at each node. Each subproblem involves enumerating the satisfying conformations (voxels) for the subchain at that node of the tree. For a leaf node, use treesearch (with voxels and with minimization to evaluate the voxels). For an internal node, perform the combine operation on the satisfying conformations (voxels) from the left and right child subchains.

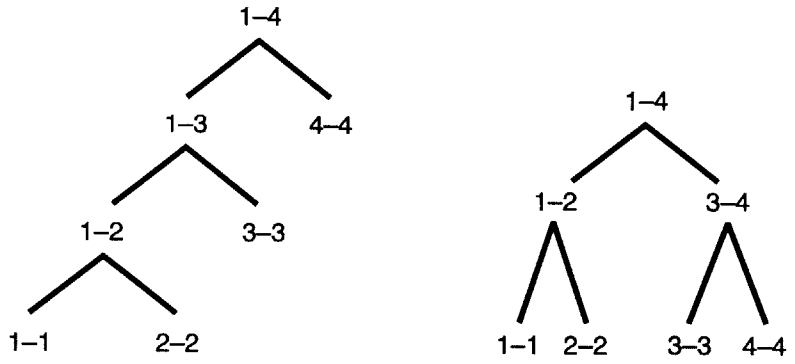

Figure 2-13: A linear or maximally-unbalanced tree on the left and a fully-balanced tree on the right.

There may be many possible merge-trees for each molecule, but in order for the average subchain size in the tree to be small, the divide-and-conquer tree should be as balanced as possible. If a whole molecule has N residues, then treesearch or divide-and-conquer according to a "linear" tree (such as in Figure 2-13) must evaluate one subproblem of each size between 1 and $N-1$ before evaluating the whole molecule. Divide-and-conquer with a fully balanced tree will evaluate many subproblems of size 1, 2 and 4, a few subproblems of size $N/4$ or $N/2$, and never any subproblems larger than $N/2$ (other than the final evaluation of the whole molecule).

If N, the number of residues in the molecule, is a power of two, then one can simply use the canonical, "full" or perfectly-balanced binary tree with $\log_2(N)+1$ levels. If N is not a a power of two, there are many possible ways to create a reasonably-balanced tree with at most $\lceil \log_2(N) \rceil + 1$ levels (where $\lceil \ldots \rceil$ denotes the operation of rounding up to the nearest integer.) For simplicity, our *default tree* will be created as follows: starting with

a full tree with $2^{\lceil \log_2(N) \rceil}$ leaves, delete the $2^{\lceil \log_2(N) \rceil} - N$ rightmost leaves and delete any internal nodes without both (left and right) children remaining. For example, if $N = 9$, build a full tree with 16 leaves, delete the 7 rightmost leaves, remove unnecessary internal nodes, and the result is the same tree as in Figure 2-18.

Distinctions between Treesearch and Divide-and-Conquer

In some sense, treesearch is a special case of divide-and-conquer where the division into pieces forms a "linear" tree instead of a balanced binary tree.

However, there is a slight difference between divide-and-conquer with a "linear" tree of subproblems and treesearch. Consider a molecule with two rotatable bonds, one bond in each of two residues. Treesearch would search the first bond, subchain 1-1, then extend the satisfying conformations of 1-1 with all possibilities for bond 2. Divide-and-conquer would also start by searching subchain 1-1, but its next step would be to search subchain 2-2, and finally it would extend the satisfying conformations of 1-1 with only the possibilities for bond 2 that have already been found to satisfy the constraints on 2-2. Figure 2-14 diagrams which subchains are defined and searched by TREESEARCH and by divide-and-conquer using either of the merge-trees in Figure 2-13.

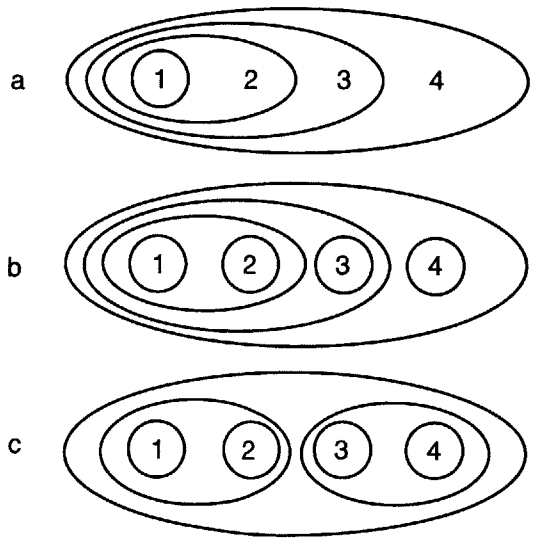

Figure 2-14: Each circle (or oval) represents a search subproblem. Treesearch would search a four-residue molecule as depicted in (a). (b) depicts divide-and-conquer with a "linear" merge-tree and (c) depicts divide-and-conquer with a balanced merge-tree.

Results

We searched three peptides with and without use of divide-and-conquer. In all cases, divide-and-conquer was far more efficient than treesearch alone.

The first case uses the same polyalanine tetrapeptide from the previous section. All atom pairs with distances between 2.5 and 6.0 Å in the helical structure were constrained to have distances within 0.1 Å of the distances found in the helical conformation. Peptide bonds were rigidly fixed at 180°. Van der Waals radii were set to 85% of half sigma, the

Search Method	Minimization to evaluate voxels?	Run Time (seconds)
Treesearch	no	31,470.550
Divide-and-conquer	no	3,009
Treesearch	yes	0.29
Divide-and-conquer	yes	0.13

Figure 2-15: Results of performing treesearch with and without divide-and-conquer, on a tetrapeptide of polyalanine, with and without minimization to evaluate each voxel. Tight constraints were used, as in Figure 2-11.

resolution was 40°, and the constraints were considered to be satisfied when the summed squared violation was ≤ 0.0005 Å^2. Run times for systematic search with and without divide-and-conquer are shown in Figure 2-15. The top two lines use gridsearch with 3.6° spacing to evaluate voxels instead of using minimization and are included only to provide support for our decision to use minimization for all voxel-evaluations (instead of just using it for evaluating treesearch voxels, which is the original context where we evaluated it). The bottom two lines show that searching with divide-and-conquer takes less than half as long as with treesearch alone. However, both algorithms run so quickly that drawing general conclusions would be premature.

The second case also uses a helix of polyalanine, but with 16 residues instead of 4. The dihedral angles of the peptide bonds are permitted to vary between 175° and 185°. Including these omega angles, there are 47 degrees of freedom in the molecule. Again, the resolution of the search is 40°, the summed squared violations are restricted to ≤ 0.0005 Å^2, and the van der Waals radii are set to 85% of half sigma. All pairs of atoms with distances between 2.5 and 6.0 Å in the helical structure were constrained to have distances within only 0.05 Å

of the distances found in the canonical helix conformation. These constraints again restrict the satisfying region of conformation space very tightly, although with the larger molecule and the added flexibility in the peptide bonds, two voxels are feasible instead of just one. (One voxel contains the canonical helix that generated the constraints and the second is the same except for a deviation in the final angle of the final residue, offset slightly by tilts of several peptide torsions.)

Search Method	Minimization to evaluate voxels?	Run Time (seconds)	Number of Minimizations
Treesearch	yes	143	660
Divide-and-conquer	yes	1.92	813

Figure 2-16: Results of performing search on a 16-residue helix of polyalanine, with and without divide-and-conquer.

Figure 2-16 shows the run time as well as the number of minimization subroutine calls for searching the 16-residue polyalanine with and without divide-and-conquer. In both cases, voxels were evaluated using up to two passes of minimization, each with up to 50 steps. Although divide-and-conquer performs more minimization calls than treesearch, divide-and-conquer is calling them with smaller subchains on average. Despite performing more minimizations, divide-and-conquer runs two orders of magnitude faster than TREESEARCH.

The third case is the 9-residue *Strep*-tag peptide from the peptide-streptavidin complex [64], entry 1RST in the protein data bank [2]. The sequence of the peptide is Ala-Trp-Arg-His-Pro-Gln-Phe-Gly-Gly. Figure 2-17 shows the atomic structure from X-ray crystallography and Figure 2-18 shows the default merge-tree.

The 1RST peptide has several long sidechains that contribute considerably to the conformational search problem. Of the 40 rotatable torsions, 32 rotate freely (as opposed to peptide bonds), which is the same number of freely-rotating torsions as in the 16-residue polyalanine. It addition, 1RST has some explicit hydrogen atoms.

We used the bond lengths and bond angles from the 1RST crystal structure, not idealized values. Because the 1RST crystal structure is highly refined for its particular conformation, using the same search parameters as for helical polyalanine would cause unavoidable van der Waals clashes. Instead of reducing the van der Waals radii or increasing the allowed threshold for the summed squared violations, we excluded van der Waals interactions between

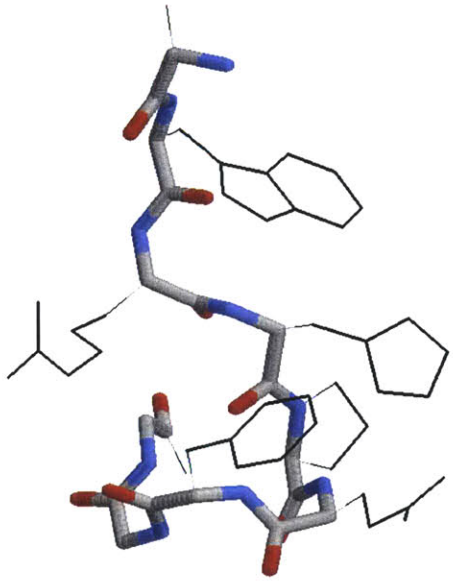

Figure 2-17: Atomic structure of the *Strep*-tag, taken from its crystal structure in complex with streptavidin (1RST in the Protein Data Bank). The backbone bonds are shown as colored cylinders and the sidechains as black lines. Residue 1 is at the top.

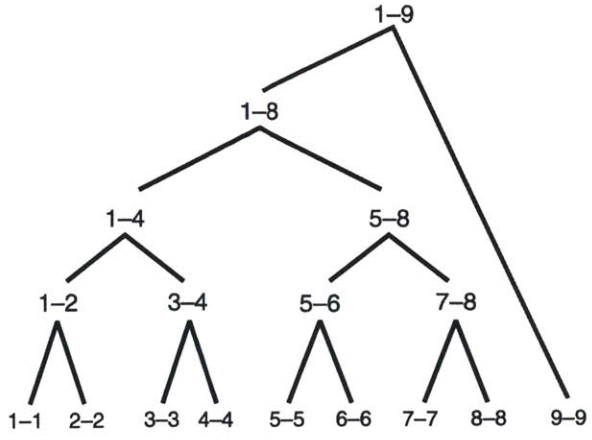

Figure 2-18: The default merge tree for 1RST.

atoms in the 1–4 positions of bonds.

The dihedral angles of the peptide bonds were permitted to vary between 175°–185°, the van der Waals radii were set to 85% of half sigma, and the summed squared violations were restricted to ≤ 0.0005 Å^2. Voxels were evaluated using up to two passes of minimization with up to 50 steps per pass. Any pair of atoms separated by a distance between 2.5 and 6.0 Å in the crystal structure was constrained to keep its distance within 0.05 Å of the measurement from the crystal structure. Because the sidechains were so flexible, even in the presence of these constraints, we chose to search them at lower resolution than the backbone angles. The resolution of the search was 40° for backbone angles and 120° for sidechains.

Search Method	Runtime (seconds)	Conformations Found	Number of Minimizations
Treesearch	13,661	648	54,848
Divide-and-conquer	2,046	657	6,712

Figure 2-19: Results of performing search on the 9-residue 1RST peptide, with and without divide-and-conquer. Figure 2-20 shows the set of structures resulting from the divide-and-conquer search. In all tests, the voxel corresponding to the crystal structure was among the answers found.

Figure 2-19 shows that the search of 1RST with divide-and-conquer was faster than with treesearch alone. In addition, treesearch performed more minimizations on 1RST than divide-and-conquer and found fewer satisfying structures. We hypothesize that these additional differences are in part because treesearch's examination of only the "prefix" fragments of the chain is ill-suited to this problem. Performing a divide-and-conquer search on 1RST with a "linear" merge-tree yielded 648 conformations after 14,353.41 seconds, thus confirming that the use of "prefix" fragments (whether in treesearch or in divide-and-conquer) is responsible for the poorer performance.

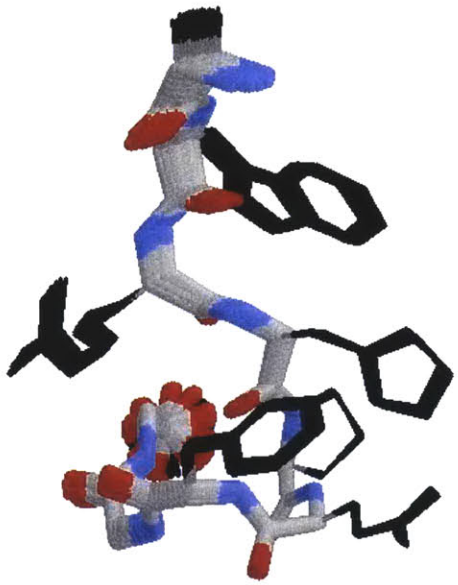

Figure 2-20: The ensemble of 657 structures found by the divide-and-conquer search of 1RST. The structures are superimposed along a central backbone bond, between the carbonyl carbon of residue 5 and the nitrogen of residue 6.

Chapter 3

The Structure of a Peptide by solid-state NMR

Article submitted to *The Proceedings of the National Academy of Sciences (USA).*

De Novo Determination of Peptide Structure with Solid-State MAS NMR Spectroscopy

Chad M. Rienstra,[1,2,5] Lisa Tucker-Kellogg,[3] Christopher P. Jaroniec,[1,2] Morten Hohwy,[2,6] Bernd Reif[1,2,7], Michael T. McMahon,[1,2] Bruce Tidor,[1,3,4,8] Tomás Lozano-Pérez,[3,8] and Robert G. Griffin[1,2,8]

[1] *Department of Chemistry*

[2] *Francis Bitter Magnet Laboratory*

[3] *Department of Electrical Engineering and Computer Science*
and
[4] *Division of Bioengineering and Environmental Health*
Massachusetts Institute of Technology,
Cambridge, MA 02139

[5] Present address: Department of Chemistry, Columbia University, New York, NY 10027.

[6] Present address: Laboratory for Physical Chemistry, ETH-Hönggerberg, CH-8093 Zürich, Switzerland.

[7] Present address: Laboratory for Organic Chemistry, TU-Munich, D-85647 Munich, Germany.

[8] Authors to whom correspondence should be addressed.

E-mail: rgg@mit.edu, tlp@mit.edu, and tidor@mit.edu

Abstract

The three-dimensional structure of the chemotactic peptide, N-formyl-L-Met-L-Leu-L-Phe-OH, was determined using solid-state nuclear magnetic resonance (SSNMR). The set of SSNMR data consisted of 16 ^{13}C-^{15}N distances and 18 torsion angle constraints (on 10 angles), recorded using uniformly ^{13}C,^{15}N- and ^{15}N-labeled samples. The peptide's structure was calculated via simulated annealing and a newly developed protocol that insures that all of conformational space, consistent with the structural constraints, is searched completely. The result is a high quality structure of a molecule that has thus far not been amenable to single crystal diffraction studies. The extensions of the SSNMR techniques and computational methods to larger systems appear promising.

Over the last two decades, multi-dimensional nuclear magnetic resonance (NMR) methods have been developed which permit determinations of globular protein structures in solution [1]. To date most structures addressed with these techniques involve proteins with molecular weights ≤20 kDa, but the continued development of new methodology shows promise for studies of larger systems [2-6]. Despite the success of these approaches, there remain fundamental limits on the size and physical state of molecules amenable to study with solution-state NMR. In contrast, high-resolution solid-state NMR (SSNMR) methods have no inherent molecular weight limit, and have for many years been employed to determine details of molecular structure for high molecular weight systems. For example, specific structural features of intact membrane proteins such as bacteriorhodopsin (effective molecular weight ~ 85 kDa) [7, 8] and large enzyme complexes such as 5-enolpyruvylshikimate-3-phosphate synthase (46 kDa) [9] and tryptophan synthase (143 kDa) [10], have been reported. SSNMR methods have also been used to examine surface-bound peptides [11] and to determine a low-resolution structure (1.9 Å backbone RMSD) of an insoluble peptide fragment from β-amyloid [12] under experimental conditions inaccessible to both solution-state NMR and crystallography.

To date, essentially all structural NMR studies of solid peptides and proteins have relied upon site-specific incorporation of a pair of spin-1/2 nuclei, such as ^{13}C–^{13}C [13] and ^{13}C-^{15}N [9]. This approach has been very successful and will likely continue to be important in experiments that address detailed mechanistic questions in large biomolecular systems. However, recent advances in solid-state NMR methodology, most notably the development of approaches to perform dipolar recoupling during magic angle

spinning (MAS) [14, 15], in principle permit multiple distance and torsion angle measurements on molecules that are uniformly ^{13}C and ^{15}N labeled [16-19]. The development of these approaches considerably simplifies preparation of samples for SSNMR experiments and concurrently opens the possibility of complete structural determinations with solid-state MAS NMR. In this paper we describe the realization of this goal with a complete structure determination of the chemotactic tripeptide, N-formyl-L-Met-L-Leu-L-Phe-OH (f-MLF-OH) [20]. The structure of the peptide is based on sets of NMR data that constrain 16 ^{13}C-^{15}N distances and 10 torsion angles derived from a series of MAS NMR experiments performed on uniformly ^{13}C,^{15}N- and ^{15}N-labeled samples. Finally, we discuss extensions of the solid-state MAS NMR techniques and computational methods employed here to larger systems.

Experimental Procedures

N-formyl-Met-Leu-Phe-OH samples were synthesized by standard solid-phase methods and HPLC purification (American Peptide Company, Sunnyvale, CA). One sample, synthesized from U-^{13}C,^{15}N-labeled amino acids (Cambridge Isotope Laboratories, Andover, MA), was employed for all resonance assignment experiments and the majority of the 3D torsion angle experiments (^{1}H-^{15}N-^{13}C-^{1}H, ^{1}H-^{13}C-^{13}C-^{1}H, ^{15}N-^{13}C-^{13}C-^{15}N). A second sample was prepared by dilution of the U-^{13}C,^{15}N-labeled f-MLF-OH peptide in natural abundance material in the ratio of 1:9, and employed for the frequency selective REDOR experiments. A third sample, synthesized from ^{15}N-labeled amino acids, was employed for the ^{1}H-^{15}N-^{15}N-^{1}H torsion angle experiments. In all cases, microcrystals of the f-MLF-OH peptides were grown by overnight evaporation from 2-

propanol, and ~15-20 mg of each polycrystalline material was packed into a 4-mm zirconia NMR rotor (Varian-Chemagnetics, Fort Collins, CO). Attempts to grow single crystals suitable for diffraction studies were not successful. The structure of the f-MLF methyl ester (f-MLF-OMe) and other analogs of f-MLF have been determined by diffraction methods [21], but the f-MLF-OH acid form has not, despite repeated efforts (E. Gavuzzo, private communication). Presumably the acid form does not form large single crystals due to small differences in crystal packing forces, relative to the methyl ester. We note that the previously published structure of f-MLF-OH includes a D-Phe residue [22] which is not present in the chemotactically active form. [20]

Magic-angle spinning (MAS) NMR experiments were performed on Cambridge Instruments spectrometers operating at 400 and 500 MHz (courtesy of Dr. D. J. Ruben), together with custom-designed 400 and 500 MHz multiple-resonance transmission line probes, or a Varian-Chemagnetics (Fort Collins, CO) 500 MHz triple resonance probe. All of the probes were equipped with 4 mm MAS spinner modules. The resonance assignment [23] and REDOR experiments [17] were performed at 500 MHz, as were most of the torsion angle experiments (^{1}H-^{15}N-^{13}C-^{1}H, ^{1}H-^{13}C-^{13}C-^{1}H, and ^{15}N-^{13}C-^{13}C-^{15}N), with the exception of ^{1}H-^{15}N-^{15}N-^{1}H (400 MHz) [24]. Typical rf field strengths were ~100-120 kHz on ^{1}H during recoupling periods, ~80 kHz TPPM during chemical shift evolution periods (using TPPM decoupling [25]), and ~60 kHz during cross polarization. Fields of ~50 kHz or less were employed on the ^{13}C and ^{15}N channels recoupling channels. Additional details regarding the pulse sequences may be found in prior publications. [17, 19, 23, 24]. Data acquisition periods are noted in the figure captions. Experiments were performed at room temperature. Tensor magnitudes along the entire

backbone (^{1}H-^{15}N and ^{1}H-$^{13}C^{\alpha}$ dipolar couplings and $^{13}C(O)$ chemical shift tensors) are consistent with rigid lattice values. Likewise the Leu side-chain is rigid, based on upon local dipolar field measurements. Signals from the Phe aromatic ring show evidence of two-site conformational exchange, and the observed Met side-chain dipolar couplings ($^{13}C^{\beta}$-$^{1}H^{\beta}$, $^{13}C^{\gamma}$-$^{1}H^{\gamma}$, $^{13}C^{\alpha}$-$^{13}C^{\beta}$) are ~25% less than the rigid lattice values, consistent with small librations of the side-chain.

Resonance Assignments

The initial step in a structural study by NMR involves the sequence-specific assignment of the chemical shifts. Several multi-dimensional chemical shift correlation methods for resolving and assigning peptide ^{13}C and ^{15}N resonances have been developed [23, 26, 27]. Experiments for ^{13}C-^{13}C assignments generally employ either a homonuclear zero-quantum recoupling sequence such as RFDR [28] or a double quantum sequence [29-31] such as SPC-5 [32]. Heteronuclear assignments are accomplished with frequency selective ^{15}N-^{13}C double cross-polarization methods [33, 34] refined with adiabatic passage techniques [35]. Slices from ^{13}C-^{13}C planes extracted from a 3D NCC experiment are shown in **Figure 1** and serve to illustrate this point [23]. The slices correspond to the three ^{15}N resonances, and the ^{13}C connected to each amide ^{15}N appear in the ^{13}C-^{13}C plane. Thus, the Leu ^{15}N slice (116.2 ppm) shows positive cross-peaks (blue) to the Met C' and Leu C^{α}. Since the ^{13}C-^{13}C correlations were established using double-quantum recoupling, the cross peaks to Leu C' and Met C^{α} are negative absorption (indicate by red cross peaks). 2D ^{13}C-^{13}C experiments and the 2D ^{13}C-^{13}C planes from ^{15}N-^{13}C-^{13}C experiments also permit the resolution and assignment of the side-chain resonances.

These methods have already been utilized at high magnetic fields (750-800 MHz ^{1}H frequency) in studies of larger proteins, yielding partial assignments in the Bovine Pancreatic Trypsin Inhibitor (BPTI) [36], LH2 light-harvesting membrane protein complex [37] and a complete *de novo* assignment of a 62-residue α-spectrin SH3 domain [38].

Torsion Angle Measurements

When spectral assignments are complete, multidimensional experiments can be used to obtain two types of structural constraints – torsion angles and internuclear distances. Measurements of backbone and side-chain torsion angles (ϕ, ψ and χ) provide constraints on the local structure and usually involve experiments that employ one or two chemical shift dimensions, and an additional dimension to record the evolution under the local dipolar interactions. Thus, the angular information is determined by the measurement of the relative orientation of two dipolar tensors. For example, we recently described a 3D experiment, for constraining the torsion angles ϕ_i, ψ_{i-1}, and χ^1_i via ^{1}H-^{15}N-^{13}C-^{1}H spectroscopy [19]. Similar 3D ^{15}N-$^{13}C^{\alpha}$-$^{13}C'$-^{15}N [39, 40] and ^{1}H-$^{13}C^{\alpha}$-$^{13}C'$-^{15}N experiments [41] can be used to constrain ψ_i, and ^{1}H-^{13}C-^{13}C-^{1}H experiments [42] constrain the side-chain χ_i angles. Finally, the projection angle $\theta_{i,i+1}$ measured in a $^{1}H_i$-$^{15}N_i$-$^{15}N_{i+1}$-$^{1}H_{i+1}$ experiment [24] further constrains ϕ_i and ψ_i. We combined data from four 3D experiments in f-MLF-OH: ^{1}H-^{15}N-^{13}C-^{1}H, ^{1}H-^{13}C-^{13}C-^{1}H, ^{15}N-^{13}C-^{13}C-^{15}N, and $^{1}H_i$-$^{15}N_i$-$^{15}N_{i+1}$-$^{1}H_{i+1}$. Each of the torsion angle measurements is most precise when the correlated dipole tensors are approximately collinear. Therefore, the ^{15}N-^{13}C-^{13}C-^{15}N experiment is very precise for $140° < |\psi| < 180°$, and the ^{1}H-^{15}N-$^{13}C^{\alpha}$-^{1}H experiment for -

$150° < \phi < -90°$. The dephasing of the Met C'-C^{α} double quantum coherence under C'-N and C^{α}-N dipolar couplings during the NCCN experiment in f-MLF-OH is illustrated in **Figure 2**. For this particular measurement, the best-fit simulation gives the torsion angle $|\psi_{Met}|=157°$. The precision ($\pm 1\ \sigma$) of this experiment is $\pm 1°$, due to the high signal-to-noise ratio of the NMR data (>1,000:1 in first data point of the dephasing trajectory), and the fact that this result falls within the most sensitive angular region of the experiment. For other torsion angle constraints (**Table 1**), the precision ranges from $\pm 2°$ to $\pm 18°$, and in all cases at least two (and sometimes four or six) solutions are consistent with the experimental data, due to mirror-plane degeneracies. Determining multiple NMR constraints on each torsion angle removes many of the degeneracies. For this reason we have combined the results from multiple 3D torsion angle experiments to provide a total of 18 constraints on 10 torsion angles in f-MLF-OH. *All* torsion angle solutions consistent with the NMR data (based upon ~10,000 iterations of Monte Carlo simulations) are allowed for purposes of searching the conformational space (see below).

Internuclear Distance Measurements

Long-range internuclear distances (3-6 Å) provide highly useful complementary constraints for the peptide structure. Determination of these distances is particularly important since small errors in the local torsion angle measurements can propagate over multiple bonds resulting in an increased uncertainty in the global fold of the peptide. Furthermore, distance measurements can provide constraints on the position of nuclei, which are inaccessible to dihedral angle measurements (e.g., Met C^{ε}). Individual ^{13}C-^{13}C and ^{13}C-^{15}N distances can be measured in site-specifically labeled samples using

techniques such as rotational resonance (R^2) [43] or its variants (e.g., R^2 tickling[44]) and REDOR [45], respectively. Recently techniques have been developed for accurate measurements of multiple ^{13}C–^{13}C [16] and ^{13}C-^{15}N [17] distances in uniformly ^{13}C,^{15}N labeled molecules. For ^{13}C-^{15}N dipolar interactions, selective recoupling is possible by combining the REDOR technique [45] with selective Gaussian inversion pulses [17]. Using this approach we have measured a total of 16 ^{13}C-^{15}N distances in f-MLF-OH (14 distances >3 Å), which are assembled in **Table 2**. Several representative distance measurements are illustrated in **Figure 3** and clearly demonstrate the strong dependence of the decay of ^{13}C magnetization on the dipolar coupling to the selected ^{15}N. The 3.12 Å Met(C^{β})–Leu(N) distance further constrains ψ_{Met}. The Met(C^{β})–Phe(N) and Leu(C^{δ})–Leu(N) distances depend on multiple torsion angles and are important in determining the shape of the turn in the f-MLF-OH backbone, and the conformation of the Leu side-chain, respectively. With 95% statistical confidence, 15 of the 16 measured distances have precision of ±0.3 Å or better.

Computational Procedures

Since this is the initial determination of a molecular structure utilizing MAS dipolar recoupling techniques, it required us to develop new approaches to calculating molecular structures from the collection of experimental distance and torsion angle constraints. Accordingly, we have explored two approaches to this problem, both of which are described below. The first is based on simulated annealing and incorporates molecular potentials configured to permit transitions among the multiple conformations consistent with the structural constraints. Thus, during the annealing protocol the

structures are biased towards the closest minima in the experimental RMSD plots at each time step. This prerequisite requires that the force constants for these potentials be sufficiently low to allow transitions among these minima. In a second approach we have addressed a problem that is often unnoticed in NMR structure calculations that sample the conformational space stochastically. In particular, these approaches do not necessarily guarantee that all regions of conformational space are examined and therefore they may lead to structures where the uncertainty in the final ensemble is anomalously low. Here we address this issue by dividing the search space into discrete non-overlapping volumes and assign each volume as allowed or disallowed, based on whether or not it contains viable structures. Ordinarily a search through such a space would be intractable for all but the smallest molecular systems. To circumvent this problem, we developed a divide-and-conquer strategy that allows us to eliminate voxels that contain conformations that violate the structural constraints. The approach condenses the search space sufficiently so that significantly larger problems may be computationally tractable with this procedure.

Simulated Annealing

An ensemble of 24 f-MLF-OH structures was calculated using the SSNMR constraints collected above incorporated into the simulated annealing protocol of Nilges et al. [46] and the program CNS [47]. The program was modified to accommodate the structural constraints generated by the SSNMR distance and torsion angle experiments described above. The internuclear distances were incorporated using a standard distance potential and the 95% confidence limits. For the experimental torsion angle data sets, the simulations based on each experimental measurement were pooled together according to

the constrained angles and the resulting joint probability distributions enclosing the 75% confidence limits were used. For example the $^{15}N^{13}C^{13}C^{15}N$, $^{1}H^{15}N^{13}C_{\alpha}{}^{1}H$, $^{1}H^{15}N_{i+1}{}^{13}C_{\alpha}{}^{1}H$ and $^{1}H^{15}N^{15}N^{1}H$ experiments were joined to define a constraint on the Met ϕ and ψ torsion angles. Harmonic square wells, with the square wells enclosing the 75% confidence limits, were incorporated directly into the source code for each pair of angles. For many of these constraints, there are several distinct minima. In these cases, the potential is written to permit switching so that during the simulated annealing the structures are biased towards the closest minima at each time step. Because of this requirement, the force constants for these potentials had to be sufficiently low to allow transitions between minima.

The results from the simulated annealing calculation are summarized and compared in Tables 2 with the experimental distance constraints from the FS-REDOR experiments. In Table 3 we summarize and compare the torsion angles from the known X-ray crystal structure of f-MLF-OMe with the torsion angles calculated with CNS. Note that in both cases there is excellent agreement between the calculations and the experimental data.

Full Structure Search

In addition we developed a new computational procedure to analyze the distance and torsion angle constraints. The simulating annealing procedure used above typically samples the space of allowed conformations stochastically, a procedure that does not assure that all regions of conformational space are sampled and thus may underestimate the uncertainty in the final structural ensemble. An alternative that overcomes this difficulty is to subdivide the search space into discrete voxels (small non-overlapping

volumes that together entirely fill the conformational space) and to assign each voxel as allowed or disallowed, based on whether or not it contains structures that satisfy the constraints. If each voxel were searched explicitly, the search space would be intractable for all but the smallest problems. We have adopted divide-and-conquer strategies to allow relatively large regions of the search space to be eliminated if they contain a substructure that violates the constraints. Such approaches can effectively prune the search tree to make even large problems computationally tractable.

The search space was constructed from 16 torsion angles, 10 of which were constrained directly by the SSNMR data. Three additional angles were peptide bonds and constrained to be within 5° of planar (either *cis* or *trans*), and the remaining three angles had no direct torsion constraints (ψ_{Phe}, χ^2_{Phe}, and χ^3_{Met}). Fixed bond lengths and angles were used to simplify the space, the values of which were determined in trial calculations involving energy minimization from an extended conformation with full freedom in the presence of intramolecular constraints alternating with the systematic search described below. Excluded volume (van der Waals) constraints were enforced using 90% of the radius values ($\sigma/2$) from the all-hydrogen protein parameters in CNS version 4.02 [47]. Divide-and-conquer was implemented through initial searches systematically performed for each residue independently at a voxel grid resolution of 5° for all but the free torsion angles, which were enumerated in 30° steps. In this initial search, each "residue" included one additional atom along the backbone chain from its neighbors to allow the join in the subsequent step. Each substructure voxel was searched by first checking the center of the voxel to determine whether this substructure satisfied the subset of constraints involving only the atoms in the substructure ("active

constraints"). If so, the substructure was retained. Otherwise, the substructure was minimized subject to the active constraints and the additional constraint that the substructure remains within the voxel boundary (using CFSQP, a constrained nonlinear programming method [48] and an objective function that included only NMR distance constraints and excluded volume). For the search of each voxel, up to three minimizations were performed. The midpoint of every voxel was always used as one of the starting points for minimization, and additional starting conformations were created by assigning each torsion angle either to the midpoint of its range in the voxel or to other values in the voxel that had previously been found, during the searches of other partial structures, to satisfy local constraints. If any minimization resulted in a structure that satisfied the active constraints, the structure and voxel was retained and no further minimizations were performed. If no satisfying substructure was found, the voxel was eliminated. This procedure yielded 1504 substructures for f-Met, 432 for Leu, and 242 for Phe. In the second phase of the divide-and-conquer strategy, these successful substructure voxels were systematically joined without regard to constraints, resulting in ~650,000 structures for the f-Met-Leu "dipeptide". Of these possibilities all but 1360 were eliminated because they violated either NMR or excluded volume constraints (see **Figure 4**). Similarly, joining the f-Met-Leu "dipeptide" structures with the 242 Phe structures and applying the constraints yielded the 56,975 allowed structures for the tripeptide.

In **Figure 5** we illustrate the family of ~60k f-MLF-OH structures consistent with the SSNMR torsion angle and ^{13}C-^{15}N distance data and excluded volume constraints. We can conclude that the experimental data defined most of the structure almost uniquely,

but that some ambiguity remains for the Phe ring and the chain carboxyl terminus. By one of the usual criteria, the quality of the f-MLF-OH structure is especially high (0.02 Å RMSD for the peptide backbone and 0.38 Å RMSD for all heavy atoms). However, it should be noted that because this peptide is small, the RMSD values are not directly comparable with values computed for proteins. Nevertheless, the backbone is fully constrained with the exception of the formyl group, which is not isotopically labeled in our samples. Note that the formyl group was allowed to assume either *cis* or *trans* conformation with low (~5°) angular fluctuations. It therefore appears in the figure as a carboxyl-like group. The C-terminal carboxyl group, for which no SSNMR torsion angle technique currently exists, is not constrained. Similarly, we presently do not have a method to constraint χ^2 and therefore the orientation of the Phe aromatic ring. However, our experiments indicate scaled C^{δ}-H^{δ} and C^{ε}-H^{ε} dipolar interactions for the Phe ring, which are consistent with twofold flipping observed in a number of cases [49, 50]. The side-chain conformations of Met have slightly greater uncertainty than the backbone, largely due to the paucity of constraints on the Met S and C^{ε}.

Conclusions

We have determined the three-dimensional structure of the chemotactic tripeptide, N-formyl-L-Met-L-Leu-L-Phe-OH, based on solid-state MAS NMR constraints (torsion angles and ^{13}C-^{15}N distances) derived from uniformly ^{13}C,^{15}N- and ^{15}N-enriched samples. Simulated annealing procedures were used to define a set of structures consistent with the NMR measurements. The prospects for extension of this work to larger systems are very promising. We note that complete ^{13}C and ^{15}N chemical shift assignments for a U-^{13}C,^{15}N

labeled 62 residue SH-3 domain from α-spectrin have been performed using solid-state NMR data alone [38] representing significant experimental progress. Since the methods employed to obtain the torsion angle constraints are already 3D, they are directly applicable to these larger systems; one structural constraint can be extracted from each resolved cross-peak in the 2D ^{13}C-^{13}C and ^{13}C-^{15}N spectra, and the most critical constraints (ϕ, ψ, $\chi 1$) are derived from the well-resolved $^{13}C^{\alpha}$, $^{13}C^{\beta}$ and ^{13}CO signals. Application of these approaches to α-spectrin [38], bacteriorhodopsin [51] and ubiquitin [52] are currently in progress. In proteins, distance measurements are crucial for determining the global fold, and improved multidimensional SSNMR methods for distance measurements in U-^{13}C,^{15}N labeled proteins are beginning to appear and will be used to measure multiple distances in a single experiment [18].

Acknowledgments

CMR acknowledges the support of a NIH NRSA (GM-20134), CPJ the support of a NSF Graduate Research Fellowship, MH the support of a Danish Natural Science Council Postdoctoral Fellowship and a postdoctoral fellowship from the European Human Frontier Science Program, and MTM the support of a NIH NRSA (GM-20818). We thank L.J. Mueller, B.A. Tounge, M. Hong, and D.J. Ruben for many helpful discussions during the course of this work. This research was supported by grants from the National Institutes of Health (AG-14366, GM-23403, and RR-00995).

Table 1. Torsion angle structural constraints in f-MLF-OH determined by 3D MAS dipolar-chemical shift experiments. Four types of 3D experiments were performed, involving sets of nuclei A-B-C-D: ^{1}H-^{15}N-^{13}C-^{1}H [19], ^{1}H-^{13}C-^{13}C-^{1}H [42], ^{15}N-^{13}C-^{13}C-^{15}N [39, 40], and ^{1}H-^{15}N-^{15}N-^{1}H [24]. In each experiment, B-C 2D chemical shift planes were recorded as a function of the dipolar mixing time between nuclei A-B and C-D. The modulation of the B-C cross peak intensity reported on the relative orientation of the A-B and C-D dipole vectors, and therefore the A-B-C-D torsion angle (assuming invariant bond lengths and angles). Each experiment yielded several types of data, as listed in column three. Due to mirror plane symmetry, multiple solutions are possible in each experiment. Monte Carlo simulations [19] were performed, with a minimum of 10,000 iterations, to determine all possible solutions. Solutions were grouped within local minima; those that occurred in more than 20% of Monte Carlo simulations are listed as most likely solutions (with ±1 σ precision), whereas those that occurred less often are indicated as less likely solutions. (The results in cases where the B and C nuclei were not directly bonded (e.g., H-N[i]-C^{β}[i]-H) depended on two intervening torsion angles (e.g., ϕ and $\chi1$) in a coupled manner; several such 2D solution spaces were included in the final calculations, but are not shown here. The determinations of ψ[i] *via* H-N[i+1]-C^{α}[i]-H data presumed a *trans* peptide bond (ω=180°).)

Residue	Angle	Data Type	Most Likely Solutions (°)	Less Likely Solutions (°)
Met	ϕ	H-N[i]-C^{α}[i]-H	-150 ± 2 -90 ± 2	-6 ± 2
	ψ	N-C^{α}[i]-C'[i]-N	±157 ± 1	n/a
		H-N[i+1]-C^{α}[i]-H^{c}	161 ± 4	108 ± 18 -151 ± 10 78 ± 5 -10 ± 8 24 ± 11
	$\chi1$	H-C^{α}[i]-C^{β}[i]-H_2	-163 ± 3 -77±3	163 ± 3 -43 ± 3
	$\chi2$	H_2-C^{β}[i]-C^{γ}[i]-H_2	±169 ± 2	n/a
Leu	ϕ	H-N[i]-C^{α}[i]-H	-94 ± 2 -146 ± 2	n/a
	ψ	N-C^{α}[i]-C'[i]-N	±91 ± 4 ±120 ± 4	±45 ± 4 ±65 ± 5
		H-N[i+1]-C^{α}[i]-H	-69 ± 4 -51 ± 4	-178 ± 7 -59 ± 10
	$\chi1$	H-C^{α}[i]-C^{β}[i]-H_2	±177 ± 3 -57 ± 3 -64 ± 3	n/a
	$\chi2$	H_2-C^{β}[i]-C^{γ}[i]-H	±173 ± 3 -65 ± 4 -56 ± 4	n/a
Phe	ϕ	H-N[i]-C^{α}[i]-H	-163 ± 2 -77 ± 2	-45 ± 6 162 ± 2
	$\chi1$	H-C^{α}[i]-C^{β}[i]-H_2	68 ± 4 52 ± 4	n/a

Table 2. Comparison of the internuclear ^{15}N-^{13}C distance constraints in f-MLF-OH, determined using frequency selective REDOR [17] and the distances in f-MLF-OMe determined with X-rays. In column three and four are the average distances determined from the 56,975 structures generated by the full search procedure developed here and the CNS calculation. Note that most of the experimental distance constraints are more precise than is customarily observed in solution NMR experiments. The excellent agreement between the experimental and calculated distances lends credence to the structural model illustrated in Figure 4.

Atoms		r_{C-N} (Å) FS-REDOR	X-ray	Full Structure	CNS
Met(N)	Met(C')	—	2.40	2.47	2.47
	Met(C^{α})	—	1.46	1.46	1.46
	Met(C^{β})	2.52±0.02	2.50	2.47	2.47
	Met(C^{γ})	3.20±0.03	3.04	3.25	3.23
	Met(C^{ε})	5.4±0.3	5.71	5.85	5.63
	Leu(C^{β})	5.7±0.7	6.03	5.97	5.92
	Leu($C^{\delta 2}$)	5.5±0.3	6.28	5.92	5.80
Leu(N)	Met(C')	—	1.33	1.33	1.32
	Met(C^{α})	—	2.44	2.41	2.37
	Met(C^{β})	3.12±0.03	3.20	3.07	3.02
	Met(C^{γ})	4.17±0.10	4.56	4.19	4.16
	Met(C^{ε})	5.5±0.3	5.93	5.52	5.56
	Leu(C')	—	2.46	2.42	2.39
	Leu(C^{α})	—	1.46	1.46	1.45
	Leu(C^{β})	2.46±0.01	2.50	2.46	2.45
	Leu($C^{\delta 2}$)	3.64±0.09	3.63	3.53	3.52
Phe(N)	Met(C')	3.4±0.2	3.41	3.54	3.59
	Met(C^{β})	4.12±0.15	4.06	4.11	4.13
	Met(C^{γ})	4.8±0.2	5.43	5.11	5.11
	Met(C^{ε})	5.2±0.3	5.62	5.03	5.08
	Leu(C')	—	1.34	1.33	1.33
	Leu(C^{α})	—	2.43	2.42	2.41
	Leu(C^{β})	3.24±0.12	3.12	3.15	3.11
	Leu($C^{\delta 2}$)	5.4±0.3	5.38	5.34	5.32
	Phe(C')	—	2.37	2.46	2.47
	Phe(C^{α})	—	1.47	1.46	1.46
	Phe(C^{β})	—	2.53	2.45	2.46

[a] Leu(C^{δ}) resonance frequency is 19.6 ppm (*20*).

Table 3. Comparison of the 14 torsion angles derived from the SSNMR structures in f-MLF-OH with the corresponding angles from the X-ray structure of f-MLF-OMe [21]. The average root mean squared deviation of the full ensemble from the representative angles shown below was ±3.5° and the average error for the CNS calculation was ±1.0°.

Residue	Angle	f-MLF-OMe x-ray	f-MLF-OH Representative	f-MLF-OH CNS
Met	ϕ	-146.0 ± 0.7	-145.5	-150.6
Met	ψ	151.3 ± 0.6	158.5	158.0
Met	χ_1	-61.2 ± 0.9	-85.0	-82.3
Met	χ_2	172.9 ± 0.6	171.4	157.8
Met	χ_3	77.5 ± 0.8	87.1	71.3
Leu	ω	169.6 ± 0.6	175	-177.2
Leu	ϕ	-67.7 ± 0.8	-89.5	-92.1
Leu	ψ	-49.1 ± 0.8	-39.5	-44.0
Leu	χ_1	-59.9 ± 0.8	-58.7	-59.4
Leu	χ_2	-178.5 ± 0.8	-178.3	-176.5
Phe	ω	175.7 ± 0.6	176.1	180.0
Phe	ϕ	-155.4 ± 0.6	-166.5	-162.9
Phe	χ_1	64.4 ± 0.8	55.7	53.1
Phe	χ_2	-78.4 ± 0.9	-76.2	-89.2

Rienstra et al., Figure 1

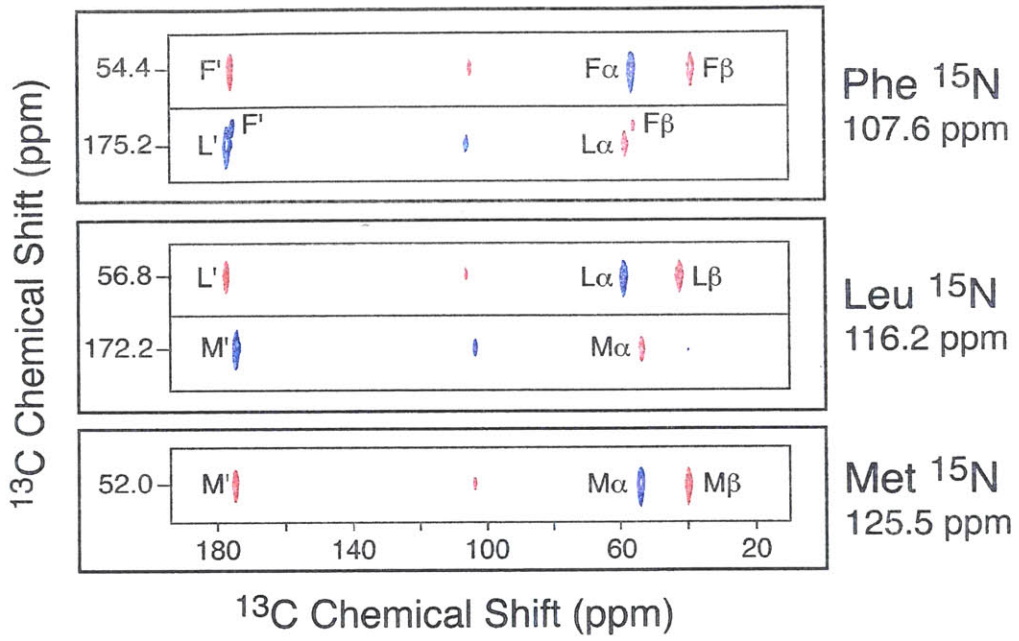

Figure 1. Strip cross sections through the ^{15}N planes of the 3D ^{15}N-^{13}C-^{13}C chemical shift correlation spectrum of f-MLF-OH, showing the backbone resonance assignments. The Met ^{15}N plane (125.5 ppm) shows only ^{13}C cross-peaks from the Met residue. In contrast, the Leu ^{15}N plane (116.2 ppm) shows Met and Leu ^{13}C cross-peaks, and the Phe ^{15}N plane (107.6 ppm) displays Leu and Phe ^{13}C cross-peaks. Since ^{13}C-^{13}C correlations were established using the SPC-5 double quantum recoupling pulse sequence [32] the cross-peaks corresponding to subsequent ^{13}C-^{13}C dipolar transfers alternate in sign [29] (blue and red for positive and negative absorption, respectively). Details of the pulse sequence and experimental parameters used to record this spectrum can be found in [23].

Rienstra et al., Figure 2

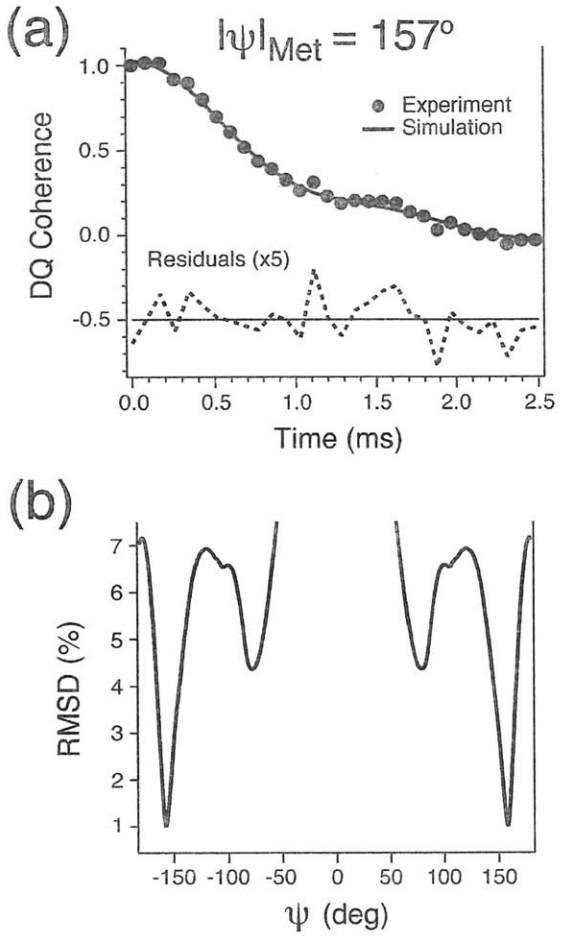

Figure 2. Measurement of ψ_{Met} in f-MLF-OH via the double quantum ^{15}N-^{13}C-^{13}C-^{15}N experiment [39]. (a) Experimental and simulated dephasing of the Met C'-C^{α} double quantum coherence under the C'-N and C^{α}-N dipolar couplings. The best simulation yields a torsion angle of ±157±1°. (b) RMSD between the NCCN simulation and experiment for the Met residue, calculated as a function of ψ.

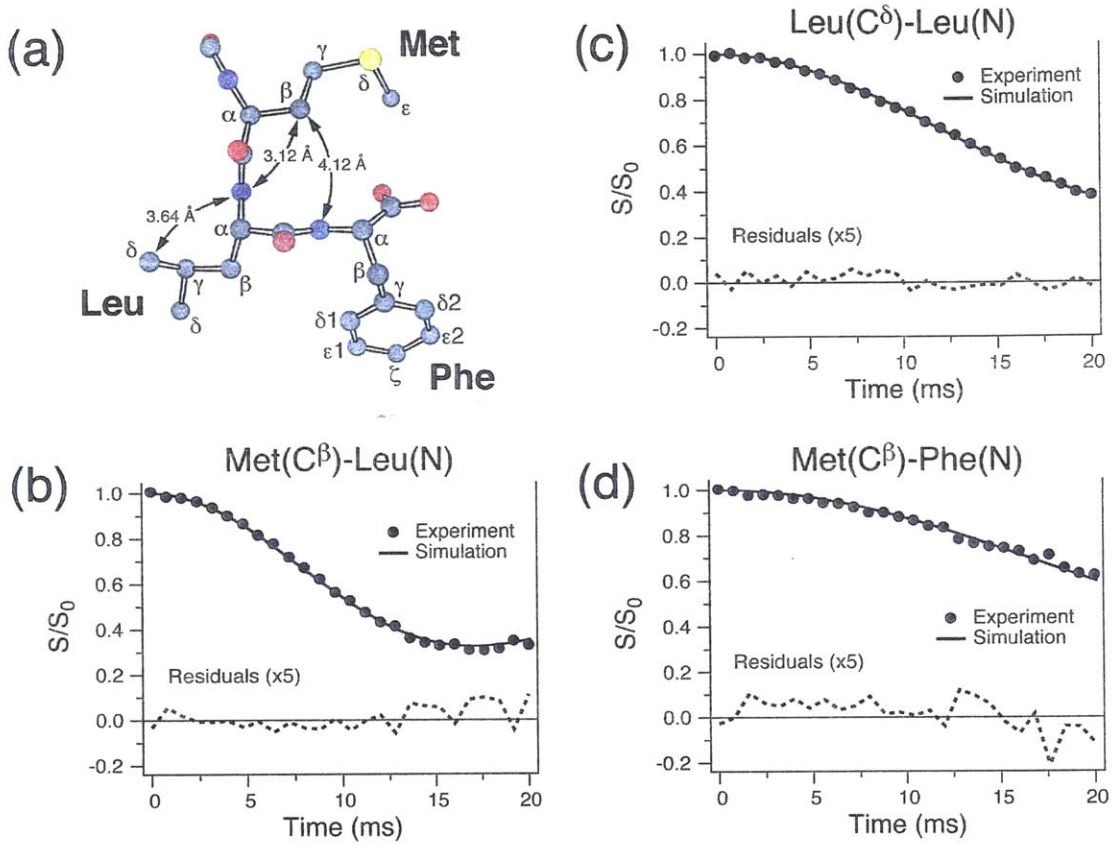

Figure 3. Measurement of carbon-nitrogen internuclear distances in U-^{13}C,^{15}N-f-MLF-OH using frequency selective REDOR [17]. (a) Structural model of f-MLF-OH displaying the distances measured in (b)-(d). Experimental REDOR S/S_0 curves (S_0 and S represent the reference and dipolar dephasing experiments, respectively) and simulations are shown for (a) Met(C^{β})-Leu(N), (b) Leu(C^{δ})-Leu(N), and (c) Met(C^{β})-Phe(N), and correspond to internuclear distances of (a) 3.12 ± 0.03 Å, (b) 3.64 ± 0.09 Å, and (c) 4.12 ± 0.15 Å. A total of 16 distances between 2.5 and 6 Å were measured in f-MLF-OH. Distance measurements were performed in a sample prepared by co-crystallizing U-^{13}C,^{15}N f-MLF-OH with natural abundance f-MLF-OH in a 1:9 ratio, in order to minimize the interference from intermolecular ^{13}C-^{15}N couplings. Details of the pulse sequence and experimental parameters can be found in [17].

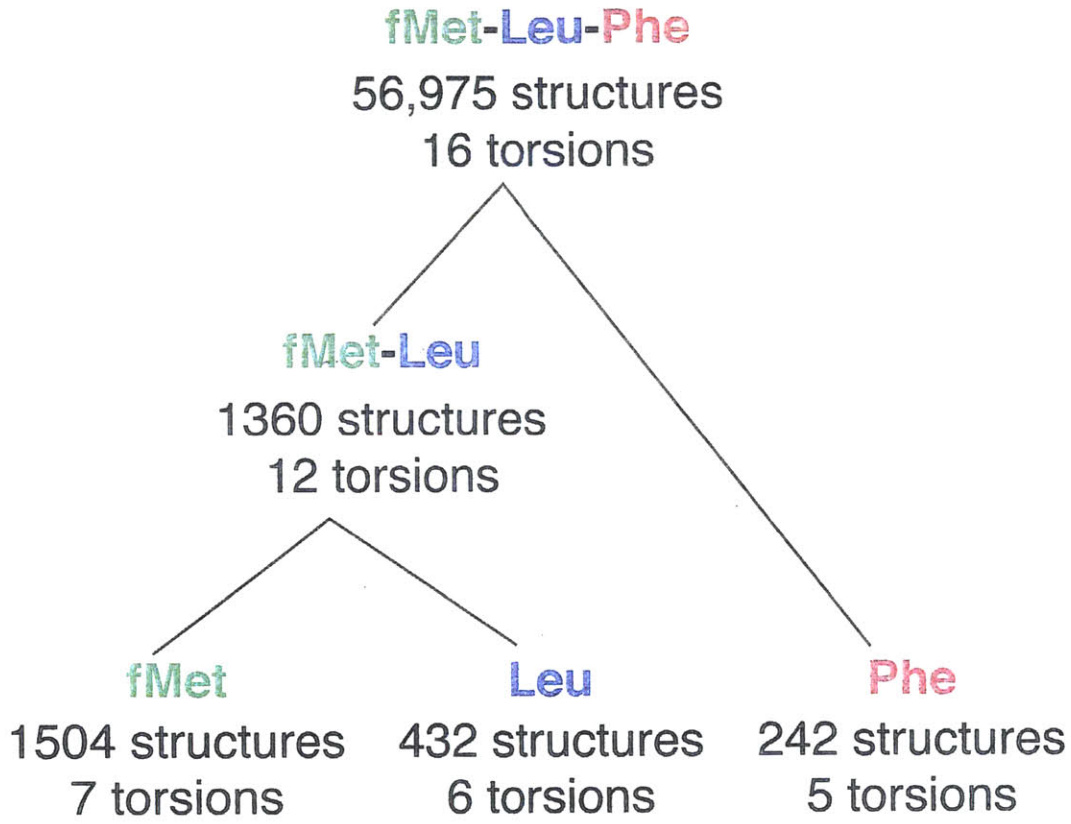

Figure 4: An illustration of the divide-and-conquer strategy used to search conformational space. Starting from the individual residues (bottom) and progressing to the tripeptide, the number of substructures satisfying the SSNMR and excluded volume constraints and the number of searchable torsions are indicated.

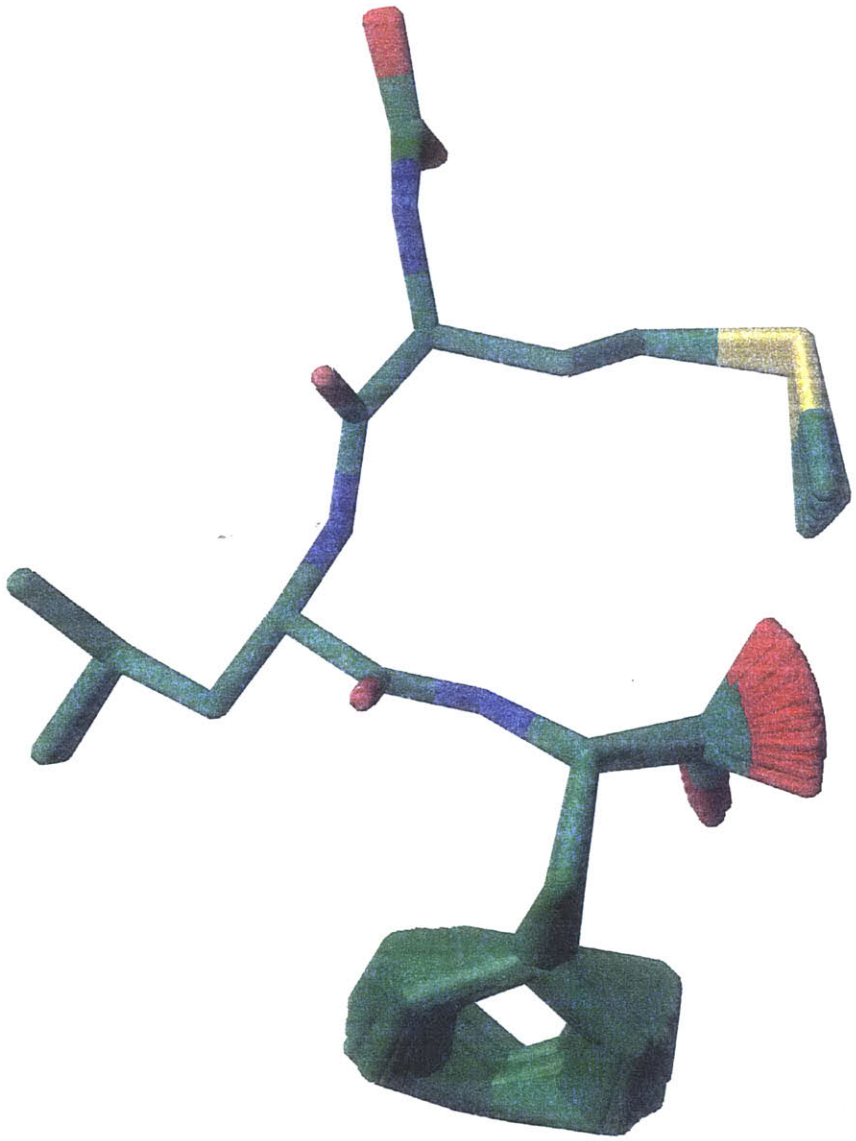

Figure 5: An illustration of a family of 56,975 f-MLF-OH structures consistent with the SSNMR torsion angle measurements, ^{13}C-^{15}N distances, and excluded volume constraints. The structure of the backbone is of especially high quality (0.02 Å RMSD). Since the formyl group was not labeled it was permitted to assume both the *cis* and *trans* conformations in the calculation, and exhibits the appearance of a carboxyl group in the figure. The C-terminus and the Phe ring appear disordered since no torsion angle methods currently exist to constrain the terminal ψ or χ^2 angle. The ring conformation is largely determined by excluded volume constraints, and it is likely undergoing twofold flips (see text). The Met and Leu side-chain conformations are also relatively well defined.

References

1. Wüthrich, K., *NMR of Proteins and Nucleic Acids*. 1986, New York: John Wiley and Sons.

2. Gayathri, C., A.A. Bothner-By, P.C.M.v. Zijl and C. MacLean, Chem. Phys. Lett., 1982. 87, 192.

3. Tolman, J.R., J.M. Flanagan, M.A. Kennedy and J. H. Prestegard, Proc. Natl. Acad. Sci. USA, 1995. 92, 9279.

4. Tjandra, N. and A. Bax, Science, 1997. 278, 1111.

5. Pervushin, K., R. Riek, G. Wider and K. Wüthrich, Proc. Natl. Acad. Sci. USA, 1997. 94, 12366.

6. Yamazaki, T., W. Lee, C.H. Arrowsmith, D.R. Muhandiram and L. E. Kay, J. Am. Chem. Soc., 1994. 116, 11655.

7. Creuzet, F., A.E. McDermott, R. Gebhard, K.v.d. Hoef, M.B. Spijker-Assink, J. Herzfeld, J. Lugtenburg, M.H. Levitt and R.G. Griffin., Science, 1991. 251, 783.

8. Thompson, L.K., A.E. McDermott, J. Raap, C.M. van der Wielen, J. Lugtenberg, J. Herzfeld and R.G. Griffin, Biochemistry, 1992. 31, 7931-7938.

9. McDowell, L.M., C.A. Klug, D.D. Beusen and J. Schaefer, Biochemistry, 1996. 35, 5395-5403.

10. McDowell, L.M., M.S. Lee, R.A. McKay, K.S. Anderson and J. Schaefer, Biochemistry, 1996. 35, 3328-3334.

11. Long, J.R., J.L. Dindot, H. Zebrowski, S. Kiihne, R.H. Clark, A.A. Campbell, P.S. Stayton and G.P. Drobny, Proc. Nat. Acad. Sci. USA, 1998. 95, 12083-12087.

12. Lansbury, P.T., Jr., P.R. Costa, J.M. Griffiths, E.J. Simon, M. Auger, K.J. Halverson, D.A. Kocisko, Z.S. Hendsch, T.T. Ashburn, R.G.S. Spencer, B. Tidor, and R.G. Griffin, Nat. Struct. Biol., 1995. 2, 990-998.

13. Creuzet, F., A.E. McDermott, R. Gebhard, K. van der Hoef, M.B. Spijker-Assink, J. Herzfeld, J. Lugtenburg, M.H. Levitt and R.G. Griffin, Science, 1991. 251, 783-786.

14. Griffin, R.G., Nat. Struct. Biol., 1998. 5, 508-512.

15. Dusold, S. and A. Sebald, Annu. Rep. Nucl. Magn. Reson. Spectr., 2000. 41, 185.

16. Nomura, K., K. Takegoshi, T. Terao, K. Uchida and M. Kainosho, J. Biomol. NMR, 2000. 17, 111-123.

17. Jaroniec, C.P., B.A. Tounge, J. Herzfeld and R.G. Griffin, J. Am. Chem. Soc., 2001. 123, 3507-3519.

18. Jaroniec, C., C. Filip and R.G. Griffin, J Amer. Chem Soc., 2002, (submitted for publication).

19. Rienstra, C.M., M. Hohwy, L.J. Mueller, C.P. Jaroniec, B. Reif and R.G. Griffin, J. Am. Chem. Soc., 2002, (submitted for publication)

20. Showell, H.J., R.J. Freer, S.H. Zigmond, E. Schiffman, S. Aswanikumar, B. Corcoran and E.L. Becker, J. Experimental Medicine, 1976. **143**, 1154-1169.

21. Gavuzzo, E., F. Mazza, G. Pochetti and A. Scatturin, Int. J. Peptide Prot. Res., 1989. **34**, 409-415.

22. Morfew, A.J. and I. Tickle, Cryst. Struct. Comm., 1981. **10**, 781.

23. Rienstra, C.M., M. Hohwy, M. Hong and R.G. Griffin, J. Am. Chem. Soc., 2000. **122**, 10979-10990.

24. Reif, B., M. Hohwy, C.P. Jaroniec, C.M. Rienstra and R.G. Griffin, J. Magn. Resonance, 2000. **145**, 132-141.

25. Bennett, A.E., C.M. Rienstra, M. Auger, K.V. Lakshmi and R.G. Griffin, J. Chem. Phys., 1995. **103**, 6951-6958.

26. Sun, B.Q., C.M. Rienstra, P.R. Costa, J.R. Williamson and R.G. Griffin, J. Am. Chem. Soc., 1997. **119**, 8540-8546.

27. Detken, A., E.H. Hardy, M. Ernst, M. Kainosho, T. Kawakami, S. Aimoto and B.H. Meier, J. Biomol. NMR, 2001. **20**, 203-221.

28. Bennett, A.E., J.H. Ok, R.G. Griffin and S. Vega, J. Chem. Phys., 1992. **96**, 8624-8627.

29. Sun, B.Q., P.R. Costa, D.A. Kocisko, P.T. Lansbury, Jr. and R.G. Griffin, J. Chem. Phys., 1995. **102**, 702-707.

30. Nielsen, N.C., H. Bildsøe, H.J. Jakobsen and M.H. Levitt, J. Chem. Phys., 1994. **101**, 1805-1812.

31. Lee, Y.K., N.D. Kurur, M. Helmle, O.G. Johannessen, N.C. Nielsen and M.H. Levitt, Chem. Phys. Lett., 1995. **242**, 304-309.

32. Hohwy, M., C.M. Rienstra, C.P. Jaroniec and R.G. Griffin, J. Chem. Phys., 1999. **110**, 7983-7992.

33. Schaefer, J. and E.O. Stejskal, J. Magn. Reson., 1979. **34**, 443-447.

34. Baldus, M.A., A.T. Petkova, J.H. Herzfeld and R.G. Griffin, Mol. Phys., 1998. **95**, 1197-1207.

35. Hediger, S., B.H. Meier and R.R. Ernst, Chem. Phys. Lett., 1995. **240**, 449-456.

36. McDermott, A., T. Polenova, A. Bockmann, K.W. Zilm, E.K. Paulsen, R.W. Martin and G.T. Montelione, J. Biomol. NMR, 2000. **16**, 209-219.

37. Egorova-Zachernyuk, T.A., A. McDermott and et al., J. Biomol. NMR, 2001. 19, 243.

38. Pauli, J., M. Baldus, B.v. Rossum, H.d. Groot and H. Oschkinat, ChemBiochem, 2001. 2, 272.

39. Costa, P.R., J.D. Gross, M. Hong and R.G. Griffin, Chem. Phys. Lett., 1997. 280, 95.

40. Feng, X., M. Eden, A. Brinkmann, H. Luthman, L. Eriksson, A. Graslund, O.N. Antzutkin and M.H. Levitt, J. Am. Chem. Soc., 1997. 119, 12006-12007.

41. Ladizhansky, V., M. Veshtort and R.G. Griffin, J. Magn. Resonance, 2002. 154, 317-324.

42. Feng, X., Y.K. Lee, D. Sandström, M. Edén, H. Maisel, A. Sebald and M.H. Levitt, Chem. Phys. Lett., 1996. 257, 314-320.

43. Raleigh, D.P., M.H. Levitt and R.G. Griffin, Chem. Phys. Lett., 1988. 146, 71-76.

44. Costa, P.R., B.Q. Sun and R.G. Griffin, J. Am. Chem. Soc., 1997. 119, 10821-10830.

45. Gullion, T. and J. Schaefer, J. Magn. Reson., 1989. 81, 196-200.

46. Nilges, M., G.M. Clore and A.M. Groenborn, FEBS LETT, 1988. 229, 317-324.

47. Brunger, A.T., P.D. Adams, G.M. Clore, W.L. DeLano, P. Gros, R.W. Grosse-Kunstleve, J.S. Jiang, J. Kuszewski, M. Nilges, N.S. Pannu, R.J. Read, L.M. Rice, T. Simonson, and G.L. Warren, Acta Cryst. D, 1998. 54, 905-921.

48. Lawrence, C.T., J.L. Zhou and A.L. Tits, Technical Report TR-94-16r1 Institute for Systems Research, University of Maryland, 1997.

49. Rice, D.M., Y. Meinwald, H.A. Scheraga and R.G. Griffin, J. Am. Chem. Soc., 1987. 109, 1636.

50. Rice, D.M., R.J. Wittebort, R.G. Griffin, E. Meirovich, E.R. Stimson, Y.C. Meinwald, J.H. Freed and H.A. Scheraga, 1981. 103, 7707.

51. McMahon, M., J. Herzfeld and R.G. Griffin, (in progress) 2002.

52. Rienstra, C.M. and A.E. McDermott, (in progress) 2002.

Chapter 4

Merge Strategy Optimization

Introduction

Chapter 2 described a systematic framework for searching the rotational degrees of freedom of a molecule and evaluating whether voxels (ranges of torsion angles) contain conformations that satisfy the user's constraints. A layer of divide-and-conquer search was constructed which searches individual residues independently, then combines the satisfying conformations to create candidate conformations for dipeptides, then combines those to form larger subchains, etc. This divide-and-conquer layer seems like it should be an improvement, and preliminary results do indicate that it is, but let us examine more closely how the divisions (the merge-trees) are chosen and how those choices influence efficiency.

We begin with an example that helps clarify the implicit assumptions of using a divide-and-conquer approach and that shows how important the choice of merge-tree can be for the run time of conformational search. Then we'll consider the full range of possible merge-trees and ask how to evaluate alternatives using available information.

Based on observations that low-dimensional searches are so much faster than higher-dimensional searches as to be practically free, we ask if there is any benefit to be had from doing "extra" searches. We will explain the benefit and then the cost of searching all possible subchains of one size before searching any subchains of a larger size. We formalize this cost evaluation as a dynamic programming table and show how an optimal merge-tree can be extracted from it.

Performing "extra" searches (searching more subchains than the minimum necessary for a particular merge-tree) sets up two additional types of flexibility that can enhance the

overall search algorithm. First and most importantly, sharing information from the result of one search can simplify the search of an overlapping subchain; we describe how constraint propagation can accomplish this sharing. Moreover, in some cases the benefit reaped is more than enough to offset the cost of searching the extra subchains in the first place. The second additional flexibility is that we may choose the order in which the subchains are searched. That sets up an opportunity to use the A* algorithm to choose which merges to perform next, as if each merge towards constructing the whole molecule were a step in a partial path towards the single destination of searching the whole molecule.

4.1 The Choice of Merge-Tree Can Be Important

We first motivate the research in this chapter by constructing an extreme example, a molecule with a set of constraints such that using a naive choice of merge-tree results in an exponential search and using a seemingly unlikely merge-tree results in a polynomial search.

Problem Framework

In order to categorize example protocols as exponential or polynomial, we will need to vary the size of the problem without changing the nature or character of the problem. Therefore we will use variable-length alpha-helices of polyalanine. (To define an alpha-helix of any length, we apply torsion angles of $\phi = -57.0°$, $\psi = -57.0°$, and $\omega = 180.0°$ to every residue in the chain.)

For the divide-and-conquer strategy in chapter 2, we used full, balanced binary merge-trees by default. This default tree is not always optimal, and we will show here that in the worst case, it could be exponentially worse than optimal.

We will design a constraint set so as to sabotage searching with the default tree but not to sabotage all possible merge-trees. Suppose our molecule has 2^k residues, the ideal number for the advantage of the default trees. (When the number of residues in a molecule is not a power of 2, especially if $N = 2^k + 1$, pathological cases for default merge-trees are easy to imagine.)

The default merge-tree

The default merge-tree for a molecule with $N = 2^k$ residues would be the full, perfectly balanced tree with $(k + 1)$ levels, defined in section 2.3. (See Figure 4-1.)

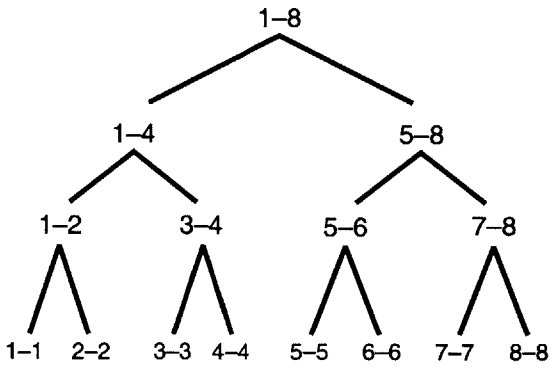

Figure 4-1: The default merge-tree for a molecule with $2^3 = 8$ residues.

Recall from section 2.3 that the most important reason for doing a divide-and-conquer search is allow more opportunities for pruning the search. By defining a tree without regard to the constraint set, there is a possibility that the chosen fragments and merges may not correspond to any active constraints and that no pruning will be possible.

The constraint set

Let us define a set of interatomic distances (for the same molecule of $N = 2^k$ residues) such that when using the "default" merge-tree, no constraints are active prior to the final merge. (The final merge is the top-most in the tree, with the whole molecule as the result.) Constrain the distance between atom i and atom j if and only if

$$\text{the residue of atom } i \text{ is } \leq \frac{N}{2},$$
$$\text{and the residue of atom } j \text{ is } > \frac{N}{2}.$$

Note that in this constraint set, distance constraints may be of any length, not necessarily less than 5 or 6 Å. For the "padding" on the constraints, let us choose plus or minus 0.05 Å from the measured distance in the helical structure.

As we have defined it, this constraint set has no constraints within the left half of the molecule nor within the right half of the molecule, but has every possible constraint

involving one atom in the left half and the other in the right. As divide-and-conquer search traverses the default merge-tree, bottom to top, no constraints will take effect prior to the final combine operation, when the two halves of the molecule are joined. Even if this set of constraints suffices to specify a unique satisfying conformation for the whole molecule, the search will have to explore and enumerate every self-avoiding conformation[1] of both $\frac{N}{2}$-sized halves. Because the number of accessible conformations for a subchain is exponential in the length of that subchain, the cost of searching this molecule with a naive binary merge tree would be $\mathcal{O}(2 * e^{\frac{N}{2}}) \in \mathcal{O}(e^N)$.

Some technicalities

When breaking a chain into two subchains at a peptide bond, we choose to include the peptide bond (and therefore the atoms that define it) in both subchain pieces. Because the C-terminal atoms of any residue are used when defining the next residue, and the N-terminal atoms of any residue are used when defining the previous residue, we also need to exclude from our constraint set any interatomic distances where atom i is at the C-terminus of residue 2^{k-1} or when atom j is at the N-terminus of residue $2^{k-1} + 1$.

Applying the triangle inequality to all the entries of the molecule's distance matrix (the "triangle smoothing" option) would allow some constraining information to trickle across the artificial boundary we're erecting between the left and right halves of the molecule, and would indirectly provide constraints that are active within the left half and within the right half of the molecule. Atoms near the boundary, thus constrained by the triangle inequality, would not contribute to the exponential growth of overall run time. The remainder of each half of the molecule would behave the same, and exponential growth would ensue, but larger examples would be required in order to display the phenomenon as clearly as it can be displayed without the use of the triangle inequality. Therefore we neglect to apply triangle smoothing in this example.

The "manually-constructed" merge-tree

A merge tree that exploits the structure of the constraints might conceivably allow a divide-and-conquer search of the same molecule with the same constraints to cost far less. Any

[1] A self-avoiding or *accessible* conformation is one that doesn't involve van der Waals clashes.

such tree would merge a subchain from residue(s) $\leq \frac{N}{2}$ with a subchain from residue(s) $> \frac{N}{2}$ at the beginning of the search, not the end.

Let us define another merge-tree such that *every* merge has active constraints. Call this the "manually constructed" merge-tree. Note that this merge-tree would under other circumstances probably be worse than the default merge-tree because its depth is linear, not logarithmic, in the number of leaves, and a substantial number of merges involve minimizing fragments almost as large as the whole molecule. In the default merge-tree, only the final merge requires such large minimizations.

Build the "manually-constructed" merge-tree so that every merge spans the artificial boundary between residues 2^{k-1} and $2^{k-1}+1$. Start with the first, lowest-level merge

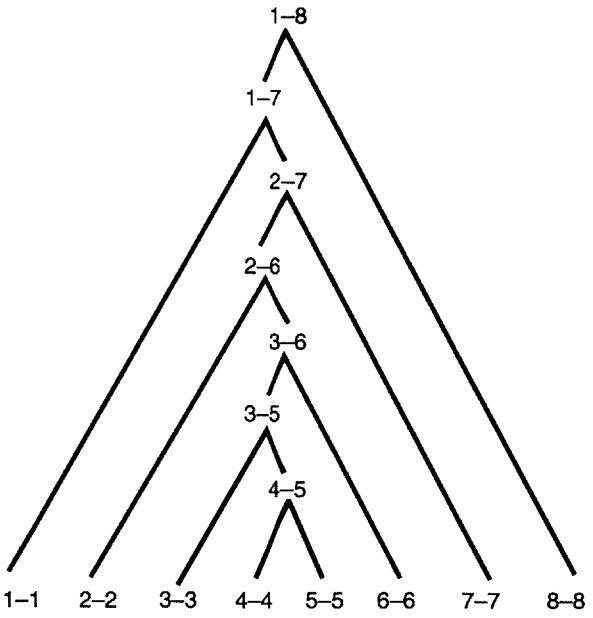

Figure 4-2: A manually-constructed merge-tree for a molecule with 8 residues.

between those two residues themselves. (See Figure 4-2.) Then merge $2^{k-1}-1$ with the dipeptide. Then merge another residue onto that tripeptide. We have chosen to alternate adding a new residue from the left and from the right but the key feature of this merge tree is that residues are added one at a time onto the subchain that spans the boundary. Therefore all subchains searched will have at least one residues on the left and at least one on the right, which means every subchain constructed (except leaf subchains) will have active constraints.

Results

We used divide-and-conquer (with VOXELIZED TREESEARCH on leaves) according to the default and "manually constructed" merge-trees, to perform a systematic search at 120° resolution on helical polyalanine subject to the set of constraints we constructed above. The van der Waals radii were set to 90% of half sigma,[2] and the threshold for summed squared violations was 0.005 Å^2. The ω torsions (for peptide bonds) were permitted to vary between 175° and 185°.

Number of Residues	Run Time by Merge-Tree	
	default	manually-computed
2	0.19	0.19
4	64.4	1.35
8	2,500,000.†	11.9
16		180

Figure 4-3: Run times with default and manually-constructed merge-trees.
†The run time for the simulation of 8 residues with the default merge-tree was extrapolated from an actual simulation that was terminated after 590333.590 seconds. In that time it had searched every node below the root and had searched 1,618,126 out of 6,638,760 candidates for the final merge.

Using the default merge-tree forces the algorithm to enumerate satisfying conformations for the left and right halves, even though there are no external constraints active on either left or right halves of the molecule. Since the left and right halves have no active constraints, and since the number of unconstrained (self-avoiding) conformations for a molecule is exponential in the length of the chain, divide-and-conquer with the default merge-tree must take an exponential amount of time. No matter how divide-and-conquer calculates its conformations, simply enumerating the conformations for the left and right halves requires exponential time. Indeed, exponential growth does accurately describe the few trial runs we were able to complete.

In contrast, each time the size of the molecule doubles (each time k increases by one), the run time of the algorithm using the "manually constructed" merge-tree increases by approximately a factor of ten. This rate of growth is polynomial. By constructing the constraint set adversarially with respect to a given merge-tree, we have created extreme

[2]half of sigma in the CNS parameter set `parallhdg.pro` [8, 19]

but vivid evidence for why the choice of merge-tree is important.

4.2 How Can We Choose Good Merge-Trees?

How can we choose merges that are appropriate to the constraint set without manual intervention? All other things being equal, small subchains can be searched more quickly than large ones. However, the availability of active constraints is not always equal, and (as shown in previous extreme example) the availability of active constraints can be a far more important consideration when choosing a merge-tree for divide-and-conquer than the sizes of subchains. The number of conformations does often correlate with the size of the subchain, but the important thing is to choose subchains with few conformations.

Simple descriptions of divide-and-conquer usually explain that a large problem is divided into pieces, the pieces are easy to solve, and the solutions for the pieces are combined to create the solution for the large problem. While that is entirely true, we emphasize that what makes divide-and-conquer especially effective for constrained searches of molecular conformations is that divide-and-conquer can isolate and search the most highly-constrained pieces of the molecule and then use the resulting small set of satisfying conformations to build satisfying conformations for the rest of the molecule.

After discussing two intuitive considerations, locality and ordering, we will define a cost function and a formal methodology for computing an optimal merge-tree.

Locality

We say there is *locality* in a constraint set if the existence of a constraint between atoms i and j means it is more likely than average that atoms near i and atoms near j will also be constrained (by non-trivial or non-redundant constraints). When a constraint set exhibits locality, it may be possible to define some subchains with particularly large numbers of active constraints, and those subchains might have particularly few satisfying conformations. Whenever possible we want to satisfy clusters of constraints with local, low-dimensional searches, so that we can avoid instantiating extra variables.

In chapter 2 we mentioned three reasons for using divide-and-conquer search over treesearch: because it evaluates individually each piece it adds, because it stores the satisfying conformations of recurring clusters of variables, and because it can allow you to define

subproblems with a smaller average size. An additional reason, related but crucial, why divide-and-conquer improves efficiency is that it has the freedom to *choose* well-constrained subproblems and to exploit the locality of the constraint set. Good merge-trees define subproblems that have as few satisfying conformations as possible. If all subproblems have the same number of satisfying conformations, then choosing a merge-tree that optimizes the average subchain length (like the default merge-tree) would make sense, but subchains with short length are not necessarily always the subchains with the smallest number of conformations.

Ordering

Consider performing treesearch at 120° resolution on a molecule with four bonds. If there

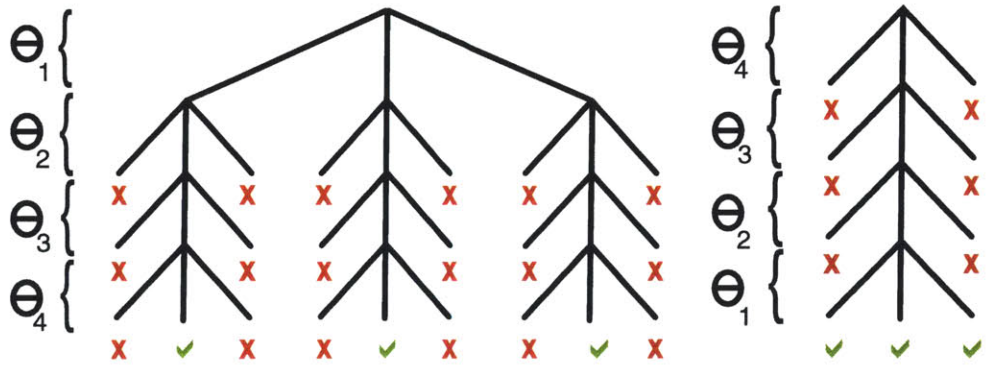

Figure 4-4: The search is less efficient if the variable with many legal assignments is instantiated first (left) instead of last (right).

are many feasible possibilities for θ_1 and few for θ_2, θ_3, and θ_4 (see Figure 4-4), then it is more efficient to search θ_1 last instead of first. Likewise, if a molecule can be broken into well-constrained subchains and poorly-constrained subchains, the search will be more efficient if we build satisfying conformations for the well-constrained subchains first and then, as late as possible, build onto them the poorly-constrained regions with higher variability. Searching well-constrained subproblems first corresponds to the "first-fail" approach to variable-ordering in generalized constraint satisfaction problems [28, 31].

An efficient merge tree isolates subproblems with well-constrained, efficiently-enumerable solutions and puts them at the early nodes of the merge-tree so that they can be solved first. It places unconstrained subchains near the root of the merge-tree (unconstrained variables

near the bottom of the search tree) so that (A) they won't have to be enumerated before needed (which is when everything is needed, in the final search of the whole molecule); and (B) they won't have to be enumerated before the little constraint they do have is applied, namely when all variables are instantiated and all constraints become active.

Cost Function

Before we attempt to design an algorithm that chooses a good merge-tree, let us clarify our measure of cost. Recall that a merge-tree depicts a divide-and-conquer search strategy: subchains corresponding to leaf nodes are searched directly and interior nodes are searched by combining the solutions of the right and left child subchains.

The run time of searching a subchain depends on factors as diverse as the number of rotatable bonds, the number and sizes of atoms attached to those bonds, the number and type of constraints, the information or the redundancy in the constraints, the number of voxels with conformations that satisfy the constraints, and the ease of finding satisfying conformations within each satisfying voxel. These factors are in addition to user-defined constants such as the resolution of the search and how long to search within each voxel before giving up.

Let us design a simple cost function that can be computed quickly and that may correlate approximately with run times. Ideally the cost function would be an accurate predictor of run time, but when in doubt about how to define the cost, we will err on the low side so that the cost will be able to serve as a lower bound on run time.

From this point forward, we consider the cost to be an abstract entity that is defined and computed independent of the run time. After the cost is defined and implemented, we will show that the cost function is useful for making decisions, such as for choosing merges and merge-trees. Although we intend the cost values to resemble run times, the cost itself an exact quantity, not an estimate. The only reason we stress this distinction is because the cost entities themselves will be estimated later in this chapter.

Let us define the cost of searching a leaf subchain as the number of feasible conformations (the number of solutions or satisfying voxels) times the cost of searching each voxel. This choice ignores the cost of ruling out regions of conformation space that have no satisfying conformations, but a cost that counted every possible voxel (as with GRIDSEARCH) would be an overestimate.

The search of a non-leaf subchain is performed by combining solutions from left and right child subchains into candidates for the larger subchain and then evaluating the candidate voxels. We define the cost of searching a non-leaf subchain to be simply the number of candidate voxels times the cost of evaluating a voxel. The number of candidates is the number of solutions for the left child times the number for the right. (This definition is not a strict lower bound because if the omega torsion at the shared peptide bond has multiple allowed ranges, (e.g., *cis* as well as *trans*), then there may be some combinations that can be ruled out immediately on the basis of incompatible ranges for the shared torsion.

The cost of evaluating a voxel is more complicated and it depends on the voxel. It can be insignificant if the initial voxel point given to the minimizer (such as the midpoint) satisfies the constraints without need for further minimization. In contrast, if the voxel contains no satisfying conformations, the algorithm will choose a starting point, minimize with perhaps 50 steps, fail, and repeat the whole process (with a fresh starting point and 50 more steps of minimization) a predetermined number of times before concluding that no satisfying conformations exist.

We have chosen to define the cost of evaluating a voxel as the number of residues in the subchain, which is roughly proportional to the number of dimensions of the voxel. We acknowledge that many simplifications are inherent in this definition. Indeed, the square of the number of residues (or the square of the number of torsions) might be more accurate in more cases than our choice of the linear cost. However, we prefer to err on the low side (providing a guaranteed lower bound) because this cost function will later be used to evaluate whether to perform additional work searching small subchains in exchange for possibly reducing the future costs of searching large subchains. If we overestimate the costs of large subchains relative to small ones, the algorithm could overestimate potential savings and waste its time. Our cost function can then be expressed as follows:

$$\begin{aligned}
\text{NODECOST(leaf subchain } m\text{–}m\text{)} &= \text{NUMSOLUTIONS(subchain } m\text{–}m\text{)} \\
&\quad \times \text{VOXELCOST(1 residue).}
\end{aligned}$$

$$\begin{aligned}
\text{NODECOST(subchain } m\text{–}n\text{)} &= \text{NUMSOLUTIONS(subchain } m\text{–}i\text{)} \\
&\quad \times \text{NUMSOLUTIONS(subchain } (i{+}1)\text{–}n\text{)} \\
&\quad \times \text{VOXELCOST}(n - m + 1 \text{ residues}).
\end{aligned}$$

$$\text{VOXELCOST}(n \text{ residues}) = n.$$

Since the cost of evaluating a non-leaf subchain was based on the assumption that its child subchains had already been searched, the entire cost of searching a non-leaf subchain from scratch is actually the cost of all the nodes in its subtree. In particular, the cost of searching the whole molecule is the sum of the costs of all the nodes n_i in the merge-tree Γ.

$$\text{TREECOST } (\Gamma) = \sum_{\mathcal{S} \in \Gamma} \text{NODECOST } (\mathcal{S}).$$

The Optimum Cost

Given the number of satisfying conformations for every subchain, we can compute the NODECOST for every subchain. Then, given enough time, we could enumerate all possible merge-trees, compute their TREECOSTs, and determine the optimum.

We claim that the following function also computes the optimum cost, over all possible choices for merge-trees, for the search of a given molecule.

BESTTREECOST (subchain m–n)

$$= \min_{m \le i < n} \begin{pmatrix} & \text{NUMSOLUTIONS(subchain } m\text{–}i\text{)} \\ \times & \text{NUMSOLUTIONS(subchain } (i{+}1)\text{–}n\text{)} \\ \times & \text{VOXELCOST}(n - m + 1 \text{ residues}) \\ + & \text{BESTTREECOST}(m\text{–}i) \\ + & \text{BESTTREECOST}((i{+}1)\text{–}n) \end{pmatrix}$$

Dynamic Programming

Computing the BESTTREECOST is an ideal case for using dynamic programming [1]. Let us build a dynamic programming table of BESTTREECOST entries such that the first residue

of a subchain gives the row of the entry and the last residue gives the column. Entry

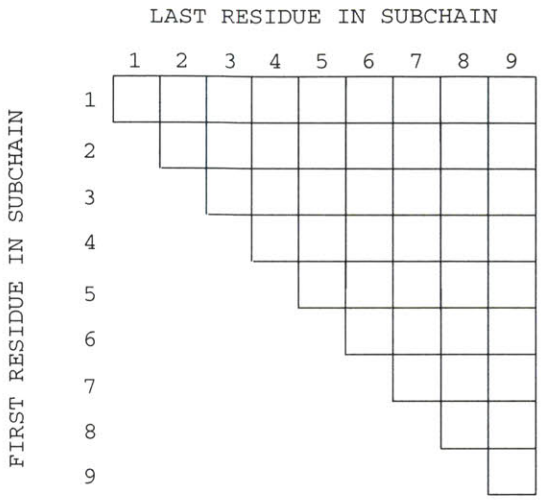

Figure 4-5: One way to set up a dynamic programming table for building up values of BESTTREECOST.

i–j of the table would contain the BESTTREECOST of the possible subtrees rooted at i–j. (See Figure 4-5.) An algorithm to fill in BESTTREECOSTs would start with the leaf subchains along the main diagonal of the table, then proceed, in order of increasing number of residues, to fill in another diagonal, using a linear number of look-ups from previous entries to compute each new entry.

If we perform a 45 degree rotation on the table in Figure 4-5, then the table entries are

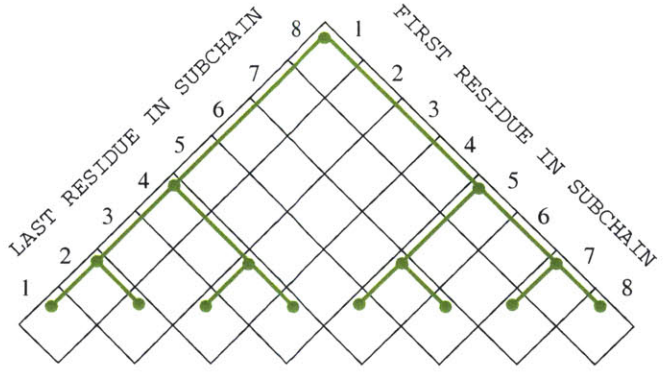

Figure 4-6: The same table as in Figure 4-5, rotated 45°, can be aligned with the default merge-tree for 8 residues.

arranged from top to bottom in order of subchain size, in such a way that merge-trees can

easily be superimposed on the tables; see Figure 4-6. The leaves are now along the bottom, in what we will call a "diagonal row." Rotating the table has the important advantage of positioning each subchain entry approximately where it would be in a merge-tree for creating that subchain, and that provides a visual cue for the relationships between the entries in the "table." Although Figure 4-6 shows a default merge-tree superimposed on the table, any merge-tree could be superimposed equally well.

The Optimal Merge-Tree

Computing a dynamic programming table of BESTTREECOSTs for a molecule gives the optimal merge-tree as a side-effect. For each subchain, m–n, the choice of i used in BESTTREECOST(m–n) gives the best division of m–n into left and right children. As with any case of dynamic programming, although cost information is computed "bottom-up" starting with small subchains, we read out the information in reverse order, "top-down," for deducing the globally optimal merge-tree. Taking the i chosen in BESTTREECOST(1–N), we break the molecule into two subchains, 1–m and (m+1)–N. Then we look up the choice of i for BESTTREECOST(1–m) to break up that subchain, and so on until reaching leaves. The set of divisions (i values) that gave rise to the merge-tree with optimal TREECOST is the optimal merge-tree.

As in any case of dynamic programming, the number of satisfying conformations for subchain 1–N cannot depend on how they were created, whether by combining, say, 1–1 $\times$ 2–N or 1–$\frac{N}{2}$ $\times$ ($\frac{N}{2}$+1)–N. The choice of i used for computing the BESTTREECOST of a child subchain does not influence the BESTTREECOST of the parent.

This dynamic programming assumption is actually a version of our previous assumption that a systematic solution to a conformational search problem does not depend on how the systematic search was performed. In other words, the voxels enumerated should be the same, except for order, whether found by by TREESEARCH, divide-and-conquer, or any other method, long as the constraints and the resolution are the same.

This assumption does turn out to be false sometimes in practice. Especially when a subchain has many degrees of freedom, the minimizer can fail to find any satisfying conformations even when a satisfying region in the voxel does exist. A difficult but potentially crucial area for future research might be to study factors that influence completeness and the probability of false negatives, such as whether the choice of merge-trees matters in prac-

tice, or whether the choice of satisfying solutions at low dimension (which are combined to create candidates at higher resolution) can improve completeness.

4.3 Searching All Subchains (OMNIMERGE)

While performing a variety of conformational searches using divide-and-conquer, we observed that a large fraction of run time is consumed during the final combine operation to create the whole molecule. The time taken to search the leaves or to perform the low-dimensional merges is utterly dwarfed by the time for performing the higher-dimensional merges. Therefore we ask if there is any conceivable benefit to be reaped during high-dimensional searches (for example during the final merge) if we performed some additional lower-dimensional searches.

Performing Locally Optimal Merges

Suppose we performed all possible dipeptide merges (1–2, 2–3, 3–4, 4–5, ...) instead of just the pairs specified in some merge-tree (such as 1–2, 3–4, ...). Obviously this would require twice as much work, but for the moment let us assume the cost of this extra work is insignificant. Could anything be gained from it? Nothing is gained if the next goal is to create tetrapeptides by combining pairs of dipeptides. If on the other hand the next goal is to create tripeptides, then we would have two different possible merges for creating each subchain and we could choose whichever one has the lower NODECOST. For example, when creating subchain 1–3, we could choose between combining subchains 1–1 × 2–3 or combining subchains 1–2 × 3–3.

Let OMNIMERGE be the algorithm that searches all possible subchains of the whole molecule, in order of increasing size, and that chooses the children for each combine operation so as to minimize the NODECOST.

BESTNODECOST(m–n)

$$= \operatorname*{MIN}_{m \le i < n} \left(\begin{array}{ll} & \text{NUMSOLUTIONS}(m\text{–}i) \\ \times & \text{NUMSOLUTIONS}((i+1)\text{–}n) \\ \times & \text{VOXELCOST}(n-m+1). \end{array} \right)$$

If we have searched all subchains of length $\le l$, we know how many satisfying conformations there are for each subchain of length $\le l$, which is enough information to evaluate the

various cost functions (NODECOSTs, BESTTREECOSTs, etc.) for subchains of length $\leq l+1$. As OMNIMERGE searches subchains of increasing length, we can trivially fill in successive diagonal rows of the BESTTREECOSTs dynamic programming table. Filling in an entry of the table requires up to l comparisons but note that we do not actually perform l merges; we compare the costs of l merges and perform the best one. The cheapest merge according to BESTTREECOST is added to to the dynamic programming table, and the cheapest merge according to BESTNODECOST dictates which combine operation OMNIMERGE actually performs. OMNIMERGE performs the locally optimal merge, without regard for the costs of any subtrees, but it keeps track of which merges would be globally optimal if the subtrees had not already been provided. In other words, because the BESTTREECOST of m–n takes into account the costs of creating the subtree of m–n, it corresponds to the globally optimal merge-tree for creating m–n. In contrast, OMNIMERGE chooses child subchains according to BESTNODECOST, without considering costs of creating the children.

Previous Work

Like OMNIMERGE, the "buildup" procedure of Scheraga *et al.* [57, 71, 22] also searches all subchains of the molecule. However, instead of choosing the most favorable left and right child subchains to merge (which is the key idea of OMNIMERGE), the buildup procedure always creates an i-residue subchain by merging the subchain for the first $i-1$ residues with the subchain for the last $i-1$ residues. The benefit of a large overlap between the right and left children is that longer-range constraints will have already been satisfied. (For example, if the merge is to create subchain 1–8, then constraints involving residues 1–7 or residues 2–8 have already been satisfied; the only newly active constraints would be between residues 1 and 8.) The disadvantage of a large overlap between the children is that longer subchains tend to have a larger number of satisfying conformations. (The number of conformations for the right child times the number of conformations for the left could be very large.) This disadvantage is not a problem for Scheraga *et al.* because they keep only a limited number of energetically-favorable conformations for each subchain. Although their approach is not systematic, it is "closer" to systematic than many other popular search methods, such as simulated annealing.

4.3.1 Mitigating the Cost of OMNIMERGE

Methods to improve the theoretical worst-case performance of an algorithm often involve paying overhead costs for a sophisticated algorithm, slowing down the answers to easy questions, in exchange for much better performance on very difficult problems. In practice one could always run a simple algorithm in parallel with a more sophisticated algorithm, and if the naive approach gives a quick answer, halt both. The worst case penalty for having the "insurance policy" of the sophisticated algorithm would only be the use of twice as many processors.

We now present several variations that may run faster than OMNIMERGE in many cases, although they generally provide less "insurance" than OMNIMERGE. Also, when a variation involves omitting some of the subchains, the BESTTREECOST dynamic programming table will not be filled in entirely and the globally optimal merge-tree can no longer be computed as a side-effect of the search. Yet, in cases where the conformational search result is more important than knowing the optimal merge-tree, one of these suggestions might be appropriate.

Imposing an upper limit on subchain sizes

The cost of searching (combining and minimizing) large subchains can be substantial. Is there some threshold size beyond which it becomes too costly to search *all* possible subchains, compared with the savings of choosing good merges? If N is the size of the whole molecule, there will be high cost for searching subchains of size $N-1$ and $N-2$, and one might argue that the likely benefit would be small. Since the whole molecule can be searched by combining two subchains that are roughly size $\frac{N}{2}$, one might prefer not to search any subchains that are significantly larger than $\frac{N}{2}$ (except of course the whole molecule).

Let LIMITED OMNIMERGE be the algorithm where we search all subchains of length up to some limit L, where L is at least large enough to include some right and left children of the root. ($L \geq \lceil \frac{N}{2} \rceil$.) In the experiments below, we chose $L = 0.667N$. LIMITED OMNIMERGE searches many more subchains than would be searched using a default merge-tree, but fewer than regular OMNIMERGE, and each subchain search (except the final merge) gets to use its optimal children.

We could also use an algorithm that stopped searching all possible subchains at a thresh-

old smaller than half the size of the molecule, to try to save more time. It could perform its final merges according to some pre-determined strategy, such as a default merge-tree or some greedy strategy that aims to create the longest possible subchain as soon as possible.

Imposing a lower limit on subchain size

As described earlier, one could use dipeptides or other fragments larger than single residues for the leaves of the divide-and-conquer algorithm. Analogously, one could construct the dynamic programming table as if amino acids were paired, thus reducing the size of the table and significantly reducing the number of subchains to be searched. If we think of the task of examining every possible subchain as itself as a type of systematic search to find well-constrained subproblems, then pairing the residues would still be a uniform and systematic search, except with coarser resolution.

Requiring divisibility of subchain sizes

Another way to reduce the cost of an OMNIMERGE-like search is to impose a divisibility requirement on the subchain sizes. Only subchains with lengths divisible by d (or smaller than d) would be searched. Assume $d|N$ and $d \ll N$. If the whole molecule has 10 residues and one imposes a divisibility limit of 2, then subchains with lengths 1, 2, 4, 6, 8, and 10 would be searched (not just subchains sizes that are powers of two). This would fill in every other diagonal row of the dynamic programming table. Requiring divisibility of subchain sizes is similar to having a lower limit on subchain sizes because it also "covers" the space of all possible subchains, but at lower density, as if decreasing the resolution of the search for well-constrained subproblems.

Removing orphaned subchains

When using one of the variants of OMNIMERGE that omits certain subchains, we can often remove additional subchains from consideration if all the larger subchains they could help create (via the combine operation) have already been removed. For example, if a molecule has 6 residues and LIMITED OMNIMERGE only searches subchains of up to 4 residues, then subchains of size 5 are removed from consideration. Subchain 1–4 is potentially useful because it could be combined with 5–6 to create the whole molecule. Likewise 3–6 might be useful because it could be combined with 1–2. In contrast, 2–5 can never be useful

because combining it with anything (such as 1–1 or 6–6) would create a subchain of size 5. Subchain 2–5 is within the size limit, but the decision not to search subchains 1–5 or 2–6 leaves subchain 2–5 "orphaned." Orphans can be removed from consideration without any risk of increasing run time. However, with the PROPAGATION algorithm introduced later in this chapter, it is conceivable that skipping an orphaned subchains could worsen the run time.

Testing limited versions of OMNIMERGE

The methods suggested so far for reducing the run time of OMNIMERGE searches involve skipping some subchains and not completing the entire dynamic programming table. Such methods are difficult to evaluate apart from some context (such as a particular molecule with a given set of constraints) because the context determines how the skipped subchain contributes to the overall solution of the problem. If the skipped subchain happens to be part of the optimal merge-tree for the whole molecule, or even just part of a locally optimal merge for some subchain, then its removal might slow the search of that subchain, possibly enough to decrease overall performance. If the skipped subchain is not part of any locally optimal merges (or if it is part of a locally optimal merge but the suboptimal alternatives are almost as good) then skipping the subchain will improve performance. The impact on performance may be a bimodal distribution as contexts vary. Methods that skip some subchains will sometimes perform better and sometimes worse than OMNIMERGES. Although it might be interesting to test many variants in many contexts, the preliminary trials in this chapter will only use one additional variant of OMNIMERGE: LIMITED OMNIMERGE with an upper threshold on subchain size of $L = 0.667 * N$, and with orphaned subchains removed.

Computing the optimal merge-tree at low-resolution

An entirely different approach to mitigating the cost of OMNIMERGE is to compute the optimal merge-tree at lower resolution than the eventual search. Assuming that the optimal merge-tree for searching a molecule at low resolution is similar to (or sometimes even the same as) the optimal merge-tree for searching the same molecule (with the same constraints) at higher resolution, then one could perform OMNIMERGE once at low resolution to compute the optimal merge-tree, and then perform an ordinary divide-and-conquer search at high resolution using the merge-tree that was found to be optimal at low resolution.

4.3.2 Performance Examples

The performance of OMNIMERGE (and LIMITED OMNIMERGE) may be better or worse than divide-and-conquer, depending on whether the default merge-tree is well suited to the molecule and the constraint set for each particular case. (Divide-and-conquer was introduced in Section 2.3.) Although we are about to present two major enhancements to OMNIMERGE (PROPAGATION in section 4.4 and A* ordering of subchains in section 4.5), we will first show how the unenhanced algorithm performs.

We used the same parameters and the same constraint set for the nine-amino acid 1RST peptide as in section 2.3.[3] When searching with 40° resolution for the backbone torsions and 120° resolution for the sidechain torsions, there are hundreds of satisfying conformations. Different search methods yielded different numbers of satisfying conformations after running for different amount of time, as reported in Figure 4-9.

We used the same merge-tree (Figure 2-18) defined in chapter 2 to be our default, but in the case of 1RST, the default merge-tree may be an unusually poor choice because the leftmost residue (combined last in the default) has more solutions by far than any other leaf. Therefore, we also inspected the 1RST crystal structure visually and designed another merge-tree that we thought might place nearby residues (residues likely to have active constraints) together at low levels of the merge-tree, while also preserving balance (which means minimizing the average subchain size) in the tree. See Figure 4-7. No objective criteria were used and we make no claims of optimality for this merge-tree.

As another comparison, we performed a full OMNIMERGE search at low resolution (120° for all torsions); then, using the dynamic programming table of BESTTREECOSTS produced as a side-effect of that search, we determined the optimal merge-tree for searching at low resolution (see Figure 4-8). Finally, we performed the regular search (40° for backbone torsions and 120° for sidechain torsions) with that merge-tree.

In terms of run time, the default merge-tree is much worse than the other methods. The manually-constructed merge-tree is competitive with OMNIMERGE, and the fastest

[3] Two passes of minimization per voxel; 40° resolution for the backbone torsions and 120° for the sidechains; summed squared violations less than 0.0005 Å^2; hard sphere van der Waals radii at 85% of half sigma; van der Waals interactions ignored between atoms in the 1–4 positions of a covalent bond; peptide bonds restricted to 175°–180°. All interatomic distances in the crystal structure were measured and for every pair of atoms between 2.5 and 6.0 Å apart, we constrained those atoms to maintain a distance within ±0.05 Å of the measured distance.

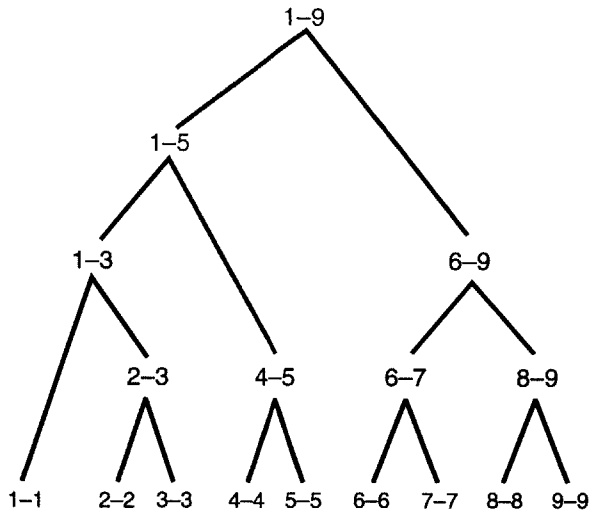

Figure 4-7: A manually designed merge tree for 1RST.

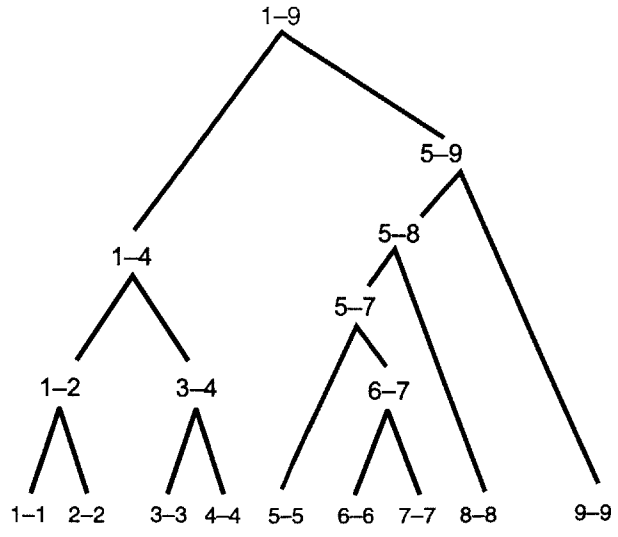

Figure 4-8: The optimal merge-tree for searching 1RST at 120° resolution.

Search Method	Runtime (seconds)	Number of Minimizations	Conformations Found
divide-and-conquer with default merge-tree	2,046	6,712	657
divide-and-conquer with manually-constructed tree	533	5,189	662
divide-and-conquer with merge-tree optimized at 120° resolution	302	3,162	679
above plus cost of OMNIMERGE at 120° resolution	320	3,654	679
OMNIMERGE	536	7,271	679
LIMITED OMNIMERGE	312	5,709	675

Figure 4-9: Run times for simulations on 1RST searching all possible subchains, compared with using particular merge-trees. (Recall that LIMITED OMNIMERGE searches all subchains with lengths $\leq 0.667N$ or up to 6 residues, and orphaned subchains are also removed.) The fastest time is colored green. The resolution of the search was 40° for backbone torsions and 120° for sidechain torsions.

algorithms are LIMITED OMNIMERGE or low-resolution OMNIMERGE followed by a divide-and-conquer search using the merge-tree that is optimal at low resolution.

The number of minimizations performed by divide-and-conquer with the manual merge-tree is lower than the number performed by LIMITED OMNIMERGE, but the divide-and-conquer run time is significantly slower. This suggests that divide-and-conquer, even when using the manually constructed merge-tree, is performing more of its minimizations on high-dimensional subchains.

After performing OMNIMERGE at the regular resolution, we saw that its optimal merge-tree is very similar to the optimum found for lower resolution searches (see Figure 4-10). Of course, the higher resolution optimum is not available in advance, but if it were, it could be searched in only 278 seconds (with 3222 minimizations).

All the methods found the voxels corresponding to the crystal structure (the structure which was used to generate the constraints and which by definition satisfies the constraints). However, many of the methods did miss some voxels. A "false negative" occurs when the voxel-evaluator disqualifies a voxel that does indeed contain a satisfying conformation. As the size of voxels increases (due to low resolution, high dimensionality, or both) the number

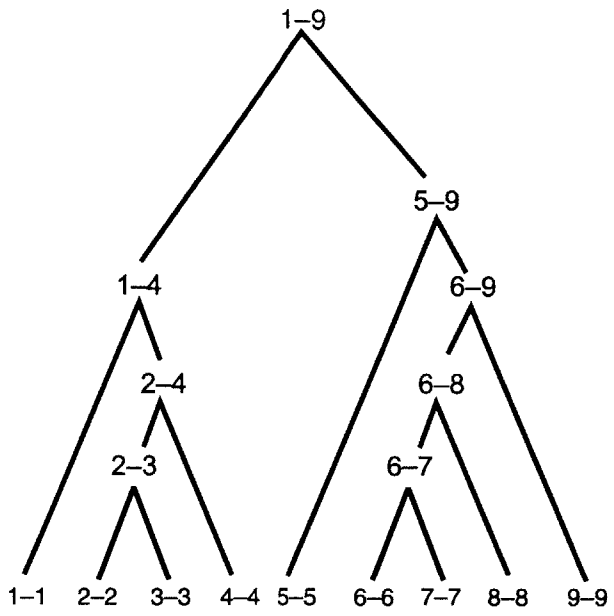

Figure 4-10: The optimal merge-tree for searching 1RST at 40° resolution.

of extrema per voxel increases, and the probability of false negatives increases. We have not yet identified any other trends to explain the varying completeness of the different searches here.

Figure 4-11 shows the manually-designed merge-tree for 1RST annotated in red with the TREECOST for each subtree. The leaf subchains are labelled in blue with the number of satisfying conformations. The blue symbols for the non-leaf subchains are the number of candidate conformations followed by how many of those conformations were found to satisfy the constraints. The diamonds, arranged as in the dynamic programming table of Figure 4-6, denote all the possible subchains of the molecule. Blank diamonds are for subchains that are not searched by this merge-tree. Appendix B has similar figures for other merge-trees.

A visual display of the performance of OMNIMERGE components is more difficult because left and right child subchains are chosen independently for each combine operation. Figure 4-12 shows which subchains were the children for the locally optimal merges that created the largest subchains of 1RST. Note a tendency of merges to reuse some of the particularly well constrained subchains (as indicated by clusters of merge endpoints), such as subchain 5–9.

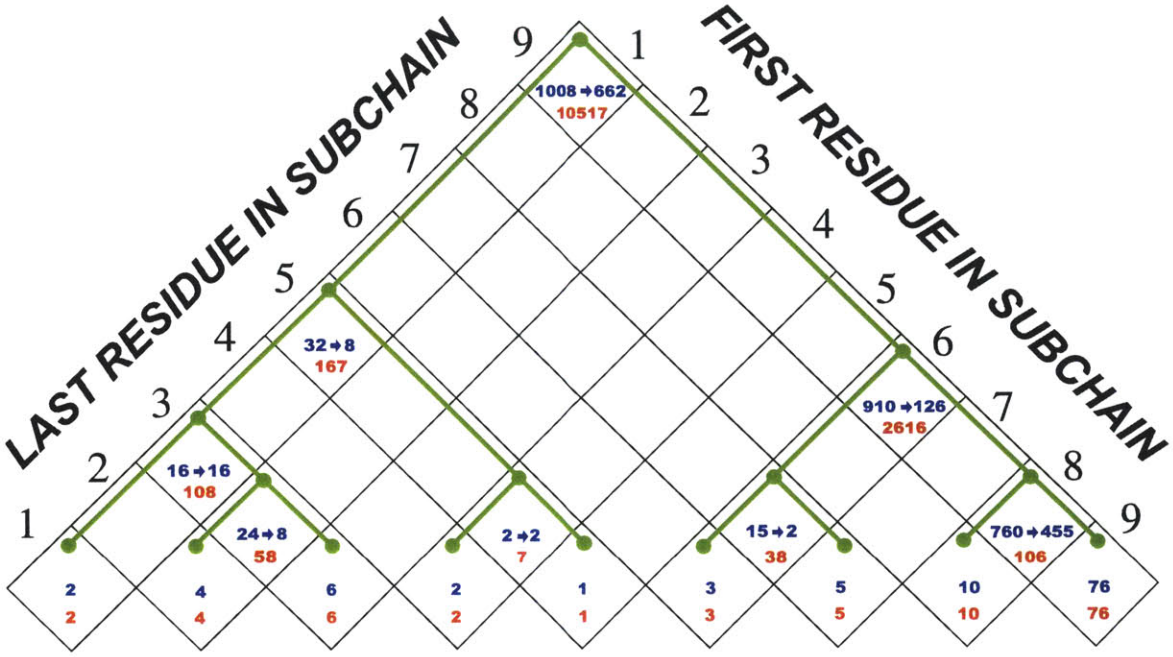

Figure 4-11: Satisfying conformations (blue) and subtree costs (red), for subchains in the manual merge-tree of 1RST. For non-leaf subchains, the blue figures include the number of combined candidates (before the arrow) and then (after the arrow) the number of candidates found to satisfy the constraints.

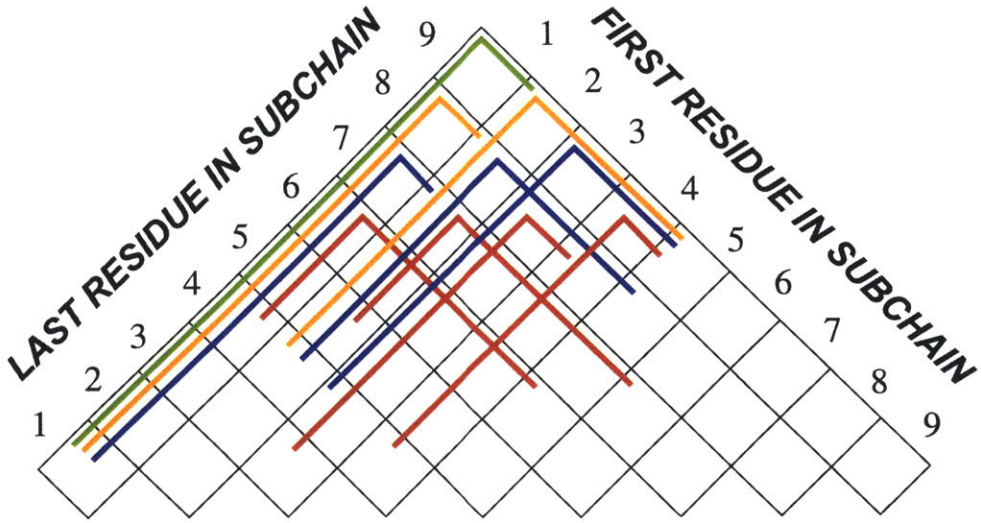

Figure 4-12: Locally optimal merges for creating subchains of size 6 are in red, for creating subchains of size 7 are in blue, for creating subchains of size 8 are in yellow, and the chosen merge for creating the whole molecule is shown in green. Merges for creating smaller subchains are omitted for clarity.

Figure 4-13 shows the number of candidate conformations OMNIMERGE evaluated for each subchain of 1RST, followed by the number of solutions. (For leaf subchains, the number of candidate subchains is not applicable and only the number of satisfying solutions is shown.)

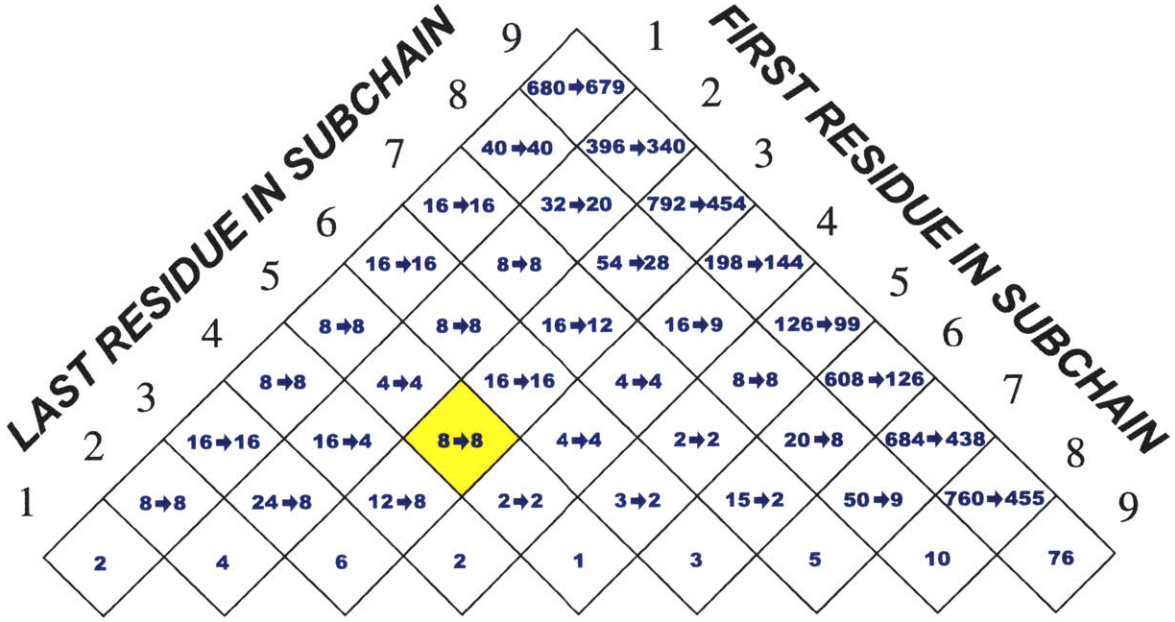

Figure 4-13: Each non-leaf entry contains the number of candidate conformations, an arrow, and the number of satisfying conformations.

All methods found the same number of satisfying conformations for all the subchains they searched in common, except the final subchain (the whole molecule) for which the disagreement in number of conformations found is reported in Figure 4-9.

The number of candidates evaluated in order to determine the number of satisfying conformations obviously does vary depending on the method. For example, in the manual merge-tree, subchain 6–9 can only be searched be combining 6–7 and 8–9, which gives $2 \times 455 = 910$ candidates. When OMNIMERGE reaches subchain 6–9, it chooses to combine 6–8 and 9–9 instead, which gives only $8 \times 76 = 608$ candidates.

Figure 4-14 shows the dynamic programming table with the BESTTREECOST of each subchain. Recall that the left and right child subchains combined by OMNIMERGE are chosen to minimize the number of candidates but the left and right children used for building up the dynamic programming table are chosen to optimize the cost of the whole subtree.

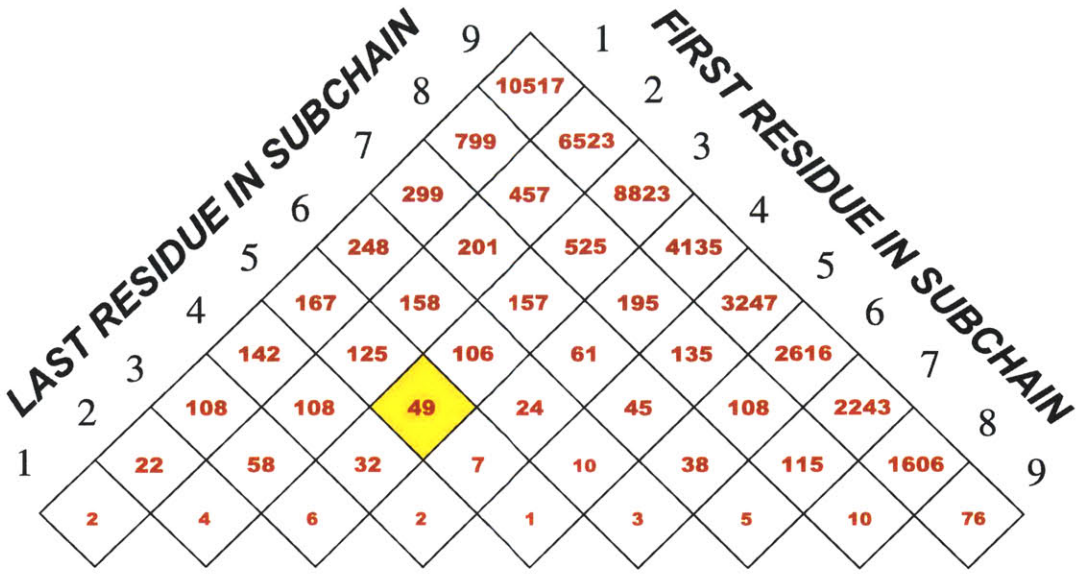

Figure 4-14: A dynamic programming table for 1RST, with each entry denoting the BESTTREECOST for the subtree rooted at that subchain. A 45° rotation of this table, as in Figure 4-5 may be more familiar.

For example, the shaded entries in Figures 4-13 and 4-14 correspond to subchain 3–5 of 1RST. The BESTTREECOST of subchain 3–5 comes from choosing to combine subchains 3–3 and 4–5, which yields a total cost 49.

$$\text{BESTTREECOST}(m,n) = \min_{m \le i < n} \begin{pmatrix} & \text{NUMSOLUTIONS}(m,i) \\ \times & \text{NUMSOLUTIONS}(i+1,n) \\ \times & \text{VOXELCOST}(n-m+1\ ) \\ + & \text{BESTTREECOST}(m,i) \\ + & \text{BESTTREECOST}(i+1,n) \end{pmatrix} = \begin{pmatrix} & 6 \\ \times & 2 \\ \times & 3 \\ + & 6 \\ + & 7 \end{pmatrix} = 49.$$

OMNIMERGE performs its search of subchain 3–5 by combining subchains 3–4 and 5–5. Subchain 3–4 might be costly to create, but since its search is already done, its results can be freely reused by OMNIMERGE.

BESTNODECOST(m,n)

$$= \underset{m \le i < n}{\text{MIN}} \begin{pmatrix} & \text{NUMSOLUTIONS}(m, i) \\ \times & \text{NUMSOLUTIONS}(i+1, n) \\ \times & \text{VOXELCOST(n-m+1).} \end{pmatrix} = \begin{pmatrix} & 8 \\ \times & 1 \\ \times & 3 \end{pmatrix} = 24.$$

4.4 PROPAGATION (Arc Consistency)

The penultimate and most promising upgrade to systematic conformational search uses the idea of arc consistency [74, 73] to propagate information about disqualified conformations from each searched subchain to any other subchains that overlap it.

Motivation

When two subchains contain overlapping residues, any conclusions deduced about the overlapping residues during the search of one subchain must also be true about the same residues in the other subchain. We want to avoid deducing the same conclusion repeatedly.

For example, suppose we searched subchain 1–5 by merging 1–3 and 4–5, and suppose we discovered after evaluating all the candidate combinations that residue 3 always clashed with residue 4 unless residue 3 was in conformation X. In such a case, it is obvious to a human that every conformation that includes these residues and satisfies the constraints (including every conformation for the whole molecule) must have residue 3 in conformation X. Suppose the next planned subchain to search involves merging subchains 2–3 and 4–6 to create subchain 2–6. The current algorithm would have to consider all possible combinations of 2–3 and 4–6 as candidates and would rediscover from scratch that residue 3 must be restricted to conformation X, which is a waste of effort.

That's one case of a general problem of how to share information. To generalize the challenge, any time a conformation (a combination of torsion ranges for a subchain) is found to violate the constraints, that combination should be pruned from consideration in all other subchains as well. Although we have not yet found a suitable algorithmic framework to distribute every kind of possibly relevant information, the notion of arc consistency between overlapping subchains addresses much of the need.

Arc Consistency

Arc consistency between variable A and variable B means that for each value of A, there exists a value for B such that the value of A and the value of B are consistent according to the constraint-arc between A and B. This terminology derives from variables being vertices in a graph and pairwise constraints on the variables being arcs between the vertices. Enforcing arc consistency (also called *constraint propagation*) can greatly accelerate solving a constraint satisfaction problem.

There are many ways to use arc consistency in the context of conformational search.[4] For our application of arc consistency, each subchain is a variable (which also makes it a vertex), each conformation is a value, and two vertices are connected by an arc if the two corresponding subchains have an overlap.

Consistency between two values according to a constraint-arc is defined as follows: given that a value is a conformation, which really just represents a voxel, and a voxel is a range of angles for each torsion, then a conformation for subchain $\mathcal{A}$ will be called *consistent* with a conformation for subchain $\mathcal{B}$ if and only if each bond that occurs in both subchains is assigned to the same range of angles (or to overlapping/compatible ranges) in the two conformations. The criterion of arc consistency can finally be translated to our problem as follows. For each conformation of subchain $\mathcal{A}$, there must exist a conformation of subchain $\mathcal{B}$ such that the two conformations assign compatible ranges of angles to each of the torsional bonds that they have in common. For example, if subchain $\mathcal{A}$ and subchain $\mathcal{B}$ both contain torsion θ_i, then a conformation of subchain $\mathcal{A}$ placing torsion θ_i in the range 30-60° would not be "consistent" with a conformation of subchain $\mathcal{B}$ placing torsion θ_i in the range 90–120°.

Pitfalls to Avoid

There is a common misunderstanding about arc consistency, and also a clash of vocabulary particular to this context. The misconception that tends to occur frequently with any application of arc consistency is that arc consistency forbids a value (conformation) for a variable (subchain) when the value is found to be inconsistent with a value for a linked

[4]Forward-checking is a limited form of arc consistency, applied during a backtracking search. The algorithm by Beusen *et al.* [5] called "treesearch with look-ahead" can also be considered backtracking with forward-checking.

(overlapping) variable. Actually, arc consistency only forbids a value when it's found to be inconsistent with *every* possible value for the linked variable.

The vocabulary problem comes from confusing the original constraints imposed on the problem by the user (such as constraints on interatomic distances) with the new requirements imposed on overlapping subchains in order to facilitate the sharing of information. Hence, we will always refer to the new requirements as constraint-arcs, and "consistent" will mean the constraint-arcs are not violated, whereas "feasible", "satisfying," or "legal" will refer to the status of the user's constraints.

In our application of arc consistency, the constraint-arcs are only enforcing compatible angles for the common bonds between overlapping subchains. A pair of perfectly good conformations that satisfy all of the user's constraints can be "inconsistent" if they don't agree on angles in the region of overlap. That could occur frequently, such as if the conformations are part of distinct alternative solutions to the overall problem. We forbid a value only when it is inconsistent with all values for an overlapping subchain. We do not forbid a value when it is inconsistent with just one value of an overlapping subchain.

PROPAGATION

We use a relatively standard algorithm for enforcing arc-consistency, which we will simply call PROPAGATION [50, 51, 70]. PROPAGATION will be run after each non-leaf subchain $\mathcal{S}_0$ (other than the whole molecule) is searched. The propagation queue holds subchains with recently-modified solution sets (in other words, subchains with new information that needs to be propagated). Therefore, the algorithm starts by initializing the queue to hold $\mathcal{S}_0$.

```
PROPAGATION
Queue ← S_0.
For each subchain, S, in the Queue {
   For each searched subchain, T, that overlaps S {
      For each conformation τ_i of T {
         For each conformation σ_j of S {
            If σ_j is consistent with τ_i at overlapping bonds,
               Then Next τ_i
         }
         Comment {There is no σ_j consistent with τ_i}
         Remove τ_i from the conformations of T.
         Queue ← Queue ∪ {T}.
      }
   }
}
```

For our implementation of PROPAGATION, we have chosen to perform PROPAGATION only once per subproblem (after the combine operation is done), rather than repeatedly during the combine operation (each time a candidate is discarded). A problem for future research might be to establish parameters to trigger PROPAGATION during a combine operation if an expected chain-reaction could reasonably reduce the number of candidates currently under consideration.

PROPAGATION has no effect when used with divide-and-conquer on a particular merge-tree because the only subchains that overlap each other have an ancestor-descendent relationship. Since information already progresses from child to parent in an orderly fashion when using a merge-tree (namely via the combine operation), there is no additional unshared information to propagate. For example, in a default merge-tree, subchain 1–4 overlaps subchain 3–4, and the solutions ruled out during the search of 1–4 might be sufficient, with PROPAGATION, to reduce the solution set for subchain 3–4, but since the only use for subchain 3–4 was for creating 1–4, nothing is gained. No information is propagated to, for example, subchain 4–5 because that subchain is never considered. Only when doing the "redundant" work of searching "extra" subchains is PROPAGATION truly applicable. Un-

less otherwise specified, we will assume PROPAGATION always occurs in conjunction with OMNIMERGE or LIMITED OMNIMERGE.

The overall effectiveness of OMNIMERGE with PROPAGATION compared with, for example, divide-and-conquer on a default merge-tree, will depend on whether the sharing of information can create enough of a "chain reaction" to offset the additional costs of searching the extra overlapping subchains, minus the regained cost for being able to use optimal merges, plus the cost of performing the PROPAGATION. In other words, we need to try it to know.

In the case of a relatively unconstrained molecule, the PROPAGATION algorithm will rarely create "chain reactions" because there are many solutions for each subchain compatible with every solution for every overlapping subchain. In a more tightly constrained system with sparse solution sets, the removal of one solution for a small subchain at one end of the molecule might dramatically reduce the solution set for a neighboring subchain; the neighboring subchain is added to the queue and its changed solution set is propagated, possibly reducing the solution set for its neighbor, and so on across the whole molecule.

Performance Results

We ran variants of OMNIMERGE, with and without PROPAGATION, on the 1RST peptide with the same constraint set and same parameters as in section 2.3, page 47. PROPAGATION allowed both OMNIMERGE and LIMITED OMNIMERGE to run significantly faster. See Figure 4-15. The reduced number of minimizations is sufficient to account for the faster speed. With PROPAGATION, OMNIMERGE ran more efficiently than the LIMITED version, even though LIMITED runs faster without PROPAGATION. This might be because the additional large, overlapping subchains searched by OMNIMERGE revealed inconsistencies, and the PROPAGATION from them allowed the search of the final subchain to run more quickly.

PROPAGATION runs quickly relative to the cost of minimization, especially with increasing subchain sizes, and so we strongly recommend using it whenever it is applicable. However, the disadvantage of using PROPAGATION during an OMNIMERGE search is that it does not provide the necessary information for determining the optimal merge-tree. Instead of computing the number of conformations of each subchain that satisfy the constraints active on that subchain, we are determining the conformations for each subchain that are both consistent with overlapping subchains and feasible with respect to the active constraints.

Search Method	With Propagation	Run Time (seconds)	Number of Minimizations	Conformations Found
divide-and-conquer with default merge-tree		2046	6712	657
divide-and-conquer with manually-constructed tree		533	5189	662
divide-and-conquer with merge-tree optimized at 120° resolution		320	3654	679
OMNIMERGE		536	7271	679
LIMITED OMNIMERGE		313	5709	675
OMNIMERGE	√	211	4390	664
LIMITED OMNIMERGE	√	245	3957	695

Figure 4-15: Run times for simulations on 1RST searching all possible subchains, with and without propagation to enforce arc-consistency. (The cost of the search using the merge-tree optimized at low resolution includes the cost of OMNIMERGE at low resolution. Compare with Figure 4-9.) The resolution of the search was 40° for backbone torsions and 120° for sidechain torsions.

Without knowing the number of solutions for each subchain, unaided by hints from overlapping neighbors, we cannot compute accurate TREECOSTs, much less BESTTREECOSTs.

Finally we note that in this case, completeness deteriorated for OMNIMERGE and improved for LIMITED OMNIMERGE. Another drawback to using PROPAGATION, not clearly evident in this example, is that PROPAGATION may easily decrease the **completeness** of a search. If the minimizer (or whatever method is used to evaluate voxels) cannot find any satisfying conformations in a voxel, that voxel is disqualified, even if there is a satisfying region of space in the voxel. Because OMNIMERGE considers all possible subchains, some combinations of torsion ranges will be considered much more often than under divide-and-conquer, sometimes in the context of high-dimensional voxels. The repetition increases the probability of a false negative occurring at some point. Without PROPAGATION, a single false negative in OMNIMERGE would have no effect on the overall results unless the subchain with the false negative participated in the merges that create the final molecule. In contrast, PROPAGATION can spread a false negative to many overlapping subchains and cause the final results to be corrupted indirectly. Even if the overlapping subchains already searched the same combination of torsions and correctly identified satisfying conformations in the voxels, having it disqualified by one subchain might be enough to have it disqualified

in others.

4.4.1 Previous Work

The PROPAGATION algorithm we use with OMNIMERGE has a conceptual similarity to the combination of "build-up" and "build-down" projections used by Gippert *et al.* in their DTAGS and binary search methods, but some background is required before the relationship between the concepts can be described.

The DTAGS (Distributed Torsion Angle Grid Search) algorithm [24, 23] searches linear chains of torsions (such as polypeptide backbones without sidechains) and constructs lookup tables of the results. For example, if a gridsearch has been performed at 10° resolution on a chain of 4 torsions $(\theta_1, \theta_2, \theta_3, \theta_4)$, then the answers (allowed/disallowed) can be stored as a four-dimensional $(36 \times 36 \times 36 \times 36)$ boolean matrix.

Whenever a conformation for a small subchain is found to violate the constraints, DTAGS *projects* that information onto a host of higher-dimensional tables. The overall idea is like the Rete algorithm [21]. There is a lookup table for every possible subchain of consecutive torsions (like our sets of all possible subchains except at the level of torsions instead of residues). If a conformation for torsions θ_1–θ_4 is infeasible, then every subchain containing torsions θ_1–θ_4 has its lookup table updated such that the sub-matrix of entries corresponding to the disallowed angles for θ_1–θ_4 are marked automatically as disallowed. For example, in the table for $(\theta_1, \theta_2, \theta_3, \theta_4)$, if the boolean entry for the conformation $(\theta_1 = 110°, \theta_2 = 120°, \theta_3 = 130°, \theta_4 = 140°)$ says disallowed, then in the table for $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$, the "column" of entries corresponding to $(\theta_1 = 110°, \theta_2 = 120°, \theta_3 = 130°, \theta_4 = 140°$, and any value of $\theta_5)$ would also get marked as disallowed.

Compared with a naive gridsearch algorithm, DTAGS is space-intensive and performs an enormous number of projection operations on bits, but it performs remarkably few floating-point operations and it never computes the interatomic distance between the same pair of atoms more than once. This is a stunningly creative trade-off. An empirical comparison of its performance or else an analysis of its asymptotic efficiency would be helpful.

In the 1995 version of the DTAGS algorithm (Gippert's doctoral thesis [23]), the term "propagation" is used to describe the repeated projection of disallowed conformations into higher-dimensional tables. That use of the term propagation is not related to our PROPAGATION algorithm nor to the "constraint propagation" algorithms within the "constraint

satisfaction" area of artificial intelligence. (The criterion used by the 1995 version of DTAGS for ruling out conformations in higher-dimensional tables is strictly weaker than the arc consistency criterion used for constraint propagation between overlapping subchains.)

In the 1998 version of the DTAGS algorithm [24], another procedure called "build-down" was added, and this brings us finally to the similarity with our PROPAGATION algorithm. Whereas the DTAGS projection in the 1995 version (called "build-up" but not with reference to the buildup procedure of Scheraga *et al.* [57, 71, 22]) uses single disallowed conformations in a low-dimensional table to rule out sub-matrices in high-dimensional tables, the new "build-down" procedure finds whole sub-matrices in high-dimensional tables that are entirely disallowed and uses that information to rule out the single corresponding conformations in lower-dimensional tables. For example, with the "build-down" procedure, if some large chain has no allowed conformations with $\theta_i = 30°$, then all smaller subchains of it can have $\theta_i = 30°$ excluded as well.

Recall the arc consistency criteria that we enforce with the PROPAGATION algorithm:

> If subchains $\mathcal{A}$ and $\mathcal{B}$ overlap,
> then for each conformation of subchain $\mathcal{A}$,
> there must exist a conformation of subchain $\mathcal{B}$ such that
> the two conformations assign compatible ranges of angles to each of the torsions
> that they have in common.

The filtering of conformations accomplished by the the DTAGS "build-down" procedure can be restated in similar terms as follows: (Differences are in boldface.)

> If $\mathcal{A}$ **is a subchain of** $\mathcal{B}$,
> then for each conformation of **subchain** $\mathcal{A}$,
> there must exist a conformation of **subchain** $\mathcal{B}$ such that
> the two conformations assign **the same angle** to each of the torsions that they
> have in common.

The two methods define legal subchains very differently, but let us assume temporarily that the concept of a subchain is not in question. If we set aside the voxel model for the moment, then the arc consistency criterion could use angles instead of ranges of angles. The only remaining difference is whether the two subchains must have a superchain-subchain relationship or an arbitrary region of overlap.

The "build-down" criterion alone is less general than the arc consistency criterion because the superchain-subchain relationship is less general than an arbitrary region of overlap. (The "build-up" criterion alone is definitely less general than the arc consistency criterion.) The combined application of both "build-up" and "build-down" might be at least as general as arc consistency because it would conceivably allow information to travel arbitrarily, including across overlaps. For example, information about disallowed values for residues 3–5 could travel from subchain 1–5 to subchain 3–8 by way of 1–8.

We conjecture (making allowances for the differences already mentioned) that any set of conformations satisfying both the "build-up" and "build-down" criteria would also satisfy our arc consistency criterion and vice versa. In other words, if the other components of two algorithms (DTAGS and OMNIMERGE with PROPAGATION) were the same, then their respective criteria for removing disallowed conformations seem like they would be equivalent, even though the motivations and implementations are very different.

Gippert *et al.* also apply the combination of "build-up" and "build-down" ideas to their divide-and-conquer method, a Cartesian-space binary tree search. The "build-down" algorithm for use with their Cartesian-space binary tree search is described using an analogy to the DTAGS method rather than in full detail. Similarly, and for lack of space, we suggest that the conceptual similarities between the Cartesian-space binary tree search and PROPAGATION are analogous to the similarities between DTAGS and PROPAGATION.

4.5 Augmenting OMNIMERGE with A*

Our method so far, OMNIMERGE with PROPAGATION, searches all possible subchains by combining small subchains into larger ones, and it propagates consistent values across overlapping subchains. The final innovation we present will be to choose the order in which subchains are searched (possibly also skipping some subchains) rather than simply searching all subchains according to the default ordering from OMNIMERGE.

In this introduction to the section, we provide some general intuitions for the ideas to be developed later. The next topic, buildable sets and the buildability graph, will provide a formal description of the space of all possible merge-strategies. Aside from the definition of a "buildable" subchain, this topic may be considered optional. The topic after that is the innovation itself, "the A* improvement" for selecting which subchain to search next. The

final topic looks at performance of OMNIMERGE with the A* improvement.

The A* improvement is a subtle idea with several layers. The basic idea was motivated by instincts acquired after observing many runs of OMNIMERGE. With some molecules, there is an early bottleneck—a "bad" subchain that is much more costly to search than other subchains of comparable length. While other subchains might have some right or left child subchains with many solutions, there is usually some combination of left and right children with reasonably low product. For the "bad" subchain, there is no good combination of right and left children.

For example, in the extreme example at the beginning of this chapter, we constructed constraints that leave the left half and right half of the molecule unconstrained even though the whole molecule is well-constrained overall. If the extreme constraints are applied to a molecule with eight residues, then subchain 5–8 would qualify as "bad." We will revisit this case after describing the A* improvement.

Human instinct is to skip the search of the "bad" subchain. Certainly time would be saved in the short run, and limited harm would seem to come of it in the long run, because many alternative subchains are available to serve as right and left children for all subsequent merges. Just as LIMITED OMNIMERGE chooses which subchains to skip based on their lengths, we would like to choose which subchains to skip based on information obtained during the searches of smaller subchains, such as information about the number of solutions for various right and left children.

One can consider exhaustively which single subchain might be best to "skip," or one can evaluate all possible merge-trees fully, but the general space of all possible merge-strategies is astronomical. (For example, the space of merge-strategies includes skipping combinations of subchains, or searching some of the subchains in one merge-tree and then searching the subchains of some other merge-tree.) Instead of restricting ourselves to using very simple merge-strategies, (such as merge-trees, or OMNIMERGE, or skipping a single choice of subchain), we would like to navigate the full space of possible merge-strategies and make choices on the fly using the information currently available. When information shows particular subchains to be "bad" or "good," we would like to omit the bad subchains, search the good subchains immediately, and otherwise take advantage of the freedom. If no useful information is available, we can follow a simple default merge-strategy, such as searching all subchains, and no harm would come from having the extra freedom.

As we wander the space of merge-strategies, we obtain some degree of new information each time another subchain is searched. If subchains that looked "good" turn out to be worse than expected, a subchain previously classified as "bad" might start to seem good by comparison. Rather than omitting the "bad" subchain permanently, we will order the subchains from good to bad and search the best subchain first. If a subchain is ranked "bad" enough for long enough, the search of the whole molecule chain will occur before the search of the "bad" subchain. Since the search of the whole molecule marks the end of the program, any subchain postponed until after the search of the whole molecule is skipped. As in life, postponing something long enough means it never gets done. Thus, we talk about how to order or prioritize subchains instead of simply talking about which subchain(s) to skip.

The two remaining ideas are about how to compute a quantitative function that evaluates the priority of a subchain. They pertain to the A^* cost function itself, but still only at an intuitive level.

We would like to prioritize a subchain not just by the immediate cost of searching it but also by the expected usefulness of the subchain towards creating other merges. Furthermore, since the true goal is to search the whole molecule, we consider defining the usefulness of a subchain in terms of its relative contribution to a merge-tree for the whole subchain. (Suppose we actually knew the costs of alternative merge-trees and suppose we were going to search the molecule using a merge-tree.) If two merge-trees for the whole molecule are chosen by optimizing over all the same freedoms, except that merge-tree α must use subchain A and merge-tree β must use subchain B, then the difference in cost between α and β can be thought of as the difference in value between subchains A and B. The immediate costs of searching A or B might dominate the comparison of their merge-trees (which would be like choosing which subchain to search next based just on the immediate costs). The more interesting case is if A is more expensive to search than B but α is less expensive than β. That might occur if A can be merged with a well-constrained subchain to create a search of the whole molecule and B would have to be merged with an expensive, unconstrained subchain before it could create a search of the whole molecule. Under such circumstances, where A is the "missing link" for reaching the final answer, we would like to choose A even though the immediate search of B would be cheaper.

The final idea comes from the supposition that we might know the costs of alternative

merge-trees and that we might search the molecule using a merge-tree. The number of conformations for each subchain is unknown, but suppose we make estimates for these numbers and use the estimates to predict an optimal merge-tree. For subchains in the distant future, our estimates will have virtually no predictive power. For subchains that might be searched in the near future (subchains for which at least one left and right child have been searched), much better estimates may be possible.

We will indeed attempt to perform the search using the optimal merge-tree. At the beginning of the search, when virtually no information is available, we will make a wildly optimistic set of estimates and pursue their logical consequences for one step, which is for the search of one subchain. When the results turn out to be disappointing, we will revise the estimates and follow their logical consequences for one step. Whenever a set of estimates is inaccurate, the next piece of incoming information (the number of solutions found for the subchain previously believed to be "best") will disprove the initial estimates. The new information causes revisions not just in the estimates for the number of solutions per subchain but also in the optimal merge-tree. We can always try to pursue the optimal merge-tree, but the optimal merge-tree may shift after each subchain.

In the absence of any estimates with predictive power, the constantly "disappointing" information and the constant shift from one optimal merge-tree to another will end up causing all subchains to be searched in order from small to large. In other words, if we have no predictive information, we will search the same subchains as OMNIMERGE! OMNIMERGE is an excellent strategy if no information is available. Meanwhile, if we can somehow estimate the optimal merge-tree correctly, then we may actually have a chance of searching according to the optimal merge-tree.

Buildable Sets

Up until now we have talked about the different merge strategies as if the question is which merge-tree to use. From that viewpoint, OMNIMERGE (without PROPAGATION) encompasses every possible merge strategy because the results of every possible merge-tree can be reconstructed easily from the results of OMNIMERGE.

However, there are yet other merge strategies that OMNIMERGE neglects. Using the same language, a more general question, instead of which merge-tree to use, is what parts of which merge-trees to explore, and in what order. This allows for the possibility of,

say, alternating between the subchains of two different merge-trees until one seems clearly better than the other. OMNIMERGE considers all possible merge-trees but it treats them uniformly; by searching all subchains in a pre-determined order, OMNIMERGE precludes the possibility of skipping ahead to a "later" subchain, such as a large subchain for which particularly promising, well-constrained left and right children have just been identified. We will now develop a more general language and framework for considering possible merge strategies.

Let N be the number of leaf subchains (residues) in the molecule being searched.

We define a *buildable set of subchains* to be a set of subchains such that every element is either a leaf subchain or else it is *buildable* from the other subchains in the set. Subchain $\mathcal{S}_i$ is buildable from subchains $\mathcal{L}_i$ and $\mathcal{R}_i$ if $\mathcal{L}_i$ and $\mathcal{R}_i$ can function as left and right children for a combine operation to create $\mathcal{S}_i$. A *complete* (or incomplete) buildable set is a buildable set that includes (or does not include) the whole molecule as one of its subchains.

Recall from the definition of a legal merge-tree (in section 2.3) that the subchains in a legal merge-tree are always a complete buildable set.

Buildability Graph

A buildability graph has a vertex corresponding to each buildable set of subchains. Let V_i denote either a vertex or its buildable set. A directed edge, $E_{\mathcal{S}}$, connects vertex V_1 to vertex V_2 if and only if there is a subchain $\mathcal{S}$ such that $\mathcal{S}$ is not in V_1, but adding $\mathcal{S}$ to V_1 yields V_2. ($\{\mathcal{S}\} \cup V_i = V_2$.) Since V_2 is buildable, then either $\mathcal{S}$ is a leaf or $\mathcal{S}$ is buildable from the subchains in V_1.

Suppose the molecule to be searched has only two residues (or two pieces of whatever size we choose to search from scratch). The set of possible subchains has three elements {1–1, 2–2, and 1–2}. The power set of the set of subchains would have $2^3 = 8$ elements, although not all of those eight would be buildable sets. Figure 4-16 shows, for a molecule with two leaves, the eight elements of its subchain power set, five of which are buildable. Each vertex (circle) contains an arrangement of open and red-filled diamonds—reminiscent of the diagonal table (Figure 4-6) with all possible subchains—to depict which subchains are absent or present in the set. The lower left diamond in each circle represents the first leaf subchain, the lower right represents the last leaf subchain, and the top diamond represents the whole molecule.

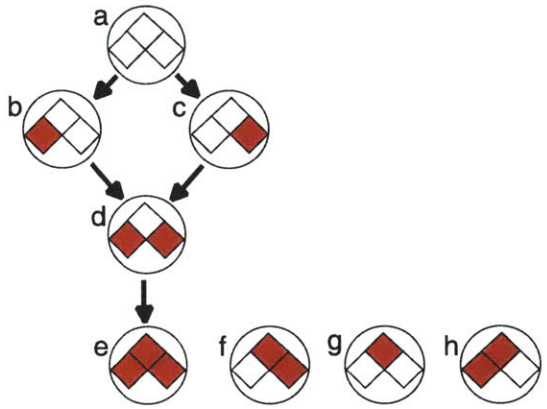

Figure 4-16: The buildability graph for a molecule with two leaves. The disconnected vertices, f, g, and h, correspond to elements of the power set that are not buildable.

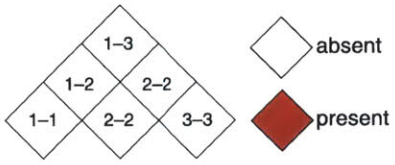

Figure 4-17: Legend for Figure 4-18.

Figure 4-18 shows a buildability graph for a molecule with three residues and six possible subchains (1–1, 2–2, 3–3, 1–2, 2–3, and 1–3). Again the diamonds in each vertex correspond to the possible subchains in a set, with the leaves in the bottom row, and the whole molecule at the apex. Figure 4-17 is a legend.

Merge Strategies as Paths

Each vertex in the buildability graph corresponds to a knowledge state. The subchains in the buildable set are the subchains that have known solutions (that have been searched). Traversing a directed edge corresponds to searching a subchain. The size of the knowledge state (the number of subchains searched) grows by one for each step taken. There are no dead-ends (no reachable vertices with zero out-degree) except when all subchains have been searched, and the longest possible path is from the vertex for the empty buildable set to the vertex for the set of every possible subchain. (OMNIMERGE explores a longest possible path.)

Finally let us define a *complete build path* as a path in the buildability graph from the empty buildable set to a complete buildable set. Each merge strategy we have considered so far in this chapter (ignoring PROPAGATION) corresponds to a merge strategy path.[5]

Contrary to first instinct, no merge strategy path could ever include both vertex b and vertex c from Figure 4-16 or 4-18 because knowledge state c indicates ignorance about the first leaf subchain, not indifference. Searching the first leaf and then the second would correspond not to a-b-c but to a-b-d in Figure 4-16 or a-b-f in Figure 4-18. Also perhaps contrary to instinct, complete build paths can terminate as soon as any complete buildable set is reached, such as nodes n or o, not just at the most complete set, node p. In a buildability graph, there is a "finish line" beyond which any node is an acceptable goal.

The path through Figure 4-18 corresponding to the strategy of divide-and-conquer with a default merge-tree is a-b-f-h-k-n. (Recall that we chose in section 2.3 to traverse merge-

[5]The set of all possible subchains except those larger than 0.667*N (but smaller than the whole subchain) is a complete buildable set because the removal of large subchains cannot prevent small subchains from being buildable, and the only subchain larger than 0.667*N, the whole molecule, is already buildable because of the size $\frac{N}{2}$ subchains. If integer d divides N evenly, then the set of all subchains that are either smaller than d or divisible by d is a complete buildable set because subchains of length kd for $k > 1$ can be built by combining subchains of length d and $(k-1)d$. For subchains of length d or smaller, all possible subchains are available for combination. Removing orphans does not alter the buildability of a set, although if other factors have been chosen poorly (such as if d does not divide N evenly) then it can alter the completeness of a set.

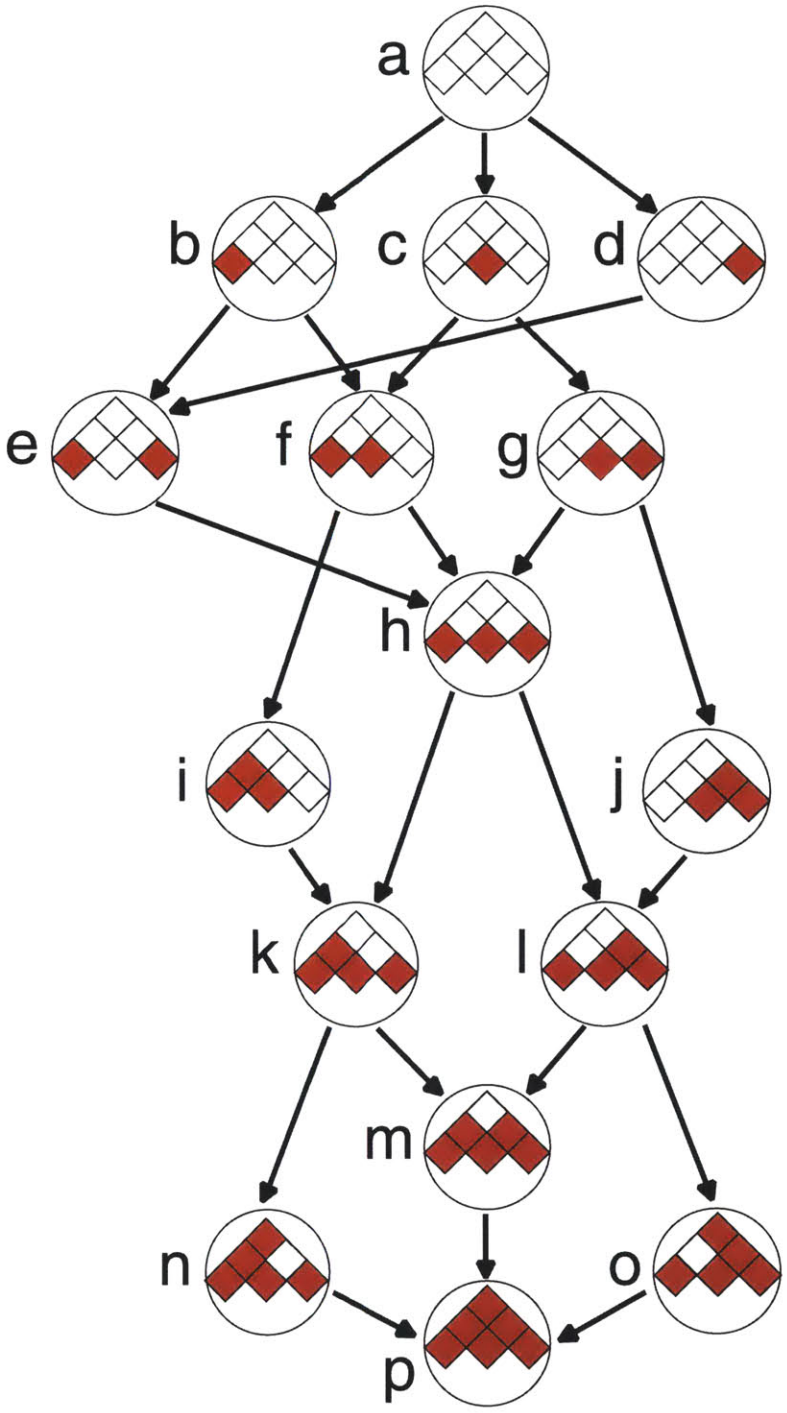

Figure 4-18: The buildability graph for a molecule with three residues. Vertices corresponding to non-buildable sets are omitted.

trees from left to right before going from bottom to top. Traversing the same merge-tree in a different order would include paths such as a-b-f-i-k-n or a-d-g-h-k-n.) A linear-merge tree is identical to the default merge-tree when there are only three leaves, so that path would be the same.

The path corresponding to OMNIMERGE is a-b-f-h-k-m-o. Instead of going straight from k to the finish line, OMNIMERGE goes from k to another incomplete state, m. As we know, OMNIMERGE searches more subchains than divide-and-conquer does before yielding the final answer, even for a molecule with only three residues. However, the final edge in a path, the search of the whole molecule, tends to be the most time-consuming step in the whole path, by far. Using OMNIMERGE and going from k to m causes an additional dipeptide to be searched. The benefits of this are, first, if the solution sets for the dipeptides (1–2 and 2–3) are not already consistent at their overlap, then PROPAGATION can reduce the number of remaining solutions; second, more than one possible combination of smaller subchains is available for the final step of searching 1-3, and the additional choice of children might allow fewer candidates to be considered.

Each diagram we have presented for visualizing search strategies emphasizes different facets of the algorithms. The buildability graph does not show which children are chosen at each step nor any costs. It just guarantees that, whenever there is an edge present, there exists some method for searching the subchain it adds. The buildability graph highlights decisions about which subchains are searched and in what order. Although we gain nothing by explicitly differentiating trivial decisions, such as the order in which the leaves are searched, this model highlights the variety of viable paths that do not correspond to any particular merge-tree or nameable strategy. Divide-and-conquer using a merge-tree never searches any "extra" overlapping subchains (it skips ahead to larger subchains at every opportunity), and OMNIMERGE always searches every overlapping subchain, never skipping anything. There is a universe of other strategies in between these two extremes.

Growth in the number of options

The total number of possible subchains for a molecule with N leaves is $N + (N-1) + (N-2) + 1 \ldots = \frac{N(N+1)}{2}$.

Every legal merge-tree with N leaves has $N-1$ internal nodes and $2N-1$ nodes total. Because the $N-1$ internal nodes of a legal merge-tree have no restrictions on arrangement

other than the binary branching factor, there is a one-to-one relationship between ordinary binary trees with $N-1$ nodes and legal merge-trees with $N-1$ internal nodes, N leaves, and $2N-1$ total nodes. Therefore, the number of different legal merge-trees with N leaves is equal to the number of binary trees with $N-1$ nodes, which is $\frac{1}{N}\begin{pmatrix} 2(N-1) \\ N-1 \end{pmatrix}$ [37].

The number of possible sets of subchains grows exponentially with the number of subchains, which itself is already quadratic in the number of residues. OMNIMERGE and its variants may exhaust the paths through the small buildability graph in Figure 4-18, but for sufficient N, the number of possible paths through the buildability graph will be far greater than the number of merge-trees. There are many ways of jumping around in the table of all subchains that do not correspond to merge-trees or other named strategies. We wish to design an algorithm that uses this insight, this flexibility, to perform a more efficient search.

4.5.1 The A* Improvement

We now describe an augmentation of OMNIMERGE that uses the A* algorithm for choosing which subchain to search next. A* is often described in conjunction with a search graph but the buildability graph is a knowledge state graph. (In a search graph, you can explore many alternative paths or partial paths by backing up and choosing another branch. In a knowledge state graph, backing up would correspond to deleting information.)

The main idea of A* is to decide among alternatives by scoring each option according to a cost function that reflects the total cost of reaching the goal by way of that option. The A* cost function typically involves a heuristic to estimate the cost from the decision point to the goal. Under A*, any uncertainty about the cost to the goal must be resolved optimistically so that the estimated cost is a guaranteed lower bound on the actual future cost if the option is chosen. Choosing which subchain to search next according to a cost function contrasts with plain OMNIMERGE, which chooses its next subchain according to a blind left-to-right, bottom-to-top ordering.

The A* cost function

The general case for defining the A* cost function assumes that some number of subchains have already been searched and multiple subchains (including the whole molecule) have not been searched. How can we compute the cost of searching the whole molecule by using a

particular subchain?

We already have a cost function, the BESTTREECOST, which gives the cost of reaching the goal (of searching the whole molecule) by way of the subchains in the optimal merge-tree. The BESTTREECOST for a molecule can be computed if one knows the number of satisfying conformations (number of solutions) for all its possible subchains. Furthermore, a guaranteed lower bound on the BESTTREECOST could be computed from guaranteed lower bounds on the number of solutions. For subchains that have not yet been searched, make the most optimistic (cheapest) guess and assume they will each have one solution. Accordingly, we can compute ESTIMATED-BESTTREECOSTS, ESTIMATED-NODECOSTS, or estimated versions of any of our cost functions using the estimated numbers of solutions.

We could define $f^*(\mathcal{S})$, the A* cost of subchain $\mathcal{S}$, to be the ESTIMATED-TREECOST for searching the whole molecule using the best merge-tree that includes subchain $\mathcal{S}$. One disadvantage of this definition is the possible inclusion of unsearched subchains in the subtree for $\mathcal{S}$. Since the purpose of this computation is to evaluate whether to search $\mathcal{S}$ next, and any immediate search of $\mathcal{S}$ would be restricted to using left and right child subchains that have already been searched, we will instead define the cost $f^*(\mathcal{S})$ to be the ESTIMATED-TREECOST for searching the whole molecule using the best merge-tree that (1) includes subchain $\mathcal{S}$ and that (2) only uses searched subchains for constructing the cost of the subtree of $\mathcal{S}$.

The A* cost function, $f^*(\mathcal{S})$, is traditionally broken into two parts, $g^*(\mathcal{S})$, the actual cost to reach $\mathcal{S}$ (without estimate), and $h^*(\mathcal{S})$, the estimated cost from $\mathcal{S}$ to the goal. In that case, if $\mathcal{S}$ corresponds to the black entry in Figure 4-19, then $g^*(\mathcal{S})$ would be computed using the costs of the subchains in the red region and $h^*(\mathcal{S})$ would be computed from costs of the subchains in the yellow region.

There are two trivial cases that we have, for simplicity, defined separately. For any leaf subchain under consideration, we define its A* cost to be small because all strategies depend on searching all leaf subchains. Therefore leaves can safely and efficiently be searched first. Conversely, unreachable non-leaf subchains should be scored so they never come next. In other words, if there aren't any searched subchains that can serve as right and left children for a combine operation to create subchain S, then the A* cost for S should be prohibitively high.

Before each non-trivial choice of which subchain to search next, we compute f^* for all

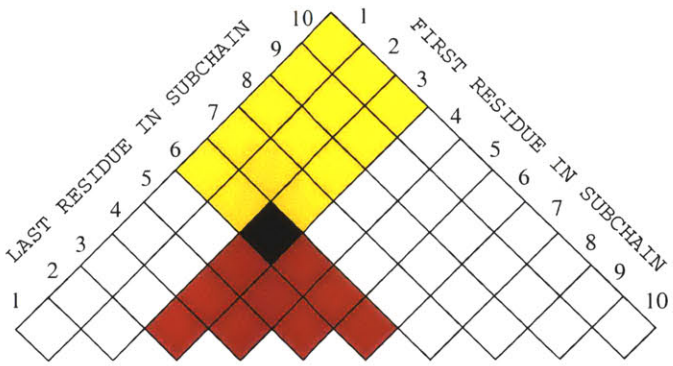

Figure 4-19: Regions influencing (red) and influenced by (yellow) the black entry when computing BESTTREECOSTS.

unsearched subchains as follows. First we compute as much as possible without making any assumptions or using any estimates. The actual number of solutions for searched subchains is used to compute TREECOSTS and g^* for as many subchains as possible. Then we compute as much as possible using optimistic estimates but without focusing on any particular choice of subchain. This gives an ESTIMATED-BESTTREECOST table including all subchains. Finally we make one table for each possible assumption that requires a particular subchain to be used. That is, for each unsearched subchain S, we make a table $\aleph_{\mathcal{S}}$.

Just as the computation of the cost of the globally best merge-tree is computed by filling in a dynamic programming table with BESTTREECOST(m, n) for each subchain up to the whole molecule, the computation of the cost of the best merge-tree restricted to $\mathcal{S}$ will be computed by filling in a dynamic programming table with an analogous function, $\aleph_{\mathcal{S}}(m, n)$, applied to each subchain up to the whole molecule. One difference is that $\aleph_{\mathcal{S}}$ is only computed for subchains that include all the the residues of $\mathcal{S}$ (the subchains shaded yellow in Figure 4-19).

$\aleph_{\mathcal{S}}(m\text{–}n)$ contains the ESTIMATED-TREECOST for searching subchain $(m\text{–}n)$ using the best merge-tree that (1) includes $\mathcal{S}$ and that (2) uses actual $g^*(\mathcal{S})$ costs to choose the best subtree rooted at $\mathcal{S}$. $\aleph_{\mathcal{S}}$ (subchain $\mathcal{S}$) is obviously just $g^*(\mathcal{S})$. We use this entry for $\aleph_{\mathcal{S}}$ to build up entries for larger subchains, as with dynamic programming to compute BESTTREECOSTS except always making sure $\mathcal{S}$ is included in any choice of merge-tree. Eventually $\aleph_{\mathcal{S}}$ (subchain (1–N)) will give us $f^*(\mathcal{S})$.

Determining each new entry $\aleph_{\mathcal{S}}(m,n)$ involves minimizing over a choice of which left and right child subchains to combine. However, with the $\aleph_{\mathcal{S}}$ computations, the choice of i (the middle residue) is also restricted so that one of the child subchains must contain all the residues of $\mathcal{S}$. The cost of that child's subtree is taken from previous entries in the $\aleph_{\mathcal{S}}$ table and the cost of the other child is taken from the ESTIMATED-BESTTREECOST table. (In the formula below, the left child is assumed to be the one containing $\mathcal{S}$.) Otherwise, if the residues of $\mathcal{S}$ were divided between the left and right children of subchain m–n, then $\mathcal{S}$ would not be a part of the implied merge-tree.

$\aleph_{\mathcal{S}}(\text{subchain } \mathcal{S}) = g^*(\mathcal{S})$

$\aleph_{\mathcal{S}}(\text{subchain } m\text{–}n) =$

$$\underset{\substack{m \le i < n \\ \mathcal{S} \subset \text{subchain } (m,i)}}{\text{MIN}} \left(\begin{array}{ll} & \text{ESTIMATED-NUMSOLUTIONS}(\text{subchain } m - -i) \\ \times & \text{ESTIMATED-NUMSOLUTIONS}(\text{subchain } i+1 - -n) \\ \times & \text{VOXELCOST}(n-m+1) \\ + & \aleph_{\mathcal{S}}(\text{subchain } m\text{–}i) \\ + & \text{ESTIMATED-BESTTREECOST}(\text{subchain } i+1\text{–}n) \end{array} \right)$$

The final entry in the table, $\aleph_{\mathcal{S}}(\text{subchain } (1,N))$ is equal to $f^*(\mathcal{S})$, the A* cost for subchain $\mathcal{S}$. After filling in tables $\aleph_{\mathcal{T}}$, $\aleph_{\mathcal{U}}$, $\aleph_{\mathcal{V}}$ for all unsearched (reachable) subchains $\mathcal{T}$,$\mathcal{U}$,$\mathcal{V}$, the subchain with best A* cost is chosen to be searched next. Although this cost function is elaborate, the cost of evaluating subchains using quadratic-sized dynamic programming tables is generally insignificant compared with the cost of actually searching those subchains.

Variations

A faster way to choose a subchain to search next would be to skip the $\aleph$ tables and just compute an ESTIMATED-BESTTREECOST table for the whole molecule. This would give both a lower bound on the search of the whole molecule and an "optimal merge-tree" corresponding to that best-case search. Then we would choose any reachable subchain in the optimal merge-tree to search next.

The A* approach to ordering subchains is not necessarily optimal, in the sense of providing a reasonable merge-strategy with best trade-off of overhead versus strategy improvement. Some searches might run faster using a greedier strategy, such as a cost function that estimates only the remaining cost to the goal instead of the total cost. More aggressive cost functions that don't provide a guaranteed lower bound might also be useful.

Using OMNIMERGE with A* (but without PROPAGATION) will evaluate all subchains necessary for computing the optimal merge-tree. Using PROPAGATION prevents the resulting table from being useful for that purpose (as described on page 107), and algorithms that don't use A*'s guaranteed lower bounds or its evaluation of total costs will also lose the guarantee of proving an optimal merge-tree as a side-effect of the search.

Results

We searched 1RST using the same constraints and parameters as in the previous test cases, but this time we included the use of the A* algorithm to choose the order in which subchains were searched. Figure 4-20 shows the results of the simulations.

Search Method	Propa-gation	With A*	Runtime (seconds)	Number of Minimizations	Conformations Found
divide-and-conquer with default merge-tree			2047	6712	657
divide-and-conquer with manually-constructed tree			533	5189	662
divide-and-conquer with optimized merge-tree			320	3654	679
LIMITED OMNIMERGE			313	5709	675
LIMITED OMNIMERGE		√	311	5709	675
LIMITED OMNIMERGE	√		245	3957	695
LIMITED OMNIMERGE	√	√	244	3604	695
OMNIMERGE			536	7271	679
OMNIMERGE		√	555	7271	679
OMNIMERGE	√		241	4390	664
OMNIMERGE	√	√	220	3968	662

Figure 4-20: Run times for simulations on 1RST searching all possible subchains, with and without PROPAGATION, with and without the A* algorithm. The resolution of the search was 40° for backbone torsions and 120° for sidechain torsions.

The search of 1RST runs faster with A* than without (except in the case of OMNIMERGE without PROPAGATION) but the improvement almost insignificant, especially when compared with the improvement in efficiency that PROPAGATION accomplished.

There are significant differences in costs between alternative merge strategies for 1RST (in part because the molecule and its constraints are very asymmetric). The A* algorithm can distinguish and exploit these differences to create a more efficient search. The addition

of A* does not seem to have much effect on the number of conformations found.

While OMNIMERGE without A* is forced to search every large subchain, and LIMITED OMNIMERGE is forced to skip every large subchain, OMNIMERGE with A* can choose to skip a large subchain (if, say, the number of candidates for it would clearly make it more expensive to search than just skipping ahead and searching the whole molecule), but it still searches large subchains that have a possibility of participating in an optimal merge-tree. Of course the vast majority of subchains examined by OMNIMERGE with A* do not turn out to participate in the optimal merge-tree, but some of them do compensate by reducing the solution sets of other subchains via PROPAGATION. In sum, we recommend the use of the A* augmentation because it can avoid searching the most poorly-constrained subchains in a molecule (a benefit with unlimited potential), and at worst it requires extra arithmetic (a delay with limited potential).

The extreme example

At the beginning of this chapter, we constructed artificial constraints on a molecule (for example, on an eight-residue helix of polyalanine) such that the right and left halves (subchains 1–4 and 5–8) are unconstrained but constraints from the left half to the right half do constrain the overall molecule tightly.

We searched an eight-residue helix with these constraints using OMNIMERGE, and the number of conformations for each subchains is shown in Figure 4-21. Note the large numbers for subchains 1–4 and 5–8. For non-leaf subchains, the two numbers provided are the number of candidate conformations to be evaluated (before the arrow) and the number of candidates found to satisfy the constraints (after the arrow).

If we searched the same molecule using PROPAGATION, the unique ranges for subchains at the interface between the two halves would propagate immediately to the rest of the molecule and the search would complete extremely quickly, with or without the A* ordering.

OMNIMERGE searches subchains in a predefined order, from left to right and from small to large. Figure 4-22 shows this order using increasing color intensity to indicate the passage of time.

If we search the same eight-residue helix with extreme constraints but using the A* function to select subchains, the order in which the subchains are searched is shown in Figure 4-23. Black entries indicate a subchain that has been skipped.

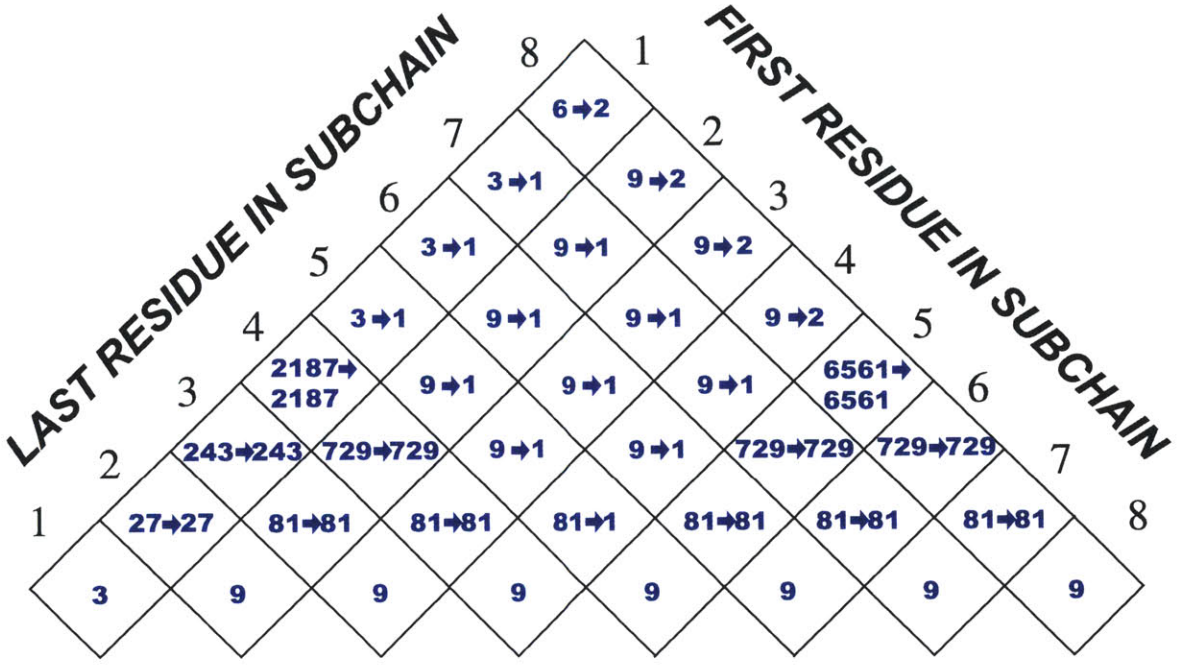

Figure 4-21: Subchain size statistics acquired by running OMNIMERGE (without any PROPAGATION) on the eight-residue helix with extreme constraints.

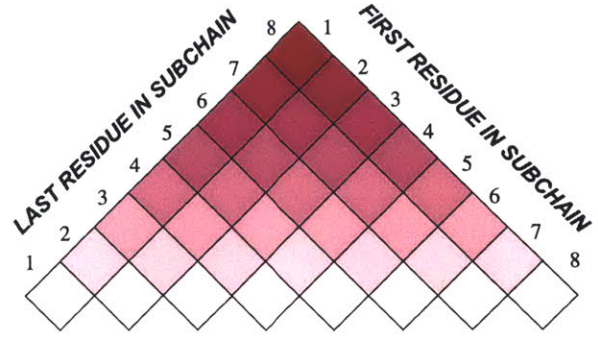

Figure 4-22: The order in which subchains are searched by OMNIMERGE is shown using increasing color intensity for the passage of time.

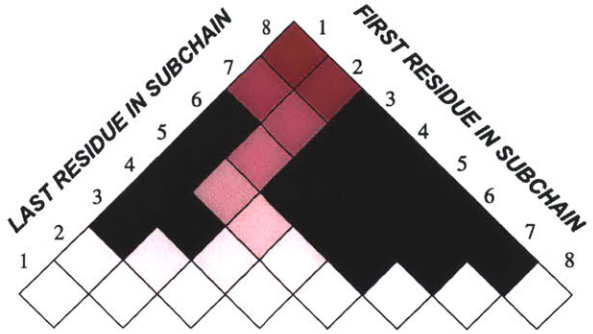

Figure 4-23: The order in which subchains are searched by OMNIMERGE with A* ordering (but without PROPAGATION) is represented by increasing color intensity.

When the number of satisfying conformations is dramatic and obvious, such as for the extreme example without PROPAGATION, using A* orders the subchains very decisively, skipping the right and left halves of the molecule and choosing subchains "up the middle" of the table.

4.6 A Difficult Case

We constructed a more difficult set of test cases using a 16-residue helix of polyalanine. (See Figure 4-24.) We chose a length of polyalanine with $2^4 = 16$ residues so that the

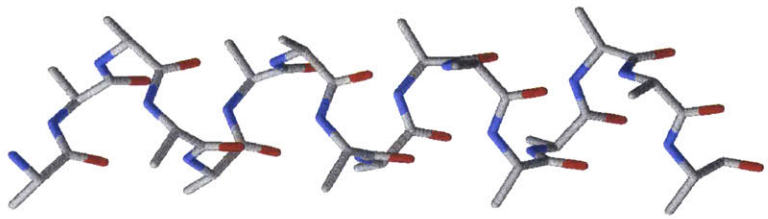

Figure 4-24: Sixteen residues of alanine with $\phi = -57.0°$, $\psi = -57.0°$, and $\omega = 180.0°$.

default merge-tree would be well-balanced and we established short-range constraints that are randomly distributed along the molecule. The constraints were generated by forming the molecule into a helix, measuring all its interatomic distances, and among all atom pairs with a distance between 2.5 and 6.0 Å, selecting some uniformly at random and constraining them to an interatomic distance within some threshold (called the "padding")

of the measurement. The selected pairs of atoms were chosen without regard for atom types, nuclear-spin properties, or covalent bonding arrangement, and therefore some constraints may be redundant with each other and/or with the rigid geometry. The pairs of atoms were selected independently for each of the trial constraint sets.

This is not expected to be a favorable case for OMNIMERGE and its variants for several reasons. Because of the complete uniformity of the molecule and the relatively uniform number of constraints for each part of the molecule, we expect there will be little difference in cost between alternative merge strategies. Thus, we expect that the merges performed by OMNIMERGE (using the optimal choices of left and right children) will not be significantly better than the merges in the default merge-tree, and OMNIMERGE will have to search many more subchains than the default merge-tree (quadratic instead of linear).

In particular, we expect the merges of large, overlapping subchains to be a waste of time because there are no constraints on pairs of atoms separated by more than five residues in the sequence. The only new constraints that will become active when performing a combine operation on medium or large subchains will be short-range constraints at the interface between the two children. If a small subchain that overlaps the interface has already been searched and its results propagated, then the high level merges will involve virtually no pruning. In that case, LIMITED OMNIMERGE will certainly outperform OMNIMERGE.

We do expect PROPAGATION to be useful, although simply not performing unnecessary searches of "extra" overlapping subchains might be superior. Finally, we expect that the A* augmentation of OMNIMERGE will not be able to suggest an order of search that is significantly better from the "default" order of plain OMNIMERGE.

Results

We searched the polyalanine peptide at 120° resolution for all bonds, with up to five passes of minimization per voxel, according to five different trial conditions and sets of constraints. All peptide bonds (ω torsion angles) were constrained to between 175° and 185°. Hydrogen atoms were excluded from the model.

Trial A constrained 70 pairs of atoms to within 0.05 Å of their measured distance in the helix and all constraints were considered satisfied when the summed squared violations were less than 0.0005 Å^2. All search methods found 6 satisfying conformations. Parallel statistics for each of the five trials appear in Figure 4-25. Figure 4-26 shows the results of

Trial	Number of Atom Pairs Constrained	Distance Constraint Padding (Å)	Sum-Sq Violation Allowed (Å^2)	Satisfying Conformations Found
A	70	0.05	0.0005	6
B	73	0.05	0.0005	15
C	503	0.5	0.005	15
D	385	0.5	0.005	30
E	294	0.5	0.005	*varies*

Figure 4-25: Five sets of parameters used for generating constraints and running searches on 16-residue peptides of polyalanine. The right column shows the number of satisfying conformations found by the searches.

searches using the conditions in trials A, B, C, and D. As expected, OMNIMERGE without PROPAGATION performs poorly; it requires an order of magnitude more time than divide-and-conquer using the default merge-tree. Also as expected, LIMITED OMNIMERGE universally outperforms OMNIMERGE, and PROPAGATION always improves performance.

The surprise is that LIMITED OMNIMERGE (with PROPAGATION) outperformed the default merge-tree in all trials. The fastest run time for each trial is colored green. LIMITED OMNIMERGE (with PROPAGATION) required between 10 percent and 50 percent as much time as the default algorithm.

Using A* to choose the order for searching subchains had mixed effects, ranging from helping significantly in trial A, and helping insignificantly in trial B, to hurting insignificantly in trials C and D.

As another comparison, we performed OMNIMERGE searches at 180° resolution (for all bonds) to obtain optimal low resolution merge-trees; then the low resolution optimal trees were used for divide-and-conquer searches at the regular 120° resolution. For the run times of these searches, the times for just the optimized divide-and-conquer searches are enclosed in parentheses, and the total times, including the time for performing OMNIMERGE at low resolution, are shown without parentheses. The lower resolution searches were initially performed with only one pass per voxel, but OMNIMERGE at low resolution failed to find any conformations that satisfied the constraints for trials A and B. (These false negatives are not too surprising because 180° resolution is coarser than the broadest rotamers, which are local minima. Multiple minima per voxel would not reliably be explored using only one pass.) Therefore, we repeated the low resolution searches for trial conditions A and B with

Search Method*	With A*	Trial A Run Time	Trial B Run Time	Trial C Run Time	Trial D Run Time
divide-and-conquer with default merge-tree		80.3	26.8	58.5	179
divide-and-conquer with merge-tree optimized at 180° resolution		(11.2)	(11.1)	(45.5)	(89.0)
above plus cost of OMNIMERGE at 180° resolution		98.3	65.8	55.3	114
OMNIMERGE without propagation		1002	749	103	199
LIMITED OMNIMERGE without propagation		633	380	75.0*	106
OMNIMERGE with PROPAGATION		20.5	47.7	13.6	63.2
LIMITED OMNIMERGE with PROPAGATION		18.2	14.4	11.0	21.0
OMNIMERGE with PROPAGATION	√	8.23	47.9	14.3	58.3
LIMITED OMNIMERGE with PROPAGATION	√	6.01	11.1	11.4	21.9

Figure 4-26: Run times for simulations on polyalanine (in seconds) according to various search methods with four different conditions. The limit of subchain size for LIMITED OMNIMERGE was two-thirds of the size of the whole molecule, or up to 10 residues.
*In trial C, LIMITED OMNIMERGE only found 14 of the 15 conformations found by the other methods.

up to 5 passes of minimization per voxel, which required much more time. Not surprisingly, the combined cost of low resolution OMNIMERGE and high resolution divide-and-conquer is greater than the default for trials A and B. However, for trials C and D, where the lower resolution searches succeeded with only one pass per voxel, the total time for low resolution optimization was faster than the default algorithm.

More False Negatives

There was one additional constraint set, "Trial E", with a larger number of looser constraints, that proved a particular challenge for any of the methods to search completely. We followed up with repeated searches using a larger limit on the number of passes per voxel. To compare the completeness of the different methods, each method was run with the maximum number of minimization passes per subchain set to 5^k, for $k = 0, 1, 2, 3, 4, 5$. Figure 4-27 shows the number of conformations found, according to the number of passes used,

Search Method	1 Pass # Confs	5 Passes # Confs	25 Passes # Confs	125 Passes # Confs	625 Passes # Confs
divide-and-conquer with default tree	124	317	323	326	329
LIMITED OMNIMERGE	60	300	323	326	331
OMNIMERGE	59	294	326	323	329
LIMITED OMNIMERGE with A*	39	275	318	327	328
OMNIMERGE with A*	48	292	322	328	328

Figure 4-27: The number of conformations found for simulations that varied both the search algorithm and the number of passes of minimization per voxel. See also Figure 4-28. All variants of OMNIMERGE shown include PROPAGATION.

and Figure 4-28 shows the corresponding run times required. The results for three of the methods (default divide-and-conquer, OMNIMERGE with A*, and LIMITED OMNIMERGE with A*) are also plotted in Figure 4-29. Divide-and-conquer has the best completeness when using a small number of passes, but OMNIMERGE performs best when using many passes.

We do not know whether the difficulty of enumerating the satisfying conformations completely is because the constraint set for trial E is particularly complicated in some way,

Search Method	1 Pass Run Time	5 Passes Run Time	25 Passes Run Time	125 Passes Run Time	625 Passes Run Time
divide-and-conquer with default tree	157.	4,623.	24,392.	122,683.	616,051.
LIMITED OMNIMERGE	18.1	211.	709.	3,252.	14,733.
OMNIMERGE	33.6	423.	987.	3,840.	13,644.
LIMITED OMNIMERGE with A*	13.7	253.	903.	3,340.	14,960.
OMNIMERGE with A*	32.0	403.	1,083.	3,129.	12,951.

Figure 4-28: The Number of seconds of run time for simulations that varied both the search algorithm and the number of passes of minimization per voxel. See also Figure 4-27. All variants of OMNIMERGE shown include PROPAGATION. Times are rounded to the nearest second or to three significant digits, whichever significance is greater.

or because the boundaries of the satisfying space lie near the boundaries of the voxels, or because of other factors. While systematic search methods are often hailed for their completeness, we find here that false negatives are a critical factor for evaluating the performance of systematic algorithms. As dimensionality increases, with larger molecules, we expect this problem to become an even more decisive factor.

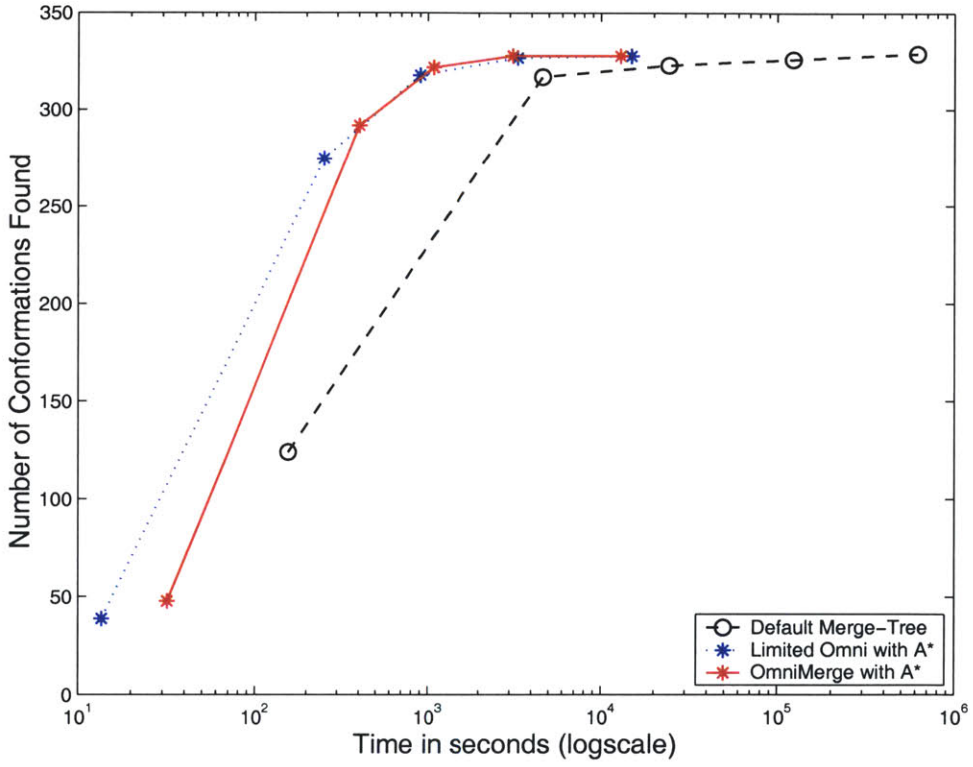

Figure 4-29: Plot of run time versus number of conformations using the data from Figures 4-27 and 4-28.

Worst Case

For OMNIMERGE (or LIMITED OMNIMERGE) with PROPAGATION and A*, the worst possible case would be if the search problem had no active constraints for any subchains smaller than the whole molecule, and every subproblem were maximally unconstrained. Although divide-and-conquer with a default merge-tree (and every other method we have considered) would perform poorly on such a case, OMNIMERGE would be by far the worst because it would evaluate the exponential number of possibilities for *every* subchain (two subchains of size $N-1$, three subchains of size $N-2$, etc.) instead of evaluating them only for two subchains of size $\frac{N}{2}$, four subchains of size $\frac{N}{4}$, etc. With no conformations to disqualify, PROPAGATION could only add overhead. With no subchains particularly well-constrained and no particular paths to the goal better than the others, A* could only add overhead. OMNIMERGE would perform a quadratic number of extra, exponentially-large searches, compared with divide-and-conquer using a default merge-tree.

However, if no constraints are active until the final merge, all methods would require time exponential in N, because the final merge would have an exponential number of candidates, no matter how few conformations actually turn out to satisfy the constraints.

At the start of this chapter we showed an example where a poor choice of merge-tree, relative to the constraint set, could result in an exponential search of a molecule even though another merge-tree exists that can be searched in polynomial time. We now conclude the chapter by postulating that no A* search will take exponential time unless *all* possible merge-trees also take exponential time.

Chapter 5

The Crystal Structure of a Protein–DNA Complex

Article appeared in *Structure* (1997) Vol. 5, No. 8, Pages 1047–1054.
The protein data bank accession code for this structure is 2HDD.

Engrailed (Gln50→Lys) homeodomain–DNA complex at 1.9 Å resolution: structural basis for enhanced affinity and altered specificity

Lisa Tucker-Kellogg, Mark A. Rould, Kristen A. Chambers, Sarah E. Ades, Robert T. Sauer, and Carl O. Pabo

Abstract

Background

The homeodomain is one of the key DNA-binding motifs used in eukaryotic gene regulation, and homeodomain proteins play critical roles in development. The residue at position 50 of many homeodomains appears to determine the differential DNA-binding specificity, helping to distinguish among binding sites of the form TAATNN. However, the precise role(s) of residue 50 in the differential recognition of alternative sites has not been clear. None of the previously determined structures of homeodomain-DNA complexes has shown evidence for a stable hydrogen bond between residue 50 and a base, and there has been much discussion, based in part on NMR studies, about the potential importance of water-mediated contacts. This study was initiated to help clarify some of these issues.

Results

The crystal structure of a complex containing the engrailed Gln50→Lys variant (QK50) with its optimal binding site TAATCC (versus TAATTA for the wild-type protein) has been determined at 1.9 Å resolution. The overall structure of the QK50 variant is very similar to that of the wild-type complex, but the sidechain of Lys50 projects directly into the major groove and makes several hydrogen bonds to the O6 and N7 atoms of the guanines at base pairs 5 and 6. Lys50 also makes an additional water-mediated contact with the guanine at base pair 5 and has an alternative conformation that allows a hydrogen bond with the O4 of the thymine at base pair 4.

Conclusions

The structural context provided by the folding and docking of the engrailed homeodomain allows Lys50 to make remarkably favorable contacts with the guanines at base pairs 5 and 6 of the binding site. Although many different residues occur at position 50 in different homeodomains, and although numerous position 50 variants have been constructed, the most striking examples of altered specificity usually involve introducing or removing a lysine sidechain from position 50. This high-resolution structure also confirms the critical role of Asn51 in homeodomain-DNA recognition and further clarifies the roles of water molecules near residues 50 and 51.

Introduction

Altered-specificity variants can provide powerful tools for studying protein-DNA recognition. The homeodomain, one of the key DNA-binding motifs used in eukaryotic gene regulation, provides a very attractive system for this type of analysis: hundreds of related homeodomain sequences are known, and there is a wealth of relevant biochemical and structural data [1]. Biochemical and genetic studies indicate that residue 50 is especially important in determining the differential specificity of homeodomain-DNA recognition [2–5], playing a role in distinguishing between binding sites of the form TAATNN. Glutamine is the most common residue at position 50, but cysteine, serine and lysine occur in other subfamilies [1]. The tightest and most specific binding occurs when lysine is present at position 50. Biochemical studies of an engrailed Gln50→Lys variant (QK50) revealed that QK50 actually binds more tightly to TAATCC than wild-type engrailed binds to TAATTA (Table 5.1) [6]. We have pursued structural studies of this Lys50 variant to understand how it forms such a stable complex, to elucidate the role of position 50 in homeodomain-DNA recognition, and—more generally—to explore the structural requirements for design-

ing altered-specificity variants. We find that the Lys50 sidechain projects directly into the major groove of the DNA and makes a set of hydrogen bonds with the guanines at base pairs (bps) 5 and 6 of the optimal TAATCC binding site. The Lys50 sidechain and these new contacts are accommodated without requiring any major changes in the overall architecture of the homeodomain-DNA complex.

Version of Engrailed	**DNA site***	
	TAATTA	**TAATCC**
Wild type	0.079	21
QA50	0.19	3.4
QK50	0.32	0.0088
*Only one strand of the DNA subsite is indicated; binding studies used 20 bp duplex DNA sites.		

Table 5.1: Equilibrium dissociation constants (in nM) for the complexes with the wild type engrailed homeodomain, the QA50 variant, and the QK50 variant.

Results

We have crystallized the engrailed QK50 variant in complex with the duplex TAATCC binding site, and have solved this structure at 1.9 Å resolution. The DNA duplex used for cocrystallization is homologous to that used in studies of the wild-type complex and gave cocrystals that are nearly isomorphous to the wild-type cocrystals studied by Kissinger et al. [7]. Our refined model (using data collected at −150°C) has a free R factor of 25.1% and a conventional R factor of 20.5% for data from 6.0–1.9 Å resolution (Table 5.2). The overall structure of the QK50 complex is very similar to that of the wild type engrailed complex (studied at 2.8 Å resolution at room temperature): alignment of the complexes by superimposing Cα atoms of the homeodomain (residues 10–55) and P and C1′ atoms of the TAATNN subsites (i.e. superimposing 24 atoms of each DNA duplex) gives a root mean square (rms) deviation of 0.48 Å, and confirms that the folding and docking are exceedingly similar. As in the wild-type complex, the homeodomain folds as a globular domain with three α helices, and helix 3, the 'recognition' helix, fits into the major groove of the DNA. An extended N-terminal arm contacts the minor groove. Given that the wild-type and QK50 complexes are so similar, we focus our attention on Lys50 and Asn51. These residues

Data Collection	
Resolution (Å)	1.9
Measured reflections	92,610
Unique reflections	27,136
Completeness to 1.9 Å (%)	94
Completeness in 1.97 − 1.90 Å shell (%)	84
R_{merge}* (%)	3.5
Refinement	
R factor† (%)	20.5
R_{free}† (%)	25.1
Rms deviation of bond lengths (Å)	0.011
Rms deviation of bond angles (°)	1.46
Number of nonhydrogen atoms	1,990
Number of water molecules	183
Rms ΔB (Å^2)‡	2.32

* $R_{merge} = \Sigma|I - \langle I\rangle|/\Sigma I$, where I =observed intensity and $\langle I\rangle$ = average intensity of multiple observations of symmetry-related reflections.

†The R factors exclude 2564 reflections for which $F < 2\sigma(F)$. Using all data from 6.0 − 1.9 Å, the R factor is 21.7% and the R_{free} is 26.5%.

‡Rms ΔB is the root mean squared difference between temperature factors of covalently bonded atoms.

Table 5.2: Data collection (−150°C) and refinement statistics.

are critical for site-specific recognition and have been the focus of much discussion when comparing NMR and crystal structures of homeodomain-DNA complexes [7–14].

Interactions between Lys50 and the DNA

Figure 5-1 shows the electron density for Lys50 from the solvent-flattened MIRAS (multiple isomorphous replacement with anomalous scattering) map (using data collected at 10°C) and also shows the final refined model (using data collected at −150°C). The overall placement of the lysine sidechain is exceedingly clear: Lys50 projects into the major groove towards the guanines of bps 5 and 6, and many of the key contacts involve hydrogen bonds to the O6 and N7 atoms of these guanines (Figures 5-1–5-4). The high resolution of our structure determination allows us to see and refine alternate conformations for the terminal atoms of Lys50. Conformation 1 (60% occupancy) places the terminal NH_3^+ group near the guanines of bps 5 and 6. In this conformation, the closest contacts involve the O6 of the guanine at bp 5 (2.76 Å) and the O6 and N7 of the guanine at bp 6 (3.27 and 3.17 Å, respectively). The N7 of the guanine at bp 5 is slightly farther away (3.92 Å), but there is a bridging water molecule that contacts the N7 of this guanine (2.88 Å) and the terminal NH_3^+ of the Lys50 sidechain (3.17 Å). Conformation 2 (40% occupancy) only moves the terminal Nζ of the lysine by 1.42 Å but the altered sidechain dihedral points the NH_3^+ somewhat more towards the guanine at bp 5 and the thymine at bp 4. In this conformation, the closest contacts involve the O6 of the guanine at bp 5 (2.78 Å) and the O4 of the thymine at bp 4 (3.04 Å). The N7 of the guanine at bp 5 is 4.11 Å away, but there are good contacts from the bridging water molecule (which in this conformation is 3.17 Å from the Nζ and 2.88 Å from the N7 of the guanine).

The role of Asn51

This high resolution cocrystal structure also provides important information about the role of Asn51 in homeodomain-DNA recognition. The structure of the wild type engrailed complex at 2.8 Å resolution [7] indicated that Asn51 makes a pair of hydrogen bonds with the adenine at bp 3 of the TA<u>A</u>TTA site, and similar contacts were seen in the α2 homeodomain-DNA complex [8]. There has been much discussion about these contacts, since NMR studies of the Antp-DNA complex have suggested that Asn51 has multiple, rapidly-interchanging conformations and have indicated that Asn51 might make water-mediated contacts with

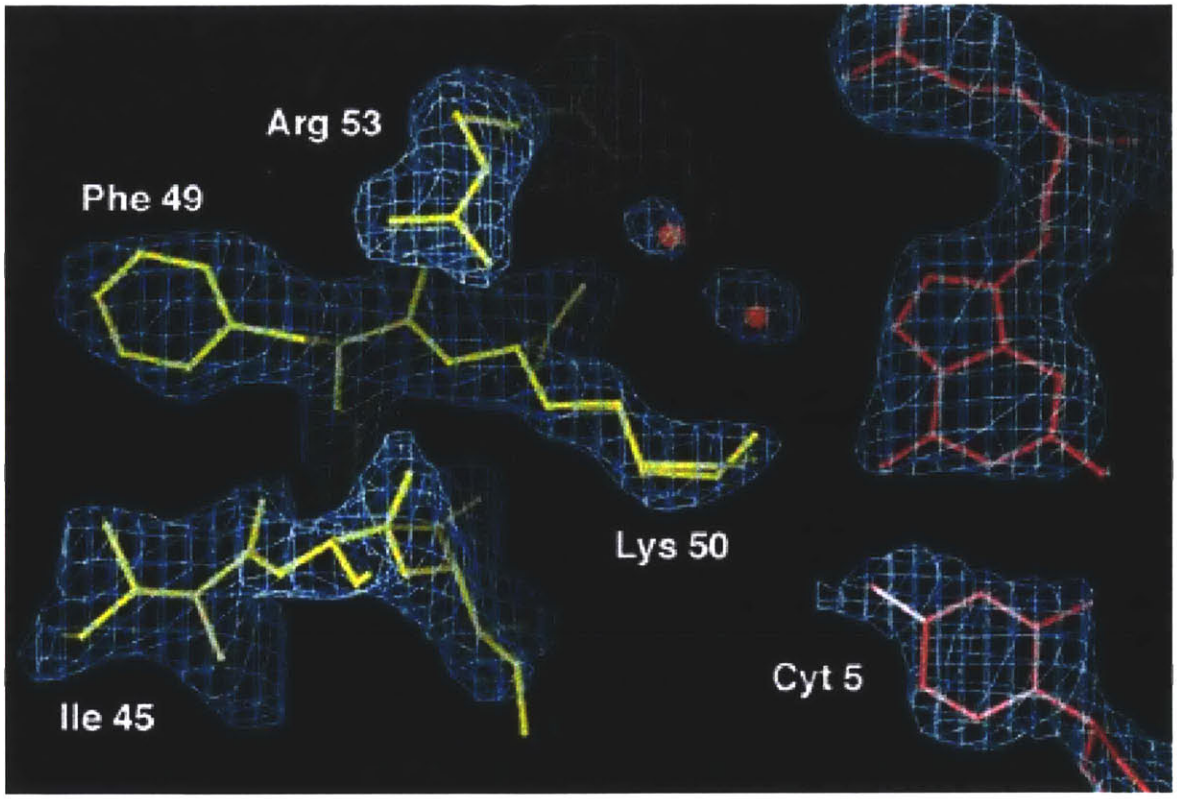

Figure 5-1: Solvent-flattened MIRAS electron-density map contoured at 1.5σ, in the vicinity of Lys50 and bp 5 of the TAATCC subsite. The model is our final refined low-temperature structure, but a rigid-body motion has been used to adjust for differences in cell dimensions (between the 10°C map and the −150°C refined structure). The protein is shown in yellow and the DNA in red. The two conformations of Lys50 are essentially superimposed when seen from this orientation.

the bases [9, 10]. Our structure of the QK50 variant confirms that Asn51 forms a pair of direct hydrogen bonds with the adenine at bp 3, with distances of 3.04 Å to the N7 atom and 3.09 Å to N6. The conformation and contacts that we observe for Asn51 are in excellent agreement with those observed in studies of the Oct-1 [11], paired [12], a1/α2 [13], and even-skipped [14] homeodomain-DNA complexes. The remarkable consistency of these structures (determined independently and in different crystal forms) suggests that these crystallographic studies are giving the correct (time-averaged) conformation of Asn51. The fact that Asn51 is so strictly conserved among the hundreds of known homeodomains (see [1] for a review) and the fact that homeodomain binding sites almost invariably have adenine at bp 3, suggests that the Asn51-adenine contacts may be similar in all homeodomain-DNA complexes.

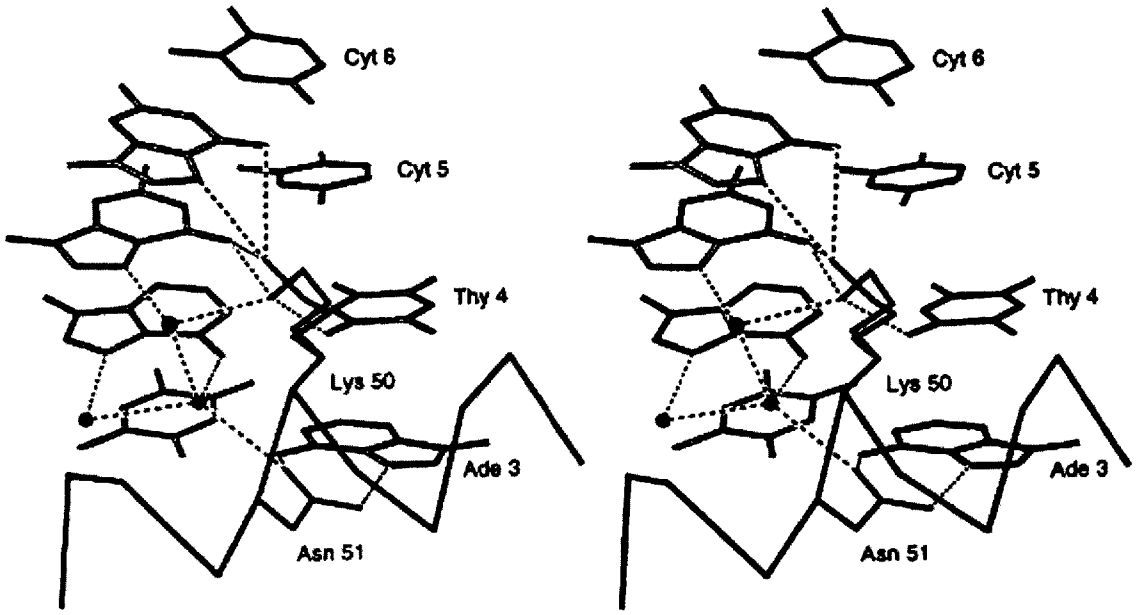

Figure 5-2: Stereo diagram showing base contacts made by Lys50, Asn51 and associated water molecules in the QK50–TAATCC cocrystals. Both conformations of the Lys50 sidechain are shown, and the three key water molecules are represented as black spheres; hydrogen bonds are shown as dashed lines. The numbering scheme for base pairs corresponds to that used in Figures 5-3 and 5-4. A Cα trace is shown for part of helix 3.

Water molecules in the binding interface

In addition to the direct hydrogen bonds made with the adenine at bp 3, Asn51 is flanked by several well-ordered water molecules at the protein-DNA interface. Perhaps the most striking interaction involves a water molecule that bridges from the Oδ1 of Asn51 to the N6 of the adenine at bp 4 (Figures 5-2 and 5-4). This water molecule has excellent hydrogen-bonding geometry, with distances of 3.01 Å to the Oδ1 and 3.11 Å to the N6 atom. This water molecule also bridges to a second water molecule which, in turn, contacts the N7 of the adenine at bp 4 (Figures 5-2 and 5-4). The distance between these two water molecules is 3.15 Å, and the distance from the second water molecule to the N7 is 3.08 Å. A third water molecule in this hydrogen-bonding network contacts the N7 of the guanine at bp 5.

We note that there is a 'tilt' in the Asn51 sidechain amide that allows Asn51 to maintain good hydrogen bonds with the adenine and yet also allows it to interact well with the bridging water molecule. (This tilt involves a rotation around χ_2, and the observed χ_2 angle ($-37°$ in our complex leaves the amide plane 27° out of the plane of the adenine.) The observed tilt of the Asn51 sidechain underscores the potential importance of the bridging water molecule. It seems quite plausible—as suggested by Wilson *et al.* [15]—that the water-mediated contacts with the adenine at bp 4 may augment the sequence specificity provided by the Ile47-thymine contact (Figure 5-3) and thus may help explain the preference of the homeodomain for the canonical TAA$\underline{\text{T}}$ subsite. Examining other refined structures of homeodomain-DNA complexes shows similar tilt angles for the Asn51 sidechains. Remarkably, we note that a conceptually analogous bridging interaction also occurs in the α2 portion of the a1/α2 homeodomain-DNA complex: here the terminal atoms of the Arg54 sidechain, rather than a water, participate in a hydrogen-bonding network that bridges from the Oδ1 of Asn51 (2.99 Å) to the O6 of the guanine at bp 4 (3.09 Å).

Discussion

The role of position 50

As emphasized in the early biochemical studies [2–5], residue 50 plays a key role in determining the differential specificity of the homeodomain, helping to explain how homeodomains can distinguish one TAATNN site from another. Correlating all the available data, however, highlights the fact that different residues at position 50 confer very different degrees

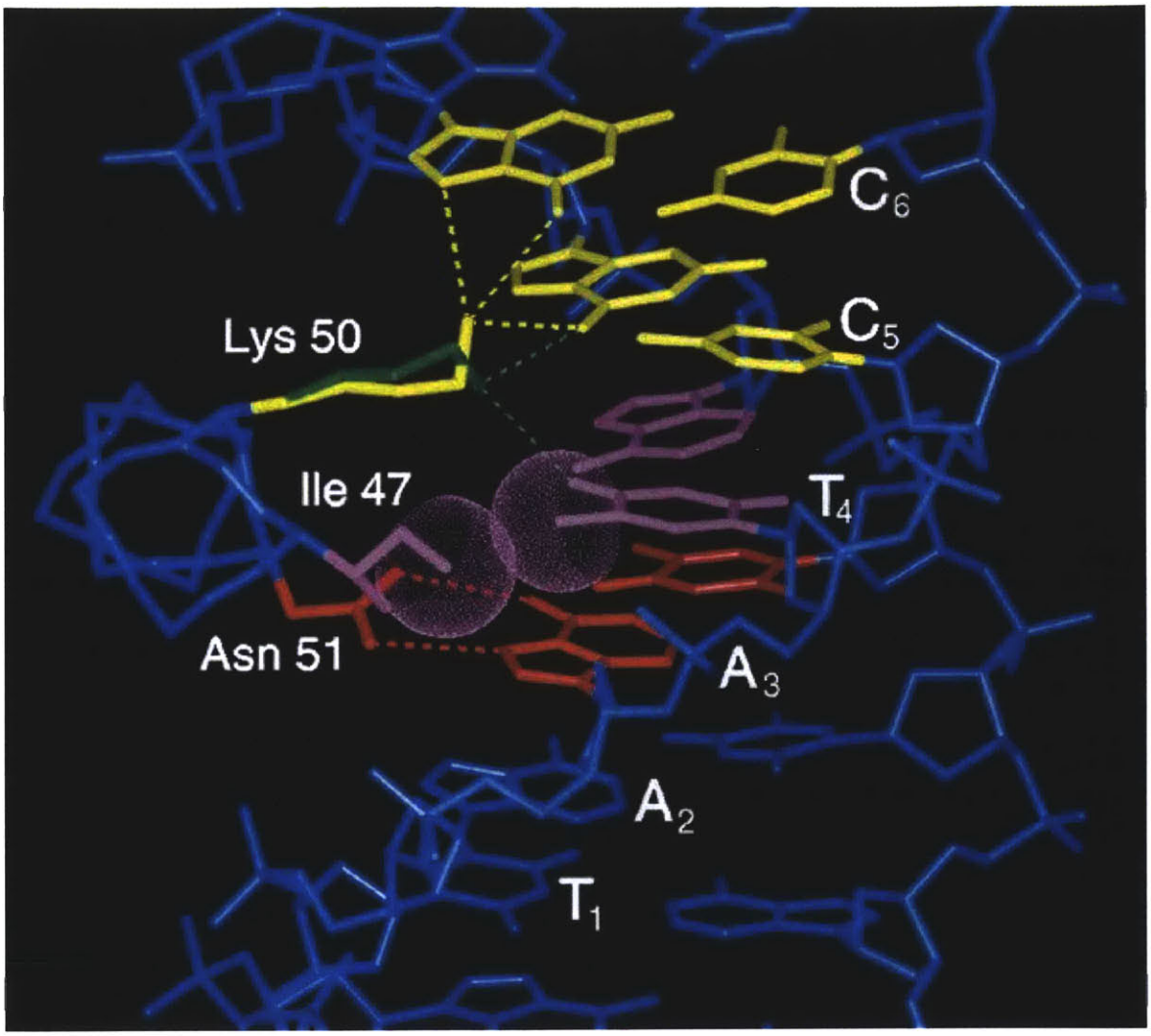

Figure 5-3: Major groove contacts of the QK50–TAATCC complex. Three residues make base contacts in the major groove: Asn51 makes a pair of hydrogen bonds with the adenine at bp 3 (red); Ile47 makes hydrophobic contacts with the methyl group of the thymine at bp 4 (purple); the primary conformation of Lys50 (yellow) makes hydrogen bonds with the O6 of the guanine at bp 5 and with the O6 and N7 atoms of the guanine at bp 6; the secondary conformation of Lys50 (green) makes hydrogen bonds with the O6 of the guanine at bp 5 and with the O4 of the thymine at bp 4. Hydrogen bonds are shown as dashed lines and van der Waals contacts are indicated with dotted spheres. For clarity, water molecules have been omitted in this figure.

of specificity for their respective sites. Reviewing the earlier papers [2–5], shows that the most striking cases of altered-specificity mutations usually involve introducing or removing a lysine residue from position 50: Key constructs are a Lys50 →Gln variant of the bicoid homeodomain [3], a Ser50→Lys variant of the paired homeodomain [4], and a Gln50→Lys mutation in the *fushi tarazu* homeodomain [5]. In every case, the Lys50 variants of these homeodomains prefer to bind a sequence of the form TAATCC and they presumably all make contacts similar to those seen in the crystal structure reported here.

There are other cases in which the sidechains at position 50 only have a marginal role in determining DNA-binding specificity. For example, the Oct-1 homeodomain, with a cysteine at position 50, shows sequence specificity for a 4 bp subsite (typically with the sequence AAAT) but shows little specificity at positions that would correspond to bp 5 and 6 of our current numbering scheme. Changing this cysteine to glutamine has little effect on DNA-binding affinity [16,17]. Even when glutamine, which is one of the most common residues at position 50, occurs in the wild-type proteins it may have only a modest energetic contribution to binding: wild-type engrailed prefers a TAATTA site, but a variant (QA50) which has alanine at position 50 binds the TAATTA site only 2.4-fold less strongly than the wild-type protein (Table 5.1) [6]. This modest energetic contribution of the Gln50 sidechain ($\Delta\Delta G = 0.5$ kcal/mol) is fully consistent with the 2.8 Å resolution crystal structure of the wild type engrailed complex, in which the only direct contact between Gln50 and the DNA bases is a van der Waals contact with the methyl group of the thymine at bp 6.

When placed at position 50, lysine seems to have a clearer sequence specificity than other residues tested and makes a greater energetic contribution to binding. Thus, comparing the QK50 and QA50 variants of engrailed shows that changing Lys50 to alanine gives a 390-fold reduction in affinity for the TAATCC sequence (Table 5.1; $\Delta\Delta G = 3.4$ kcal/mol) [6]. The numerous direct contacts made by Lys50 in our QK50 structure, and the way that these lysine-guanine contacts fit so well within the conserved structure of the complex, provide a simple explanation for the efficacy of this residue in site-specific recognition. Remarkably, these are the first direct hydrogen-binding contacts reported for residue 50 in any homeodomain-DNA complex.

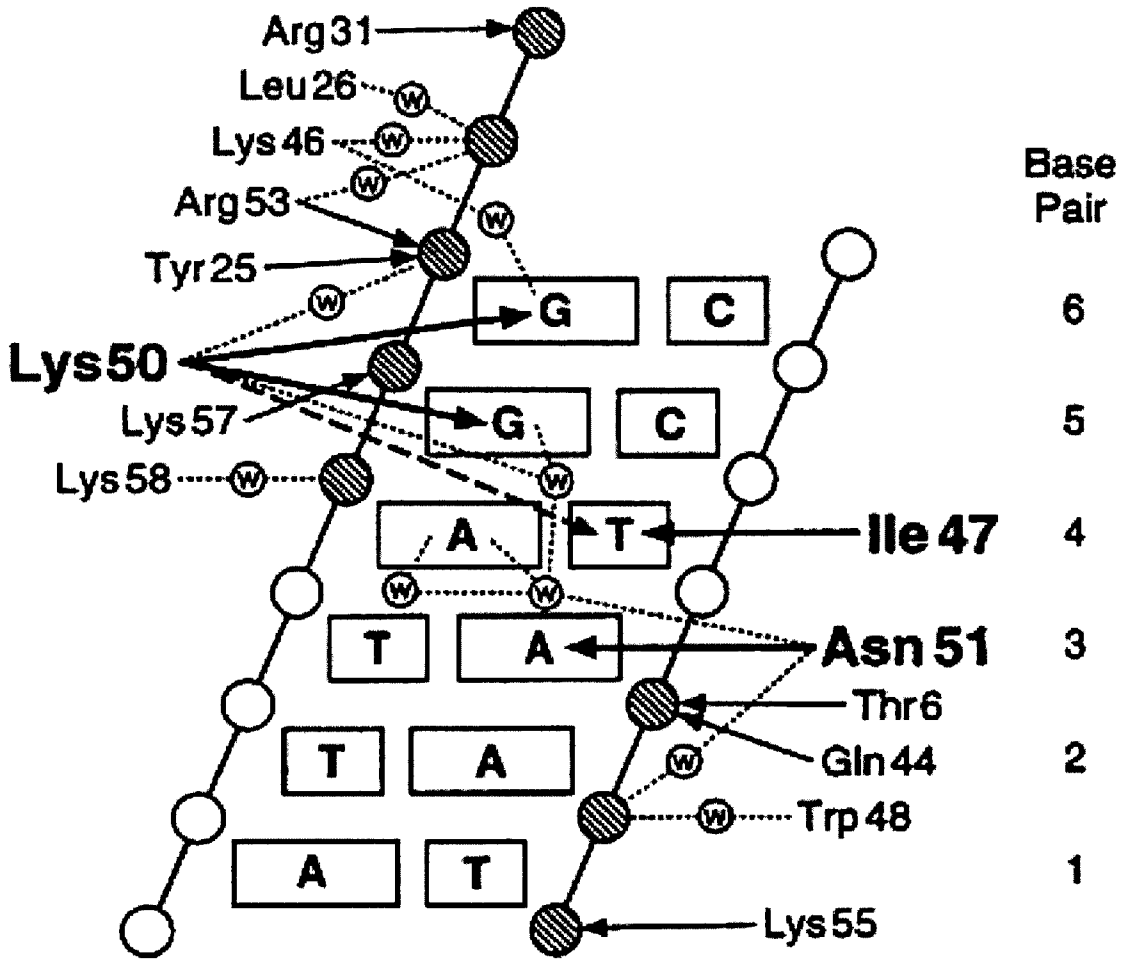

Figure 5-4: Sketch of major groove contacts in the QK50–TAATCC complex. Residues that make base contacts in the major groove are shown in boldface. Phosphates are represented with circles, and hatched circles mark phosphates that are contacted by the homeodomain. Arrows represent direct protein-DNA contacts. Small circles marked 'W' denote water molecules, and finely-dotted lines represent water-mediated contacts. The numbering scheme for the base pairs corresponds to that used in Figures 5-2 and 5-3; base pairs are numbered to represent a typical homeodomain binding site in the form TAATNN.

Base-specific electrostatic interactions

Our structure of the QK50–DNA complex also highlights the role that 'electrostatic readout' of the major groove may play in site-specific recognition. Recent surveys of sidechain-base interactions in the known protein-DNA complexes show that arginine-guanine and lysine-guanine interactions are remarkably common [18, 19]. As hydrogen bonds involving one charged partner can be very strong [20], and because the N7 of guanine is the most electronegative region in the major groove [21], such contacts may make a major contribution to site-specific recognition. (A key lysine residue in the N-terminal arm of λ repressor also forms hydrogen bonds with a pair of guanines in the major groove [22, 23].) We presume that the binding affinity of the QK50 variant reflects both the intrinsic affinity of these lysine-guanine interactions and the very favorable structural framework (provided by the rest of the homeodomain-DNA complex), which holds lysine in an ideal position for making these contacts.

Prospects for other altered-specificity variants

One of the underlying structural issues in protein-DNA recognition—and one that is especially important when thinking about altered-specificity variants—involves the complex interrelationship between the overall folding and docking arrangement of a protein and the geometric requirements for particular sidechain-base interactions (COP and L Nekludova, unpublished data). To what extent does the overall folding and docking determine which sidechain-base interactions are geometrically plausible at a given position? How often will a strictly local substitution, such as the Gln50→ Lys change in engrailed, allow new DNA contacts that are as favorable or more favorable than the wild-type contacts?

In thinking about these issues, it is interesting to re-examine the role of Gln50 in the engrailed homeodomain. After seeing the important role that glutamine-adenine contacts play in other protein-DNA complexes, one might have imagined that glutamine would make a pair of hydrogen bonds with one of the adenines in the preferred TAATTA binding site. However, (as mentioned above) the only direct contact between Gln50 and the DNA is a van der Waals contact with the methyl group of the thymine at bp 6, and changing Gln50 to alanine only gives a modest (0.5 kcal/mol) reduction in affinity for the TAATTA site.

Modeling studies readily confirm the problems that would be involved in trying to make

canonical glutamine-adenine contacts from position 50 of the homeodomain, and modeling thus helps explain why these contacts do not occur in the wild-type complex. Surveying known protein-DNA complexes shows that the most favorable glutamine-adenine interactions (such as those seen in the λ repressor [23, 24] and the 434 repressor [25] complexes) involve a pair of hydrogen bonds between the sidechain and the base, and in these situations the terminal atoms of the sidechain (Cγ, Cδ, Nϵ and Oϵ) are roughly coplanar with the adenine base. While keeping the protein and DNA backbones fixed in the conformation of our QK50 crystal structure, we attempted to superimpose the same 'canonical' glutamine-adenine contacts seen in the phage repressors onto residue 50 and bps 5 or 6 of the engrailed complex (L Nekludova and COP, unpublished data). Regardless of what sidechain χ angles are used during modeling, there is no way that canonical glutamine-adenine contacts can fit into the structural context provided by the homeodomain. The position and orientation of the polypeptide backbone at position 50 (vis-à-vis the DNA) provides an ideal geometric arrangement for the lysine-guanine interactions, but simply does not work as well for optimizing potential glutamine-adenine interactions.

Other studies have revealed similar limitations in the design and selection of DNA-binding proteins with altered specificity. For example, biochemical and genetic studies involving systematic variation in position 51 of the Oct-1 homeodomain and in bp 3 of the AA<u>A</u>T binding site failed to reveal any other sidechain-base combination that would work as well as the wild-type Asn51-adenine arrangement [26]. Glutamine was particularly disruptive when placed at position 51 (causing a 1,100-fold reduction in binding to the AA<u>A</u>T subsite), and it appears that the structural context provided by the rest of the homeodomain-DNA complex plays a critical role in determining which sidechain-base interactions will be possible at any given position. The basic idea is very simple and yet has broad implications for our understanding of protein-DNA recognition: given the distinct sizes, shapes, and conformational preferences of the sidechains, only one or two may fit well at a given position in a complex. The overall folding and docking arrangement of the protein (and the overall structure of the DNA) will help to determine which contacts are possible.

Conclusions

These structural and biophysical studies of the QK50 variant provide an interesting perspective on current studies of protein-DNA recognition. The analysis of multiple conformations and of water-mediated contacts has some meaningful role in the understanding of homeodomain-DNA interactions, but we find that a lysine variant which can make direct hydrogen bonds with the DNA bases binds more tightly and specifically than the wild-type engrailed homeodomain. The crystal structure of this altered-specificity complex shows there is nothing mysterious about the tight binding: the homeodomain presents the Lys50 sidechain in a very favorable geometric and structural context (fixed by the conserved folding and docking arrangement of the homeodomain), and the lysine can make a set of direct, sequence-specific hydrogen bonds with the O6 and N7 groups of the guanines. (There is also a water-mediated contact and an alternative conformation of the terminal atoms that allows a hydrogen bond with the O4 of a thymine.) Our structure gives a satisfying explanation for the affinity and specificity of the lysine QK50 variant, but challenging problems remain as we try to understand the limits of altered-specificity variants. How often will such favorable substitutions be possible? How do the overall folding and docking arrangements help to determine what sidechain-base interactions will be possible at a given position?

Biological Implications

Homeodomains are one of the most important eukaryotic DNA-binding motifs, and they occur in many transcription factors that control differentiation and determine cell fate. Homeodomains contain 60 amino acids, which fold to form a module with three α helices and an extended N-terminal arm. Homeodomain-DNA interactions have been studied intensively both because of the intrinsic importance of the homeodomain, and because the homeodomain has become a paradigm for the analysis of protein-DNA interactions. Previous structural studies have shown that helix 3, the 'recognition' helix, docks into the major groove and makes many of the base contacts. Biochemical and genetic studies have suggested that residue 50 of the homeodomain is especially important for differential recognition, distinguishing among sites of the form TAATNN. However, none of the previously determined structures of homeodomain-DNA complexes has provided evidence for a stable hydrogen bond between residue 50 and a base, and there has been much discussion about

the potential significance of water-mediated contacts in homeodomain-DNA recognition.

Biochemical data, showing that a Gln50→Lys (QK50) variant of the engrailed homeodomain has very high affinity and specificity for a TAATCC site, motivated solving the structure of this complex, and we find a set of very favorable Lys50-guanine contacts that readily explain the biochemical data. The QK50-DNA structure also confirms the conserved docking arrangement of the homeodomain and the critical Asn51-adenine contacts seen in the crystal structures of other homeodomain-DNA complexes. The fact that there is a rigidly conserved docking arrangement may help explain why other sidechains (including the wild-type glutamine) cannot make such energetically favorable contacts from position 50. More generally, our analysis suggests limits (only certain sidechains will fit at certain positions) that may occur in the design and selection of altered-specificity DNA-binding mutants. Finally, our data suggest that direct sidechain-base interactions, when geometrically compatible with the other contacts in a complex, can provide greater affinity and specificity than water-mediated contacts.

Materials and Methods

Protein expression and purification

The engrailed QK50 domain used in these studies contains 60 amino acids from the *Drosophila* engrailed protein, but the glutamine residue at position 50 of the wild-type homeodomain is replaced by lysine and an N-terminal methionine is introduced in cloning. (Thus the sequence of our peptide is the same as that shown in Figure 1 of reference [7], except that lysine is present at position 50.) The QK50 variant was expressed in *Escherichia coli* strain BL21 cells containing the DE3 plasmid [18]. Cultures were induced for 2.5 h with 0.3 mM isopropyl-β-D-thiogalactopyranoside (IPTG) at 37°C. Soluble protein was purified using ion exchange and reverse phase chromatography, and the purity of the peptide was confirmed by gel electrophoresis, mass spectroscopy, amino acid analysis, and protein sequencing (William Lane, Harvard Microchemistry Facility).

DNA complex formation and crystallization

The complex was formed in 1 M ammonium acetate (to keep it soluble), with a 2:1 molar ratio of QK50 peptide to duplex DNA. The DNA used for cocrystallization, contains one

5′—**T T T** T G C C A T G **T A A T C C** C C **G G A**
A A A C G G T A C **A T T A G G** G G **C C T A**—5′

TAATCC subsite, and when these DNA duplexes stack in the crystal, a related subsite with the sequence AAATCC is formed by the juxtaposed DNA duplexes. Crystals of the QK50-DNA complex were grown using the hanging-drop vapor diffusion method [27]. Well buffer contained 0.73–0.80 M ammonium acetate (pH 8.0) and 1% PEG 400. The best crystals grew in three days at room temperature from a 2 μl hanging drop containing the complex at a concentration of 10 mg/ml. Note that these conditions are somewhat different from those used for crystallizing the wild-type complex, which had been studied at 2.8 Å resolution [7]. As with the wild-type crystals, the QK50-DNA complex crystals form in space group C2, but have cell parameters of a = 129.9 Å, b = 45.45 Å, c = 72.75 Å, β = 118.7° at 10°C. (The wild-type crystals have a = 131.2 Å, b = 45.45 Å, c = 72.9 Å, β = 119.0° at room temperature [7].) Under cryo conditions (−150°C), the cell parameters of the QK50-DNA crystals are a = 127.7 Å, b = 45.3 Å, c = 72.5 Å, β = 119.5°. As expected from studies of the wild-type complex, the asymmetric unit of the QK50 crystals contains one DNA duplex, one homeodomain at the TAATCC site, and a second homeodomain at the AAATCC site that is formed by juxtaposed duplexes.

Phasing and refinement

Structure determination of the QK50-DNA cocrystals proceeded in two stages: MIRAS was used for initial phasing and model refinement at 2.0 Å resolution using data collected at 10°C; and final refinement to 1.9 Å used data collected under cryo conditions. For MIRAS phasing, data were collected on a Rigaku R-Axis IIC detector equipped with Yale/MSC mirrors. Two crystals of the native and of each double- or triple-iodinated DNA derivative were used and data were processed with DENZO/SCALEPACK (written by Z Otwinowski and W Minor and distributed by Molecular Structure Corp.). A constant temperature of 10°C was maintained at the crystal with an FTS AirJet crystal cooling system. Derivatives were local scaled to the native using MAXSCALE [28]. Cross-phased heavy-atom refinement (each derivative is refined separately using phases derived only from the other three derivatives) was carried out with the program PHARE [29]. Solvent flattening [30] was used to improve the phases. Heavy-atom parameters were then re-refined to convergence using the

solvent-flattened phases as parent phases (without updating the phases during refinement), and new MIRAS phases were recalculated [31]. This process of refining the heavy-atom parameters using the solvent-flattened phases, recalculating MIRAS phases, and solvent flattening was repeated four times to give the final electron-density map, free of any model bias (Figure 5-1; Tables 5.3 and 5.4). The 2.8 Å model for the wild-type complex [7], with the appropriate changes in amino acid and nucleotide sequences, was rebuilt into this density using TOM/FRODO [32, 33] and refined to 2.0 Å with XPLOR [34]. Higher resolution data, extending to 1.9 Å, were obtained using cryocrystallographic methods. Crystals were cryoprotected by adding glycerol to the hanging drop, with a final concentration of 30% (v/v) immediately prior to flash cooling. Data were collected at −150°C and processed as before. In rebuilding to the cryo data, rigid-body refinement was used to adjust for the differences in cell parameters. Throughout refinement we made repeated use of simulated annealing omit maps and monitored the free R factor to avoid overfitting the experimental data. The same free R list was used for the 10°C data and the −150°C data. Local scaling was used to correct for absorption errors and anisotropic diffraction. The two monomers were refined independently and have almost identical DNA contacts, but discussion in this paper focuses on the homeodomain at the TAATCC site as this binding site has the same sequence as that used in the biochemical studies [6] and because this complex does not have a nick in the DNA. (As in the wild-type complex, the other homeodomain binds to a 'nicked' site formed by the juxtaposition of neighboring DNA duplexes.)

	Native	Derivative 1	Derivative 2	Derivative 3	Derivative 4
Iodinated bases*	–	T2, T8′, C17	T2, T8′, C18	T2, C17	T2, C18
R_{merge} (%)	4.4	4.7	5.5	4.8	5.0
R_{cross} on I (%)	–	18.3	27.6	16.6	25.5
R_{cross} on F (%)	–	13.1	20.2	12.3	18.7
Completeness to 2.25 Å (%)	90	88	76	99	90
Completeness to 2.0 Å (%)	76	79	65	86	81
R_{cullis}	–	0.495	0.551	0.529	0.564
Phasing power	–	3.25	2.59	3.01	2.40

*Numbering scheme for bases corresponds to that used in Figure 1b of reference [7].
R_{cross} on I = sum($|I_N - I_D|$)/sum(I_N). R_{cross} on F = sum($|F_N - F_D|$)/sum(F_N).
R_{cullis} = sum($||F_{PH} + / - F_P| - |F_{H_{calc}}||$)/$sum$($|F_{PH} + / - F_P|$), for centric reflections only.
Phasing power = (sum($F_{H_{calc}}^2$)/sum(($|F_{PH}| - |F_{PH_{calc}}|$)^2))$^{0.5}$.

Table 5.3: MIRAS phasing statistics (data collected at 10°C).

Resolution shell (Å)	9.4	6.2	4.6	3.6	3.0	2.6	2.2	2.0	20–2.0
Figure of merit	0.93	0.95	0.93	0.89	0.86	0.79	0.65	0.46	0.73*

*Subsequent solvent flattening increases the mean figure of merit to 0.84.

Table 5.4: MIRAS figure of merit versus resolution.

Accession numbers

Coordinates are being deposited with the Brookhaven Data Bank. While they are being processed, a set of coordinates may be obtained by sending an e-mail message to pabo@mit.edu.

Acknowledgements

This project was supported by an NIH grant (GM31471) to COP and used equipment purchased with support from the PEW Charitable Trusts. We thank Lena Nekludova for help with some of the modeling studies cited in this paper, and we thank Ernest Fraenkel and Joel Pomerantz for helpful comments on this manuscript. LT-K was supported by an NSF Pre-doctoral Fellowship.

References

1. Gehring W.J., Affolter M., Bürglin T.: 1994, Homeodomain proteins. *Annu. Rev. Biochem.* 63: 487–526.

2. Hanes S.D., Brent R.: 1989, DNA specificity of the bicoid activator protein is determined by homeodomain recognition helix residue 9. *Cell* 57: 1275–1283.

3. Hanes S.D., Brent R.: 1991, A genetic model for interaction of the homeodomain recognition helix with DNA. *Science* 251: 426–430.

4. Treisman J., Gönczy P., Vashishtha M., Harris E., Desplan C.: 1989, A single amino acid can determine the DNA binding specificity of homeodomain proteins. *Cell* 59: 553–562.

5. Percival-Smith A., Müller M., Affolter M., Gehring W.J.: 1990, The interaction with DNA of wild-type and mutant *fushi tarazu* homeodomains. *EMBO J.* 9: 3967–3974.

6. Ades S.E., Sauer R.T.: 1994, Differential DNA-binding specificity of the engrailed homeodomain: the role of residue 50. *Biochemistry* 33: 9187–9194.

7. Kissinger C.R., Liu B., Martin-Blanco E., Kornberg T.B., Pabo C.O.: 1990, Crystal structure of an engrailed homeodomain-DNA complex at 2.8 Å resolution: a framework for understanding

homeodomain-DNA-interactions. *Cell* 63: 579–590.

8. Wolberger C., Vershon A.K., Liu B., Johnson A.D., Pabo C.O.: 1991, Crystal structure of a MAT $\alpha 2$ homeodomain-operator complex suggests a general model for homeodomain-DNA interactions. *Cell* 67: 517–528.

9. Billeter M., Qian Y.Q., Otting G., Müller M., Gehring W., Wüthrich K.: 1993, Determination of the nuclear magnetic resonance solution structure of an *Antennapedia* homeodomain-DNA complex. *J. Mol. Biol.* 234: 1084–1097.

10. Billeter M., Guntert P., Luginbühl P., Wüthrich K.: 1996, Hydration and DNA recognition by homeodomains. *Cell* 85: 1057–1065.

11. Klemm J.D., Rould M.A., Aurora R., Herr W., Pabo C.O.: 1994, Crystal structure of the Oct-1 POU domain bound to an octamer site: DNA recognition with tethered DNA-binding modules. *Cell* 77: 21–32.

12. Wilson D.S., Guenther B., Desplan C., Kuriyan J.: 1995, High resolution crystal structure of a paired (Pax) class cooperative homeodomain dimer on DNA. *Cell* 82: 709–719

13. Li T., Stark M.R., Johnson A.D., Wolberger C.: 1995, Crystal structure of the MATa1/MAT$\alpha 2$ homeodomain heterodimer bound to DNA. *Science* 270: 262–269.

14. Hirsch J.A., Aggarwal A.K.: 1995, Structure of the even-skipped homeodomain complexed to AT-rich DNA: new perspectives on homeodomain specificity. *EMBO J* 14: 6280–6291.

15. Wilson D.S., Sheng G., Jun S., Desplan C.: 1996, Conservation and diversification in homeodomain-DNA interactions: a comparative genetic analysis. *Proc. Natl. Acad. Sci.* USA 93: 6886–6891.

16. Verrijzer C.P., Alkema M.J., van Weperen W.W., van Leeuwen H.C., Strating M.J.J., van der Vliet P.C.: 1992, The DNA binding specificity of the bipartite POU domain and its subdomains. *EMBO J.* 11: 4993–5003.

17. Ingraham H.A. *et al.*, Rosenfeld M.G.: 1990, The POU-specific domain of Pit-1 is essential for sequence-specific, high-affinity DNA binding and DNA-dependent Pit-1–Pit-1 interactions. *Cell* 61: 1021–1033.

18. Ades S.E.: 1995, *The Engrailed Homeodomain: Determinants of DNA-Binding Affinity and Specificity.* PhD Thesis. Massachusetts Institute of Technology, USA.

19. Mandel-Gutfreund Y., Schueler O., Margalit H.: 1995, Comprehensive analysis of hydrogen bonds in regulatory protein-DNA complexes: in search of common principles. *J. Mol. Biol.* 253: 370–382.

20. Fersht A.R. *et al.*, Winter G.: 1985, Hydrogen bonding and biological specificity analyzed by protein engineering. *Nature* 314: 235–238.

21. Saenger W.: 1984, Principles of Nucleic Acid Structure. Springer-Verlag, New York, NY, USA.

22. Clarke N.D., Beamer L.J., Goldberg H.R., Berkower C., Pabo C.O.: 1991, The DNA binding arm of λ repressor: critical contacts from a flexible region. *Science* 254: 267–270.

23. Beamer L.J., Pabo C.O.: 1992, Refined 1.8 Å crystal structure of the λ repressor-operator complex. *J. Mol. Biol.* 227: 177–196.

24. Jordan S.R., Pabo C.O.: 1988, Structure of the lambda complex at 2.5 Å resolution: details of the repressor-operator interactions. *Science* 242: 893–899.

25. Aggarwal A., Rodgers D., Drottar M., Ptashne M., Harrison S.C.: 1988, Recognition of a DNA operator by the repressor of phage 434: a view at high resolution. *Science* 242: 899–907.

26. Pomerantz J.L., Sharp P.A.: 1994, Homeodomain determinants of major groove recognition. *Biochemistry* 33: 10851–10858.

27. Blundell T.L., Johnson L.N.: 1976, Protein Crystallography. Academic Press, San Diego, CA, USA.

28. Rould M.A.: 1997, Screening for heavy atom derivatives and obtaining accurate isomorphous differences. *Methods Enzymol.* 276: 461–472.

29. Collaborative Computational Project Number 4: 1994, The CCP4 suite: programs for protein crystallography. *Acta Cryst. D* 50: 760–763.

30. Wang B.C.: 1985, Resolution of phase ambiguity in macromolecular crystallography. *Methods Enzymol.* 115: 90–112.

31. Rould M.A., Perona J.J., Steitz T.A.: 1992, Improving MIR phasing by heavy atom refinement using solvent-flattened phases. *Acta Cryst. A* 48: 751–756.

32. Jones T.A.: 1978, A graphics model building and refinement system for macromolecules. *J. Appl. Cryst.* 11: 268–272.

33. Israel M., Chirino A.J.: 1991, *TOM/FRODO version 3.0.* University of Alberta, Alberta, Canada: California Institute of Technology, CA, USA.

34. Brünger A.T.: 1992, *X-PLOR Version 3.1: a System for X-ray Crystallography and NMR.* Yale University Press, New Haven, CT, USA.

Author Contacts

L. Tucker-Kellogg, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. MA Rould, and KA Chambers, Howard Hughes Medical Institute, Department of Biology, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. SE Ades, and RT Sauer, Department of Biology, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. CO Pabo (corresponding author), Howard Hughes Medical Institute, Department of Biology, Mas-

sachusetts Institute of Technology, Cambridge, MA 02139, USA. e-mail: pabo@mit.edu.

Keywords

altered-specificity mutation, DNA binding, homeodomain, protein-DNA interactions, X-ray crystallography.

Received / Accepted

Received: 12 May 1997

Revisions requested: 5 June 1997

Revised: 7 July 1997

Accepted: 9 July 1997

Chapter 6

Conclusion

We have presented a core algorithm for systematic conformational search (Chapter 2), an application of that algorithm to structure determination by solid-state NMR (Chapter 3), some new ideas for improving how the subproblems in conformational searches are defined (Chapter 4), and a separate structure determination project involving X-ray crystallography (Chapter 5).

6.1 Outlook for Searching Larger Molecules

We have made some progress at applying systematic search to larger problems, but with moderate resolution, the number of voxels that satisfy the constraints can be prohibitive to enumerate. Even when using very tight constraints, the search time for moderate resolution can still be very large. For lower resolution searches, incompleteness becomes the primary obstacle, particularly with large molecules.

We have found cases with large molecules where a low-resolution search fails to find any satisfying conformations, even if the constraints have been designed around some known conformation. In other words, low resolution gives us the expected problem of false negatives. Then, when we increase the resolution of the search infinitesimally, the search finds an astronomical number of conformations for the whole molecule. This may be surprising but the reasons for it become clear with hindsight. Many boundaries of conformation space are hyperplanes instead of rugged manifolds, and if the orientation of a boundary is degenerate or exactly parallel to the definition of the voxels (such as $\theta \leq 120°$), then the size of the solution set will not necessarily be continuous as a function of resolution. The simplest case

is when the feasible region of conformation space is a tall, thin rectangular slab. When the search fails to find any voxels with satisfying conformations, that means the feasible region, if it exists, must be narrower than the interval of the search. When the search explodes (when the solution sets become intractable), the feasible region must be wider than the interval of the search in several dimensions. Both situations can occur simultaneously in large molecules because here can be both overconstrained and underconstrained degrees of freedom in the same molecule at the same time.

Preliminary trials (and tribulations) with Engrailed

We performed some preliminary searches of engrailed QK50 (residues 5–55, without hydrogens, as in Chapter 5), which has over three hundred torsional degrees of freedom. The parameters were 40° resolution for all torsions, 85% of half sigma for hard-sphere atomic radii, peptide bonds constrained to within 5° of planar, and summed squared constraint violations less than 0.001 $Å^2$. We first tried to search the protein with distance constraints alone, constraining up to 100% of all atom pairs that have a distance in the crystal structure between 2.5 and 6.0 Å.

The final stage of the search, the merge that brings together the N-terminus and C-terminus, requires satisfying many newly-active constraints. The "almost correct" conformations for the left and right pieces had to be adjusted within the combined voxel via minimization, but searching over 304 possible dimensions of dihedral variation proved too difficult. Very tight distance constraints left so small a region of feasible space that it could not be found in the high-dimensional voxel. Loose constraints caused obvious problems as well.

We tried adding some dihedral constraints, which we structured to resemble the dihedral constraints provided by the solid-state NMR experiments on fMLF. First, the angle value for the torsion to be constrained is measured in the crystal structure and a 10° range is defined around that value. Then, two other 10° ranges were defined at random in the remaining 350° span of angles (overlapping ranges were discarded and randomized again). Finally, the torsion angle was constrained to be in one of the three narrow ranges. We applied constraints of this form to 61 torsions selected at random from among the non-peptide bonds.

The combination of interatomic distance constraints and torsion constraints was suffi-

cient to allow our algorithm to search the engrailed protein. With constraints allowing a 0.1 Å range in interatomic distances for all short-range pairs of atoms, and with 61 torsions constrained as described, our method identified 54 voxels containing satisfying conformations in 31,429 seconds. These trials were preliminary and there may be any number of more practical circumstances (such as a change in atomic radii) that would allow our algorithm to complete the search. We are eager to try allowing larger overlaps between the right and left children of a merge; larger overlaps the the merge might reduce the difficulty we encountered in the final merge of having many newly active constraints and of needing to adjust many of the torsions within their voxels. Nonetheless, initial indications are that increasing the number of degrees of freedom will provide greater challenges to systematic search methods than distance constraints alone can immediately counterbalance.

Implications for future goals in systematic search

Even if an algorithm had perfectly complete and instantaneous knowledge, simply listing the satisfying voxels for a protein in reasonable time requires that the number of underconstrained degrees of freedom be small. Also, because of how we have defined the problem, a single erroneous constraint that cannot be satisfied will cause the whole search to fail. (Halting with no solutions is called "failing" in constraint satisfaction parlance, but the term indicates the absence of solutions, not a judgment about the quality of the algorithm or about whether the result is a false negative.) Although both these "features" are what we originally wanted, the results (and the frequency with with these results occur) are frustrating "bugs" to users in practical circumstances.

Infeasible constraints. Our method can easily allow for small violations of a constraint by modifying the criteria of acceptance, for example $\theta \leq 120° + \epsilon$ instead of $\theta \leq 120°$. This is what we do when setting a small but nonzero threshold on the summed squared constraint violations. Allowing k constraints to be grossly violated but strongly enforcing the remaining constraints is far more complex but it is an approachable and intellectually-appealing area for future research.

Explosion of solutions. Our method can assign different resolutions to different degrees of freedom, and we found with the fMLF structure that this contributes considerably to reducing the number of voxels without sacrificing much information content. With the current implementation, the user must specify the choices of resolution. When we began

these studies, defining the resolution as a strictly independent variable seemed desirable, but a helpful feature might be an option for the algorithm to suggest resolution intervals for each torsion based on results of a preliminary search.

One of the most promising directions for future research in systematic conformational search methods may be to explore how the search problem might be recast in different terms. A very useful contribution would be a compact representation for the simultaneous variations in different torsions, although there are questions about which information to sacrifice and how inaccuracy could become amplified over the size of the molecule. Torsion angle representations can become unwieldy for large chains as a small change in one backbone angle can have a large effect on the position of distant atoms. However, Cartesian coordinate or distance matrix representations are often prone to poor covalent geometry on a local scale. Perhaps future methods will find a way to combine the best aspects of torsion angle search for small subchains with the best of Cartesian or distance-based methods for combining larger subchains.

6.2 Contributions of the Doctoral Candidate

The contributions in this thesis involving structure determination (the journal articles of Chapters 3 and 5) contain their own references, introductions, and summaries of contributions. Here I highlight my previously unpublished results on systematic conformational search.

I designed, implemented, and demonstrated the practical impact of four fundamental, interrelated, and truly novel improvements to the state of the art in systematic conformational search:

- The OMNIMERGE algorithm searches all possible subchains and **uses the best choice of left and right child subchains for each combine operation.** The greatest significance of OMNIMERGE may be that it makes possible many of the other innovations in this thesis. By itself it also can improve the search of molecules for which default methods get trapped using needlessly expensive components at the final merge of the search.

- The merge-tree **cost functions** evaluate alternatives for how large problems may be broken into subproblems. A simple application of dynamic programming to those

functions, in conjunction with results from any previous OMNIMERGE search of the same molecule, yield the **optimal merge-tree** for the molecule.

- PROPAGATION **enforces compatibility between overlapping subchains** by filtering out infeasible conformations from their solution sets.
- An **A* function** selects which subchain to search next based on current information about the sizes and costs of other subchains, and based on estimates for the relative contribution of each subchain towards the search of the whole molecule. Costly subchains may be skipped entirely but low-cost subchains (ones that could be part of an optimal merge-tree for the whole molecule) are never skipped.

With Tomás Lozano-Pérez, I demonstrated the utility of **the voxel model**, which allows efficient but non-systematic searches to be used locally within a systematic context.

In addition, I adapted systematic search methods to a real dataset. By searching the formyl-Met-Leu-Phe-OH peptide at an exceedingly high resolution, we provided unprecedented confidence in the interpretation of the constraints while determining a previously unknown atomic structure.

Appendix A

Systematic Search Implementation

A.1 Definitions and Preprocessing

Bonds

Each rotatable bond in a molecule defines a local reference frame, also called an internal coordinate system [9]. Atoms that have invariant relative positions (invariant over any rotation of torsion bonds) are defined to be in the same *bond frame* (also called an "aggregate" [5]). Figure A-1 shows the rotatable bonds (the torsions) of an amino acid residue

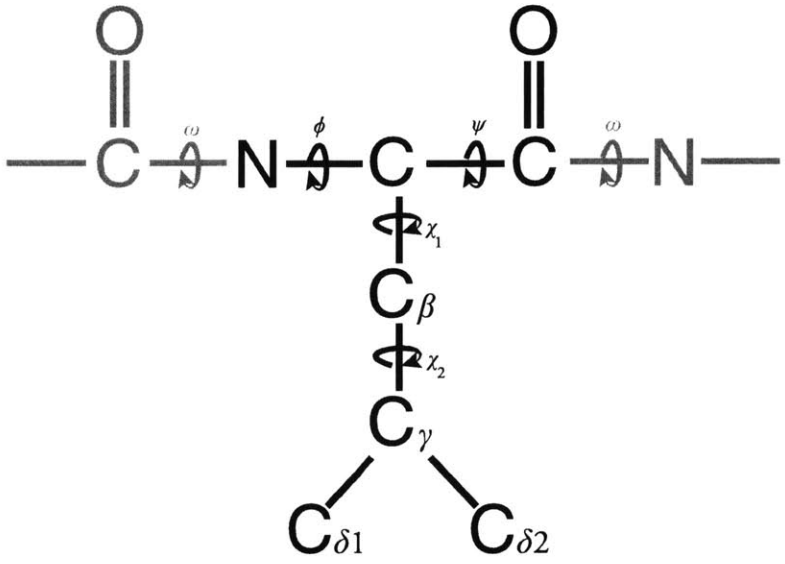

Figure A-1: The rotatable bonds in a leucine amino acid.

and Figure A-2 shows how the literal bonds can be converted into a tree datastructure of

bond frames. The default TREESEARCH strategy uses an ordering of the torsions $t_1, t_2 t_3 \ldots$

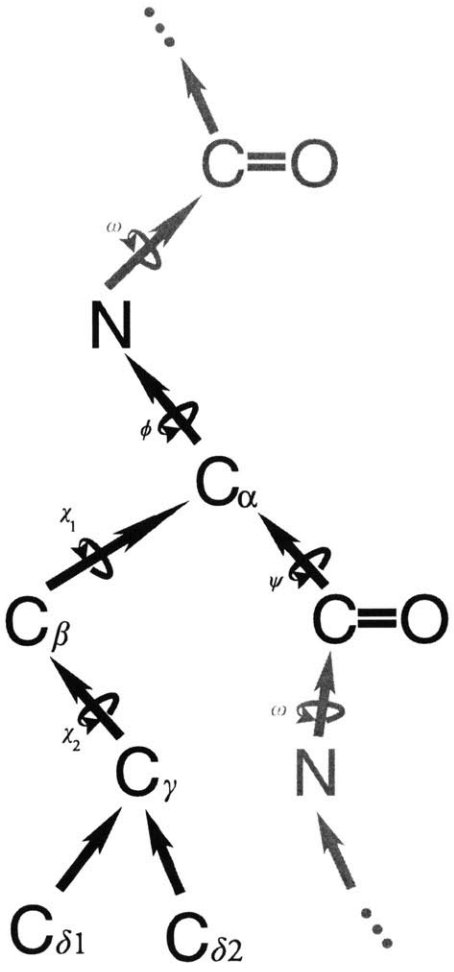

Figure A-2: A tree of bonds.

such that if t_i is the parent of t_j in the bond tree, then t_i occurs before t_j in the ordering. The parent of t_1 is the invariant coordinate frame; the parent of t_2 must be t_1; the parent of t_3 may be either t_1 or t_2, and so on.

The bond tree should not be confused with the search tree. The bond at the root of the bond tree is instantiated first, at the root of the search tree, and the bonds lower in the bond tree are searched later, lower in the search tree. However, a bond corresponds to a whole level of the search tree, but it is only one edge in the bond tree.

Atoms

Given the coordinate frame of the parent bond frame and the rotation of the bond connecting the child frame to the parent frame, one can compute the child coodinate frame. Bond frames and the torsion vectors defining their relative orientations are the primary representations used by our algorithm. Cartesian coordinates for atoms only exist for the duration of subroutines that evaluate a conformation on the basis of constraints on the atoms.

Atoms are specified as coordinates relative to their bond frame. Two torsions are *adjacent in the bond tree* if there are no *rotatable* bonds separating them. When adjacent torsions in the bond tree are instantiated, the atoms in their respective bond frames have an exact relative orientation, which permits calculation of the distance between atoms in the two frames. For atoms in two arbitrary bond frames, ALL intervening torsions must be instantiated before the relative orientation of the frames can be computed and before the relative positions of the atoms can be calculated.

A.2 Options and Optimizations

Atom hash

Another opportunity to improve efficiency is in the check for interatomic distances that violate the user's distance constraints or the van der Waals constraints. Evaluating the constraint violations of a conformation is part of the inner loop of the search algorithm. Therefore, efficiency gains in this stage can have a significant impact on total runtime, even when the non-optimized algorithm is polynomial. Contrary to popular opinion, the check of interatomic distances need not consume quadratic time. We can exploit the short-range nature of most distance constraints to design a procedure that is more efficient than checking all pairs of atoms.

Van der Waals constraints, a special category of distance constraints, are derived from the chemical structure of the molecule rather than input by the user. They prohibit any atom from entering the volume of another atom, unless the two atoms are covalently bonded or meet other chemical criteria. Van de Waals constraints, although numerous, only concern atoms in close proximity.

The user-defined distance constraints could theoretically require two atoms to be separated by a large distance, but more often they tend to require atoms to be close to each

other.

Note that distance violation checks are performed only after a particular vector of torsions angles has been chosen (for example, the midpoint of a voxel). Since the torsions define a unique conformation, absolute three-dimensional coordinates are available for each atom.

Rather than computing the distances between all pairs of atoms and checking those distances against the various constraints, we can divide the 3-dimensional space of atomic coordinates into small cubes (like the way we divide torsion space into small voxels), and hash the atoms according to their cubes. (So far this only requires time linear in the number of atoms.) Then we can iterate through the non-empty hash cubes instead of iterating through pairs of atoms or through the Cartesion space. Each hash cube will contain at most a constant number of atoms. For each hash cube, compute distances for all pairs of atoms that are in the same or in nearby cubes, and check only those distances against the constraints. This avoids considering the majority of atoms pairs, which are in distant cubes.

The number of occupied hash cubes grows with the size of the molecule, but the number of pairs of atoms grows with the square of the size of the molecule. Therefore, for large enough molecules, the hash is a more efficient way to check van der Waals and other short-range distance constraints.

For each user-defined constraint involving a long interatomic distance (such as if two atoms were required to be at least 20 angstroms apart), that pair of atoms would have to be considered explicitly, in addition to the iteration over hash squares. But as long as the number of constraints requiring long interatomic distances is much smaller than the number of pairs of atoms, it will be more efficient to check for short-distance violations using the hash and then for the remaining constraints separately than to check all pairs of atoms.

Minimization options

We provide a variety of extra options for "tweaking" the minimizer because the minimization is so important. However, we have not yet determined what combinations of options is optimal, for this algorithm or in general.

A popular heuristic for augmenting local minimization methods is to repeat the minimization several times using randomized initial values. We allow the user to specify the number of such restarts, called *passes*. There are a variety of methods for choosing initial

points (starting torsion values) for the minimization, including complete randomization, partial randomization (often called a "random kick"), or reuse of angle values found during lower dimensional-searches). In the extreme case where a user chooses many passes with totally random initial values, but limits the minimization to zero iterations (beyond the evaluation of the initial values), our method to evaluate voxels becomes identical to random sampling.

The user may also choose for the number of passes to be a function of the number of dimensions of the voxel. The number of passes can be quadratic or exponential in the number of dimensions. These options for additional sampling were designed in response to the greater occurrence of false negatives in higher-dimensional voxels.

Distances without Square Roots

We provide the option of using squared distances and squared bounds throughout the method (including in the distance matrix) rather than true distances, so that time-consuming square root operations may be avoided. Let b be a bound on the distance between two atoms, and let d be a measured distance between those atoms that has been determined to violate the bound. For simplicity of notation, assume b is an upper bound. Then, the magnitude of the violation, v, is approximated as follows:

$$\begin{aligned} v &= d - b \\ v_{approx} &= \frac{d^2 - b^2}{2b}. \end{aligned}$$

The above approximation for the violations of distance constraints is also used by the DIANA program for NMR interpretation [27].

Appendix B

Merge-Trees for 1RST

Below are figures showing various merge-trees for 1RST annotated with the TreeCost for each subtree in red. Each leaf subchain is annotated in blue with the number of conformations for the subchain, and each non-leaf subchain has two blue numbers, the number of candidate conformations (before the arrow) and the number of candidates found to satisfy the constrains (second).

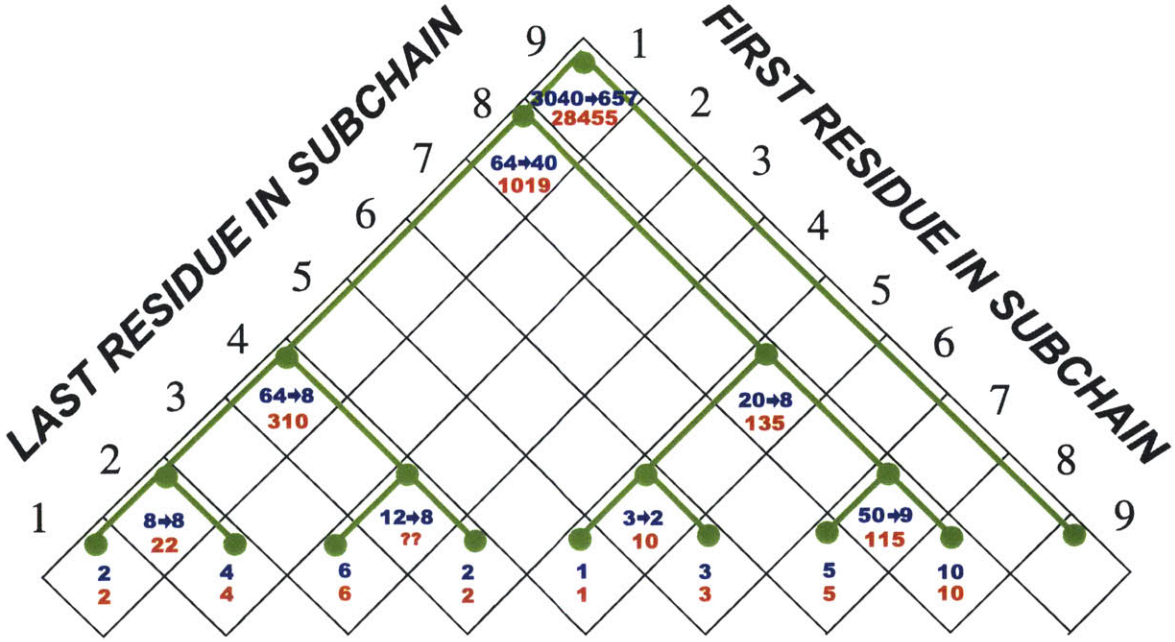

Figure B-1: Subchains in the default merge-tree of 1RST.

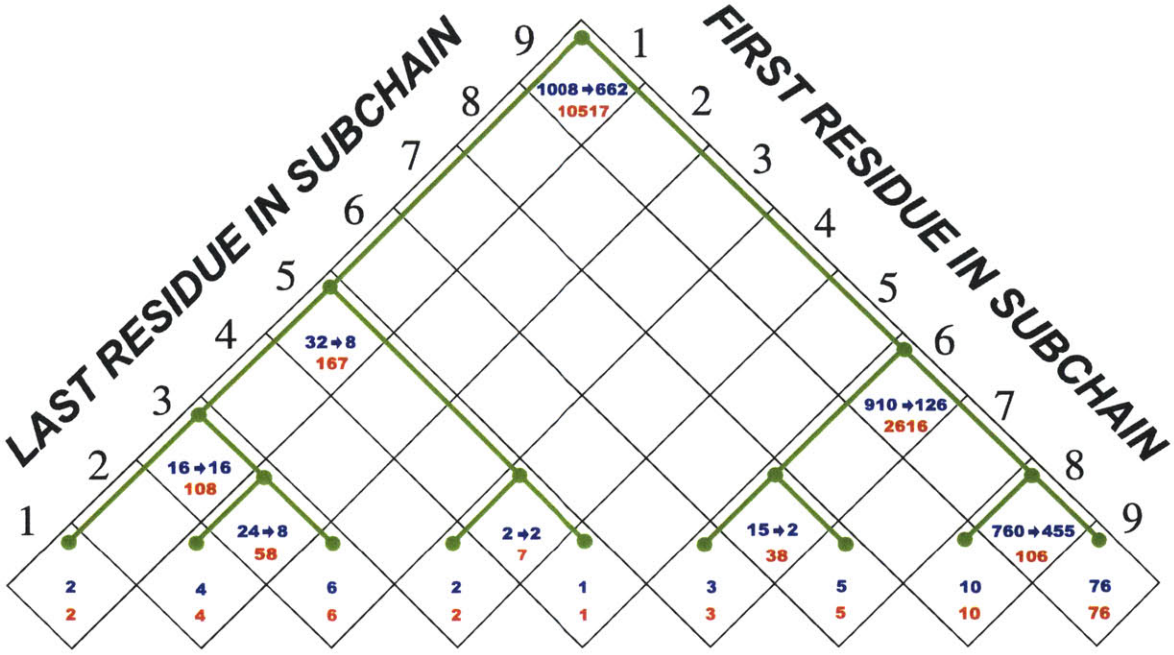

Figure B-2: Subchains in the manual merge-tree of 1RST

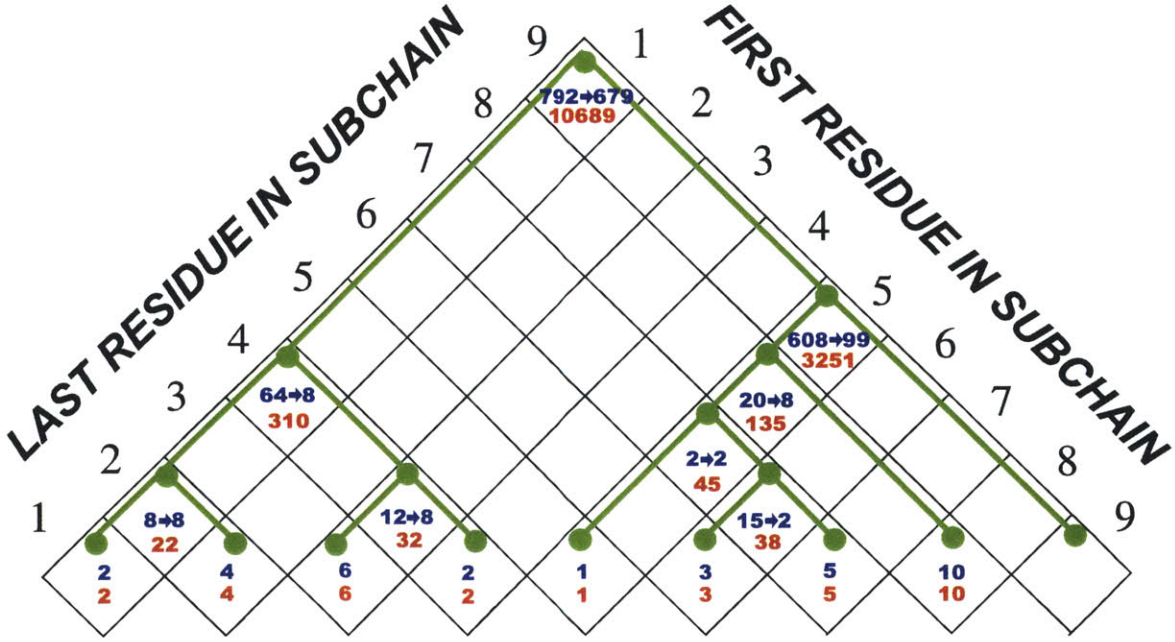

Figure B-3: Subchains in the merge-tree that was determined to be optimal for a low-resolution search of 1RST (120° resolution for all bonds).

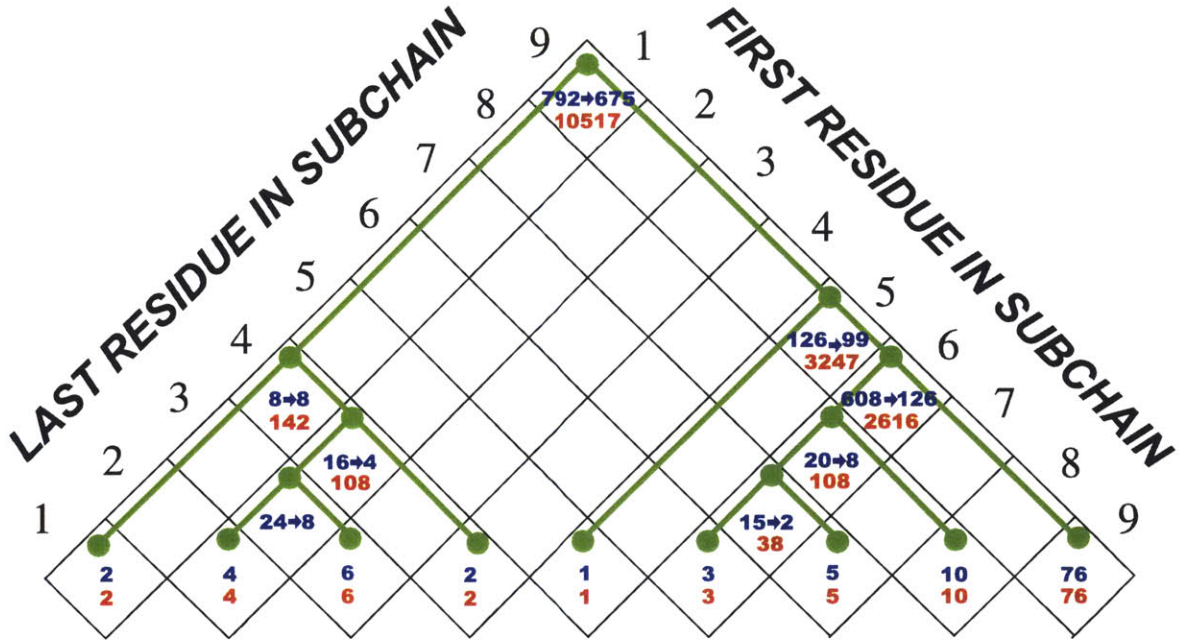

Figure B-4: Subchains in the merge-tree that was determined to be optimal for a moderate-resolution search of 1RST (40° resolution for backbone bonds and 120° resolution for sidechain bonds).

Bibliography

[1] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[2] H.M. Berman, J. Westbrook, Z. Feng *et al.*, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[3] D.D. Beusen, H. Iijima, and G.M. Marshall. Structures from NMR distance constraints. *Biochem. Pharm.*, 40:173–175, 1990.

[4] D.D. Beusen and E.F.B. Shands. Systematic search strategies in conformational analysis. *Drug Discovery Today*, 1(10):429–437, 1996.

[5] D.D. Beusen, E.F.B. Shands, S.F. Karasek, G.R. Marshall, and R.A. Dammkoehler. Systematic search in conformational analysis. *Journal of Molecular Structure (Theochem)*, 370:157–171, 1996.

[6] H. Bohm, G. Klebe, T. Lorenz, T. Mietzner, and L. Siggel. Different approaches to conformational analysis: A comparison of completeness, efficiency, and reliability based on the study of a nine-membered lactam. *Journal of Computational Chemistry*, 11(9):1021–1028, 1990.

[7] A.T. Brunger. Free R value: a novel statistical quantity for assessing the accuracy of crystal structures. *Nature*, 355:472–475, 1992.

[8] A.T. Brunger, P.D. Adams, G.M. Clore *et al.*, and G.L. Warren. Crystallography and NMR system (CNS): A new software system for macromolecular structure determination. *Acta Crystallographica D*, 54:905–921, 1998.

[9] G. Chang, W.C. Guida, and W.C. Still. An internal coordinate monte carlo method for searching conformational space. *Journal of American Chemical Society*, 111:4379–4386, 1989.

[10] F.E. Cohen and M.J.E. Sternberg. On the prediction of protein structure: The significance of the root-mean-square deviation. *J. Mol. Biol.*, 138:321–333, 1980.

[11] T.F. Coleman, D. Shalloway, and Z. Wu. A parallel build-up algorithm for global energy minimizations of molecular clusters using effective energy simulated annealing. Technical Report 130, Cornell University, 1993.

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

[13] J. Desmet, M. DeMaeyer, B. Hazes, and I. Lasters. The dead end elimination theorem and its use in protein side-chain positioning. *Nature*, 356:539–542, 1992.

[14] J.F. Doreleijers, J.A.C. Rullmann, and R. Kaptein. Quality assessment of NMR structures: a statistical survey. *Journal of Molecular Biology*, 281:149–164, 1998.

[15] J.-P. Doucet and J. Weber. *Computer-Aided Molecular Design: Theory and Applications*, chapter Knowledge-based prediction: model building from homology, pages 431–447. Academic Press, 1996.

[16] J.-P. Doucet and J. Weber. *Computer-Aided Molecular Design: Theory and Applications*, chapter 7.2 Exploring the conformational space, pages 206–220. Academic Press, 1996.

[17] M.J. Dudek, K. Ramnarayan, and J.W. Ponder. Protein structure prediction using a combination of sequence homology and global energy minimization: II. energy functions. *J. Comput. Chem.*, 19(5):548–573, 1998.

[18] P.L. Easthope and T.F. Havel. Computational experience with an algorithm for tetrangle inequality bound smoothing. *Bull. Math. Biol.*, 51:173–194, 1989.

[19] R.A. Engh and R. Huber. Accurate bond and angle parameters for x-ray protein structure refinement. *Acta Crystallographica, Section A*, A47:392–400, 1991.

[20] A. Fahmy and G. Wagner. Treedock: a tool for protein docking based on minimizing van der waals energies. *J Am Chem Soc*, 124(7):1241–1250, 2002.

[21] C.L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, pages 17–37, 1982.

[22] K.D. Gibson and H.A. Scheraga. Revised algorithms for the build-up procedure for predicting protein conformations by energy minimization. *Journal of Computational Chemistry*, 8:826–834, 1987.

[23] G.P. Gippert. *New Computational Methods for 3D NMR Data Analysis and Protein Structure Determination in High-Dimensional Internal Coordinate Space.* PhD thesis, Scripps Research Institute, La Jolla, California, 1995.

[24] G.P. Gippert, P.E. Wright, and D.A. Case. Distributed torsion angle grid search in high dimensions: a systematic approach to NMR structure determination. *J. Biomol. NMR*, 11(3):241–263, 1998.

[25] N. Gō and H.A. Scheraga. Ring closure and local conformational deformation of chain molecules. *Macromolecules*, 3:178–187, 1970.

[26] P. Güntert, M. Billeter, O. Ohlenschläer, L.R. Brown, and K. Wüthrich. Conformational analysis of protein and nucleic acid fragments with the new grid search algorithm found. *Journal of Biomolecular NMR*, 12:543–548, 1998.

[27] P. Güntert, W. Braun, and K. Wüthrich. Efficient computation of three-dimensional protein structures in solution from nuclear magnetic resonance data using the program diana and the supporting programs calibas, habas and glomsa. *J. Mol. Biol.*, 217:517–530, 1991.

[28] R. M. Haralick and L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.

[29] T.F. Havel. An evaluation of computational strategies for use in the determination of protein structure from distance constraints obtained by nuclear magnetic resonance. *Prog. Biophys. Molec. Biol.*, 56:43–78, 1991.

[30] J.B. Hendrickson. Molecular geometry. i. machine computation of the common rings. *J. Am. Chem. Soc.*, 83:4537–4547, 1961.

[31] P. Van Hentenryck. *Constraint Satisfaction in Logic Programming.* MIT Press, 1989.

[32] B.E. Hingerty, S. Figueroa, T.L. Hayden, and S. Broyde. Prediction of DNA structure from sequence: A buildup technique. *Biopolymers*, 28:1195–1222, 1989.

[33] L. Holm and C. Sander. Database algorithm for generating protein backbone and side-chain co-ordinates from a c alpha trace application to model building and detection of co-ordinate errors. *Journal of Molecular Biology*, 218(1):183–194, 1991.

[34] A.E. Howard and P.A. Kollman. An analysis of current methodologies for conformational searching of complex molecules. *Journal of Medicinal Chemistry*, 31(9):1669–1675, 1988.

[35] T.A. Jones and S. Thirup. Using known substructures in protein model building and crystallography. *EMBO Journal*, 5(4):819–822, 1986.

[36] S.R. Krystek Jr., D.A. Bassolino, R.E. Bruccoleri, J.T. Hunt, M.A. Porubcan, C.F. Wandler, and N.H. Andersen. Solution conformation of a cyclic pentapeptide endothelin antagonist. comparison of structures obtained from constrained dynamics and conformational search. *FEBS Lett.*, 299(3):255–261, 1992.

[37] D.E. Knuth. *The Art of Computer Programming, Second Edition*, volume 1, chapter 2.3.4.4 Enumeration of Trees, pages 385–389. Addison-Wesley, 1973.

[38] R.A. Laskowski, M.W. MacArthur, D.S. Moss, and J.M. Thornton. PROCHECK: a program to check the stereochemical quality of protein structures. *Journal of Applied Crystallography*, 26:283–291, 1993.

[39] R.A. Laskowski, J.A.C. Rullmann, M.W. MacArthur, R. Kaptein, and J.M. Thornton. AQUA and PROCHECK-NMR: Programs for checking the quality of protein structures solved by NMR. *Journal of Biomolecular NMR*, 8:477–486, 1996.

[40] I. Lasters and J. Desmet. The fuzzy-end elimination theorem: correctly implementing the side chain placement algorithm based on the dead-end elimination theorem. *Protein Engineering*, 6(7):717–722, 1993.

[41] I.M. Lasters, DeMaeyer, and J. Desmet. Enhanced dead-end elimination in the search for the global minimum energy conformation of a collection of protein side chains. *Protein Engineering*, 8(8):815–822, 1995.

[42] C. T. Lawrence, J. L. Zhou, and A. L. Tits. User's guide for CFSQP version 2.5: A C code for solving (large scale) constrained nonlinear (minimax) optimization problems, generating iterates satisfying all inequality constraints. Technical Report Technical Report TR-94-16r1, Institute for Systems Research, University of Maryland, College Park, MD 20742, 1997.

[43] A.R. Leach. *Molecular Modeling: Principles and Practice.* Addison–Wesley, 1996.

[44] A.R. Leach and A.P. Lemon. Exploring the conformational space of protein side chains using dead-end elimination and the A* algorithm. *PROTEINS: Struct. Funct. Genet.*, 33(2):227–239, 1998.

[45] J. Lee, H.A. Scheraga, and S. Rackovsky. New optimization method for conformational energy calculations on polypeptides: Conformational space annealing. *Journal of Computational Chemistry*, 18:1222–1232, 1997.

[46] H. Li, R. Tejero *et al.*, and G.T. Montelione. Homology modeling using simulated annealing of restrained molecular dynamics and conformational search calculations with CONGEN: application in predicting the three-dimensional structure of murine homeodomain Msx-1. *Protein Science*, 6(5):956–970, 1997.

[47] O. Lichtarge, C.W. Cornelius, B.G. Buchanan, and O. Jardetzky. Validation of the first step of the heuristic refinement method for the derivation of solution structures of proteins from NMR data. *PROTEINS: Struct. Funct. Genet.*, 2(4):340–358, 1987.

[48] M. Lipton and W.C. Still. The multiple minimum problem in molecular modeling. Tree searching internal coordinate conformational space. *Journal of Computational Chemistry*, 9(4):343–355, 1988.

[49] S.C. Lovell, J.M. Word, J.S. Richardson, and D.C. Richardson. The penultimate rotamer library. *PROTEINS: Struct. Funct. Genet.*, 40(3):389–408, 2000.

[50] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[51] A.K. Mackworth and E.C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–74, 1985.

[52] N. Majeux, M. Scarsi, J. Apostolakis, C. Ehrhardt, and A. Caflisch. Exhaustive docking of molecular fragments with elecrostatic solvation. *PROTEINS: Struct. Funct. Genet.*, 37:88–105, 1999.

[53] S. Makino and I.D. Kuntz. Automated flexible ligand docking method and its application for database search. *Journal of Computational Chemistry*, 18(14):1812–1825, 1997.

[54] M.J. McGregor, S.A. Islam, and M.J.E. Sternberg. Analysis of the relationship between side-chain conformation and secondary structure in globular proteins. *Journal of Molecular Biology*, 198:295–310, 1987.

[55] EU 3-D Validation Network. Who checks the checkers? Four validation tools applied to eight atomic resolution structures. *Journal of Molecular Biology*, 276:417–436, 1998.

[56] J.T. Ngo and M. Karplus. Pseudosystematic conformational search. Application to cycloheptadecane. *J. Am. Chem. Soc.*, 119:5657–5667, 1997.

[57] M.R. Pincus, R.D. Klausner, and H.A. Scheraga. Calculation of the three-dimensional structure of the membrane-bound portion of melittin from its amino acid sequence. *Proceedings of the National Academy of Sciences (USA)*, 79:5107–5110, 1982.

[58] J.W. Ponder and F.M. Richards. Tertiary templates for proteins: Use of packing criteria in the enumeration of allowed sequences for different structural classes. *Journal of Molecular Biology*, 193:775–791, 1987.

[59] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*, chapter 10.7 "Variable Metric Methods in Multidimensions", pages 324–328. Cambridge University Press, 1988.

[60] R.J. Read and J. Moult. Fitting electron density by systematic search. *Acta Crystallographica A*, 48:104–113, 1992.

[61] M. Saunders. *Encyclopedia of Computational Chemistry*, volume Volume 4, chapter Systematic and Random Search Methods for Finding Conformers of Molecules, pages 2948–2950. John Wiley and Sons, 1998.

[62] M. Saunders, K.N. Houk, Y.-D. Wu, W.C. Still, M. Lipton, G. Chang, and W.C. Guida. Conformations of cycloheptadecane. a comparison of methods for conformational searching. *Journal of the American Chemical Society*, 112:1419–1427, 1990.

[63] K.J. Schleifer, E. Tot, and H.D. Holtje. Pharmacophore and pseudoreceptor modelling of class ib antiarrhythmic and local anaesthetic lidocaine analogues. *Pharmazie*, 53(9):593–602, 1998.

[64] T.G. Schmidt, J. Koepke, R. Frank, and A. Skerra. Molecular interaction between the strep-tag affinity peptide and its cognate target, streptavidin. *Journal of Molecular Biology*, 255:753–766, 1996.

[65] A. Smellie, S.D. Kahn, and S.L. Teigg. Analysis of conformational coverage. 1. validation and estimation of coverage. *J. Chem. Inf. Comput. Sci.*, 35(2):285–294, 1995.

[66] G.M. Smith and D.F. Veber. Computer-aided, systematic search of peptide conformations constrained by NMR data. *Biochem. Biophys. Res. Commun.*, 134(2):907–914, 1986.

[67] Y. Takeuchi, E.F.B. Shands, D.D. Beusen, and G.R. Marshall. Derivation of a three-dimensional pharmacophore model of substance p antagonists bound to the neurokinin-1 receptor. *J. Med. Chem.*, 41(19):3609–3623, 1998.

[68] R. Tejero, D. Monleon, B. Celda, R. Powers, and G.T. Montelione. HYPER: A hierarchical algorithm for automatic determination of protein dihedral-angle constraints and stereospecific $C^{\beta}H2$ resonance assignments from NMR data. *Journal of Biomolecular NMR*, 15:251–264, 1999.

[69] W. Tong, E.R. Collantes, W.J. Welsh, B.A. Berglund, and A.C. Howlett. Derivation of a pharmacophore model for anandamide using constrained conformational searching and comparative molecular field analysis. *J. Med. Chem.*, 41(22):4207–4215, 1998.

[70] E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[71] M. Vásquez and H.A. Scheraga. Use of buildup and energy-minimization procedures to compute low-energy structures of the backbone of enkephalin. *Biopolymers*, 24(8):1437–1447, 1985.

[72] G. Vriend. WHAT IF: a molecular modeling and drug design program. *Journal of Molecular Graphics*, 8(1):52–56, 1990.

[73] D. Waltz. *The Psychology of Computer Vision*, chapter Understanding line drawings of scenes with shadows, pages 19–91. MIT Press, Cambridge, Massachusetts., 1975. edited by P.H. Winston.

[74] D.L. Waltz. *Generating semantic descriptions from drawing of scenes with shadows.* PhD thesis, Massachusetts Institute of Technology, 1972.

[75] C.-S. Wang. Efficient algorithm for conformational search of macrocyclic molecules. *Journal of Computational Chemistry*, 18(2):277–289, 1997.

[76] C.E. Wang. Confmatch: automating electron-density map interpretation by matching conformations. *Acta Crystallogr D Biol Crystallogr*, 12:1591–1611, 2000.

[77] J. Wang, P.A. Kollman, and I.D. Kuntz. Flexible ligand docking: A multistep strategy approach. *Proteins: Structure, Function, and Genetics*, 36:1–19, 1999.

[78] M. Zacharias and H. Sklenar. Conformational analysis of single-base bulges in a-form DNA and RNA using a heirarchical approach and energetic evaluation with a continuum solvent model. *J. Mol. Biol.*, 289(2):261–275, 1999.

Previous Degrees and Awards

M.S. in Computer Science and Electrical Engineering, MIT, 1993.

Thesis: *A Local Rule Paradigm for the Self-Assembly of Icosahedral Viruses.*

Advisor: Daniel Kleitman

B.S., *Summa cum laude*, in mathematics and computer science, Yale University, 1990.

Thesis: *Efficient Algorithm for reassembly of DNA restriction fragments.*

Advisor: Richard Beigel.

National Science Foundation Graduate Fellowship, 1991.

Anthony Stanley Prize in pure and applied mathematics, Yale University, 1990.

DeForest Prize in pure and applied mathematics, Yale University, 1990.