Determining Molecular Conformation from Distance or Density Data

by

Cheuk-san (Edward) Wang

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2000

Author . . . .
Department of Electrical Engineering and Computer Science
January 5, 2000

Certified by . . . .
Tomás Lozano-Pérez
Professor, Electrical Engineering and Computer Science
Thesis Supervisor

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

Determining Molecular Conformation from Distance or Density Data

by

Cheuk-san (Edward) Wang

Submitted to the Department of Electrical Engineering and Computer Science
on January 5, 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The determination of molecular structures is of growing importance in modern chemistry and biology. This thesis presents two practical, systematic algorithms for two structure determination problems. Both algorithms are branch-and-bound techniques adapted to their respective domains.

The first problem is the determination of structures of multimers given rigid monomer structures and (potentially ambiguous) intermolecular distance measurements. In other words, we need to find the the transformations to produce the packing interfaces. A substantial difficulty results from ambiguities in assigning intermolecular distance measurements (from NMR, for example) to particular intermolecular interfaces in the structure. We present a rapid and efficient method to simultaneously solve the packing and the assignment problems. The algorithm, *AmbiPack*, uses a hierarchical division of the search space and the branch-and-bound algorithm to eliminate infeasible regions of the space and focus on the remaining space. The algorithm presented is guaranteed to find all solutions to a pre-determined resolution.

The second problem is building a protein model from the initial three dimensional electron density distribution (density map) from X-ray crystallography. This problem is computationally challenging because proteins are extremely flexible. Our algorithm, *ConfMatch*, solves this "map interpretation" problem by matching a detailed conformation of the molecule to the density map (conformational matching). This "best match" structure is defined as one which maximizes the sum of the density at atom positions. The most important idea of ConfMatch is an efficient method for computing accurate bounds for branch-and-bound search. ConfMatch relaxes the conformational matching problem, a problem which can only be solved in exponential time (NP-hard), into one which can be solved in polynomial time. The solution to the relaxed problem is a guaranteed upper bound for the conformational matching problem. In most empirical cases, these bounds are accurate enough to prune the search space dramatically, enabling ConfMatch to solve structures with more than 100 free dihedral angles.

Thesis Supervisor: Tomás Lozano-Pérez
Title: Professor, Electrical Engineering and Computer Science

Acknowledgments

I have benefited greatly from the advise of my thesis supervisor, Tomás Lozano-Pérez. I truely appreciate the freedom and support he has given me through all my years at MIT. I also thank Bruce Tidor for his help and comments, especially for the work in Chapter 2. Many other people have helped me during my thesis research. In particular: Paul Viola, Oded Maron, Lisa Tucker Kellogg, J. P. Mellor, William M. Wells III, and all current and former members of the Tidor lab.

I am grateful to the Merck/MIT graduate fellowship for supporting my thesis research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The determination of molecular structures is of growing importance in modern chemistry and biology. Advances in biophysical techniques result in data on more ambitious structures being generated at an ever increasing rate. Yet, finding structures that are consistent with this data presents a number of formidable computational problems. A structure determination problem usually has many degrees of freedom and hence an extremely large state space. Traditionally, scientists use stochastic algorithms such as simulated annealing [22] or genetic algorithm [16] to solve these problems. They usually define an objective function that scores structures more consistent with data lower than those less consistent. This problem space usually has a large number of local minima. The goal is either to find the global minimum or enumerate all states where the objective function is below a given threshold. The stochastic methods then operates to minimize the objective function. The advantage of this approach is that one can obtain an approximate solution efficiently because only a small portion of the state space is explored. The disadvantage is that there is no guarantee that the result is the global minimum. In practice, the stochastic methods often produce local minima solutions or fail to enumerate all consistent solutions.

Systematic methods are fundamentally different from stochastic ones. Systematic algorithms examine the entire state space. They can therefore guarantee to find the globally optimal solution. Unfortunately, efficient systematic algorithms for structure determination problems are hard to develop because they require searching very large

	AmbiPack	ConfMatch
Application area	NMR	X-ray crystallography
Experimental data	Ambiguous distance constraints	Electron density distribution
Input	Two rigid substructures	Amino acid sequence
Output	All satisfying configurations of substructures	A single best-matching conformation
Degrees of freedom	3 rotational, 3 translational	The number of free dihedral angles plus 6 rigid degrees of freedom

Table 1.1: Comparison of the AmbiPack and ConfMatch algorithms.

state spaces. Branch-and-bound is a technique well suited to this type of problem. The challenges in applying branch-and-bound are intelligent formulation of the problems and accurate heuristics for searching. Without a good formulation or heuristics, the search is intractable even for simple problems. On the other hand, if there is an accurate set of bounds, one can avoid most of the search space and reach the solution efficiently.

This thesis presents two practical, systematic algorithms for two structure determination problems. Table 1.1 summarizes the differences between these two algorithms.

1.1 AmbiPack: Packing with Ambiguous Constraints

The first problem is the determination of structures of multimers. The structure(s) of the individual monomers must be found and the transformations to produce the packing interfaces must be described. A substantial difficulty results from ambiguities in assigning intermolecular distance measurements (from NMR, for example) to particular intermolecular interfaces in the structure. Chapter 2 presents a rapid and efficient method to simultaneously solve the packing and the assignment problems given rigid monomer structures and (potentially ambiguous) intermolecular distance measurements. A potential application of this algorithm is to couple it with a monomer searching protocol such that each structure for the monomer that is consistent with *in-*

tramolecular constraints can be subsequently input to the current algorithm to check whether it is consistent with (potentially ambiguous) *intermolecular* constraints. The algorithm, *AmbiPack*, finds a set of transforms of rigid substructures satisfying the experimental constraints. Though the problem domain has a modest 6 degrees of freedom, the computational challenge lies in the ambiguity of the constraints. Each input constraint has 2 possible interpretations (Figure 2-1). Thus the total number of interpretations is exponential in the number of constraints. These large number of interpretations present a formidable obstacle to finding the right structures. AmbiPack uses a hierarchical division of the search space and the branch-and-bound algorithm to eliminate infeasible regions of the space. Local search methods are then focused on the remaining space. The algorithm generally runs faster as more constraints are included because more regions of the search space can be eliminated. This is not the case for other methods, for which additional constraints increase the complexity of the search space. The algorithm presented is guaranteed to find all solutions to a pre-determined resolution. This resolution can be chosen arbitrarily to produce outputs at various level of detail.

1.2 ConfMatch: Matching Conformations to Electron Density

The second problem is building a protein model from the initial three dimensional electron density distribution (density map) from X-ray crystallography (Figure 3-2). This task is an important step in solving an X-ray structure. The problem is computationally challenging because proteins are extremely flexible. A typical protein may have several hundred degrees of freedom (free dihedral angles). The space of possible conformations is astronomical. Chapter 3 presents an algorithm, *ConfMatch*, that solves this "map interpretation" problem by matching a detailed conformation of the molecule to the density map (conformational matching). This "best match" structure is defined as one which maximizes the sum of the density at atom positions. ConfMatch quantizes the continuous conformational space into a large set of discrete conformations and finds the best solution within this discrete set by branch-and-bound search. Because ConfMatch samples the conformational space very finely, its solution is usually very close to the globally optimal conformation. The output of ConfMatch, a chemically feasible conformation, is both detailed and high quality. It is detailed because it includes all non-hydrogen atoms of the target molecule. It is high quality because the conformation satisfies various commonly-accepted chemical constraints such as bond lengths, angles, chirality, etc.

To find the "best match" structure, it is necessary to systematically explore a search tree exponential in the number of degrees of freedom. The most important idea of ConfMatch is an efficient method for computing accurate bounds. ConfMatch relaxes the conformational matching problem, a problem which can only be solved exactly in exponential time, into one which can be solved in polynomial time. The relaxed problem retains all local constraints of conformational matching, but ignores all non-local ones. The solution to the relaxed problem is a guaranteed upper bound for the conformational matching problem. When the input data has sufficiently good quality, the local constraints can lead to accurate bounds. In most empirical cases, these bounds are accurate enough to prune the search space dramatically, enabling ConfMatch to solve structures with more than 100 free dihedral angles.

In addition to solving the "map interpretation" problem, ConfMatch is potentially applicable to rational drug design. Instead of fitting a single structure to the electron density, ConfMatch can be adapted to fit a family of structures simultaneously to any kind of stationary field (Section 3.4.5). The result would be the best-fit structure within the family. For example, one may calculate the optimal electrostatic field for binding of a disease causing protein [25]. This optimal field specifies the electrostatic charges at different regions of space potentially occupied by a ligand. The space may be partitioned into regions charged positively, negatively, or neutrally. ConfMatch can at once fit all peptides of a certain length to this field. The output structure will give the best theoretical peptide ligand to this protein, as well as its most favored conformation.

Though the problem domains of AmbiPack and ConfMatch are quite different, both algorithms are systematic techniques based on branch-and-bound searches. They are the first systematic techniques in their respective areas. AmbiPack and ConfMatch are described in Chapters 2 and 3 of the thesis respectively. Chapter 4 summarizes the lessons learned while developing these algorithms.

Chapter 2

AmbiPack: Packing with Ambiguous Constraints

2.1 Introduction

The determination of structures of multimers presents interesting new challenges. The structure(s) of the individual monomers must be found and the transformations to produce the packing interfaces must be described. This chapter presents an efficient, systematic algorithm for the second half of the multimer problem: finding packings of rigid predetermined subunit structures that are consistent with ambiguous intermolecular distance measurements from NMR experiments.

Whereas X-ray crystallography essentially provides atomic-level information in *absolute* coordinates, NMR spectroscopy provides *relative* distance and orientation information through chemical shifts, coupling constants, and especially distances estimated from magnetization transfer experiments. In NMR spectroscopy, the identity of an atomic nucleus is indexed by its chemical shift (in 2D experiments) and also that of its neighbors (in higher dimensional experiments). Thus, two atoms that occupy exactly the same environment (e.g. symmetry-mates in a symmetric dimer) cannot generally be distinguished and distances measured to them can be ambiguous. For instance, in the symmetric dimer case, intra- and intermolecular distances are ambiguous. This type of ambiguity can generally be removed through isotopic labeling

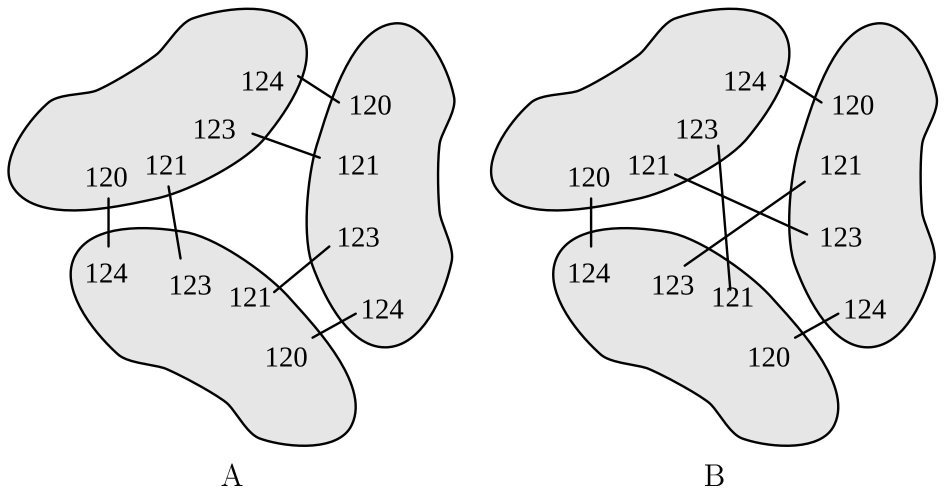


Figure 2-1: Two ambiguous intermolecular distances can have two different interpretations. The P22 tailspike protein is shown schematically with two different interpretations for the proximity pairs 120–124 and 121–123.

schemes. However, for higher-order multimers, where different types of intermolecular relationships exist, each intermolecular distance remains ambiguous. Furthermore, in solid-state NMR experiments [21], one can obtain unambiguous intramolecular distances but generally only ambiguous intermolecular distances. This kind of problem is evident with symmetric coiled coils [18], the trimeric P22 tailspike protein [39], and the fibrils formed from fragments of the Alzheimer precursor protein [21].

This type of ambiguity is illustrated in Figure 2-1 with the P22 tailspike protein, which forms a symmetric homotrimer. The intermolecular distances between residues 120 and 124 are short, as well as those between residues 121 and 123. Arrangement A assigns the intermolecular distances to the correct pairs of residues. Arrangement B differs from A by switching the assignment of residues 121 and 123. Many experimental techniques cannot distinguish between residues on different subunits. Thus A and B are both valid interpretations of the experimental data. For every intermolecular distance measurement, there are two such possible interpretations. When multiple ambiguous intermolecular distances are given, one has to solve the "assignment problem" — for each intermolecular distance, assign the residue pair to the correct subunits such that a structure can be generated to match all the distances.

One conceivable solution to the assignment problem is enumeration. One could attempt to enumerate all possible assignments and test each one by trying to generate a structure. Unfortunately, this is impractical in almost all cases. Consider that each intermolecular distance measurement may be assigned in two different ways to any pair of subunits and all combinations of assignments must be explored[1]. This means that, given n ambiguous intermolecular distances in a symmetric homomultimer, there are at least 2^{n-1} assignments. Furthermore, not all measurements need to hold between all pairs of subunits, that is, there may be more than one type of "interface" between the subunits of a homomultimer (see Section 2.4.2). This further increases the number of combinations that need to be explored. Since the number of assignments to be tested grows exponentially with the number of ambiguities, this approach is not feasible for realistic numbers of distances. For example, later we will be dealing with 43 ambiguous measurements for the P22 homotrimer. The size of this assignment problem is 2^{42}, which is approximately 4×10^{12}; this is clearly too many combinations to enumerate.

A different approach is to design a potential function that has the effect of performing a logical "OR" over the possible solutions for the ambiguous constraints. For example, this function can be a sum of terms reflecting a penalty for unsatisfied distance measurements. Each term can contribute zero when the corresponding distance is satisfied in any way consistent with its labeling ambiguity. The penalty function may increase monotonically with the magnitude of the distance violation so that global optimization techniques, such as simulated annealing, may be utilized to search for solutions. If multiple packed structures exist that are consistent with the measurements, there would be many minima with a zero potential. Nilges' *dynamic assignment* strategy [31, 32] uses a smooth function with these properties for ambiguous inter- and intramolecular distances. Dynamic assignment has the significant advantage of not assuming a rigid monomer. Instead, the monomer is assumed to be flexible and restrained by intramolecular distances. O'Donoghue et al. [33] successfully applied this technique to the leucine zipper homodimers, where the monomer

[1]The first assignment can be made arbitrarily since all measurements are relative.

structure is known. Unfortunately, this approach must contend with the multiple local minima problem; there are many placements of the structures that satisfy only a subset of the distances but such that all displacements cause an increase in the potential. As the number of ambiguous distances increases, the minimization takes longer to find valid solutions, due to increasing ruggedness of the potential landscape. Furthermore, since this approach is a randomized one, it is not guaranteed to generate all packings satisfying the constraints. Likewise, if no structure can possibly match all distances, this method will not be able to prove that conclusively.

Yet another approach is to systematically sample rigid transformations, apply them to the subunit, and then test whether the resulting structures match all distances. Since a rigid transformation has six parameters (three translations and three rotations), one needs to test n^6 transforms where n is the number of samples for each transformation parameter. This will take a great deal of computer time even for a moderate size n, such as 30, since $30^6 = 729,000,000$). Furthermore, this approach may miss solutions that are "between" the sampled transformations [28, 6]. So, to have a fair degree of confidence that no solutions have been missed requires very fine sampling, that is, a large value of n (generally much greater than 30).

We have developed a new algorithm, AmbiPack, that generates packed structures from ambiguous (and unambiguous) intermolecular distances. AmbiPack is both exhaustive and efficient. It can find all possible packings, at a specified resolution, that can satisfy all the distance constraints. This resolution can be chosen by the user to produce packings at any level of detail. It gives a null answer *if and only if* there is no solution to the constraints. In our implementation, AmbiPack takes minutes to run[2] on a problem with more than forty ambiguous constraints. Its running time does not depend significantly on the size of the subunits. Furthermore, while most other techniques run slower when more constraints are added, AmbiPack generally runs *faster* with more constraints because this allows earlier pruning of a greater number of solutions and requires detailed exploration of a smaller number of solution. Therefore, it is quite practical to apply AmbiPack to a family of NMR-derived subunit

[2]All runs were on a Sun Ultra 1 workstation.

structures to obtain a corresponding family of packed structures. Moreover, it can be used in tandem with a subunit generating procedure (which satisfies *intrasubunit* distances) to filter out those subunit models incompatible with the set of *intersubunit* distances.

2.2 Problem Definition

We now define the packing problem more precisely; we will start by assuming only two structures and generalize the definition later.

2.2.1 Two Structures

The inputs to the AmbiPack algorithm are:

1. Two rigid structures (S and S') that are to be packed. Without loss of generality, we assume that S' is fixed in the input configuration and S has an unknown rigid transformation relative to S'. S is the same as S' for identical structures, which is frequently the case.

2. A set of constraints on the intermolecular distances. These constraints specify the allowed ranges of distances between atoms, e.g. 3 Å$<PQ'<$6 Å where P and Q' are atoms on S and S' respectively. The constraints can be specified *ambiguously*, i.e. only one of several bounds needs to be satisfied. Suppose P and Q are atoms on S while P' and Q' are correspondingly atoms on S'. One ambiguous constraint may be

$$(PQ' < 6 \text{ Å}) \text{ OR } (QP' < 6 \text{ Å}),$$

   which requires only one of the two distances to be shorter than 6 Å.

In principle, the input constraints to AmbiPack may have many possible forms; each constraint can be a boolean combination of an arbitrary number of inequalities which can put limits on any intermolecular distances. In practice, experiments usually generate two types of constraints, called *positives* and *negatives*. They correspond to positive and negative results from solution or solid-state NMR. A positive result means that a pair of atoms is closer than some distance bound. However, due to the labeling ambiguity present in current experiments of this variety, a positive constraint has the form ($PQ' < x$ Å) OR ($QP' < x$ Å), which has a two-fold ambiguity. The constraint also need not be satisfied at all between a given pair of monomers, which introduces additional ambiguity.

On the other hand, a negative experimental result means that a pair of atoms are farther apart than some bound. *All* such intermolecular pairs must satisfy the requirement. There are no ambiguous interpretations. A negative constraint has the form ($PQ' > x$ Å) AND ($QP' > x$ Å).

The output of AmbiPack is a set of rigid transformations. When any of the output transformations is applied to the structure S, the resulting complex with S' satisfies the specified constraints.

2.2.2 More Than Two Structures

The description above applies to structures with two subunits, but it can be extended to structures with more than two identical subunits. There are two classes of problems involving more than two structures, depending on whether all of the distance constraints hold at all interfaces among monomers or not.

The simpler case is when all of the ambiguous (positive) distance constraints hold at the interface between any pair of structures. In this situation, there is only one type of interface between pairs of monomers. This case is quite common; it is illustrated by the P22 tailspike trimer (Figure 2-1), which is treated in detail in Section 2.4.1. For such a symmetric trimer, in which there is two-fold ambiguity between all intermolecular constraints and each intermolecular constraint is satisfied at least once between each pair of monomers, the structure of the trimer can be constructed through successive application of an output transformation (T) to the input structure (S). That is,

$$S, T(S), T^2(S)$$

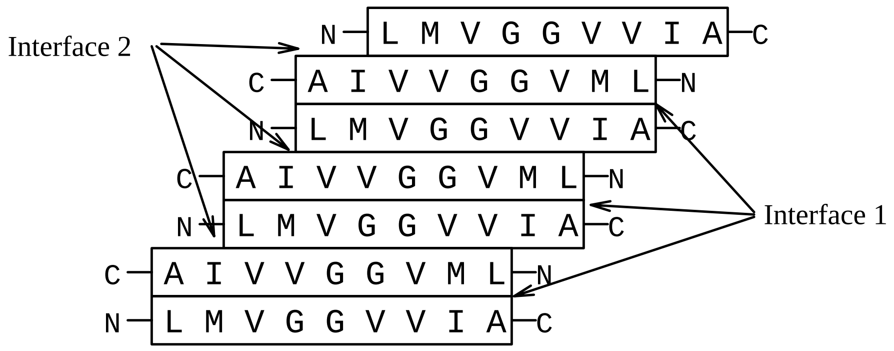


Figure 2-2: Structure of β amyloid fibril proposed by Lansbury et al. [21]

together form a candidate trimer packing. The constraints should also be satisfied across the $T^2(S) : S$ interface, which needs to be verified for each candidate T. A similar approach can be taken for symmetric homomultimers with m subunits, but only one type of interface.

The more complex case is when the positive distance constraints are not all satisfied between every pair of structures. Figure 2-2 shows the structure of a C-terminal peptide (β34-42) of the β amyloid protein (β1-42). [21] This infinitely repeating structure forms an ordered aggregate. There are two types of interfaces in this structure. Solid-state ^{1}3C NMR experiments have produced 8 positive and 12 negative constraints. Either interface satisfies all negative constraints but only a subset of the positive ones. Together the interfaces satisfy all positive constraints. A direct approach to this type of problem is to enumerate subsets of the constraints that may hold between different pairs of structures. AmbiPack can be used to find solutions for each of these subsets of constraints. A valid multimer can be constructed from combinations of output transformations, applied singly or successively, such that each constraint is satisfied at least once in the multimer. This is the strategy illustrated in Section 2.4.2. This strategy is only feasible when the number of ambiguous constraints is relatively small since the number of constraint subsets also grows exponentially with the number of ambiguous constraints. A more efficient variant of this strategy that exploits the details of the AmbiPack algorithm is discussed in Section 2.3.4.

2.3 The AmbiPack Algorithm

We now describe our approach to the packing problem — the AmbiPack algorithm. For ease of exposition, we first present a simplified version for solving unambiguous constraints, i.e. constraints without OR's, and a single interface, i.e. all the constraints hold between the given structures. In Section 2.3.4 we will generalize this description to ambiguous constraints and multiple interfaces.

2.3.1 Algorithm Overview

AmbiPack is based on two key observations:

1. Suppose there are some constraints of the form $PQ' < x$ Å, where P and Q' are atoms on S and S' respectively. This constraint specifies the approximate location of P. Specifically, it describes a sphere of radius x Å around Q' in which P must be found.

2. If we fix the positions of three non-collinear atoms on S, we have specified a unique rigid transformation.

These observations suggest that one may approach the problem of finding a packing consistent with a given set of (unambiguous) input constraints as follows:

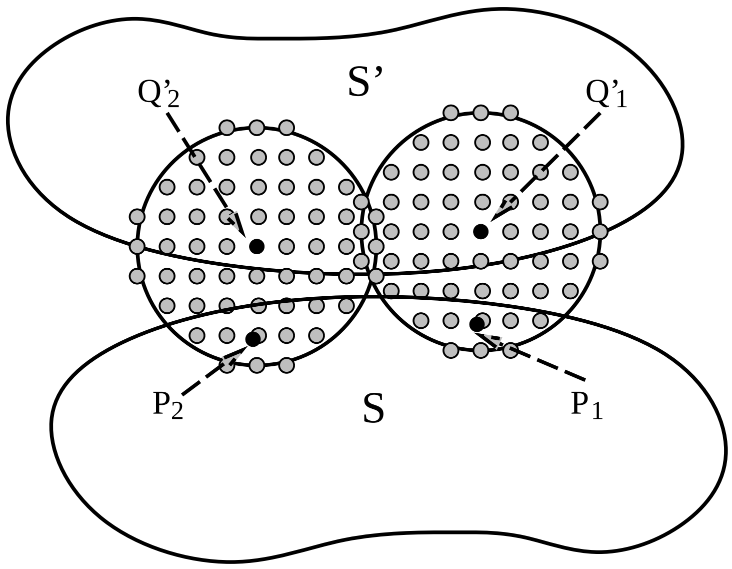
1. Select three (unambiguous) constraints ($P_iQ'_i < x_i$ Å, $i = 1, 2, 3$) from the input set.

2. For each P_i, uniformly sample its possible positions inside the sphere with radius x_i Å centered on Q'_i.

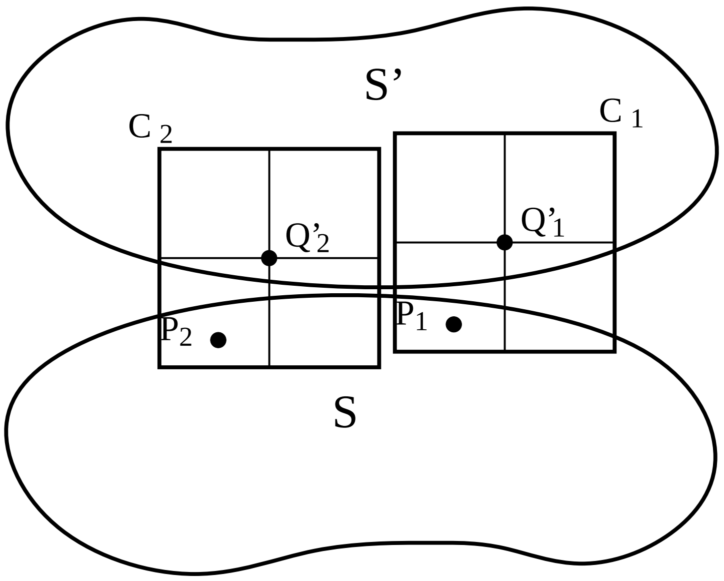
3. Calculate rigid transformations based on the positions of P_i's. Test whether these transformations satisfy all the input constraints.

A two dimensional example of this approach is shown in Figure 2-3 A.

Note, however, that this approach is *not* guaranteed to find a legal packing whenever one exists. In particular, it misses solutions that would require placing any of



A



B

Figure 2-3: Two approaches to generating transforms (illustrated here in two dimensions, where two points are sufficient to place a structure): (A) matching points P_i from S to sampled points in spheres centered on Q'_i, or (B) placing points P_i from S somewhere within cubes contained in spheres centered on Q'_i. The first of these may miss solutions that require placing the P_i away from the sampled points.

the P_i away from the sampled points. Of course, by sampling very finely we can reduce the chances of such a failure, but this remedy would exacerbate the two other problems with this approach as it stands. One problem is that the method needs to test m^3 transformations where m is the number of sampled points in each of the three spheres. Typically we would sample hundreds of points in each sphere and thus millions of transformations are to be generated and tested. The other, related problem is that the finer the sampling, the greater the number of transformations, many nearly identical, that will be produced, since the constraints seldom define a single solution exactly. To alleviate these latter problems, we want a relatively coarse sampling.

AmbiPack is similar to the method above but instead of sampling points at a fixed spacing within the spheres, AmbiPack explores the possible placements of the P_i within the spheres in a coarse-to-fine fashion. To achieve the advantages of exploration using coarse sampling while maintaining a guarantee of not missing solutions, we must replace the idea of sampling points with that of subdividing the space. Consider placing the P_i not at fixed points within the spheres but rather *somewhere* inside (large) cubes centered on the sampled points (Figure 2-3 B). We can now pose the following question: "Can we *disprove* that there exists a solution in which the P_i are inside the chosen cubes?" If we can, then this combination of cubes can be discarded; no combination of points within these cubes can lead to a solution. If we cannot disprove that a solution exists, we can subdivide the cubes into smaller cubes and try again. Eventually, we can stop when the cubes become small enough. Each of the surviving assignments of points to cubes represents a family of possible solutions that we have not been able to rule out. Each of these potential solutions is different from every other in the sense that that at least one of the P_i's is in a different cube. We can then check, by sampling transformations or by gradient-based minimization, which of these possible solutions actually satisfy all the input constraints.

The key to the efficiency of the algorithm, obtained without sacrificing exhaustiveness, is the ability to disprove that a solution exists when the three P_i are placed anywhere within the three given cubes, C_i. Since the P_i's are restricted to the cubes, the possible locations of other S atoms are also limited. If one can conservatively

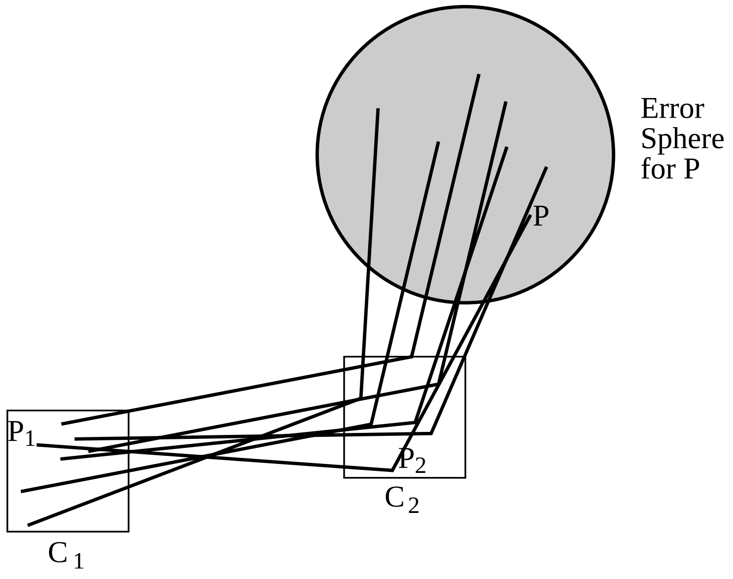


Figure 2-4: The error spheres for points in S when the P_i are constrained to be somewhere within cubes contained in spheres centered on Q_i'. Illustrated here in two dimensions.

bound the locations of other atoms, one can use the input constraints to disprove that a solution can exist. AmbiPack uses *error spheres* to perform this bounding. For each atom on S, its error sphere includes all of its possible positions given that the P_i's lie in C_i's (Figure 2-4). The details of the error sphere computation are given in Sections 2.3.2 and 2.3.2.

Up to this point we have not dealt with ambiguous constraints. However, we only need to modify the algorithm slightly to deal with them. Note that once we have a candidate transformation, checking whether ambiguous constraints are satisfied is no more difficult than checking unambiguous constraints; it simply requires dealing with constraints including OR's as well as AND's. So, the only potential difficulty is if we cannot select an initial set of three unambiguous constraints in Step 1 of the algorithm. If the constraints are ambiguous, we cannot tell whether the atoms referred to in the constraints are drawn from S or S'. In that case, however, we can enumerate the possible interpretations of the ambiguous constraints and find the solutions for each one. Assuming that all the distance constraints hold between the given structures and since we are dealing with at most three ambiguous measurements, this generally involves a small number of iterations of the algorithm.

2.3.2 Algorithm Details

AmbiPack is an example of a branch-and-bound tree-search algorithm [1]. During the search, it prunes away branches that are ruled out by the bound function. Figure 2-5 illustrates the algorithm. Initially, AmbiPack selects three constraints, $P_iQ'_i < x_i$ Å, $i = 1, 2, 3$. Each node in the search tree corresponds to three cubes in space — C_1, C_2, and C_3; the position of P_i is limited to be inside C_i. At the root of the tree, each C_i is centered at Q'_i and has length $2x_i$ in each dimension. Thus, all possible positions of P_i's satisfying the constraints are covered. At every internal node, each C_i is subdivided into 8 cubes with half the length on each side. Each child has three cubes 1/8 the volume of its parent's. Each parent has 512 children because there are $8^3 = 512$ combinations of the smaller cubes. At each level further down the search tree, the positions of P_i's are specified at progressively finer resolution. If one calculates transformations from all nodes at a certain level of the tree, one systematically samples all possible solutions to the packing problem at that level's resolution. The method of computing a transformation for a node is described below.

The very large branching factor (512) of the search tree means that an effective method for discarding (*pruning*) solutionless branches is required. Otherwise the number of nodes to be considered will grow quickly — 512^d, where d is the depth of the tree — precluding exploration at fine resolution. AmbiPack uses two techniques to rule out branches that cannot possibly satisfy all input constraints.

The first technique is to exploit the known distances between the P_i, since the monomers are predetermined structures. Between any pair of atoms in S, the distance is fixed because S is a rigid structure. Suppose C_1, C_2 and C_3 are the cubes corresponding to a tree node. Let $\max(C_1, C_2)$ and $\min(C_1, C_2)$ be the maximum and minimum separation, respectively, between any point in C_1 and any point in C_2. A

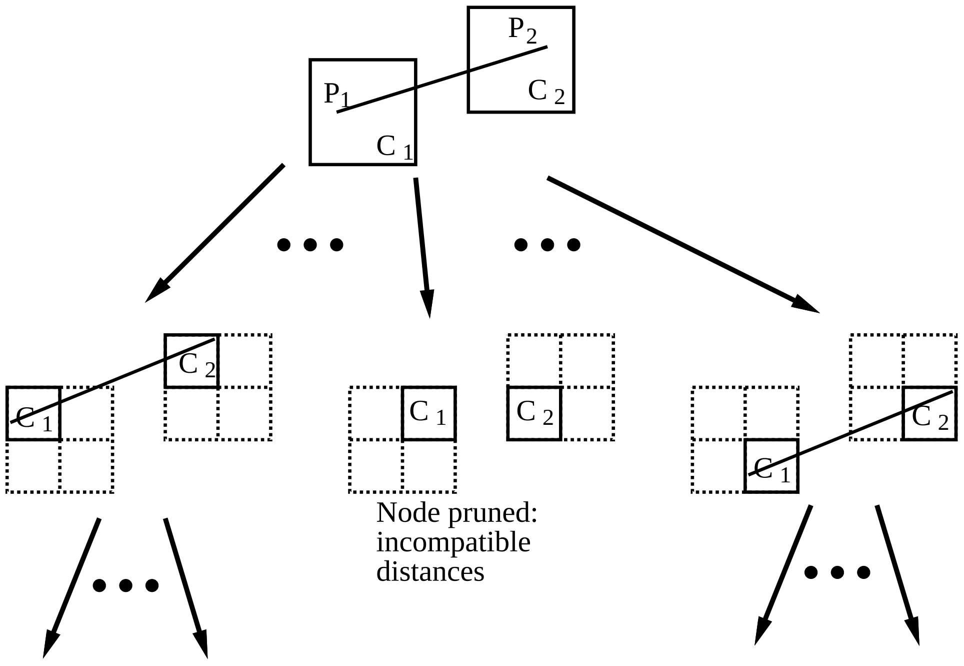


Figure 2-5: The AmbiPack algorithm explores a tree of assignments of the three P_i to three cubes C_i. Illustrated here in two dimensions.

necessary condition for P_1 to be in C_1 and P_2 to be in C_2 is

$$\min(C_1C_2) \leq P_1P_2 \leq \max(C_1C_2).$$

Similarly, for the other two pairs of atoms, we require $\min(C_2C_3) \leq P_2P_3 \leq \max(C_2C_3)$ and $\min(C_1C_3) \leq P_1P_3 \leq \max(C_1C_3)$. If any of the three conditions are violated, the node can be rejected; since the P_i's cannot be simultaneously placed in these cubes.

The second pruning technique makes use of the error spheres mentioned in Section 2.3.1. For each atom on S, its error sphere includes all of its possible positions given that the P_i's lie in the C_i's. Let E and r be the center and radius, respectively, of the error sphere of an atom located at P on S (the computation of E and r is discussed below). Since we want to discard nodes which cannot lead to a valid solution, we want to ensure that no possible position of P (the points within the error sphere) can satisfy the constraints. We can do this by replacing all input constraints on P with constraints on E (the center of the error sphere), with the constraints "loosened" by the error sphere radius r. Suppose $PQ' < x$ is an input constraint. This pruning technique requires that $EQ' < x + r$. Similarly, $PQ' > x$ will translate into $EQ' > x - r$. Given these loosened constraints, we can implement a conservative method for discarding nodes. We can compute one transformation that maps the P_i so that they lie in the C_i; any transformation that does this will suffice. We can then apply this transform to the centers of the error spheres for all the atoms of S. If any of the input constraints fail, when tested with these transformed error sphere centers and loosened by the error sphere radii, then the node can be discarded. Note that if there are more constraints, this technique will impose more conditions; thus more nodes will be rejected. This is why AmbiPack is more efficient if more constraints are given.

The key remaining problem is efficiently finding the centers and radii of error spheres for the specified P_i's and C_i's.

Centers of error spheres

We want to make the error spheres as small as possible, since this will give us the tightest constraints and best pruning. We can think of the center of the error sphere as defined by some "nominal" alignment of the P_i to points with the C_i. The points within the error sphere are swept out as the P_i are displaced to reach every point within the C_i. The magnitude of the displacement from the error sphere center depends on the magnitude of the displacement from the "nominal" alignment. This suggests that we can keep the error sphere small by choosing a "nominal" alignment that keeps the displacement of the P_i needed to reach every point in C_i as small as possible. That is, we want the "nominal" alignment to place the P_i close to the centers of the C_i.

We find the centers of the error spheres by calculating a transformation, T, that places the P_i's as close as possible to the centers of the C_i's. For every atom P in S, $T(P)$ is taken as the center of its error sphere. There are many well-known iterative algorithms for computing transformations that minimize the sum of distances squared between two sets of points, e.g. [13]. However, in our case, since we are dealing with only three pairs of points, we can use a more efficient analytic solution.

The points P_i define a triangle and so do the centers of the C_i. Therefore, we are looking for a transformation that best matches two rigid triangles in three-dimensional space. It should be clear that the best (minimal squared distance) solution has the triangles coplanar, with their centroids coincident. Suppose these two conditions are met by two triangles $x_1x_2x_3$ and $y_1y_2y_3$ whose centroids are at the origin. x_i's and y_i's are vectors and each x_i is to match y_i. Let the y_i's be fixed but x_i's be movable. The only unknown parameter is θ, the angle of rotation of x_i's on the triangles' plane about the origin (Figure 2-6). The optimal θ can be found by writing the positions of the x_i's as a function of θ, substituting in the expression for the the sum of squared distances and differentiating with respect to θ. The condition for this derivative being zero is:

$$\tan\theta = \frac{\sum_{i=1}^{3} |x_i \times y_i|}{\sum_{i=1}^{3} x_i \cdot y_i}.$$

With θ found, the required transformation that matches P_i's to the centers of C_i's is

$$T = T_4 R_3 R_2 T_1$$

where

- T_1 translates the centroid of P_i's to the origin;
- R_2 rotates the P_i's about the origin to a plane parallel to that of the centers of C_i's;
- R_3 rotates the P_i's about their centroid by the optimal θ;
- T_4 translates the P_i's centroid from the origin to the centroid of the C_i's centers.

For every atom P in S, $T(P)$ is defined to be the center of its error sphere.

Radii of error spheres

The radii of error spheres are harder to find than the centers. For each sphere, its radius must be larger than the maximum displacement of an atom from the error

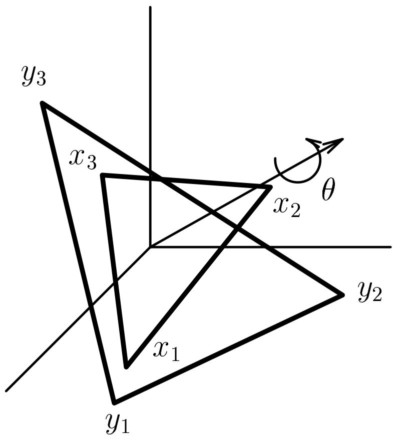


Figure 2-6: When two triangles are coplanar and their centroids are coincident, there is only one angle of rotation, θ, to determine.

sphere center. The sizes of the spheres depend not only on the C_i's, but also the locations of atoms on S relative to the P_i. The range of motion of an atom increases with the dimension of the C_i's, as well as its separation from the P_i's. For efficiency, AmbiPack calculates a set of radii for atoms of S depending on the sizes of the C_i's, but not on the C_i's exact locations. Thus all nodes at the same level of the search tree share the same set of error sphere radii; it is not necessary to recalculate them at every node.

Suppose the largest C_i has length d on each side. A sphere of radius $\sqrt{3}d$ centered at any point in the cube will contain it, regardless of the cube's orientation. Therefore we can restate the problem of finding error sphere radii as: Given S and the P_i's where each P_i may have a maximum displacement of $\sqrt{3}d$, find the maximum displacements of all other atoms in S. These displacements will be used as the radii of the error spheres. There are two possible approaches to this problem — analytical and numerical. One can calculate an analytical upper bound of the displacement of each atom, but it is quite difficult to derive a tight bound. A loose bound will result in excessively large error spheres and ineffective pruning. We choose a simple randomized numerical technique to find the maximum displacements. A large number (1000) of random transformations, which displace the P_i's by $\sqrt{3}d$ or less, are generated. These transformations are applied to all atoms on S. For each atom, we simply record its maximum displacement among all transformations. Empirically, this technique converges very quickly on reliable bounds. The performance data in the next section also shows that the resulting radii are very effective in pruning the search tree.

2.3.3 Algorithm Summary

Figure 2-7 summarizes the basic AmbiPack algorithm. Note that for a problem where some solutions exist, the search tree is potentially infinite. There will be a feasible region in space for each P_i. If we do not limit the depth of the tree, it will explore these regions and will continuously subdivide them into smaller and smaller cubes. Thus, we have an issue of choosing a maximum depth for exploration of the tree. On the other hand, the search tree is finite if a problem has no solution. As one searches deeper down the tree, the C_i's and error spheres become smaller. With error sphere pruning, the conditions on the nodes become more stringent and closer to the input constraints. Eventually, all nodes at a certain level will be rejected.

The simplest strategy for using the AmbiPack algorithm is:

1. Select a desired resolution for the solutions, which corresponds to a level of the search tree.
2. Search down to the specified level with pruning.
3. If there are no leaf nodes (that is, if every branch is pruned due to an inability to satisfy the constraints), there is no solution to the problem. Otherwise, calculate transformations from the leaf nodes and test against the input constraints. One would typically use a local optimization technique, such as conjugate gradient, to adjust the leaf-node transformations so as to minimize any violation of the input constraints.
4. If some transformations satisfy all constraints, output them. Otherwise, the resolution chosen in Step 1 may not be fine enough, or the problem may not

```
1. Select three constraints ($P_iQ'_i < x_i$, $i = 1, 2, 3$) from the constraint set.
2. Construct $C_1$ to be a cube centered at $Q'_1$ with $2x_1$ on each side.
        Similarly, construct $C_2$ and $C_3$.
3. For all level $l$ in the search tree do
4.      For all atom $p$ on $S$ do
5.          Calculate $r(p, l)$, the radius of error sphere of atom $p$ at depth $l$.
6. Call Search($C_1$,$C_2$,$C_3$,1).

7. **Procedure** Search($C_1$,$C_2$,$C_3$, $Search\_level$)
8.      Calculate transformation $T$ by matching $P_i$'s to $C_i$'s.
9.      For all input constraints $pq < x$ do
10.         Check $T(p)q < x + r(p, Search\_level)$
11.     For all input constraints $pq > x$ do
12.         Check $T(p)q > x - r(p, Search\_level)$
13.     If $Search\_level < Search\_depth$ then
14.         Split $C_i$ into $C_i^1, C_i^2, \ldots, C_i^8$, $i = 1, 2, 3$.
15.         For $j$ from 1 to 8 do
16.             For $k$ from 1 to 8 do
17.                 If $\min(C_1^jC_2^k) \leq P_1P_2 \leq \max(C_1^jC_2^k)$ then
18.                   For $l$ from 1 to 8 do
19.                       If $\min(C_2^kC_3^l) \leq P_2P_3 \leq \max(C_2^kC_3^l)$ and
20.                         $\min(C_1^jC_3^l) \leq P_1P_3 \leq \max(C_1^jC_3^l)$ then
21.                         Call Search($C_1^j$,$C_2^k$,$C_3^l$, $Search\_level + 1$).
22.     else
23.         Output $T$.
```

Figure 2-7: The AmbiPack algorithm. *Search_depth* is the limit on the depth of the tree.

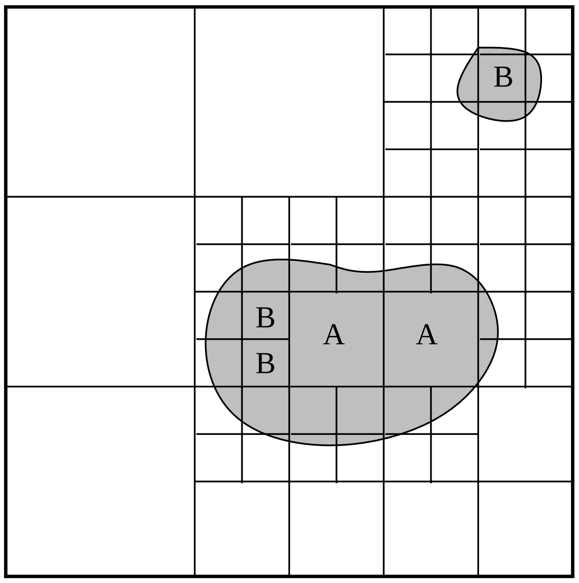


Figure 2-8: Solutions will generally be clustered into regions (shown shaded) in a high-dimensional space, characterized by the placements of the P_i. Each leaf of the search tree maps into some rectangular region in this space whose size is determined by the resolution. AmbiPack samples one point (or at most a few) for each leaf region in this solution space. At a coarse resolution, only leaves completely inside the shaded solution region are *guaranteed* to produce a solution, e.g. the regions labeled A. As the resolution is improved, new leaves may lead to solutions, some of them will be on the boundary of the original solution region containing A, others may come from new solution regions not sampled earlier.

have any solution. Select a finer resolution (deeper level in the search tree) and go to Step 2.

This strategy is generally quite successful in determining whether solutions exist for a given problem. If solutions exist, it will typically find all of them at the specified resolution, determined by the maximum search depth. However, this strategy is not completely systematic since it is relying on step 3 to find a solution if one exists, but this is not guaranteed since only one, or at most a few, transformations will be sampled for each leaf (see Figure 2-8). Our experience is that this works quite well in practice. Most of the results reported in the next section use this simple strategy.

However, if stronger guarantees are required then a more sophisticated variant of this strategy can be followed. As we discussed in Section 2.3.1, there are two

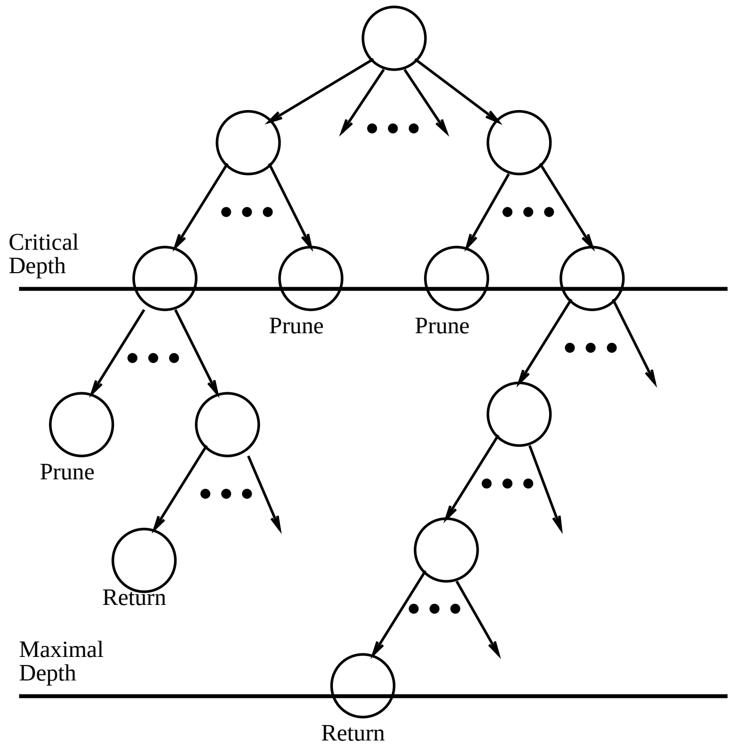


Figure 2-9: Illustration of the search using a critical depth and maximal depth.

meaningful, and conflicting, resolution limits in this type of problem. One arises from the goal of finding solutions if they exist. There is a minimum resolution determined by the accuracy of the measurements below which it makes no sense to continue partitioning space in search of a solution. Therefore, there is a *maximal depth* in the tree beyond which we never want to proceed. However, we do not want to expand all the leaves of the tree to this maximal depth. The other search limit stems from desire to avoid generating too many nearly identical solutions. This defines a *critical depth* in the tree beyond which we want to return at most a single transform if one exists, rather than returning all the transforms corresponding to leaves. Therefore, we can modify the AmbiPack algorithm so that, below the critical depth and above the maximal depth, it attempts to find a solution (using minimization) for every node that is not pruned. If a solution is found, it is stored and search resumes with the next subtree at the critical depth. In this way, at most one solution is stored per subtree at the critical resolution but the subtree is searched to the maximal resolution.

2.3.4 Algorithm Extensions

Assume that all the specified constraints must hold between the given structures. In Section 2.3.1 we outlined how the algorithm above can be extended to cope with ambiguous constraints. There are two parts of the algorithm in Figure 2-7 that need to be changed:

Line 1 : Ideally, we select three constraints that have no ambiguity. If all constraints are ambiguous, we select the three least ambiguous ones and enumerate all possible interpretations of them. This requires adding an outer loop which runs the search a^3 times, where a is the ambiguity in each constraint. Typical experiments produce constraints with 2-fold ambiguity. They are solved by AmbiPack efficiently. However, if each constraint has a large number of ambiguous interpretations, AmbiPack may not be appropriate.

Lines 9 through 12 : If some constraints are ambiguous, we simply add appropriate OR's to the inequalities derived from those constraints. This modification does not slow execution.

These extensions mean that we may need to run the search a^3 times, which is usually a small number. If the constraints have some special properties, this number can be reduced even further. For example, if S is the same as S' and all constraints are either positives or negatives, we need to search only 4 instead of 8 times. The positives and negatives are symmetrical. If transformation T is a solution satisfying a set of inequalities, T^{-1} is also a solution satisfying the complementary set of inequalities. Making use of this symmetry, we choose the interpretation of one positive constraint arbitrarily and, therefore, only need to calculate half of the solutions.

Because AmbiPack needs to select three constraints initially, it is limited to problems with three or more "less-than" constraints. This is not a severe restriction because most practical problems have a large number of "less-than" constraints. On the other hand, the choice of a particular set of three constraints will have a large impact on efficiency. Given C_i's of the same size, atoms on S will have smaller displacements if P_i's are farther apart from each other. This will lead to smaller error spheres and more effective pruning. In our implementation, we select the constraints that maximize the area of the triangle $P_1P_2P_3$.

Now, consider the case where all the constraints need not be satisfied between the given structures. This may be due to labeling ambiguity or simply due to measurement error. As we mentioned earlier, one approach to this problem is to enumerate subsets of the ambiguous constraints and solve them independently. The difficulty with this approach is that the number of subsets grows exponentially with the number of constraints. An alternative approach is, instead of requiring that all input constraints be satisfied, to specify a minimum *number* of constraints that must be satisfied.

Once again, it is Line 1 and Lines 9 through 12 that need to be changed. The easy change is that Lines 9 through 12 can be readily changed from checking that all constraints are satisfied into *counting* the satisfiable constraints. We reject a node if the count is less than the minimum number. If we just make this enhancement, without changing Line 1, we can use the algorithm to constrain "optional" chemical properties such as the minimum number of feasible hydrogen bonds or favorable van der Waal's contacts in a structure.

The more difficult problem is what to do when all of the constraints do not need to hold. If one wants to guarantee that all possible solutions are found, then one needs to consider *all* possible triples of constraints in Line 1 instead of just choosing *one* initial triple. The number of such triples grows roughly as n^3 when there are n ambiguous constraints. This is a great improvement over the exponential growth in the number of constraint subsets, but it is still the limiting factor in applying the AmbiPack algorithm to problems with large numbers of ambiguous constraints and multiple interfaces.

2.4 Results and Discussions

We have carried out two detailed studies using the AmbiPack algorithm to explore its performance and illustrate its range of applicability. The first study involves a large protein (the P22 tailspike protein — a homotrimer involving 544 residues), a large number (43) of (simulated) ambiguous measurements and a single type of interface between three subunits. The second study involves a nine-residue peptide from β-amyloid, a small number (8) of ambiguous measurements and two different interfaces between an indefinitely repeating group of subunits.

In all the tests discussed below, the AmbiPack algorithm is implemented in Common Lisp [20], running on a Sun Ultra 1 workstation. All constraints used involved carbon–carbon distances because they are commonly measured in solid-state NMR experiments, rather than distances to hydrogen, which are currently more common in solution NMR spectroscopy.

2.4.1 P22 Tailspike Protein

The first test of the AmbiPack algorithm is the P22 tailspike protein [39] (PDB [3] code 1TSP). In its crystal structure, the positions of 544 residues are determined. The protein forms a symmetric homotrimer. Each subunit of the homotrimer contains a large parallel β helix. We use AmbiPack to find the relative orientation of two subunits; the third subunit can be placed by applying the solution transformation twice, as discussed in Section 2.2.

First, we measured the intermolecular distances between C_α carbons at the interface of the subunits. There were 43 C_α-C_α distances less than 5.5 Å (a typical upper bound for distance measurements in some NMR experiments), giving 43 positive constraints. To be conservative, we specified each constraint with an upper bound of 6 Å. For example, one of the constraints was ($C_\alpha^{120} C_\alpha'^{124} < 6.0$ Å) OR ($C_\alpha^{124} C_\alpha'^{120} < 6.0$ Å). The 43 ambiguous constraints and the two identical subunit structures were given to the algorithm. The constraints have a total of 2^{42} possible interpretations. Our program solved this problem in 279 seconds to a maximal resolution where the C_i's are 2 Å on each side. Most of the computer time was spent in the recursive search procedure. When the 47 solution transformations were applied to a subunit, the results had an average RMSD of 0.6078 Å from the X-ray structure. The worst solution had an RMSD of 2.853 Å. This error is much smaller than the ranges of input con-

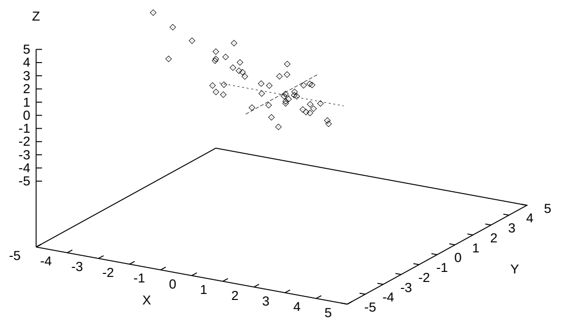
Critical Res.	Maximal Res.	No. Solutions	Avg. RMSD	Time (sec)
2.0	2.0	47	0.6078	279
2.0	0.5	743	1.073	3801
2.0	0.25	827	1.033	11497

Table 2.1: Results of AmbiPack running on the P22 tailspike trimer with different values of the maximal resolution for the search.

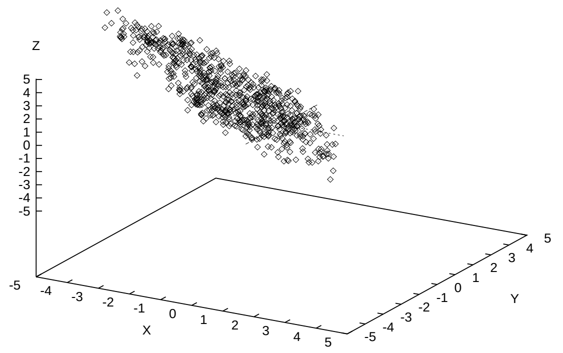
straints. The four search trees (arising from exploring the ambiguous assignments of the first three constraints chosen, given identical subunits) had a total size of $57,425$ nodes. The effective branching factor is 24.3 instead of the theoretical worst case of 512. The branching factor becomes smaller as we search deeper because the pruning techniques become more powerful.

We investigated the effect of using different values for the maximal resolution during the search, while leaving the critical resolution at 2 Å (see Section 2.3.3). The results are shown in Table 2.1. Note that there are many more leaves that lead to a solution when using finer critical resolutions. However, we found that there were no distinctly different solutions introduced, rather one obtains more samples near the boundary of a single solution region (see Figure 2-10). The gradual increase in the average RMSD is consistent with the fact that the new solutions obtained with improved maximal resolution are from the boundary of a relatively large solution region.

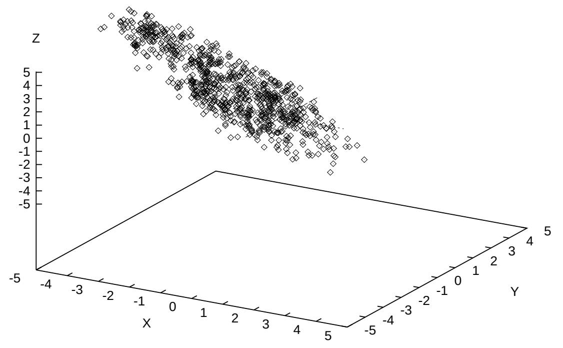
These solutions are obtained without using any steric constraint of the protein. Following a suggestion of an anonymous reviewer, we filtered these structures by steric constraints. Figure 2-11 shows solutions that contain 5 or fewer severe steric clashes. We define a severe steric clash as having two atoms' van der Waal's spheres overlapping by 1Å or more. These selected solutions are confined to a small region around the packing of the crystal structure. This shows that steric constraints are effective as a filter of packing solutions. The user would have the choice of provisionally accepting solutions that violate steric and using refinement methods to relax the structures while satisfying all constraints or, alternatively, filtering out such solutions



A



B



C

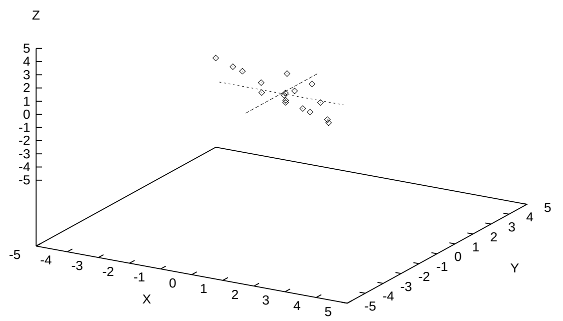
Figure 2-10: The translation components of solutions from AmbiPack running on the P22 tailspike trimer with critical resolution of 2.0Å and maximal resolution of (A)2.0Å, (B)0.5Å, and (C)0.25Å. The solution from the crystal structure, (0,0,0), is marked by a cross. The rotation components are mostly identical for all solutions.

and requiring the monomer generating procedure to provide more accurate structural models. Presumably data rich problems would be amenable to the latter solution and data poor problems are more efficiently solved with the former.

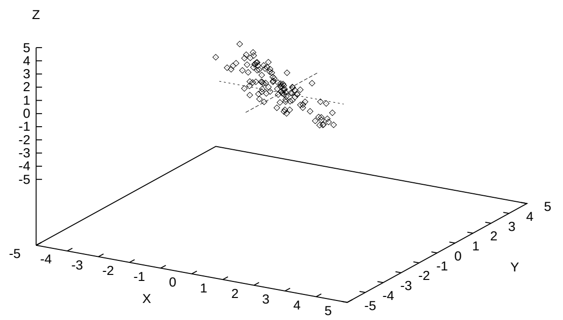
As one would expect, changes in the maximal resolution (and therefore the maximal depth of the search tree) have a substantial impact on the running time. Subsequent experiments were done with both the critical and maximal resolution set to 2 Å.

We also used the tailspike protein to investigate the effect of quantity and type of constraints on protein structures. We constructed various sets of constraints and measured the computational resources required to find solutions for each set as well as the accuracy of structures. Results of the runs are plotted in Figures 2-12 and 2-13. Each data point on the plots is the average over 25 randomly selected sets of constraints. Initially, we used different subsets of the 43 positive constraints. These runs produced the uppermost lines on the plots. Figures 2-12 shows the computational resources approximated by the number of nodes in the search trees. Clearly, the computer time decreases rapidly as more positive constraints are added. This reflects the effectiveness of early pruning in the AmbiPack algorithm. Figure 2-13 shows the worst RMSD in the solutions. Solutions improved rapidly in quality with more positive constraints. In general the plots show diminishing returns when adding positive constraints.

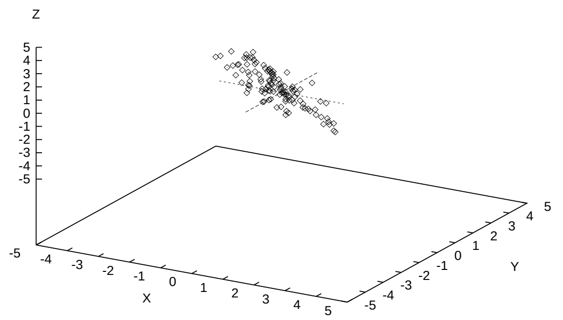
Next we studied the effect of negative constraints. In the crystal structure, we found more than 100,000 pairs of C_α's with distances greater than 7 Å. Thus there are many more potential negative constraints than positive ones. We specified each negative constraint with a lower bound of 6 Å, e.g. $(C_\alpha^{114} C_\alpha'^{117} > 6.0$ Å$)$ AND $(C_\alpha^{117} C_\alpha'^{114} > 6.0$ Å$)$. However, we found that these negative constraints had almost no effect on the computer time or solution quality. We believe that most negative constraints, whose C_α pairs are very far apart in the crystal, do not affect the packing solutions. We randomly selected 500 negative constraints whose C_α pairs are farther than 20 Å and added them to the 43 positive constraints. The size of search tree and resulting solutions were identical to those when using only the positive constraints.



A



B



C

Figure 2-11: The translation components of solutions from AmbiPack running on the P22 tailspike trimer, with 5 or fewer severe steric clashes. The critical resolution is 2.0Å and the maximal resolutions are (A)2.0Å, (B)0.5Å, and (C)0.25Å. The solution from the crystal structure, (0,0,0), is marked by a cross.

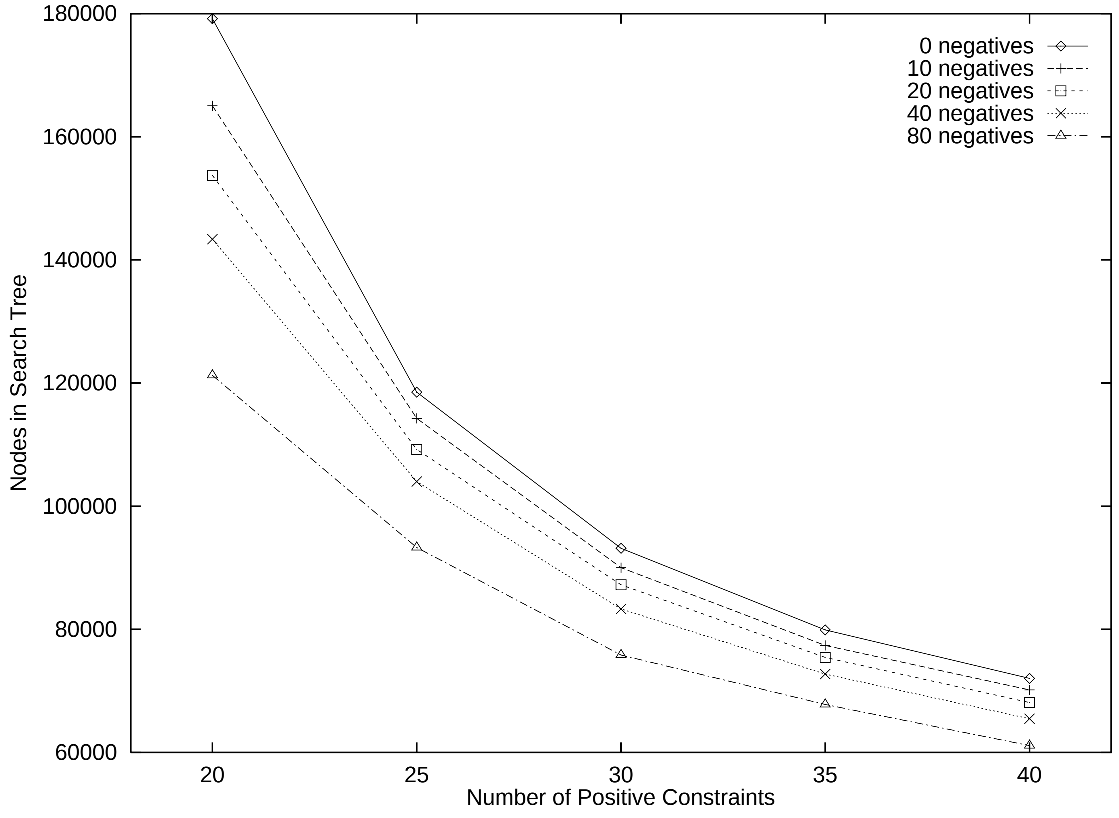


Figure 2-12: Computational resources used by various combinations of positive and "near-miss" negative constraints.

Therefore, these negative constraints from far-apart atom pairs do not contain new information for structure determination.

In order to explore the potential impact of well-chosen negative results, in the later runs, we used only "near-miss" negative constraints from C_α pairs whose actual distances are just a little above the lower bound. In the P22 tailspike protein, we found 589 "near-misses" from measuring the crystal structure with an upper bound of 10 Å. These constraints did affect the computer time and solution accuracy. The results of runs with "near-misses" are also shown on Figure 2-12 and 2-13. Computational efficiency and solution quality improved as more "near-miss" negative constraints were added, though their effect is not as significant as the same number of positive constraints. In these simulations, positive constraints contain ambiguous information, but they are more valuable for structure determination than the unambiguous negative constraints. These results also suggest that experiments should be designed to obtain negative data close to the boundary of detection, thus maximizing information on the structure. For example, if a small set of constraints is known, we can use the triangle inequality to establish upper bounds on other distances[8]. Further experiments can be directed towards measurements with small upper bounds. It should be noted that in certain circumstances, strategically chosen negative constraints may be especially useful; the results here suggest that randomly selected negative constraints are unlikely to be as useful as randomly selected positive constraints.

2.4.2 C-terminal Peptide of β Amyloid Protein

The second test of the AmbiPack algorithm is a nine-residue peptide. This peptide (β34-42) is from the C-terminus of the β amyloid protein (β1-42). Lansbury, Griffin et al.[21] have applied solid-state ^{13}C NMR to this peptide and measured intra- and intermolecular ^{13}C-^{13}C distances. Their experiments produced 8 positive and 12 negative intermolecular constraints. A pleated antiparallel β sheet was proposed as the structure that satisfies all constraints, although the data can only define the structure to a relatively low resolution. There are two types of interfaces in their proposed structures. They alternate among the β-strands in the sheet (Figure 2-2).

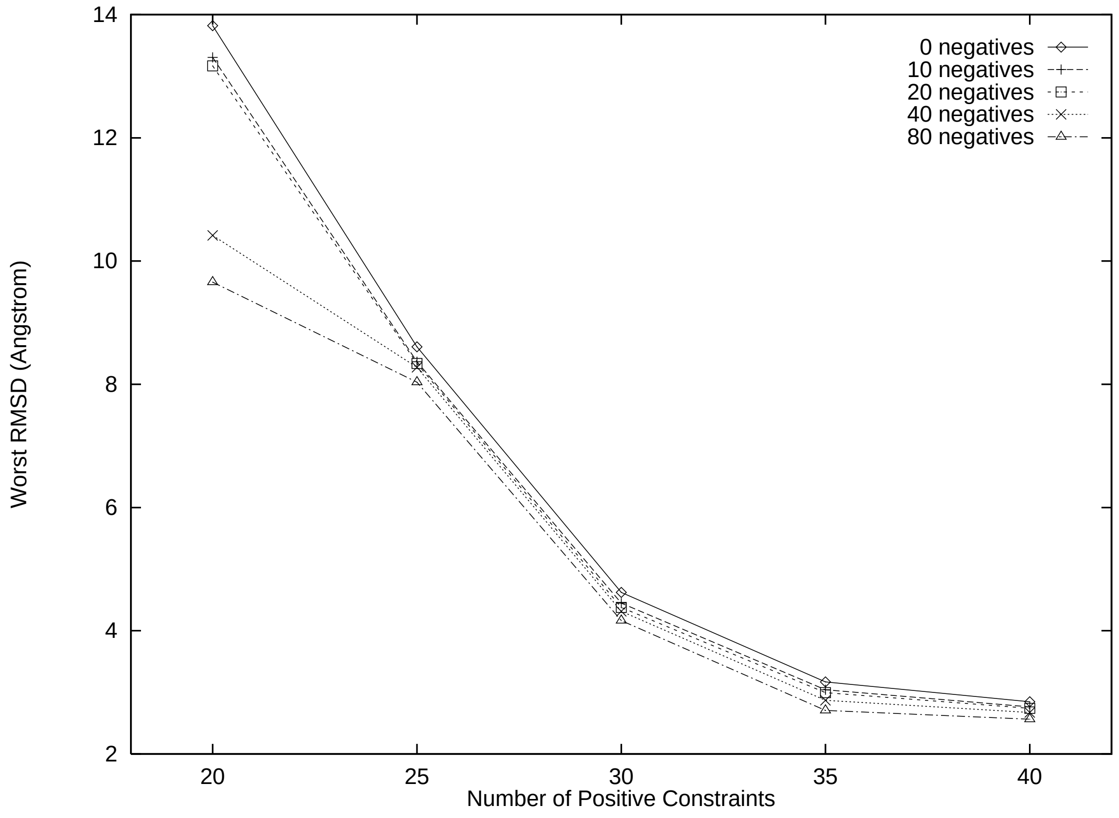


Figure 2-13: Quality of structures produced by various combinations of positive and "near-miss" negative constraints. The RMSD is measured by applying the computed transforms to one of the P22 tailspike monomers (from the X-ray structure) and measuring the displacement from this monomer to the nearby monomer in the X-ray structure.

Either interface satisfies all negative constraints but only a subset of the positive ones. Together the interfaces satisfy all positive constraints.

A nine-residue poly-alanine idealized β-strand was energy minimized subject to the intramolecular backbone carbon–carbon distance restraints and used as the monomer[3]. AmbiPack was given the 20 measured constraints and additional constraints that eliminate steric clashes. The positive constraints have very lenient upper bounds of 6.5 Å and the negatives also have lenient lower bounds of 5.5 Å. Still, AmbiPack could not find any solution to this problem, meaning that there is no single packing that satisfies all the constraints; two or more different packings are required. This result is consistent with Lansbury et al.'s two-interface structures.

If there are two or more packing interfaces, each one must satisfy a subset of the positive constraints and together they must satisfy all. We ran our program on all subsets down to three positive constraints, which is the minimum requirement for AmbiPack. Because of the symmetry of the constraints, we search only four times for each subset to a resolution where the C_i are 2 Å on each side. The results are shown in Table 2.2. There are many subsets with solutions; we investigate only the largest ones. There is one set with 7 constraints (set A). There are 9 sets with 6 constraints, but 7 of the 9 are subsets of A. We call the other two sets B and C. They are given in Table 2.3. When given set A plus all negative constraints, our program found four solutions (Figure 2-14). By symmetry, there are four other solutions due to the inverse transforms. They are not shown in the figure. One of the four solutions is antiparallel to the stationary strand. Three others are tilted with respect to the stationary one. AmbiPack found three solutions to constraint set B (Figure 2-15). In this case, one solution is antiparallel and two are tilted. C gives a single tilted solution (Figure 2-16).

In order to find the full structure of the peptide, we need to combine the solutions from A with those from B or C. $A \cup B$ or $A \cup C$ gives the complete set of positive constraints. Lansbury et al. have shown that this peptide forms a noncrystalline,

[3]Initial experiments had suggested that the monomer was a bent β-strand due to a *cis* peptide bond, but more recent evidence is consistent with an all-*trans* structure.

Constraint set size	Number of sets	Sets with solutions	Running time per set (sec)
8	1	0	63
7	8	1	90
6	28	9	92
5	56	29	106
4	70	50	135
3	56	50	201

Table 2.2: Results of AmbiPack running with multiple subsets of positive constraints.

All Positive Constraints	A	B	C
C_α^{37}, C^{38}	C_α^{37}, C^{38}	C_α^{37}, C^{38}	C_α^{37}, C^{38}
C^{37}, C_α^{39}	C^{37}, C_α^{39}		
C^{36}, C_α^{39}	C^{36}, C_α^{39}	C^{36}, C_α^{39}	C^{36}, C_α^{39}
C^{36}, C_α^{40}	C^{36}, C_α^{40}		C^{36}, C_α^{40}
C^{34}, C_α^{39}		C^{34}, C_α^{39}	C^{34}, C_α^{39}
C^{34}, C_α^{40}	C^{34}, C_α^{40}	C^{34}, C_α^{40}	C^{34}, C_α^{40}
C_α^{36}, C^{38}	C_α^{36}, C^{38}	C_α^{36}, C^{38}	C_α^{36}, C^{38}
C_α^{36}, C^{39}	C_α^{36}, C^{39}	C_α^{36}, C^{39}	

Table 2.3: The complete positive constraints and three subsets.

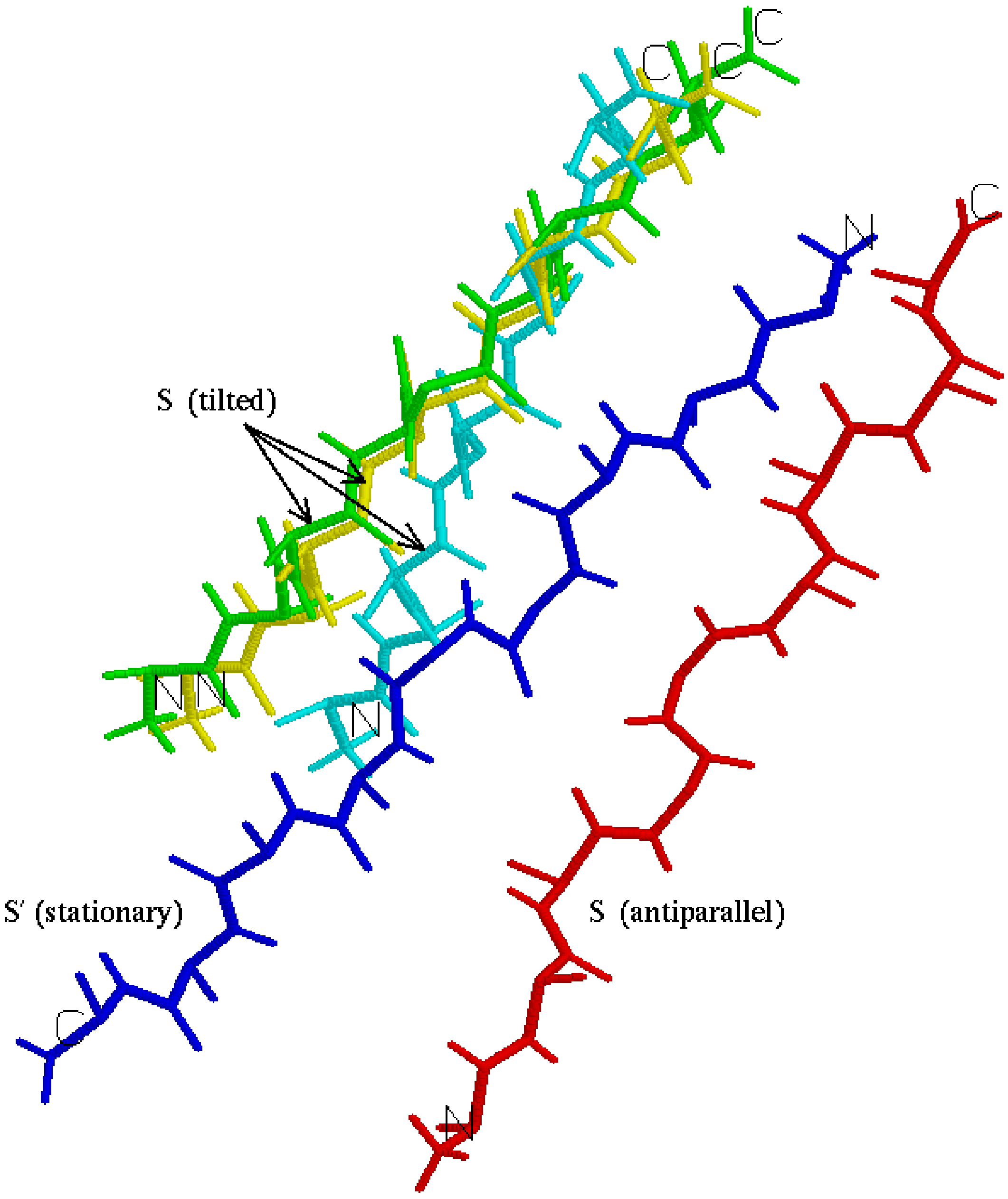


Figure 2-14: 4 packing solutions to constraint set A.

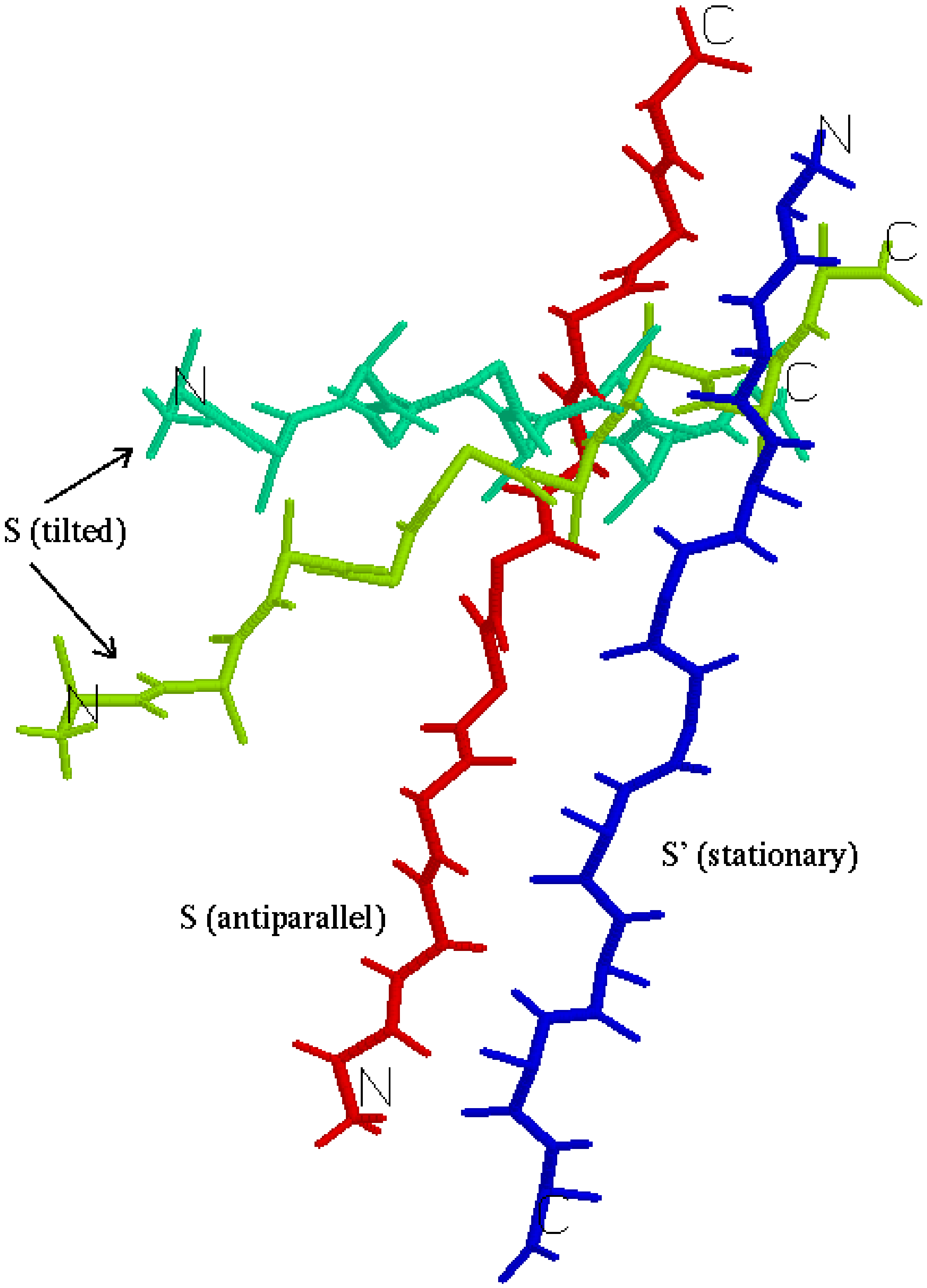


Figure 2-15: 3 packing solutions to constraint set B.

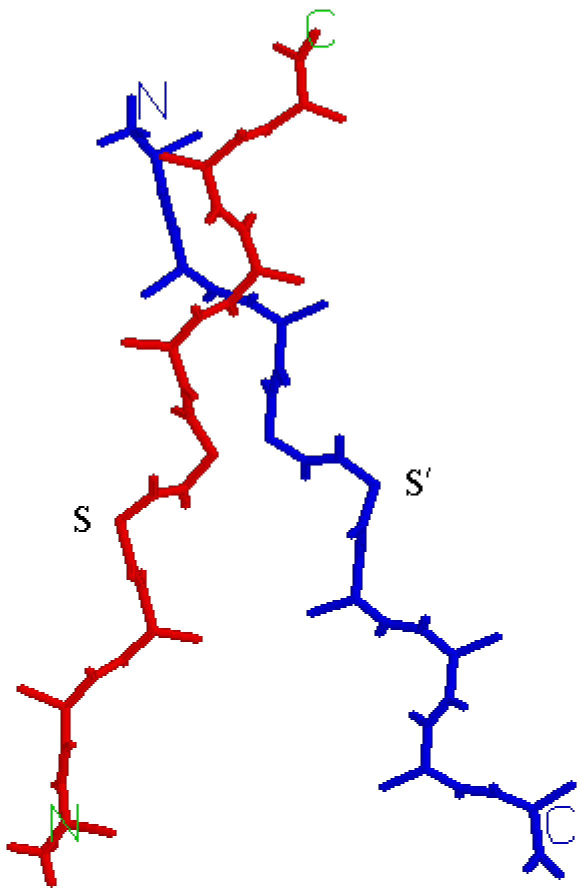


Figure 2-16: A single solution to constraint set C.

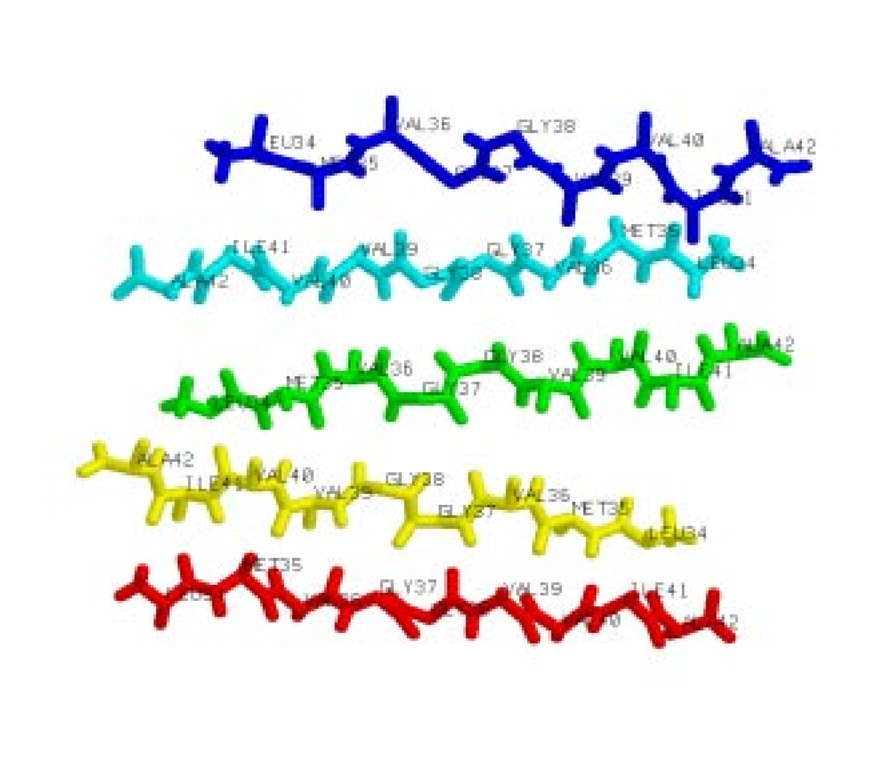
yet ordered aggregate. The most plausible structure consists of an infinite number of subunits which "tile" the space in a regular fashion. Therefore, we try to calculate a continuous structure with alternating interfaces from A and B or A and C. First, we focus on A and B. Suppose T_1 and T_2, two rigid transformations, are solutions to A and B respectively. Let S be the subunit. Then

$$S, T_1(S), T_1T_2(S), T_1T_2T_1(S), T_1T_2T_1T_2(S), \ldots$$

is the series which forms a continuous structure from T_1 and T_2. The interface between S and $T_1(S)$ satisfies A, whereas the interface between $T_1(S)$ and $T_1T_2(S)$ satisfies B, and so forth. There are $4 \times 2 \times 3 \times 2 = 48$ such structures possible. Again, by symmetry of the transformations, we need consider only half of the structures. 22 of the 24 have steric clashes among the subunits. The two structures without steric clashes are shown in Figure 2-17. Structure 1 is an antiparallel β sheet that is compatible with the hydrogen-bond pattern of Lansbury et al.'s model. In this structure the hydrogen-bonding partners of Lansbury et al.'s model are properly aligned, albeit too distant, for good hydrogen bonds. This structure can be a starting point for structure refinement. Structure 2 is a non-sheet structure which does not form regular hydrogen bonds. Combining solutions from A and C, there are four solutions, all non-sheet like.

2.5 Conclusions

The AmbiPack algorithm has been developed to pack pre-conformed monomer structures into multimers using interatomic distance constraints. A novel feature of the approach taken here is that it efficiently and accurately deals with the labeling ambiguity inherent in symmetric multimers due to a lack of knowledge about which atom in an intermolecular distance constraint comes from which monomer. The branch-and-bound method is applied to a search tree defined for progressively finer levels of resolution in the placement of three points on one of the monomers. Efficient pruning


Structure 1

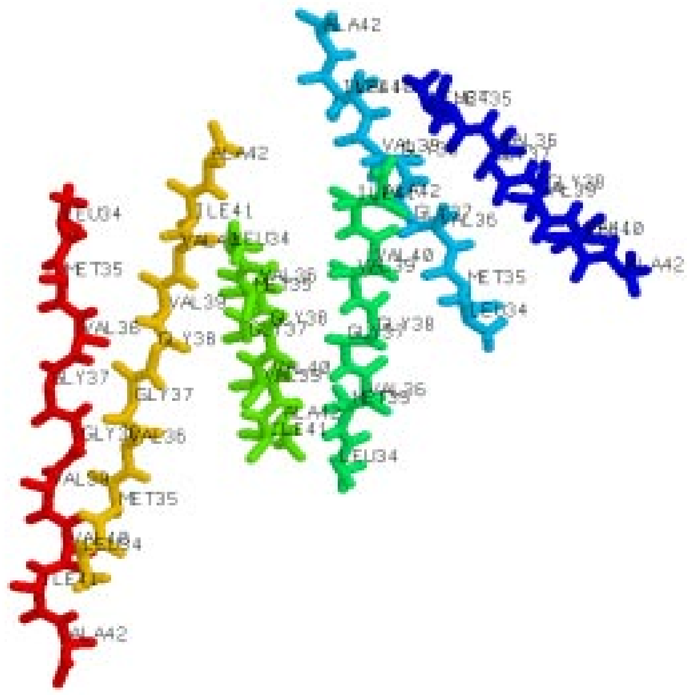

Structure 2

Figure 2-17: Two continuous structures with alternating interfaces satisfying A and B, but without steric clashes.

dramatically reduces the branching factor for this tree from a theoretical value of 512 to values typically 20-fold lower. Improved pruning causes the algorithm to run faster when more constraints are present. While the algorithm is exhaustive at any desired level of resolution, we have found that it is generally sufficient to stop the search at relatively coarse resolution of 2 Å. In our tests, resolutions down to 0.25 Å did not generate distinct new solutions. Methods based on this algorithm could be especially useful in instances where it is important to establish the uniqueness of a packing solution or to find all possible solutions for a given set of constraint data.

Chapter 3

ConfMatch: Matching Conformations to Electron Density

3.1 Introduction

Determining the structures of proteins is essential to understanding molecular biology of cells. X-ray crystallography is the "gold standard" for protein structure determination. Appendix A explains the terminology and summarizes the process of X-ray crystallography. This chapter describes ConfMatch, a systematic algorithm for an important step in solving an x-ray structure—building a model from the initial three dimensional electron density distribution (density map). ConfMatch solves this "map interpretation" problem by matching a detailed conformation of the molecule to the density map. This problem is computationally challenging because proteins are extremely flexible. A typical protein may have several hundred degrees of freedom. The space of possible conformations is astronomical. If one defines a function that evaluates how well a conformation matches the density map, this function will have many local minima over the space of possible conformations. Any non-systematic algorithm may produce a local optimum instead of the global optimum. ConfMatch quantizes the continuous conformational space into a large set of discrete conformations and finds the best solution within this discrete set. Because ConfMatch samples the conformational space very finely, its solution is usually very close to the globally optimal conformation.

ConfMatch's approach is based on fitting a chemically feasible molecular structure to an imperfect density map. It finds this "best match" structure by a systematic branch-and-bound search. The output of ConfMatch, a chemically feasible conformation, is both detailed and high quality. It is detailed because it includes all non-hydrogen atoms of the target molecule. It is high quality because the conformation satisfies various commonly-accepted chemical constraints such as bond lengths, angles, chirality, etc.

ConfMatch has several important potential applications:

1. When a scientist tries to solve a structure by multiple isomorphous replacement (MIR) or multiple wavelength anomalous dispersion (MAD), he/she must spend a long time manually fitting and refining the molecular structure to the experimental density map. This manual step can be mostly automated if an algorithm can find a high quality structure matching well to the density map. Therefore ConfMatch can be a time-saving tool for protein crystallographers.

2. Since the molecular structure found by ConfMatch is chemically reasonable, this structure can in turn produce an improved density map which takes into account the chemical constraints. Therefore, our approach is a scheme to optimize the electron density distribution. If one integrates it with other optimization methods operating on the phase set (Section B.4), one may be able to develop a purely computational solution to the phase problem. Therefore ConfMatch may be a part of a direct method for protein crystallography.

3. Instead of fitting a single structure to the electron density, ConfMatch can be adapted to fit a family of structures simultaneously to any kind of stationary field (Section 3.4.5). The result would be the best-fit structure within the family. For example, one may calculate the optimal electrostatic field for binding of a disease causing protein [25]. This optimal field specifies the electrostatic charges at different regions of space potentially occupied by a ligand. The space may be partitioned into regions charged positively, negatively, or neutrally. ConfMatch

can at once fit all peptides of a certain length to this field. The output structure will give the best theoretical peptide ligand to this protein, as well as its most favored conformation. Therefore ConfMatch can be a useful tool for rational drug design.

3.2 Related Work

The process of building a model from the initial electron density map (map interpretation) is an important part of solving an x-ray structure. For small molecules with high resolution data, the most common method for map interpretation is *peak picking*. This algorithm simply finds the w highest peaks (local maxima) in the map, where w is the expected number of atoms in a unit cell, and declare them the atom positions. The identities of different peaks are usually labelled manually. Peak picking has long been used in small-molecule direct methods because crystals of small molecules diffract to very high resolution. Atoms are located at the peaks of density maps at sufficiently good resolution. However, peak picking becomes ineffective with worse than 1.2Å data. At lower resolution, atoms rarely locate at the peaks of the electron density map. Most macromolecules of interest diffract to 2.0Å or worse. Therefore the applicability of peak picking to protein crystallography is limited.

Currently, there is no fully automated solution that can derive a detailed molecular structure from a density map for protein-size molecules. Much manual intervention is required to construct such a model. There are several computational approaches that automate different aspects of this process. Most of these techniques attempt to detect within the density map certain structural features, which may guide or facilitate the human model builder.

Skeletonization [17] is a widely used method for building protein models. First, the map is searched to locate three kinds of features—peaks, ridges, and join points. Peaks are the same features defined in the previous method. A ridge is the highest density path joining two peaks. A join point is the lowest point on a ridge. In other words, a join point is the the lowest point on the highest path joining two peaks.

The output of this method is a set of ridges forming the "skeleton," which may trace the main and secondary chains of the molecule. Usually, only ridges with high-value join points are included in the output. Similar to peak picking, the skeleton is an unlabelled structure. The atom labels are usually added manually to the skeleton, which is then refined.

Molecular scene analysis [26] is a new approach to map interpretation. At medium (≈3Å) resolution, this algorithm searches the map to locate two kinds of features—peaks and passes. Peaks are defined the same way as in the previous methods. A pass is a saddle point in the map where the three first derivatives are zeroes, but only two out of the three second derivatives are positive. A pass is very similar to a join point in the previous method. Leherte et al. observed that, at medium resolution, the peaks correspond to amino acid residues, while the passes correspond to the adjacency of the residues in the primary sequence. The protein backbone can thus be viewed as a sequence of alternative peaks and passes. Given the peaks and passes features, the molecular-scene-analysis method calculates a minimal spanning tree of alternating peaks and passes. The peaks are declared as the locations of either the residues or large side chains. The next stage of the algorithm finds the most plausible way to superimpose the amino acid sequence onto the spanning tree by protein threading methods.

Zou and Jones developed an alternative approach to matching a protein sequence to a model structure [41]. Their method requires the crystallographer to build at least a polyalanine model through the density map. For each of the 20 residue types, their program optimizes the fit of the side-chain atoms to the density by pivoting the side chain around each C_α atom. For the best fitting rotamer of each residue type, a score is calculated which is used as an index of how well that amino-acid type fits the density. Once the scores are obtained for every residue type at every position, Zou and Jones' method calculates how well a sequence of amino acids matches the backbone model by combining the individual scores. The output of their program defines the placements of subsequences of the protein on the polyalanine structure.

Template convolution [23] is a new approach to detecting large structural features in the density map. A template is a set of atoms, usually an ideal, short α-helix or β-strand. This algorithm rotates the template around a pivot point for each point in the map, and a score is calculated which reflects how well the atoms fit the density for each orientation at each point. This is equivalent to a six dimensional rigid-body search. The highest scores indicate the locations and orientations of the templates. The structural features detected by this method can guide the human model builder or enhance the electron-density map.

The biggest difference between ConfMatch and existing approaches is that ConfMatch is fully automated. No human guidance is required in constructing a detailed, high-quality molecular structure. ConfMatch generates its output conformation directly from the density map. It removes human subjectivity from the map interpretation process.

Unlike most existing techniques, ConfMatch does not use any local features. Without the aid of local features, ConfMatch is usually more computationally intensive than feature-based algorithms. On the other hand, ConfMatch's output achieves a global property—the entire conformation is a “best match” to the density map. The use of a global property instead of local features may allow ConfMatch to interpret less accurate density maps than other algorithms.

3.3 The Conformational Matching Problem

This section describes an approach to interpreting an electron density map by solving the *conformational matching* problem—finding a conformation that best matches the density. At resolutions typical of protein crystals, the peaks of the density map usually do not correspond to atom positions, but the high density regions still follow the main and side chains of the protein. Thus it is quite possible to find the correct conformation from a medium resolution density map. To overcome the inaccuracies of the density map, we make use of the commonly-accepted chemical constraints such as bond lengths, angles, chirality, etc. These constraints limit a molecule to its chemically feasible conformations. The possible distribution of atoms is much more restricted if they must obey the basic rules of chemistry. By applying more chemical constraints, we hope to produce a fully automated solution to electron-density map interpretation.

The definition of the conformational matching problem is: *Given an electron density map and the primary structure of a molecule, assuming fixed bond distances and angles, find a feasible conformation such that the sum of the density at (non-hydrogen) atom positions is maximized.* A feasible conformation is one which satisfies constraints such as cis/trans and planarity of double bonds, chirality, and excluded volume (Section 3.4). The objective function, the sum of the density at non-hydrogen atom positions, ignores the different identities of atoms. A carbon atom occupying a position is valued the same as if a nitrogen atom occupies it. If all non-hydrogen atoms have similar atomic numbers, their electron density distributions will be very similar. This objective function is adequate if the molecule is composed of C, N, O, and H only. However, if some non-hydrogen atoms have much higher atomic numbers than others, the objective function may need to be modified. One possible solution (Section 3.5.2) is to separate the atoms into two classes: heavy atoms and light atoms. Each class has its own electron density distribution that the atoms will measure from. The modified objective function is to maximize the sum of density, measured from an atom's particular density distribution, at positions of all atoms.

Instead of maximizing the sum of density at atom locations, there are other possible measures for the best structure. For example, one could minimize the R-factor, or the electron density unaccounted for by the structure. However, it is difficult to develop an efficient algorithm for these objective functions because they are calculated based on the entire density distribution, not just at the atom positions. The conformational matching problem as defined above strikes a good balance between computational efficiency and the accuracy of results.

Conformational matching is a constrained global optimization problem. One cannot solve this problem by finding local features in the density map, for a locally optimal conformation may not be part of the globally optimal solution. This is especially true in the presence of errors in the density map. In order to find the global

Notation	Description	Section
p	Number of free dihedral angles	3.3
n	A node in the fragment tree	3.4.1
$\mathbf{j}$	Bond placement	3.4.1
$E_{n,\mathbf{j}}$	An upper bound of the density sum of n's sub-fragment-tree, where n's in-bond is placed at $\mathbf{j}$.	3.4.1
s	Number of torsional samples	3.4.1
$e^i_{n,\mathbf{j}}$	The density sum of the i-th sample of fragment n	3.4.1
$S^i_{n,\mathbf{j}}$	An upper bound of the density sum of n's sub-fragment-tree, provided that sample i is chosen for fragment n.	3.4.1
R	A rotational transformation in Cartesian coordinates	3.4.2
θ_L	Sampling interval of torsional angles (pseudo uniform)	3.4.2
t	A state in conformational search	3.4.3
$f(t)$	An upper bound of the density sum of the entire structure given the current partial structure at t	3.4.3
$g(t)$	Density sum of the partial structures at t	3.4.3
$h(t)$	An upper bound of the density sum of the remaining structure	3.4.3
f_{lim}	f-value limit for a depth-first search	3.4.3
M	An upper bound of the density sum of the entire structure	3.4.3
d	The density sum of a solution structure	3.4.4
ϵ	The minimal improvement in solution we can accept	3.4.4
C	A transformation from fractional into Cartesian coordinates	3.5.2

Table 3.1: These symbols are defined in Chapter 3 and listed in the order of their appearance.

optimum, some form of global search is required. If one assumes fixed bond angles and bond lengths, the number of degrees of freedom of a conformation is $6+p$, when p is the number of free dihedral angles. (Table 3.1 lists all symbols defined in this chapter.) The extra 6 degrees of freedom comes from the rigid displacements. Even for very small proteins, p can run into the hundreds. The number of possible conformations, exponential in p, is astronomical. Exhaustive conformational search, such as uniform torsional search, is impractical without an intelligent way to vastly reduce the search space.

3.4 The ConfMatch Algorithm

ConfMatch is a systematic algorithm for solving the *discretized* conformational matching problem. The discretization specifies a 3D grid in space. For ease of implementation, we require that the 3 axes of the grid follow the axes of the unit cell. The grid axes are thus orthogonal in fractional space but not necessarily in Cartesian space. All atoms are required to locate on grid points. This grid will allow us to use discrete combinatorial techniques. The size of the grid also determines the local quality of the initial solution structure. Given a fine grid, the resulting structure is very close to the continuous solution. However, local constraints such as bond lengths and angles may be violated slightly as a function of grid size. To improve the local quality and remedy these violations, one may simply apply local optimization techniques, such as conjugate gradient in a continuous search space, on the output of ConfMatch. Usually, we choose 0.5Å as the grid spacing. In the rest of this chapter, all references to bond lengths, angles, and planarity have some implicit tolerance that permits the use of the grid.

ConfMatch is a branch-and-bound method with two stages: the *bound-preprocessing* stage and the *search* stage. The bound-preprocessing stage runs in time proportional to a function polynomial in p, the number of free dihedral angles (polynomial time). It calculates a table of upper bounds on the possible density sum. These upper bounds are based on all conformations that have the correct bond lengths and angles, but may or may not satisfy the excluded volume constraints. This set of bounds will allow the second stage to avoid most of the search space. The search stage performs a systematic conformational search of the target molecule. Each torsion angle of a single bond is searched through a series of possible values, similar to the internal coordinate tree search [28]. However, it is much more efficient than internal coordinate search because of the bounds: At every step of the conformational search, ConfMatch retrieves from the bounds table an accurate estimate of the remaining density sum. If the estimate is too low, the particular search direction is terminated. Therefore it can explore only a small portion of the search space and find the solution conformation.

Like other back-tracking search techniques, this stage can take time proportional to a function exponential in p (exponential time) in the worst case. Although, in practice, the search stage may take much less time than the bound-preprocessing stage given good density data.

Because any molecule must obey the basic rules of chemistry, a molecule's primary structure translates into a large number of geometric constraints, including:

1. bond angles,
2. bond lengths,
3. cis/trans and planarity of double bonds[1],
4. chirality of all chiral centers,
5. intramolecular excluded volume constraints, i.e. any pair of non-bonded atoms in the same molecule must be apart further than the sum of their hard sphere radii,
6. intermolecular excluded volume constraints, i.e. any pair of atoms in different molecules (including symmetry mates) must be apart further than the sum of their hard sphere radii.

Although the bond angles or lengths do vary a small amount among different molecules, their variation is not nearly as large as the grid spacing. They can be assumed fixed for the conformational matching problem. We call the above constraints the **full** set. The conformational matching problem is equivalent to maximizing the total density at atom positions while satisfying the **full** constraints. ConfMatch separates these geometric constraints into two sets, **local** and **non-local**, one for each stage of the algorithm. (**full** = **local** $\cup$ **non-local**) The bound preprocessing stage satisfies the **local** set:

1. angles of all bonds except flexible-ring-forming ones,

[1]If it is not known whether a double bond is cis or trans, ConfMatch can calculate the most likely isomer. Section 3.4.5 describes a simple extension to the algorithm that handles this case.

2. lengths of all bonds except flexible-ring-forming ones,
3. cis/trans and planarity of all double bonds except flexible-ring-forming ones,
4. chirality of all chiral centers.

A flexible ring must have at least one rotatable bond. For proteins, the flexible-ring-forming bonds refer to disulfide bonds only. In other molecules, we need to remove one bond from each flexible ring. The aromatic rings, such as the phenol ring, are rigid and not broken apart. If a rigid ring is puckered, such as the one in proline, but its exact puckering is unknown, ConfMatch can calculate the most likely one. A simple extension to the algorithm that handles this case is described in Section 3.4.5. Figure 3-1 shows how a bond is removed from a flexible ring of a complex molecule. Note that the number of atoms remains unchanged and the molecule is still a connected structure.

The search stage satifies the remaining constraints, the **non-local** set:

1. angles, lengths, and planarity of ring-forming bonds,
2. intramolecular excluded volume constraints,
3. intermolecular excluded volume constraints.

3.4.1 The Bound-preprocessing Stage

A molecule without any flexible rings has a tree-like structure. In a tree-structured molecule, the constraints in **local** do not impose any limit on the dihedral angles because steric clashes are allowed and there are no rings to close. The key observation enabling ConfMatch is: *Without any constraints on the dihedral angles, the optimization of density sum can be solved in time polynomial in the number of dihedrals by dynamic programming.* We will describe the optimization method later in this section. Because **local** is a subset of **full**, this maximized density sum must be an *upper bound* on the solution to the complete problem. In fact, the bound-preprocessing stage is solving a *relaxed* conformational matching problem because constraints in **non-local**

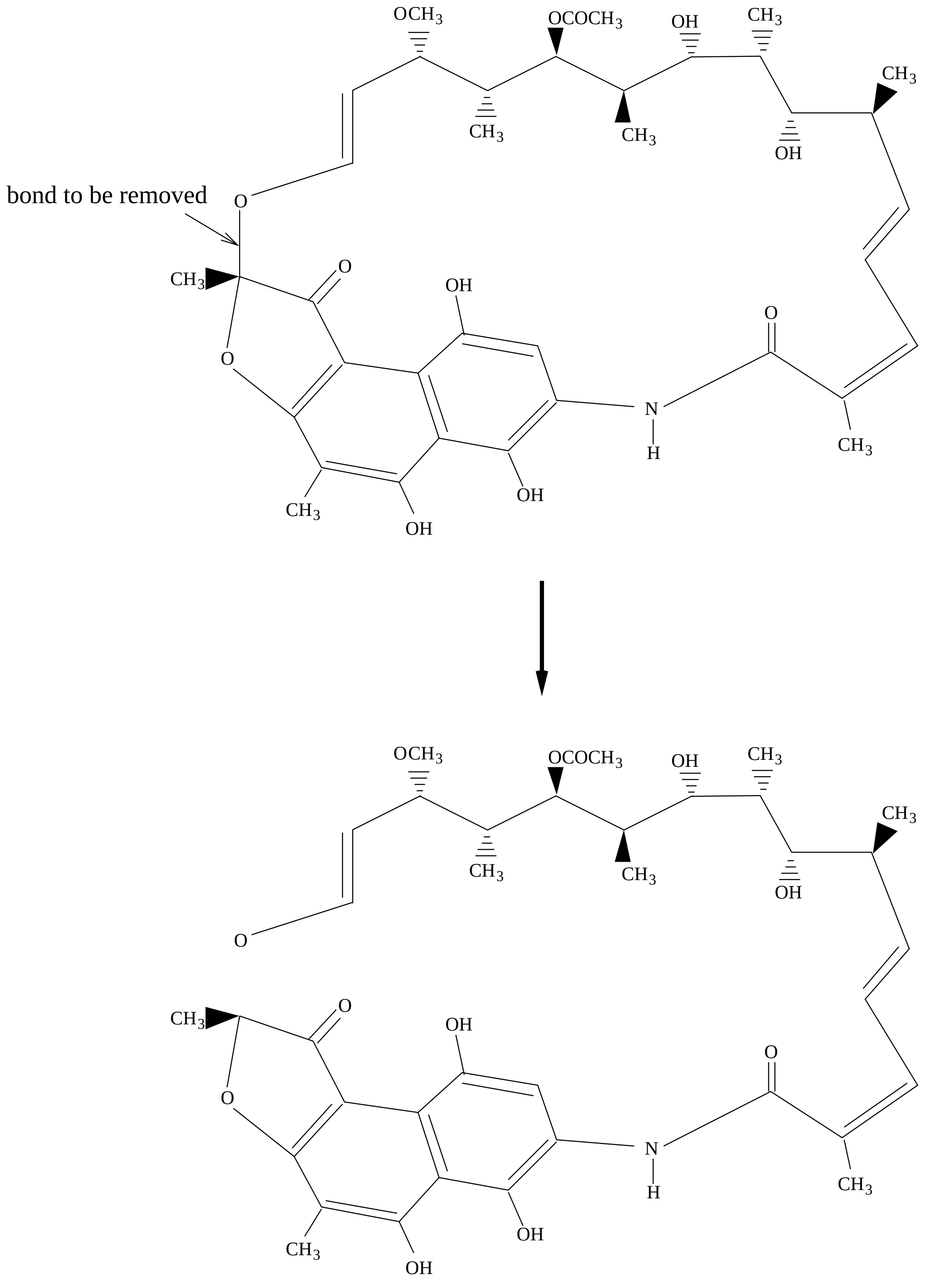


Figure 3-1: A bond is removed from the flexible ring of rifamycin SV [2], a macrocyclic molecule. The smaller rings are rigid and not broken apart. The **local** constraint set specifies the local geometries of the reduced molecule.

are ignored. Introducing these **non-local** constraints can never increase the optimal density, only possibly decrease it.

We are guaranteed that the solution (maximum density sum) to the **local** set is an upper bound of the **full** set. In order to maximize its usefulness, we also want the upper bound to be as tight as possible. Given a reasonable electron density distribution, this upper bound from **local** is likely to be close to the actual value for the following reasons:

- Most bonds are included in the calculation. The ring forming ones constitute a small percentage of the bonds in a typical macromolecule.
- All important local geometries, including angles, lengths, planarity, chirality, are considered in this stage.
- A reasonable density distribution has most of its density near the correct conformation. Figure 3-2 shows the crystal structure of a peptide and its density distribution with substantial error added. Most atoms do not locate at the peaks of the density map, but the density does tend to coalesce around the correct structure. This kind of distribution does not tend to induce many steric clashes even when they are allowed. However, as the quality of the density data deteriorates, more and more steric clashes will be induced. There will be more conformations with steric clashes and that have higher density sums than the best conformation without clashes. As a result, the gap between the upper bound and the actual value will grow.

Results in Section 3.5 will show that the difference between the upper bound and the solution is usually very small for data with low to medium level of error. This difference is sometimes less than the density of a single atom.

The bound-preprocessing stage uses a bottom-up, dynamic programming approach to calculate the upper bound. This approach begins with the observation that all molecules can be viewed as an assembly of small, rigid *fragments*. For instance, Figure 3-3 shows how a glycine molecule can be formed from two rigid fragments.

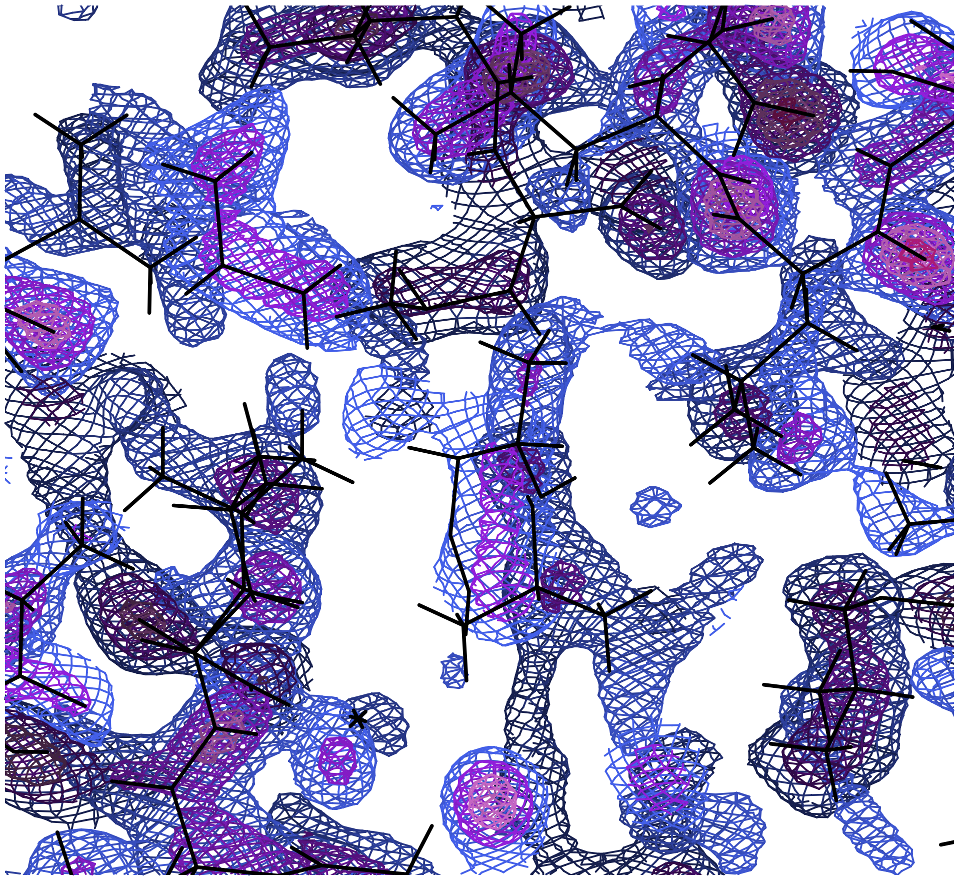


Figure 3-2: A partial structure of Alpha-1 [35], a designed alpha-helical peptide, and its electron density distribution at 2Å resolution with 50° (standard deviation) phase error.

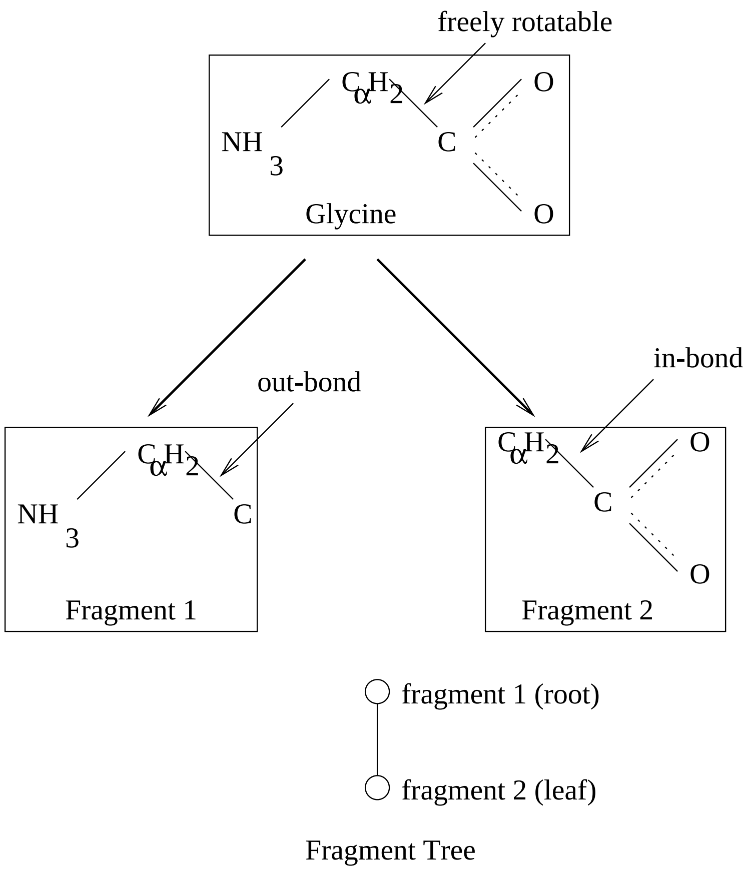


Figure 3-3: A glycine molecule can be separated into 2 rigid fragments. Its fragment tree has 2 nodes.

Note that all hydrogen atoms are "unified" with their heavier neighbors because we do not calculate the positions of hydrogens explicitly. Hydrogen atoms are not resolvable in typical electron density maps. The bond where the two fragments join is freely rotatable, giving the molecule an extra degree of freedom. A protein would have hundreds of these rigid fragments and hence hundreds of degrees of conformational freedom. For each rigid fragment, we define one bond to be its *in-bond* and some other bonds to be its *out-bonds*. In general, a fragment can have at most one in-bond and any number of out-bonds. For example, the fragment centered at the α-carbon of valine would have two out-bonds—one along the protein backbone and the other along the valine sidechain. Two fragments can be linked if one's in-bond coincides with the other's out-bond.

Given the input to ConfMatch—the primary structure of a molecule, the first step of the algorithm removes the ring-forming bonds and divides the rest of the structure into rigid fragments. If exact coordinates of the fragments are not given, one may use a library of standard fragments. Each fragment is connected to the others through in-bond-out-bond relationships. The fragments form a *fragment tree* that matches the tree structure of the molecule. The forks in the tree are formed by fragments with multiple out-bonds. The fragment tree of a protein would have a long stem corresponding to the main chain, as well as many short branches corresponding to the sidechains. The structure of fragments and their in-bond-out-bond relationships assure that the local geometry of the molecule is correct. Thus the **local** constraint set is satisfied by all conformations derived from the fragment tree.

The output of the bound-preprocessing stage is a large table of bounds. Each entry is written as $E_{n,\mathbf{j}}$ where n is a particular node in the fragment tree; $\mathbf{j}$ is the position of a pair of grid points ($\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix}$ and $\begin{pmatrix} x'_j \\ y'_j \\ z'_j \end{pmatrix}$). $E_{n,\mathbf{j}}$ stores the maximum density sum (satisfying only the **local** set) of the sub-fragment-tree of n (Figure 3-4), where n's in-bond is located at $\mathbf{j}$, that is, the first and second atom of the in-bond are located at $\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix}$ and $\begin{pmatrix} x'_j \\ y'_j \\ z'_j \end{pmatrix}$ respectively. There is an entry for every node in the fragment tree and every pair of grid points separated by the right bond length. Because a unit cell's neighbors are exact copies of itself, we need to consider only the grid points within a single cell. However, for pairs that cross the unit cell boundary, the out-of-bound point is translated back into the cell. This table's size is equal to

$$\boxed{\text{the size of the fragment tree}} \times$$
$$\boxed{\text{the number of pairs of grid points at bonding distance in a unit cell}}.$$

The number of fragments in a structure is equal to 1 plus its torsional degrees of freedom p. A typical bond is between 1 and 2Å, which is quite small compared with the unit cell of a crystal. Therefore the number of pairs of grid points at bonding distance is a small constant[2] multiple of the size of the grid. We can rewrite the table size as

$$\boxed{\text{a small constant}} \times (1+p) \times \boxed{\text{the number of grid points}}.$$

It is also the space complexity of the bound-preprocessing phase. This table can take a large amount of storage if the input molecule has many degrees of freedom and a large unit cell. For instance, the bounds table has more than 600 million entries for a short 12-residue peptide with an 11,000Å^3 unit cell. Section 3.4.2 describes several techniques that reduce the table size by a constant factor while retaining most of the information.

The bounds table is calculated node by node. The iteration over n, the node in the fragment tree, is the outer loop, while the iteration over $\mathbf{j}$, the bond placement, is the inner loop. This calculation is done in a bottom-up fashion. Initially, the bounds of the leaf fragments are computed. Since the subtree of a leaf node is the leaf itself, we only need to calculate the bound of a single rigid fragment. At every grid-point-pair, we simply do a uniform torsional sampling about the in-bond and store the maximum density sum. Figure 3-5 shows a torsional sampling of the second fragment of glycine. Suppose s torsional angles are uniformly sampled for a leaf node n whose in-bond is located at $\mathbf{j}$. The i-th sample is generated by a rotation of $\theta_i = \frac{2\pi i}{s}$ around n's in-bond. This rotation corresponds to the following rigid transform [7]:

$$\begin{bmatrix} k_xk_xv\theta_i + c\theta_i & k_xk_yv\theta_i - k_zs\theta_i & k_xk_zv\theta_i + k_ys\theta_i & x_j \\ k_xk_yv\theta_i + k_zs\theta_i & k_yk_yv\theta_i + c\theta_i & k_yk_zv\theta_i - k_xs\theta_i & y_j \\ k_xk_zv\theta_i - k_ys\theta_i & k_yk_zv\theta_i + k_xs\theta_i & k_zk_zv\theta_i + c\theta_i & z_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_j \\ 0 & 1 & 0 & -y_j \\ 0 & 0 & 1 & -z_j \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[2]Actually this constant is the number of grid points on a sphere with radius equal to the bond length. This number is proportional to the square of grid spacing, or 2/3 power of the number of grid points if we assume uniform grid spacing.

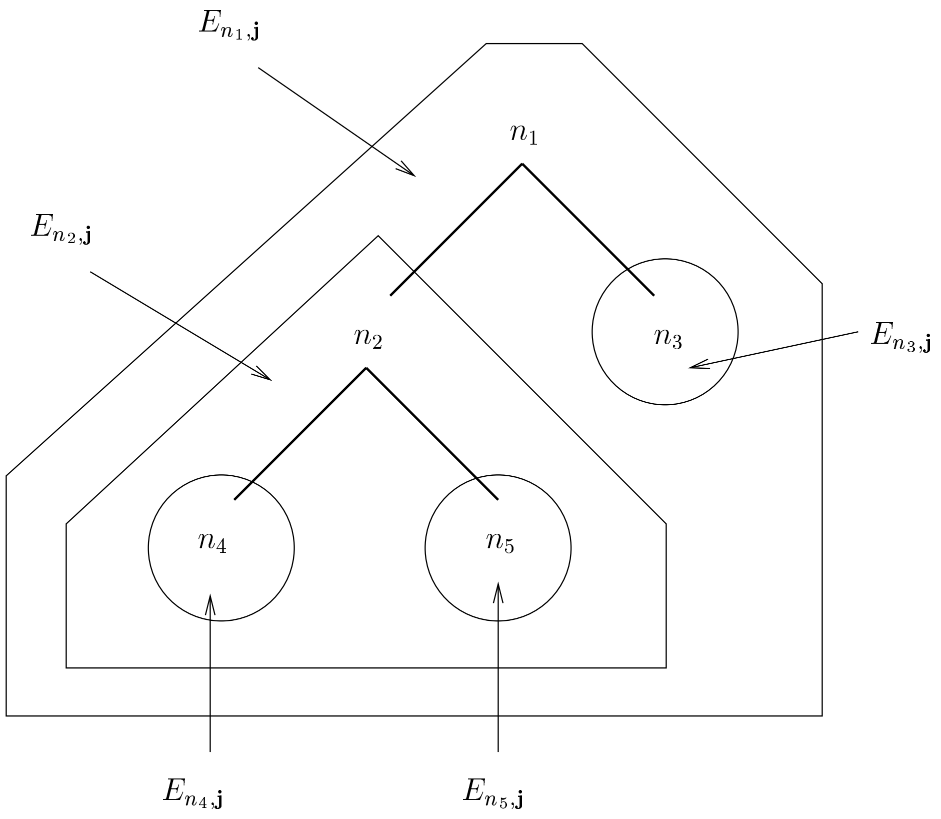


Figure 3-4: A fragment tree and its entries in the bounds table. Each set of entries stores the upper bounds of a sub-fragment-tree.

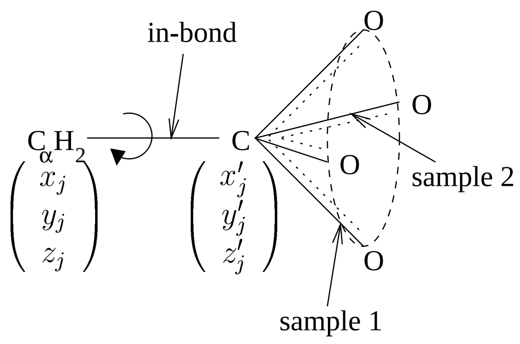

in-bond
O
O
O
O
$C_\alpha H_2$
C
$\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix}$
$\begin{pmatrix} x'_j \\ y'_j \\ z'_j \end{pmatrix}$
sample 2
sample 1
Fragment 2

Figure 3-5: A torsional sampling about the in-bond of the second fragment of glycine.

Where $\begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix}$ is the unit vector in the direction of $\begin{pmatrix} x'_j - x_j \\ y'_j - y_j \\ z'_j - z_j \end{pmatrix}$; $c\theta_i = \cos\theta_i$, $s\theta_i = \sin\theta_i$, and $v\theta_i = 1 - \cos\theta_i$. Section 3.4.2 describes some techniques which perform the torsional sampling without using these transformation matrices.

Let $e^i_{n,\mathbf{j}}$ be the density sum of the i-th sample. This value is the density sum of a single fragment at a particular configuration. It can be calculated in time proportional to the number of atoms in the fragment minus 2. To avoid double counting, we do not include the 2 atoms of the in-bond. These atoms are included in the density sum of the parent fragment. Then

$$E_{n,\mathbf{j}} = \max_{i=1}^{s} e^i_{n,\mathbf{j}}. \tag{3.1}$$

The inner nodes's bounds can be calculated based on their children's values. Suppose n is an inner node whose children are nodes $n_1, n_2, \ldots, n_m$. At every grid-point-pair, we also do a uniform torsional sampling about the in-bond of n. At each sample, we also find the positions of out-bonds 1 to m. Let $\mathbf{j}^i_l$ be the position of the lth out-bond at sample i of node n. Instead of maximizing n's density sum alone, we maximize the sum plus the bounds of its children:

$$E_{n,\mathbf{j}} = \max_{i=1}^{s}(e^i_{n,\mathbf{j}} + \sum_{l=1}^{m} E_{n_l,\mathbf{j}^i_l}) \tag{3.2}$$

If we define $S^i_{n,\mathbf{j}} = e^i_{n,\mathbf{j}} + \sum_{l=1}^{m} E_{n_l,\mathbf{j}^i_l}$,

$$E_{n,\mathbf{j}} = \max_{i=1}^{s} S^i_{n,\mathbf{j}}.$$

$S^i_{n,\mathbf{j}}$ can be consider the maximum density sum (satisfying only the **local** set) of n's sub-fragment-tree, provided that sample i is chosen for fragment n. Since we calculate the bounds from the leaves up towards the root, we can simply look up $E_{n_l,\mathbf{j}^i_l}$ from the table. Let u be the time required to calculate each entry in the table.

$$u \propto s \times (\boxed{\text{average number of atoms in a fragment}} - 2 + \boxed{\text{average branching factor of the fragment tree}}).$$

The bound-preprocessing stage is finished after we calculate the bounds of the root node. One can prove that the entries are indeed the maximal density sum by an induction on the fragment tree. The running time of this stage is

$$u \times \boxed{\text{the size of the fragment tree}} \times \boxed{\text{the number of pairs of grid points at bonding distance}}.$$

3.4.2 Optimizations of the Bound-preprocessing Stage

Although the bound-preprocessing stage is a polynomial time and space method, it can take days of CPU time and gigabytes of storage for even a small protein. This section describes several techniques that reduce the space requirement and the running time by a constant factor. Table 3.2 is a comparison of these optimization methods. ConfMatch integrates all of them to solve large problems efficiently.

Common Subtree Elimination

From the semantics of the bounds table, we see that two nodes with identical sub-fragment-trees would have the same bounds. That is, if the subtree of n is identical to that of n', then

$$E_{n,\mathbf{j}} = E_{n',\mathbf{j}}$$

Technique	Time Optimization	Space Optimization
Common Subtree Elimination	Depends on molecular structure	
Precomputing Torsional Sampling	Some	None
Utilizing Crystallographic Symmetry	By the number of asymmetric units in the unit cell	
Reducing the Size of Each Table Entry	None	By a factor of 2

Table 3.2: Comparison of different optimizations of the bound-preprocessing stage.

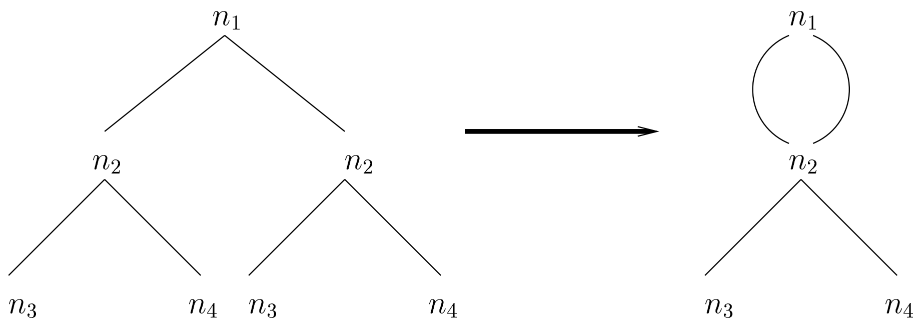


Figure 3-6: 3 nodes of a fragment tree are eliminated by common subtree elimination.

for all $\mathbf{j}$. We can avoid redundant calculations if common subtrees can be discovered and merged. Figure 3-6 shows this operation on the fragment tree graphically. We call this procedure *common subtree elimination*, analogous to the common subexpression elimination technique in compiler optimization. This operation can be applied repeatedly on the fragment tree to reduce its size. Thus decreasing the size of the bounds table and the computation time proportionately.

If we apply common subtree elimination to a protein, none of the nodes on the main chain can be eliminated because each has a unique subtree. However, all sidechain nodes can be merged according to their amino acid labels. All valines will be merged into a single branch, all leucines into another, etc. In effect, we calculate the bounds of each amino acid type only once, regardless of the size of the protein. The advantage of this optimization grows with the level of repetition in the amino acid sequence. For a large protein, the calculation of sidechain bounds is amortized over many residues. Each additional residue usually adds only 2 nodes to the tree, corresponding to the additional ϕ and ψ angles on the main chain.

Precomputing Torsional Sampling

Calculating each entry in the bounds table requires a torsional sampling about an in-bound. With some precomputation, we can avoid the expensive multiplication by a 4×4 transformation matrix at every sample. The precomputation involves a pseudo uniform rotational sampling using Lattman's method[24]. This method generates a number of rotational matrices $(R_1, R_2, \ldots)$ that are approximately uniformly distributed over the space of all rotational transforms. Each rotation differs from its adjacent neighbors by a fixed angle θ_L. After the molecular structure is divided into rigid fragments, we apply these rotations to each fragment. It gives us a pseudo uniform sampling of the rotational configurations of all fragments. Let N be the initial configuration of a fragment. Its rotational configurations are stored as $R_1(N), R_2(N), \ldots$ These configurations are classified based on the orientations of their in-bonds. Let $\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$ and $\begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix}$ be the locations of the two in-bond atoms of $R_i(N)$. Each $R_i(N)$ is classified according to the vector $\begin{pmatrix} x'_i - x_i \\ y'_i - y_i \\ z'_i - z_i \end{pmatrix}$. To sample the torsional space of a particular in-bond, we simply select the subset with the correct in-bond orientation. For example, if we are to sample an in-bond at $\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix}$ and $\begin{pmatrix} x'_j \\ y'_j \\ z'_j \end{pmatrix}$, $R_i(N)$ would be selected if $\begin{pmatrix} x'_i - x_i \\ y'_i - y_i \\ z'_i - z_i \end{pmatrix} = \begin{pmatrix} x'_j - x_j \\ y'_j - y_j \\ z'_j - z_j \end{pmatrix}$. The required configuration is generated

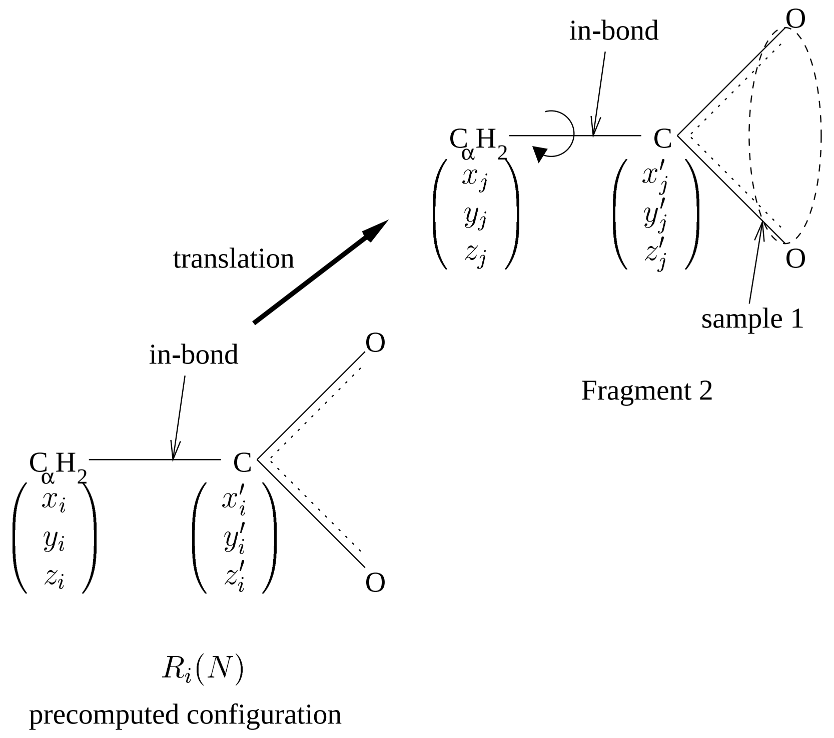

in-bond
O
$C_\alpha H_2$
C
$\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix}$
$\begin{pmatrix} x'_j \\ y'_j \\ z'_j \end{pmatrix}$
O
translation
sample 1
Fragment 2
in-bond
O
$C_\alpha H_2$
C
$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$
$\begin{pmatrix} x'_i \\ y'_i \\ z'_i \end{pmatrix}$
O
$R_i(N)$
precomputed configuration

Figure 3-7: A torsional sample of the second fragment of glycine is generated from a precomputed configuration.

by translating $R_i(N)$ by $\begin{pmatrix} x_j - x_i \\ y_j - y_i \\ z_j - z_i \end{pmatrix}$. Figure 3-7 shows an example configuration generated by this technique. For every torsional sample, we need to calculate only a simple translation instead of the full 4×4 transform. These configurations sample each torsional angle at intervals of about θ_L.

Both θ_L and the grid spacing can affect the sampling of conformations. For most applications, a fixed θ_L of about 20^o is sufficient. If one wants to assure that every possible conformation on the grid is sampled (completeness), one must choose θ_L based on the grid spacing and the geometries of fragments: Given a fragment, we find the atom farthest away from the in-bond axis—the axis of rotation (Figure 3-8). Let D be the distance between this atom and the axis. Let g be the grid spacing. We need to guarantee that between two adjacent samples, all atoms move by distance g

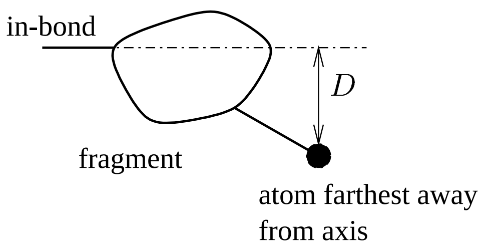


Figure 3-8: To assure completeness, it is necessary to choose θ_L based on the distance between the in-bond axis and the atom farthest away from it.

or less:

$$\theta_L D \leq g.$$

We need to choose $\theta_L \leq g/D$. Larger fragments usually gives a bigger D value. Using this scheme thus implies choosing small θ_L for a fine grid with large fragments, and vice versa.

Utilizing Crystallographic Symmetry

If a crystal has rotational or screw symmetry (crystals of all space groups except P1), its unit cell is composed of several copies of the asymmetric unit. The bounds table would have the same symmetry as the crystal if we preserve the symmetry property throughout our calculation. Specifically, preserving the symmetry has the following requirements:

- The electron density distribution has the same symmetry as the crystal. This is always true with appropriate input data.
- The grid has identical symmetry as the crystal. i.e. the grid is invariant under symmetry operations of the crystal, as well as translation by one unit cell length along any of the 3 axes. If the unit cell has 2, 3, 4, or 6 fold axes of rotation, the grid must have the same axes. This requirement does not reduce the generality of ConfMatch because one can always find an appropriate grid for any kind of unit cell.

- The rotational sampling preserves the rotational symmetry of the crystal. This property, together with the symmetry of the grid, assures that if a particular configuration of a fragment is sampled, its symmetric counterparts will also be sampled.

The bounds table will have identical symmetry as the crystal if all of the above requirements are fulfilled. We need to calculate and store only the asymmetric unit of the bounds table, cutting the time and space requirement to a fraction of the original. For space group $P2_1$, this optimization results in a 2-fold reduction in time and space of the bound preprocessing stage.

Reducing the Size of Each Table Entry

Electron density values are usually stored as floating point numbers, which usually takes 4 bytes of memory. The only operations on these values by ConfMatch are additions (for summing density values) and comparisons (for choosing the maximum sum). If we use fixed point instead of floating point values, these operations only propagate errors linearly. In other words, the magnitude of the error is proportional to the number of operations performed. Without loosing much accuracy, we may use a properly normalized integer representation of density values. ConfMatch uses 2-byte short integers to represent these values as well as entries in the bounds table. This representation cuts down the size of the bounds table by one half.

3.4.3 The Search Stage

The output of the bound preprocessing stage is a large table of upper bounds. Without any search, it is possible to calculate a "greedy" structure that maximizes the density sum. This "greedy" structure, satisfying **local** but not **non-local**, is formed by tracing the highest density path through the bounds table. First, one finds the best location of the in-bond of the root fragment. Let n_{root} be this root node. $\mathbf{j}^* = \arg\max_{\mathbf{j}} E_{n_{\text{root}},\mathbf{j}}$ is the pair of grid points where the in-bond locates. Then one finds $\arg\max_{i=1}^{s} S^i_{n_{\text{root}},\mathbf{j}^*}$ which gives the torsion angle of the in-bond. This provides the exact configuration of the root fragment, from which n_{root}'s out-bonds' positions can be derived. If one applies this procedure recursively down the fragment tree, all torsional angles can be selected and the "greedy" structure is found. Unfortunately, this "greedy" structure may have rings that do not close, or atoms that clash with one another. Therefore, it is necessary to perform a conformational search to find a structure without any of these violations.

The constraints to be satisfied in the search stage, the **non-local** set, are embodied in two distance matrices [8]—one intramolecular and one intermolecular. Each distance matrix describes a lower and an upper bound of the distance between every pair of atoms. Obviously, the diagonal entries of the matrices are zeroes. The intramolecular and intermolecular matrices represent the intramolecular and intermolecular constraints respectively.

The intramolecular distance matrix simply specifies the ranges of distances within a single molecule. This matrix is derived from the intramolecular excluded volume constraints, as well as the local geometries of all bonds (including ring-forming ones). The excluded volume constraints involve every pair of atoms more than 3 bonds apart in the structure. The lower bound between a pair of atoms is set to be the sum of their van der Waal's radii minus a small tolerance. The local geometries, such as bond lengths and angles, of all bonds become tight upper and lower bounds of the right atoms in the matrix. We use the triangle inequality to smooth and propagate the bounds to every entry in the matrix.

The intermolecular distance matrix specifies the distances between one molecule and all of its symmetry mates. To verify the compliance of the intermolecular matrix, it is necessary to calculate and check several copies of the molecule. This matrix is derived from intermolecular excluded volume constraints alone. Unlike the intramolecular constraints, these excluded volume constraints involve all pairs of atoms. The lower bound between a pair of atoms is set to be the sum of their van der Waal's radii minus a small tolerance. The intermolecular matrix consists of no upper bounds, only lower bounds.

The goal of the search stage is to place the root fragment and find a set of dihedral angles for the fragment tree such that the distance matrices are satisfied and the density sum is maximized. Since our problem is discretized, it can be formulated as one of searching a state space:

- The initial state is the null structure.
- The successor states of the initial state consist of all **j** pairs which can potentially be the in-bond of the root fragment.
- Every intermediate state is a partial structure of the molecule with a number of *open out-bonds*. These open out-bonds are ones that do not yet have fragments connected to them. The successor states of an intermediate state consist of every torsional sampling of every open out-bond.
- The goal states are structures that satisfy the distance matrices and do not have any open out-bonds. All fragments' positions are specified.
- The value of a state is the sum of density of fragments in the partial structure. The value is independent of the path taken from the initial state.

The problem for the search stage is to move from the initial state to the goal state with the highest value (Figure 3-9). Since the value is path independent, the problem is equivalent to finding the highest value goal state. This goal state can be reached through multiple paths which differ because they order the open out-bonds differently. In fact, all goal states can be reached by any ordering of the open out-bonds. Because of the path independence property, we simplify this problem from a graph search into a tree search: Every intermediate state commits to a particular open out-bond and branches on only its torsional samples. All other open out-bonds are deferred. This commitment assures that there are no alternative orderings to generate a particular structure. There is only one path from the initial state to any other state. We have reduced the graph to a search tree. The heuristic for selecting an open out-bond will be described in Section 3.4.4.

One possible approach to the problem is to use an informed search method like the A* algorithm [37]. When A* reaches a state t, it calculates an evaluation function,

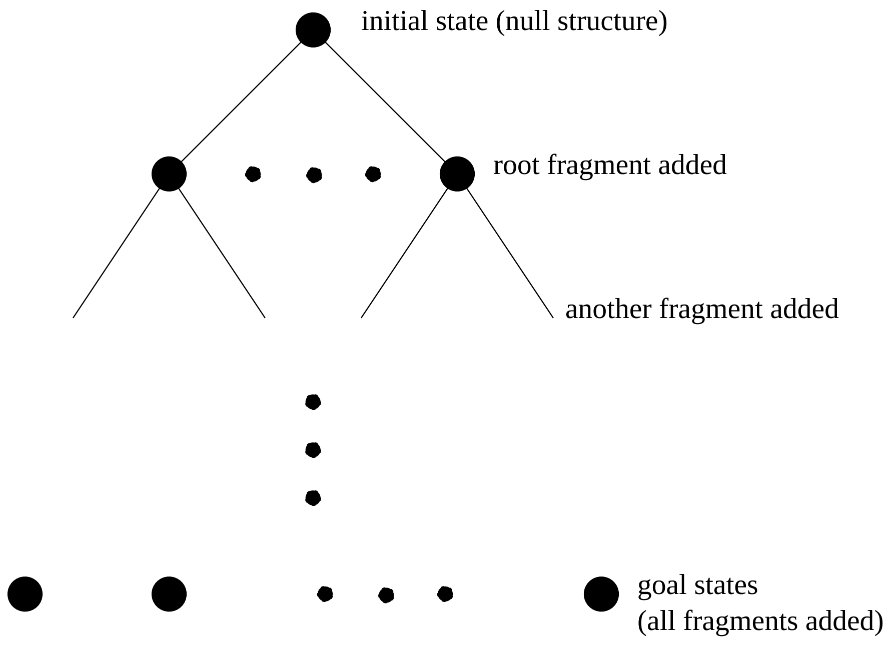


Figure 3-9: Starting from the initial state, the goal of the search stage is to find the highest-value goal state.

$f(t)$, that is the sum of the value of the current state, $g(t)$, and the estimated difference between the current state and the best reachable goal state, $h(t)$. ($f(t) = g(t) + h(t)$) Here $g(t)$ is the density sum of the partial structure, whereas $h(t)$ is the upper bound calculated from the bound preprocessing stage. Suppose state t has b open out-bonds, corresponding to fragments $n_1, \ldots, n_b$, whose in-bonds locate at $\mathbf{j}_1, \ldots, \mathbf{j}_b$ respectively.

$$h(t) = \sum_{i=1}^{b} E_{n_i, \mathbf{j}_i}$$

h is an admissible heuristic because the upper bounds $E_{n_i, \mathbf{j}_i}$ never underestimates. From the theory of A*, we are guaranteed that the search is complete, optimal, and optimally efficient[3]:

Completeness A* is guaranteed to find a goal state when there is one.

[3]Traditionally, A* finds the *lowest cost* goal state with an admissible heuristic that never *overestimates*. Here we reverse both the objective and the admissibility property, but the theory still applies. Because the search tree/graph is acyclic, it is impossible to increase the path value $g(t)$ indefinitely. Thus all upper bounds, $h(t)$, are well defined.

Optimality A* finds the highest-value goal state when there are several different goal states.

Optimally efficient No other optimal algorithm is guaranteed to search fewer states than A*.

The optimality property implies that the highest density structure would be found. Unfortunately, A*'s optimal efficiency requires storing the entire f-value contour—all intermediate states whose f-value is within a certain range. The f-value contour's size is exponential in the search depth because our heuristic function, $h(t)$, has large errors[4]. The search depth is equal to the size of the fragment tree, $1 + p$. A*'s memory requirement is $O(s^{1+p})$ where s is the number of torsional samples. On a large problem, this contour may need more memory than practically available.

Iterative deepening A* (IDA*)[37] is a variant of A* that has the same completeness and optimality properties, but is not optimally efficient. ConfMatch uses IDA* for the conformational search because it can trade more CPU time for using less memory. IDA* performs multiple iterations of depth-first searches. Each iteration uses a particular f-value limit (f_{lim})—a guess of the best possible value. Each depth-first search determines whether a solution exists above f_{lim}. If a solution is found, IDA* terminates. Otherwise, we reduce the guess, f_{lim}, and perform another iteration.

During every depth-first search, a state t is expanded only if

$$f_{\text{lim}} \leq g(t) + h(t).$$

Thus, each iteration expands all nodes inside the contour of the current f_{lim}. If f_{lim} can be set to the value of the best goal state, the depth-first search will explore exactly the same nodes as A*. Once the search inside a given contour has been completed, a

[4]The f-value contour will grow exponentially if the error in the heuristic function grows faster than the logarithm of the actual value. In mathematical notation, the condition for exponential growth is that

$$|h(t) - h^*(t)| > O(\log h^*(t))$$

where $h^*(t)$ is the *true* difference between t and the best reachable goal. Here $h(t)$'s error is at least proportional to t's uninstantiated fragments. i.e. $|h(t) - h^*(t)| \geq O(h^*(t))$.

new iteration is started using a lower f_{lim} for the next contour. IDA* terminates if one iteration finds some solutions. IDA*'s space requirement is the same as depth-first search. It requires $O(s(1+p))$ storage, proportional to the longest path it explores, which is much less than that of A*.

IDA* explores some states multiple times during different iterations. However, the time overhead of IDA* over depth-first search is rather small [34]. The reason is that in an exponential search tree, almost all of the nodes are in the bottom level, so it does not matter that the upper levels are expanded multiple times. IDA* spends most of the CPU time on the last search tree. If the last f_{lim} is close to the best goal value, depth-first search will search a few more nodes than A*. IDA*'s efficiency will be close to optimal. However, if the last f_{lim} is much smaller than the best goal value, IDA* will search many more nodes than A*.

ConfMatch uses an ad hoc heuristic for choosing f_{lim}'s. Since $E_{n_{\text{root}},\mathbf{j}}$ is an upper bound for the root fragment at a particular in-bond orientation, $M = \max_{\mathbf{j}} E_{n_{\text{root}},\mathbf{j}}$ must be the upper bound for the entire structure. Clearly, $f_{\text{lim}} \leq M$. ConfMatch selects f_{lim}'s following the sequence $(1-\beta)M, (1-2\beta)M, \ldots$, where β is a small constant about 0.001.

3.4.4 Optimizations of the Search Stage

Conformational search is intrinsically very expensive in the worst case. With the bounds table and the IDA* algorithm, it is still very time consuming to solve any large structure. This section describes several techniques which accelerate the search. None of these techniques can change the fact that the problem is exponentially complex in the worst case. However, in practice, the search stage with these optimizations can take much less time than the bound-preprocessing stage given good density data.

Constraining the Search by the Distance Matrices

From the state space of the search, we notice that every path adds a fragment at every step. No fragment is ever removed along a valid path. If an intermediate state's partial structure violates some constraints in the distance matrices, it can never lead to any goal state. It can be safely discarded from the search. Because of this observation, we check every partial structure against the distance matrices and terminate a branch if any distance bounds are violated.

The Most-constrained-variable Heuristic

During the search, every intermediate state must select an open out-bond to instantiate. This selection has a drastic effect on the efficiency of search. We found that the most-constrained-variable heuristic [37] is the most efficient one among several candidates. At every state, we count how many options are still available for each open out-bond, given the choices made so far. We keep track of the options allowed by the f-value limit and the distance matrices. Suppose state t has b open out-bonds, corresponding to fragments $n_1, \ldots, n_b$, whose in-bonds locate at $\mathbf{j}_1, \ldots, \mathbf{j}_b$ respectively. A torsional sample l is an available option of an open out-bond n_i at $\mathbf{j}_i$ if it does not have any distance violations with the existing structure, and satisfies the condition

$$g(t) + S^l_{n_i,\mathbf{j}_i} + \sum_{k \neq i} E_{n_k,\mathbf{j}_k} \geq f_{\text{lim}}.$$

$g(t)$ is the density of the current partial structure. $S^l_{n_i,\mathbf{j}_i}$ is the upper bound of n_i's sub-fragment-tree if sample l is chosen. $\sum_{k \neq i} E_{n_k,\mathbf{j}_k}$ is the upper bound of all other open out-bonds. The sum of the 3 terms is an upper bound of the solution density if l is chosen at t. This inequality ensures that sample l can potentially lead to a solution better than f_{lim}. At each point in the search, the open out-bond with the fewest such options is chosen to have its fragment assigned. In this way, the branching factor in the search tends to be minimized. Intuitively, this heuristic selectively instantiates the fragment with many close previously-placed neighbors and little conformational freedom.

Utilizing Crystallographic Symmetry

Just like in the previous stage, the search stage can be accelerated by exploiting crystallographic symmetry. If a structural solution exists, its symmetry mates are also solutions with identical density sums. We can limit the root fragment to be in one of the asymmetric units, instead of the entire unit cell. The search is reduced by a factor equal to the number of asymmetric units in the unit cell.

Improving the Bounds Table by Memoization

The technique of memoization speeds up programs by saving the results of computation and retrieving them when needed later. During the conformational search, all of the dead ends are caused by violations of the distance matrix constraints. Many dead end structures may share a common "core" where the violations occur. If we can extract some knowledge from every dead end, much of the search may be avoided. This "learning" is achieved by updating the bounds table by memoization. During a single depth-first search, some entries in the bounds-table may be read multiple time. Many paths in the search tree may have the same open out-bond placement and hence require the same upper bound[5]. If an entry can be lowered after the first access, subsequent reads will obtain a more accurate value and may avoid some dead ends.

Recall that an entry in the bounds table, $E_{n,\mathbf{j}}$, stores the greedy maximal sum of the sub-fragment-tree of n, where n's in-bond is located at $\mathbf{j}$. Memoization defines a slightly different $E^l_{n,\mathbf{j}}$ to be the upper bound of the density sum of the sub-fragment-tree of n, *satisfying the distance matrices*, where n's in-bond is located at $\mathbf{j}$, i.e. $E^l_{n,\mathbf{j}}$ is an upper bound of valid solutions of the sub-fragment-tree. This change in semantics does not affect other parts of the search, but allows a tighter upper bound. We maintain the invariant that $E^l_{n,\mathbf{j}} \leq E_{n,\mathbf{j}}$ for all $n, \mathbf{j}$.

[5]The search tree's size is exponential in the number of fragments, but the bounds table's size is only linear. As the number of fragments increases, the ratio between the two sizes will grow rapidly. Therefore, on average, every entry in the table will be accessed more and more times as the size of molecule increases.

Initially, $E^l_{n,\mathbf{j}} = E_{n,\mathbf{j}}$ for all $n, \mathbf{j}$. IDA* performs depth-first searches with different f-value limits (f_{lim}). During the search, suppose a particular state t has b open out-bonds, corresponding to fragments $n_1, \ldots, n_b$, whose in-bonds locate at $\mathbf{j}_1, \ldots, \mathbf{j}_b$ respectively. We focus on the sub-search-tree of t. If the precondition

$$f_{\text{lim}} \leq g(t) + \sum_{i=1}^{b} E^l_{n_i,\mathbf{j}_i} \tag{3.3}$$

is satisfied, the sub-search-tree will be explored depth-first.

Without loss of generality, we assume n_1 is selected for instantiation. During the search, if a structural solution is found with density d, this solution is recorded and f_{lim} is immediately raised to $d + \epsilon$ where ϵ is the smallest improvement in solution we accept. While traversing the sub-search-tree, we record every pair of fragments that are involved in violations of the distance matrices.

After searching the sub-search-tree, f_{lim} is raised above all solutions. If we were to perform the search again with the updated f_{lim}, no solution would be found. If all violations of the distance matrices occur within the sub-fragment-tree of n_1, the sub-fragment-tree is the "core" of violations. Other parts of the fragment-tree have not had any violations and their bounds shall not be changed. We may lower $E^l_{n_1,\mathbf{j}_1}$ to the level that Equation 3.3 will not be satisfied with the updated f_{lim}. This requires $E_{n_1,\mathbf{j}_1} \leq f_{\text{lim}} - g(t) - \sum_{i=2}^{b} E^l_{n_i,\mathbf{j}_i}$.

ConfMatch uses the following rules to update the bounds table after searching the sub-search-tree of t.

1. If all violations of the distance matrices occur within the sub-fragment-tree of n_1, $E^l_{n_1,\mathbf{j}_1}$ is updated to be

$$\begin{aligned} \min(\; & f_{\text{lim}} - g(t) - \textstyle\sum_{i=2}^{b} E^l_{n_i,\mathbf{j}_i}, \\ & E^l_{n_1,\mathbf{j}_1}, \\ & \textstyle\max_{i=1}^{s} S^i_{n_1,\mathbf{j}_1}). \end{aligned} \tag{3.4}$$

2. If some violations involve fragments outside of the sub-fragment-tree of n_1, $E^l_{n_1,\mathbf{j}_1}$

is updated to be

$$\begin{aligned}\min(\quad & E^{l}_{n_1,\mathbf{j}_1}, \\ & \max_{i=1}^{s} S^{i}_{n_1,\mathbf{j}_1}).\end{aligned} \tag{3.5}$$

The updated entries are always upper bounds of valid solutions. The term $E^{l}_{n_1,\mathbf{j}_1}$ ensures that the bounds are monotonically decreasing. The term $\max_{i=1}^{s} S^{i}_{n_1,\mathbf{j}_1}$ follows Equation 3.2. These terms are necessary for the consistency of the bounds table. Appendix C gives a rigorous correctness proof of these update rules.

Memoization lowers the upper bounds continually during the search. The better bounds means Equation 3.3 is satisfied less frequently. IDA* is able to avoid many dead ends it would otherwise need to explore. On large molecules, memoization sometimes results in speedup of an order of magnitude.

3.4.5 Algorithm Extension

Sometimes details of a molecule's local covalent structure is not know perfectly before its crystal structure is solved. There may be some ambiguities in parts of the molecule. For example, a double bond can be either cis or trans; a rigid ring can pucker in one of several ways. ConfMatch can resolve these ambiguities by incorporating the different isomers into a single search. The output of ConfMatch will be the "best matching" isomer at its "best matching" conformation.

We illustrate this extension by a simple example. Suppose it is not known whether a fragment n is a cis or trans double bond (Figure 3-10). Obviously, we could have run ConfMatch twice, once with n fixed to the cis configuration and once to trans. Then we simply pick from the two solutions the one with the higher sum of density. Let us assume the cis conformation has a higher sum of density.

The exact same solution will be found by an extension of ConfMatch, but it will use only a little more resources than a single run of the algorithm. We call the cis and trans configurations n_{cis} and n_{trans} respectively. Their entries in the bounds table, $E_{n_{cis},\mathbf{j}}$ and $E_{n_{trans},\mathbf{j}}$, are calculated separately. The bounds of their parent fragment, n_{parent}, are calculated from the maximum of n_{cis}'s and n_{trans}'s bounds. Without

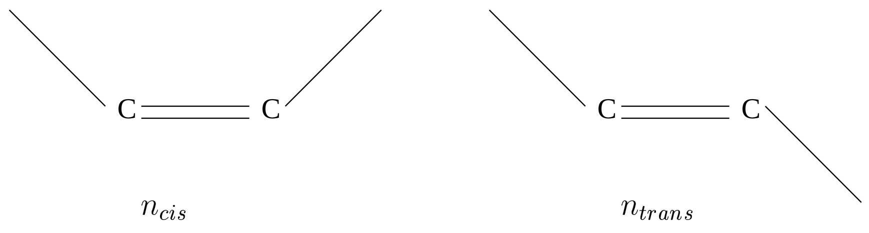


Figure 3-10: A fragment is ambiguously defined as either a cis or a trans double bond.

loss of generality, we assume that n is the only child of n_{parent} in the fragment tree. Following Equation 3.2, the bounds of n_{parent} are calculated by

$$E_{n_{parent},\mathbf{j}} = \max_{i=1}^{s}(e^{i}_{n_{parent},\mathbf{j}} + \max(E_{n_{cis},\mathbf{j}^i}, E_{n_{trans},\mathbf{j}^i}))$$

Thus $E_{n_{parent},\mathbf{j}}$ stores the upper bound of both cis and trans fragments. The extra time and space required for the bounds preprocessing stage is equivalent to adding a single fragment.

The search stage is little changed. When ConfMatch needs to find the torsional angle of the in-bond of n, it simply searches the torsional angles of both n_{cis} and n_{trans}. It appears as if n has twice as many samples as other fragments. However, if we have good density data, the cis solution is likely to have much higher density sum than the trans solution. The bounds of the best n_{cis} samples will be much higher than those of n_{trans}. Thus the n_{trans} options will not be explored. The time spent on the search stage will increase only a little. In fact, the conformational search will probably explore a few more nodes than the search with n fixed to the cis configuration, but their solutions are exactly the same. The worst case scenario occurs when the cis and trans solutions have identical density sums, in which case we must explore both sets of options. The search stage will take time equal to searching for the two solutions separately.

Similarly, we can generalize this extension to molecules with multiple ambiguous fragments. An entire sub-fragment-tree may also be ambiguously defined. For instance, we may specify a sidechain as one of several possible amino acids. Furthermore, we can specify only the length of a peptide, but leave all sidechains ambiguous. The output will be the "best matching" peptide at its most favored conformation. This application may be a useful tool for rational drug design (Section 3.1).

3.5 Results and Discussion

We have carried out two detailed studies using the ConfMatch algorithm to explore its performance and illustrate its range of applicability. The first study involves a designed alpha-helical peptide. The second study involves a small protein (crambin). The ConfMatch algorithm is implemented using 3000 lines of C and the XtalView crystallography library [30]. The results given below were run on a 533MHz Digital Alpha 21164PC processor with 512MB of RAM.

3.5.1 Alpha-1: A Designed Alpha-helical Peptide

The first test of the ConfMatch algorithm is a designed alpha-helical peptide—Alpha-1 [35] (PDB [3] code 1BYZ). This peptide has 12 residues:

Ac-GLU-LEU-LEU-LYS-LYS-LEU-LEU-GLU-GLU-LEU-LYS-GLY

The N terminus of the peptide is acetylated. Alpha-1's native structure is a 4-helix bundle. The crystal of Alpha-1 diffracts to 0.9Å resolution with 23681 structure factors. The space group of the crystal is P1. The unit cell dimensions are $|\mathbf{a}| = 20.846$Å, $|\mathbf{b}| = 20.909$Å, $|\mathbf{c}| = 27.057$Å with angles $\alpha = 102.40^o$, $\beta = 95.33^o$, and $\gamma = 119.62^o$. There is a single bundle with 4 chains in each unit cell.

Since the 4 chains are mostly identical, ConfMatch tries to determine only one of them. It simply chooses the chain with the highest density sum. This target molecule has 102 non-hydrogen atoms, 55 free dihedral angles, and 61 degrees of freedom total. Alpha-1 has no flexible rings and therefore no ring-forming bonds. We use a set of fragments with standard geometries, as well as standard van der Waal's radii for inter and intramolecular excluded volume constraints. Before common subtree elimination, the fragment tree has 56 fragments. After the elimination, only 34 fragments are left.

Common subtree elimination has reduced the time and space of the first stage by 39%. Unfortunately, the crystal's P1 symmetry means that its asymmetric unit is the entire unit cell. We are unable to use the crystal's symmetry to reduce the size of the bounds table further.

A grid of $42 \times 42 \times 55$ was selected for the Alpha-1 unit cell. The grid spacing is approximately 0.5Å in every dimension. We use Lattman's method to generate 3686 rotational transforms for each fragment. The spacing of the rotations, θ_L, is 16.36°. The size of the bounds table of each fragment ranges from 13,291,740 to 20,956,320 entries. The total size of the table is 608,121,360 entries, taking 1.216G bytes of storage.

We tested ConfMatch with diffraction data at 2.0Å. The density distribution is generated from 2548 structure factors with their published phases. This input is merely 10.8% of the data at 0.9Å. Using these ideal phases means that we are matching a conformation to the perfect density, but with the high frequency information removed. The bound preprocessing stage and the search stage takes 14,700 and 17 seconds of CPU time respectively. The overwhelming majority of the running time is spent on the first stage. Figure 3-11 shows the solution structure from ConfMatch superimposed with the published 0.9Å structure. ConfMatch's result has an RMSD (root-mean-squared-deviation) of 0.812Å from the target structure. The difference between the global upper bound from the first stage, M, and the density of the solution structure is very small. It is equivalent to just 0.32 of the average density of an atom.

We investigated the effect of using data at various resolutions, while keeping all other parameters unchanged. In doing so, we try to find the minimum amount of experimental data necessary to calculate a useful structure. The results are shown in Table 3.3. In general, all performance measures worsen with the data resolution, because less information is available in the density map. The running time of the bound preprocessing stage is constant for all resolutions, but that of the search stage varies with the difficulty of the conformational search. However, the bound preprocessing stage always dominates the search stage in CPU time. The quality of the

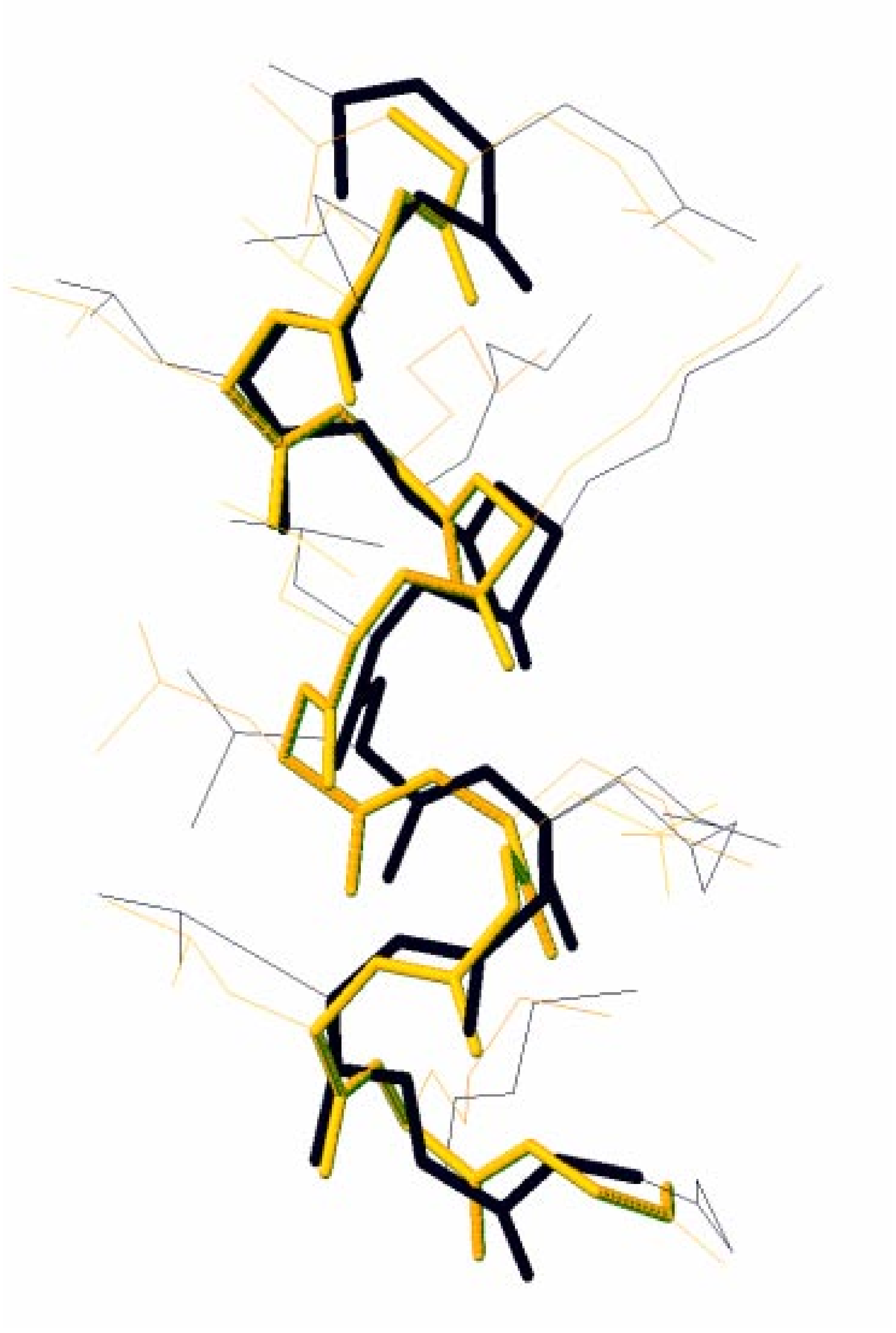


Figure 3-11: ConfMatch's solution structure (in black) of Alpha-1 from 2.0Å resolution data and the published 0.9Å structure (in yellow). The thicker portions are the backbones of the structures.

Resolution (Å)	Number of Reflections	RMSD (Å)	Search Stage Last Iteration Time (sec)	DIFF
2.0	2548	0.812	17	0.32
2.1	2190	0.759	20	0.03
2.2	1925	0.817	14	0.18
2.3	1669	0.960	20	0.18
2.4	1475	0.964	20	0.0
2.5	1309	0.968	15	0.16
2.6	1144	0.939	14	0.38
2.7	1036	0.866	20	0.07
2.8	933	0.831	15	0.0
2.9	831	0.827	20	0.0
3.0	740	1.003	25	0.81
3.1	691	1.030	42	1.15
3.2	629	1.386	20	1.23
3.3	566	1.979	704	2.04
3.4	515	5.774	253	2.82

Table 3.3: Alpha-1's conformation is matched to data at various resolutions with ideal phases. The running time of the last iteration of the search stage is close to that of the entire stage because IDA* is dominated by the last depth-first search. DIFF: Difference between the global upper bound, M, and solution density (equivalent number of atoms).

solution structure (as measured by RMSD to the correct solution) and the bounds table (as measured by the difference between the global upper bound, M, and the actual solution density) both deteriorate with worse resolution of the data. There is a big jump in RMSD from 3.3 to 3.4Å. Figure 3-12 shows the 3.4Å solution structure superimposed with the target structure. The backbones of those two structures are significantly different. For Alpha-1, 3.3Å seems to be the resolution limit where Conf-Match can calculate an accurate structure. This limit is sufficiently generous because it includes almost every set of published data. The number of structure factors at 3.3Å is merely 2.39% of the original experimental data. It may also be the limit of chemical constraints in the **full** set. To push this boundary further, an algorithm needs to acquire more chemical and biological knowledge about the molecule.

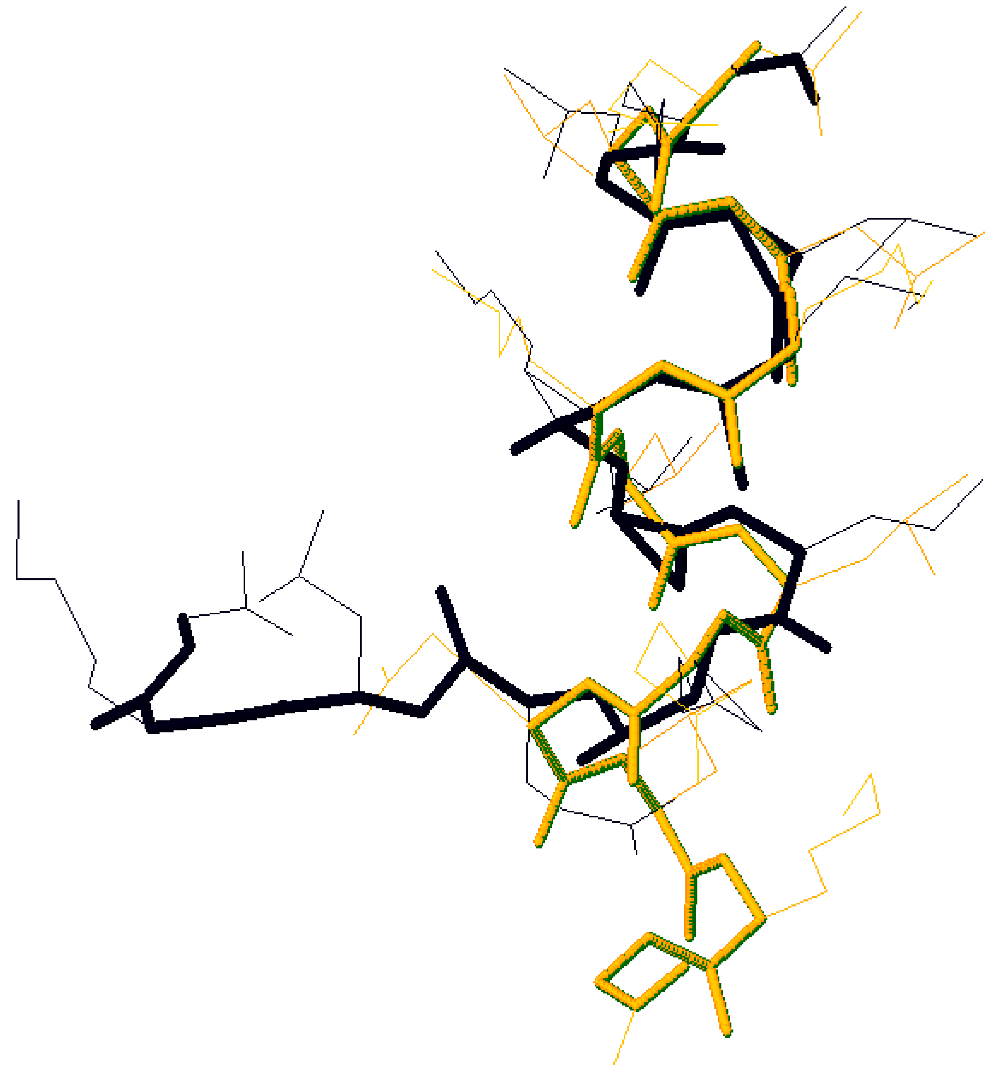


Figure 3-12: ConfMatch's solution structure (in black) of Alpha-1 from 3.4Å resolution data and the published 0.9Å structure (in yellow). The thicker portions are the backbones of the structures.

Phase Error (degree standard deviation)	RMSD (Å)	Search Stage Last Iteration Time (sec)	DIFF
0	0.812	17	0.32
5	0.702	20	0.38
10	0.793	20	0.62
15	0.806	25	0.27
20	0.841	20	0.23
25	0.718	100	0.48
30	1.002	310	0.88
35	1.322	15	0.72
40	0.951	21	0.30
45	1.013	980	7.12
50	1.416	67	1.18
55	11.370	21240	7.16

Table 3.4: Alpha-1's conformation is matched to phases with various level of error using 2Å resolution data. DIFF: Difference between the global upper bound, M, and solution density (equivalent number of atoms).

We have also investigated the effect of phase error on ConfMatch. Both experimental and direct methods for structure determination generate phases with substantial errors. Being able to tolerate phase error is essential for ConfMatch's practical applications. We model these errors by adding random Gaussian noise to the perfect phases[6]. By varying the standard deviation of the Gaussian distribution, we can measure ConfMatch's tolerance. The results from 2Å resolution data are shown in Table 3.4. As expected, all performance measures worsen with increasing phase error. The RMSD generally increases with phase error. There is a big increase in both RMSD and search time from 50 to 55^o. At 55^o phase error, the search tree of the last iteration has 161,828,154 nodes. The search stage uses more CPU time than the bound-preprocessing stage, but can only find a low quality structure. 50^o may be the limit of ConfMatch's error tolerance of Alpha-1 at 2Å. We expect this tolerance to improve with higher resolution data, and shrink with worse data.

[6]Following a suggestion of William M. Wells III, the Gaussian distribution is approximated by summing 3 uniform random variables within the range $[-\sigma, \sigma]$, where σ is the desired standard deviation of the Gaussian distribution.

From Tables 3.3 and 3.4, we observe a positive correlation between RMSD and DIFF—the difference between M and solution density. In other words, the quality of the solution correlates with the quality of the bounds table. The only exception occurs at 45^{o} phase error, where DIFF is quite large, but the answer is tolerable. This suggests a possible use of DIFF as a confidence measure: If we apply ConfMatch on a real crystallography project, we cannot calculate the RMSD because the target structure is not known. Similarly, we have no knowledge of the amount of error in the phase set. Under this circumstance, DIFF may be substituted as a measure of confidence in the solution conformation. The smaller the DIFF, the more confident we are in the solution, and vice versa.

3.5.2 Crambin

The second test of the ConfMatch algorithm is crambin [40] (PDB code 1AB1), a small, 46 residue protein. The crystal of crambin diffracts to 0.89Å resolution with 19803 structure factors. The space group of the crystal is $P2_1$. The unit cell dimensions are $|\mathbf{a}| = 40.759$Å, $|\mathbf{b}| = 18.404$Å, $|\mathbf{c}| = 22.273$Å with angles $\alpha = 90.00^{o}$, $\beta = 90.70^{o}$, and $\gamma = 90.00^{o}$. This molecule has 326 non-hydrogen atoms, 141 free dihedral angles, and 147 degrees of freedom total.

Modifying the Objective Function of Conformational Matching

Crambin has 6 cysteine residues which form 3 disulfide bonds. Sulphur has a higher electron density than N, C, or O, because it has a larger atomic number than the others. The overall density distribution has several large peaks corresponding to the sulphur positions. Figure 3-13 plots the density of the highest peaks of a typical crambin distribution. There is a significant gap between the 6th highest peak and the 7th one because of the difference between the 6 sulphur atoms and others. If we use the simple objective function, the sum of density at atom locations, and ignore the different identities of atoms, the sulphur locations will become strong "attractors" of all other atoms. Consequently, the bound preprocessing stage will

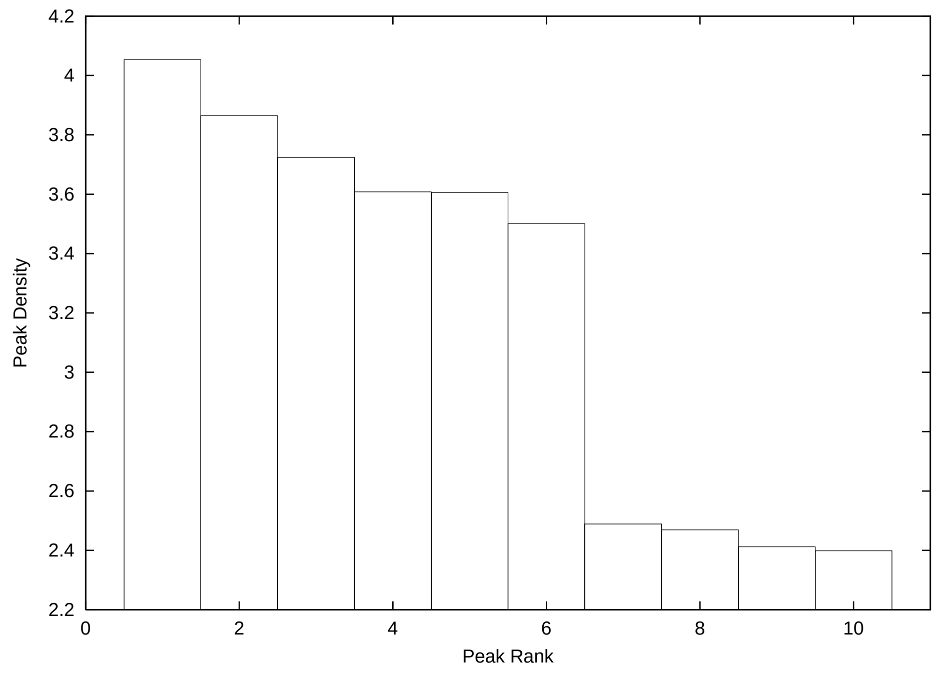


Figure 3-13: The highest peaks in crambin's 2.0Å resolution density distribution.

place multiple atoms at the sulphur positions and the upper bounds in the bounds table will be very loose. This problem can be overcome by a small modification to the objective function: We separate the atoms into different classes according to their atomic numbers. Each class has its own electron density distribution that the atoms will measure from. These different distributions are biased toward their respective classes of atoms. The new objective of conformational matching is to maximize the sum of density, measured from an atom's particular density distribution, at positions of all atoms. The ConfMatch algorithm can accommodate this modification without any major change.

In the case of crambin, the atoms are separated into 2 classes: (1) sulphur atoms and (2) all other non-hydrogen atoms. The sulphur atoms, using the original density distribution, will preferably locate at the highest density regions. All atoms other than sulphur use a density distribution modified from the original one. This modified distribution is more uniform than the original because it has the highest density regions suppressed: First, we find the 7 highest peaks of the original distribution, one more than the number of sulphur atoms. Let $P_1, \ldots, P_7$ be the densities of the 7 peaks. The difference between P_i and P_7 is a measure of our confidence that the i-th peak shall be occupied by sulphur. We suppress the i-th peak by subtracting a typical sulfur-atom density distribution, scaled to height $2(P_i - P_7)$, from the original distribution. After the 6 highest peaks are suppressed, we have removed much of the influence of the sulfur atoms. This modification is robust in spite of low data resolution or phase errors. For instance, several neighboring peaks of heavier atoms often merge together at low resolution. It appears as if there are fewer heavy atoms than expected. If these or other errors in the density distribution cause a wrong peak i to be chosen, it is very unlikely that $P_i - P_7$ will be large. Therefore our miscalculation will have only a minor impact on the density distribution.

Utilizing Crambin's Crystal Symmetry

The crambin crystal's $P2_1$ symmetry means that there are 2 asymmetric units per unit cell. This allows us to reduce the time and space requirement of our program by 50% if the conditions outlined in Section 3.4.2 can be satisfied. The $P2_1$ space group has a 2-fold screw axis parallel to **b** (Figure A-3). If a particle is located at fractional coordinates $\begin{pmatrix} u \\ v \\ w \end{pmatrix}$, there must be an identical particle symmetrically located at $\begin{pmatrix} -u \\ v + 1/2 \\ -w \end{pmatrix}$. The symmetry operation can be represented by the matrix $\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1/2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. This symmetry places a restriction on the dimension of the grid. Let b be the number of grid points in the **b** direction. The grid spacing in the **b** direction is $1/b$ in fractional coordinates. Since the symmetry operation includes a translation of $1/2$ in the **b** direction, $1/2$ must be divisible by $1/b$ in order for the grid to be invariant. Therefore b must be an even number. We selected a grid of $83 \times 38 \times 45$ for the crambin unit cell ($b = 38$), with grid spacing approximately 0.5Å in every dimension. The unit cell can be separated into 2 asymmetric units by dividing along $u = 1/2$. We perform all of our calculations on the asymmetric unit where $u \in [0, 1/2]$. The grid within this asymmetric unit is $42 \times 38 \times 45$, about half the size of the original grid.

The other condition of utilizing the crystal symmetry is that the rotational sampling preserves the rotational symmetry. Extracted from the symmetry operations, the rotational symmetry can be represented as $S = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$. If we are given a rotational transform R in Cartesian coordinates, its symmetric transform will be $CSC^{-1}R$. For crambin, $C^{-1} = \begin{bmatrix} 0.024534 & 0 & 0.0003 \\ 0 & 0.054336 & 0 \\ 0 & 0 & 0.044901 \end{bmatrix}$, which implies $CSC^{-1} = S$. Thus R's symmetric transform is simply SR. This property is true for all $P2_1$ unit cells, but it remains to be proven for other space groups.

Initially, we use Lattman's method to generate 2062 rotational transforms, with $\theta_L = 20.0^o$. (The branching factor of conformational search is around 18.) Each transform then generates a symmetry mate by the equation above. A total of 4124 rotational transforms are generated. Note that these transforms are no longer uniformly distributed, but are twice as dense around **b** than around other axes. Rao et al. [36] have described a method which generates the rotational transforms more uniformly for various space groups.

Crambin Results

In a typical x-ray experiment, the exact disulfide-bonding residue pairs are not know beforehand. This knowledge is usually obtained from the density map. We have modeled this lack of knowledge by removing all of crambin's disulfide bonds from the intramolecular distance matrix, but allowing any cysteine-pairs to form such bonds:

We reduce the intramolecular lower bounds among all cysteine sulphur atoms to the bond length of a typical disulfide bond. The upper bounds, on the other hand, are not set. Thus, any cysteine pair can form or not form a disulfide bond.

With the modifications above, we tested our program on crambin. After removing the disulfide bonds, crambin does not have any flexible-ring-forming bonds. Before and after common subtree elimination, the fragment tree has 142 and 111 fragments, respectively. Common subtree elimination reduces the time and space of the first stage by 22%. The crystal's $P2_1$ symmetry reduces the time and space by another factor of 2. The size of the bounds table of each fragment ranges from 7,756,560 to 13,933,080 entries. The total size of the table is 1,380,954,960 entries, taking 2.762G bytes of storage.

As in the previous experiment, we tested ConfMatch with diffraction data at 2.0Å. The density distribution is generated from 2360 structure factors with their published phases. This input is merely 11.9% of the data at 0.89Å. The bound preprocessing and the search stage takes 42,900 and 34 seconds of CPU time respectively. Again, the vast majority of the running time is spent on the first stage. The last iteration of IDA* explored a search tree with 46,453 nodes to reach the solution. The effective branching factor of the search tree is only 1.08, which is much smaller than the worst case branching factor of 18. This is mostly due to the accuracy of the upper bounds.

Figure 3-14 shows the solution structure from ConfMatch superimposed with the published 0.89Å structure. ConfMatch's result has an RMSD of 0.588Å from the target structure. The difference between the global upper bound, M, (calculated from the bound-preprocessing stage) and the density of the solution conformation is equivalent to 0.73 of the average density of a single carbon atom.

As the previous experiment, we investigated the effect of using data at various resolutions, while keeping all other parameters unchanged. The results are shown in Table 3.5. The running time of the bound preprocessing stage is constant for all resolutions, but that of the search stage varies greatly. In general, all performance measures worsen with the data resolution. ConfMatch was able to calculate an accurate structure at 2.6Å resolution. At 2.7Å, however, ConfMatch could not

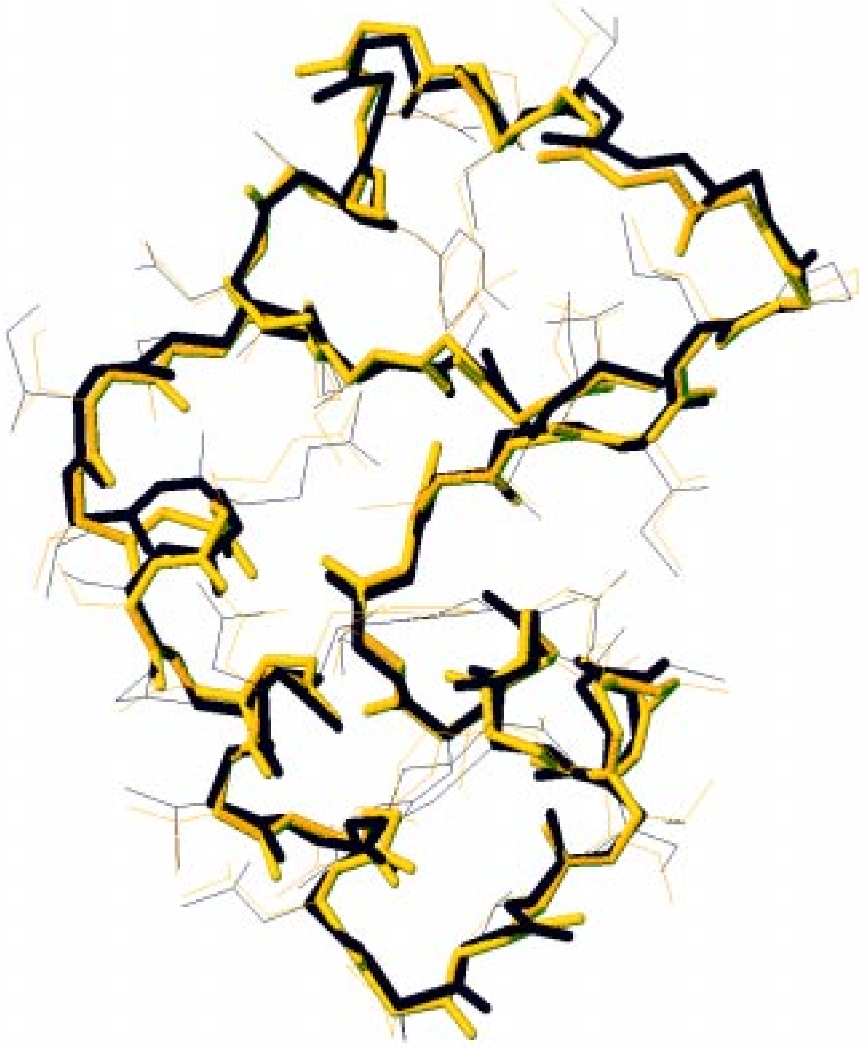


Figure 3-14: ConfMatch's solution structure (in black) of crambin from 2.0Å resolution data and the published 0.89Å structure (in yellow). The thicker portions are the backbones of the structures.

Resolution (Å)	Number of Reflections	RMSD (Å)	Search Stage Last Iteration Time (sec)	Last Search Tree Size	Effective Branching Factor	DIFF
2.0	2360	0.588	34	46453	1.08	0.73
2.1	2049	0.611	16	6724	1.06	0.46
2.2	1783	0.724	1176	2507739	1.11	1.71
2.3	1565	0.774	21	21660	1.07	1.80
2.4	1378	0.677	1695	4763224	1.11	2.04
2.5	1228	0.707	1580	4207880	1.11	1.05
2.6	1102	0.794	5926	17764058	1.12	2.04
2.7	987	unknown	> 176369	> 463342204	> 1.15	> 2.11

Table 3.5: Crambin's conformation is matched to data at various resolutions with ideal phases. The running time of the last iteration of the search stage is close to that of the entire stage because IDA* is dominated by the last depth-first search. DIFF: Difference between the global upper bound, M, and solution density (equivalent number of atoms).

find a solution. In this case, ConfMatch is not limited by the chemical constraints, but by the available computational resources—the search stage requires more CPU time than we can afford. After spending 176,369 CPU seconds (2.04 CPU days) on the last iteration of IDA*, no solution was found. There were too many structures with steric clashes and had higher density than the best solution. At 2.7Å, the exponential-time search stage requires far more resources than the polynomial-time bound-preprocessing stage. Finding a solution at 2.7Å will require more computational resources for searching, or a more efficient algorithm.

We have also investigated the effect of phase error on ConfMatch. We model these errors by adding varying degree of random Gaussian noise to the perfect phases. The results from 2Å resolution data are shown in Table 3.6. ConfMatch was able to calculate an accurate conformation with 15^o phase error. However, ConfMatch could not find a solution with 20^o phase error. After spending 105,998 CPU seconds (1.23 CPU days) on the last iteration of IDA*, no solution was found. Once again, we are limited by the computational resources, not by the chemical constraints.

While measuring the error within ConfMatch's output, we notice that the 3 asparagine sidechains usually have larger RMSD than others. It points out a limitation

Phase Error (degree standard deviation)	RMSD (Å)	Search Stage Last Iteration Time (sec)	Last Search Tree Size	Effective Branching Factor	DIFF
0	0.588	34	46453	1.08	0.73
5	0.588	580	735865	1.10	1.20
10	0.631	470	153694	1.08	1.08
15	0.681	23	2726	1.06	0.18
20	unknown	> 105998	> 310132939	> 1.14	> 1.42

Table 3.6: Crambin's conformation is matched to phases with various level of error using 2Å resolution data. DIFF: Difference between the global upper bound, M, and solution density (equivalent number of atoms).

of our geometric constraints: The atoms at the end of asparagine, OD1 and ND2, are very different chemically but nearly identical geometrically. Their positions are often swapped by ConfMatch. The same problem will occur in proteins containing glutamine or histidine sidechains. In order to solve this problem, it will be necessary to introduce more chemical knowledge such as electrostatics or hydrogen bond donor/acceptor geometries into our algorithm.

3.6 Conclusions and Future Work

We have demonstrated that ConfMatch, a branch-and-bound algorithm, can find the globally optimal solution of a problem (discretized conformational matching) that has more than 100 degrees of freedom. The solution space of this problem includes the grid-based conformations generated from sampling all free dihedral angles, as well as the 6 rigid degrees of freedom. (To ensure that ConfMatch covers all possible conformations on the grid, one may follow the sampling scheme in Figure 3-8.) To reach the global optimum, it is necessary to systematically explore a search tree exponential in the number of degrees of freedom. The most important idea of ConfMatch is an efficient method for computing accurate bounds. ConfMatch relaxes the conformational matching problem, a problem which can only be solved in exponential time (NP-hard [14]), into one which can be solved in polynomial time. The relaxed problem retains all local constraints of conformational matching, but ignores all non-local ones. The solution to the relaxed problem is a guaranteed upper bound for the conformational matching problem. When the input data is of sufficiently good quality, the local constraints can lead to accurate bounds. In most empirical cases, these bounds are accurate enough to prune the search space dramatically, making ConfMatch a practical algorithm for the NP-hard problem.

On the practical side, ConfMatch is the first algorithm to fully automate the interpretation of electron density maps. This important task normally requires much interactive fitting and refining of the molecular structure. Now ConfMatch may be able to transfer part of the workload from the crystallographer to computers. This may remove human subjectivity from map interpretation and accelerate the crystal structure solution process. This technology may have particular impact in protein structure solution efforts.

Presently, ConfMatch can solve the conformation of a 40-50 residue protein with moderate error in the phase set. If one needs to solve the structure of a larger protein or to use a density map with larger error, one may need to provide some guidance to the program. One possible technique is to split a large protein into smaller, 20-30 residue peptides and solve each segment independently. This effectively converts the global optimization problem into several sub-problems. If the density distribution has good quality, the various segments may merge at appropriate positions and form a good conformation overall. In other words, the solutions of the sub-problems can combine to be an approximate solution to the global problem. Our results with Alpha-1 using sufficiently high quality data support this possibility. Searching density for 4 chains with a single chain correctly identified a single, connected chain in the density.

A different kind of human assistance can be incorporated using ConfMatch through seeding the structure or restricting the locations of some parts of the conformation. Traditionally, crystallographers initiate the map interpretation process by locating large, distinct sidechains such as tryptophan, then gradually filling in the rest of the structure. The same kind of information can greatly improve the efficiency of ConfMatch by accelerating the search stage. For example, if a user specifies some positions and orientations of tryptophans, ConfMatch can assign some small regions of the unit cell to tryptophans only. Within these special regions, the tryptophan fragments will have their density boosted, but all other fragments will have very low values. This modification creates a big gap (in density sum) between those conformations that place tryptophans at the specified locations and those that do not. This gap will be reflected in the bounds table. Most conformations that do not place tryptophans at the specified locations are eliminated by the bounds. As a result, the conformational search will explore far fewer conformations than the naive approach.

Obviously, the best approach to solving large proteins or data sets with big errors is to improve fundamentally the computational efficiency of ConfMatch. One promising technique is to extract more information from the density map. Obtaining more information has effects similar to improving the resolution of the density map. It will automatically lead to better overall performance. Specifically, the current objective function of ConfMatch evaluates the density of an atom at a single point—the center of the atom. If we measure the density within a local neighborhood of the atom, we may detect a more accurate signal. For example, Jones et al. [19] developed the *real-space R factor* which measures the difference between the expected and the observed density distribution within a region. The real space R factor is defined as

$$\frac{\sum |\rho_{\text{obs}} - \rho_{\text{calc}}|}{\sum |\rho_{\text{obs}} + \rho_{\text{calc}}|}$$

where ρ_{obs} and ρ_{calc} are the observed and expected density, respectively, at various grid points. The sum is over all grid points within an envelope. For ConfMatch, every fragment has a unique density distribution. An appropriate envelope can be defined for each fragment. A small real-space R factor will imply that the fragment is likely to be at the center of the envelope, and vice versa. ConfMatch can be modified to minimize the sum of real-space R factor over all fragments. If the real-space R factor or other measures is a more accurate detector than the density at atom centers, it will automatically lead to better bounds and more efficient searches. ConfMatch may then be able to solve bigger and harder problems.

Chapter 4

Discussions and Conclusions

We have presented two systematic algorithms, AmbiPack and ConfMatch, for two very different structure determination problems. They are both based on branch-and-bound search, but with different problem formulations and heuristics. From the experience of developing these algorithms, we may learn some lessons for investigations into other problems.

The goal of AmbiPack is finding all satisfying solutions to the packing problem. There are no objective functions to optimize for. Thus there are no bounds to calculate. The focus is on making the search more efficient by using the ambiguous distance constraints early. There are many ways to formulate the search. The obvious one is branching on the alternative interpretations of ambiguous constraints, i.e. at every node, one selects an ambiguous constraint and specify its interpretation. If the current set of interpretations become inconsistent, one can rule out a branch of the search tree. If the constraints have very narrow ranges and the monomers are quite large, this may be a good approach because most interpretations will be inconsistent. One can use the simple triangle inequality to prune most of the search space. Unfortunately, in practice, the distance constraints have very large ranges relative to the monomer size. Thus almost no interpretation can be ruled out by the triangle inequality. This approach will explore almost every node in the search tree and be every inefficient. The key observation in the development of AmbiPack is that one can fix a rigid structure by specifying the positions of only 3 atoms. Once the configuration of the substructures are fixed, one can simply check the ambiguous constraints. It is unnecessary to enumerate the numerous alternative interpretations. Therefore, it becomes natural to search base on the placement of 3 atoms. Using a hierarchical division of the physical space, the constraints can eliminate large sets of placement early in the search. This approach makes AmbiPack an efficient algorithm.

The goal of ConfMatch is finding a single optimal conformation. With more than 100 degrees of freedom, it initially seems impossible to develop a practical systematic algorithm. For any branch-and-bound algorithm, the bounds must be very accurate to rule out most of the search space early. Otherwise, the search will be hopelessly inefficient. We design the objective function, the sum of density at atom positions, because it is simple and intuitive. The key insight of ConfMatch is that one can calculate a set of upper bounds of this objective function in polynomial time using dynamic programming. These bounds obey the local geometrical constraints but not the non-local ones. To our pleasant surprise, these bounds are quite accurate given good density data. In addition, the optimal conformation based on our objective function is indeed quite close to the published structure. The accurate bounds enable ConfMatch to prune the search space dramatically. Thus ConfMatch is able to solve several large structures in reasonable time.

Structure determination problems are computationally challenging because of their enormous search space. However, our experience shows that it is still possible to develop practical systematic algorithms to solve them. Branch-and-bound is usually a good approach for these problems. It is often necessary to extensively customize the algorithm for the specific type of problem and input data. Given enough effort by the scientists, systematic algorithms can be competitive with stochastic ones in this application domain.

Appendix A

Outline of Protein Structure Determination by X-ray Crystallography

In order to determine the structure of a protein [12], one needs to first grow a crystal of the material. A crystal of organic material is a three-dimensional periodic arrangement of molecules. In the regular packing of the molecules three repeating vectors **a**, **b** and **c** can be recognized with the angles α, β, and γ between them. These three vectors define a unit cell in the crystal lattice (Figure A-1). The crystal can be regarded as a three-dimensional stack of unit cells with their edges forming a lattice (Figure A-2). The regular packing of molecules in a crystal lattice often leads to a symmetric relationship between the molecules. One characteristic of a crystal is that it has three-dimensional translational symmetry, corresponding to the repetition of the unit cells. Often, additional symmetry within the unit cell such as 2- or 3-fold (screw) rotation is encountered. For a biological macromolecule, there are 65 different ways to combine the symmetry operations in a crystal, leading to 65 space groups. These space groups are named P1, P2, P2_1, C2, P222, etc. For instance, the P2_1 space group has a 2-fold screw symmetry with the screw axis parallel to **b**. The symmetry operation of P2_1 is a rotation by π radian about the **b** axis and then a translation along the **b** axis by $\frac{|\mathbf{b}|}{2}$. Figure A-3 illustrates the screw axis of P2_1.

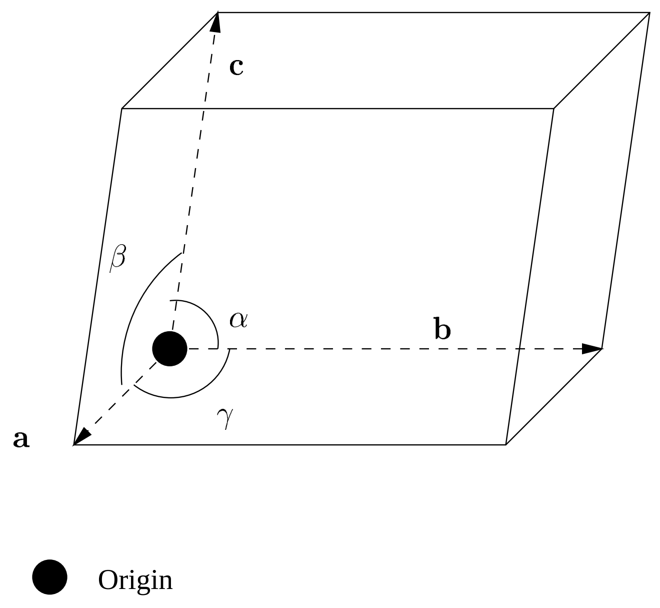


Figure A-1: One unit cell in the crystal lattice.

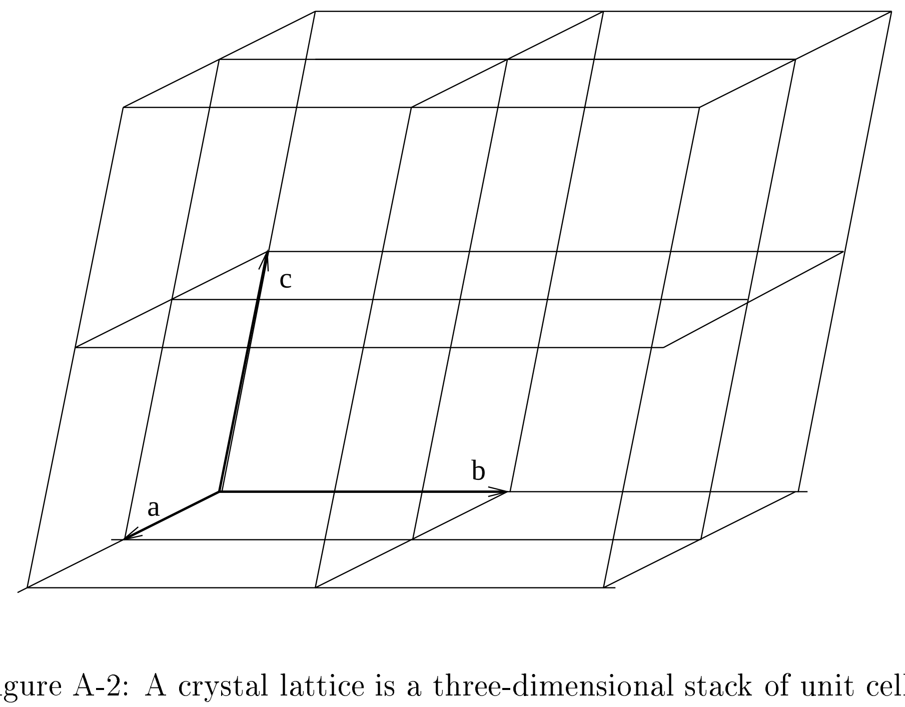


Figure A-2: A crystal lattice is a three-dimensional stack of unit cells.

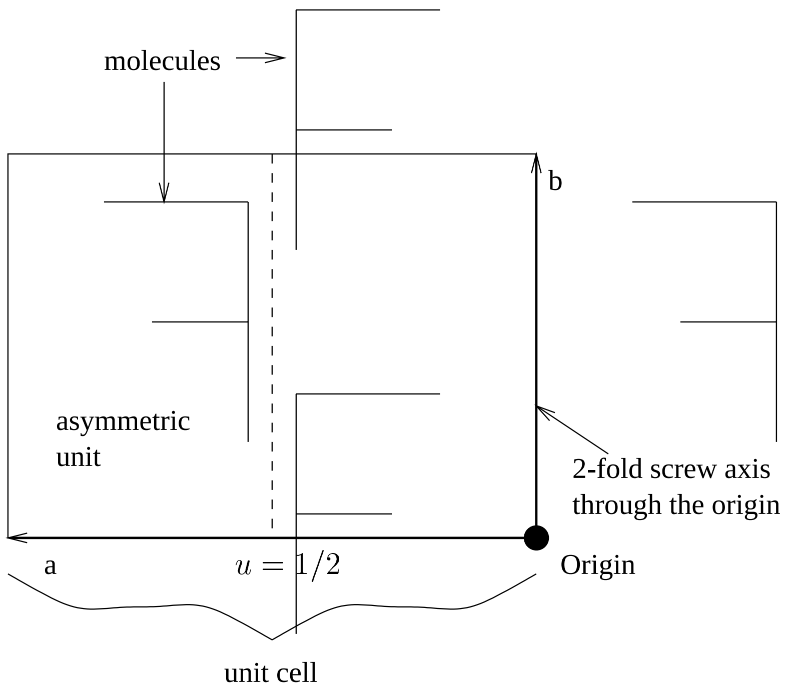


Figure A-3: The 2-fold screw axis of the $P2_1$ space group in 2 dimensions. The **c** vector projects into the plane. The asymmetric unit is half of the unit cell, divided along $u = 1/2$.

In all space groups except P1, each particle in the cell will be repeated a number of times as a consequence of the symmetry operations. For example in space group P2_1 one can always expect at least two equal particles in the unit cell related by the symmetry operations. This unit cell is comprised of two equivalent asymmetric units. An asymmetric unit is a fraction of the unit cell but contains all of the cell's information. One can regenerate the unit cell from its asymmetric unit by simply applying the symmetry operations. The asymmetric unit of P2_1 is also illustrated in Figure A-3.

The position of a point P in the unit cell is given by its position vector $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$. In terms of its fractional coordinates u, v, and w with respect to the crystal axes $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, P's position is given by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{a}u + \mathbf{b}v + \mathbf{c}w.$$

The position of P can thus be described by its "fractional" coordinates $\begin{pmatrix} u \\ v \\ w \end{pmatrix}$, with $0 \leq u, v, w \leq 1$. The 3×3 matrix $[\ \mathbf{a} \quad \mathbf{b} \quad \mathbf{c}\ ]$, denoted as C, transforms a vector from fractional into Cartesian coordinates. Conversely, C^{-1} transforms from Cartesian into fractional coordinates. In the rest of the chapter, u, v, and w will denote fractional coordinates, whereas all other symbols will denote Cartesian coordinates.

After a crystal is produced, it is placed in a beam of x-rays to record diffraction data (Figure A-4). The diffracted x-rays emerge from the crystal at different angles and have different intensities. The angles and intensities can be recorded on a piece of photographic film. Each diffracted x-ray makes a spot or "reflection" where it intersects the x-ray film. The entire pattern of diffraction can be thought of as a three-dimensional lattice of spots. Crystallographers refer to this lattice as the

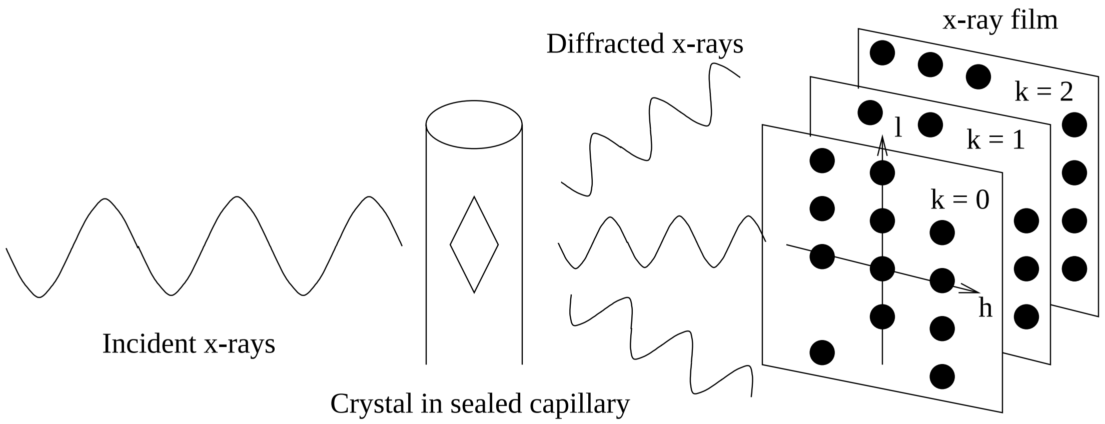


Figure A-4: Recording of x-ray data. A beam of x-rays is incident on a crystal. Diffracted x-rays emerge and can be detected on photographic film. Each spot or "reflection" on the film arises from the intersection of a diffracted ray with the film. The pattern of diffraction spots may be thought of as a three-dimensional lattice.

reciprocal lattice.

The intensities of reflections are the primary data of x-ray crystallography. Each reflection is identified by three indices, h, k, and l, that specify its place in the reciprocal lattice. From each reflection, the *structure-factor magnitude*, $|F(h,k,l)|$, can be measured. $|F(h,k,l)|$, a scalar, is the magnitude of the complex number $F(h,k,l)$.

One of the end products of a crystallographic analysis of molecular structure is a plot of the electron density, $\rho(u,v,w)$, of one unit cell of the crystal as a function of the three coordinate axes of the unit cell. One must also identify each feature of the electron-density plot with a chemical group, effectively assigning a position for every atom in the unit cell through fitting the electron density. Once this process of identification is complete, the structure has been determined.

Unfortunately, the structure-factor magnitudes, $|F(h,k,l)|$, actually comprise only part of the information needed to determine $\rho(u,v,w)$. The other, more important part consists of a "phase" for each reflection, which is denoted $\alpha(h,k,l)$. $\alpha(h,k,l)$ is the phase angle of $F(h,k,l)$. The $|F(h,k,l)|$s and $\alpha(h,k,l)$s can be thought of respectively as the amplitudes and phases of component waves that are summed to describe the crystal structure. The structure factor equation

$$F(h,k,l) = \int_0^1 \int_0^1 \int_0^1 \rho(u,v,w) \exp[2\pi i(hu+kv+lw)]dudvdw$$

relates by Fourier transform the electron density of the crystal, $\rho(u,v,w)$, with the amplitudes and phases of the diffraction, $F(h,k,l)$. Conversely, $\rho(u,v,w)$ can be calculated from $F(h,k,l)$ by an inverse Fourier transform. The diffraction pattern of a crystal gives the amplitudes of the Fourier transform of the crystal content, but the phase angles are lost. It is because the magnitude of a particular diffraction spot, $|F(h,k,l)|$, can be measured but its phase, $\alpha(h,k,l)$, cannot.

Deriving a structure from its X-ray diffraction pattern is a hard inverse problem (Figure A-5): It is easy to check a structure's correctness by computing its Fourier transform and matching the amplitudes, but it is very hard to find the correct structure from amplitudes alone. Since amplitudes and phase angles are both necessary to determine a structure, a key step in protein crystallography is solving the *phase problem*[1]—recovering the phase angles of the reflections.

For small molecules the phases often can be determined from information within the $|F(h,k,l)|$s. In principle, given the covalent structure of a molecule and enough unique diffraction amplitudes, its three dimensional structure is overdetermined. Small molecules sometimes have a number of unique diffractions that is more than 100 times the number of atoms. Because of this tremendous overdeterminacy, purely computational techniques for solving the phase problem (direct methods) have long existed for small molecule crystallography [15]. As the molecule's size increases, the accuracy of experiments generally deteriorate. Since the number of diffraction measurements decreases rapidly with worsening experimental resolution, the level of overdeterminacy also decreases as the size of the molecule increases. These classical direct methods require measurement accuracies that are currently not achievable for macromolecules

[1]The phase problem has an analogous counterpart in image processing [27]. There the problem is to reconstruct a two dimensional image from its Fourier transform magnitudes. Unfortunately, this is also an extremely hard inverse problem. None of the algorithms developed to date are capable of solving any image of reasonable size.

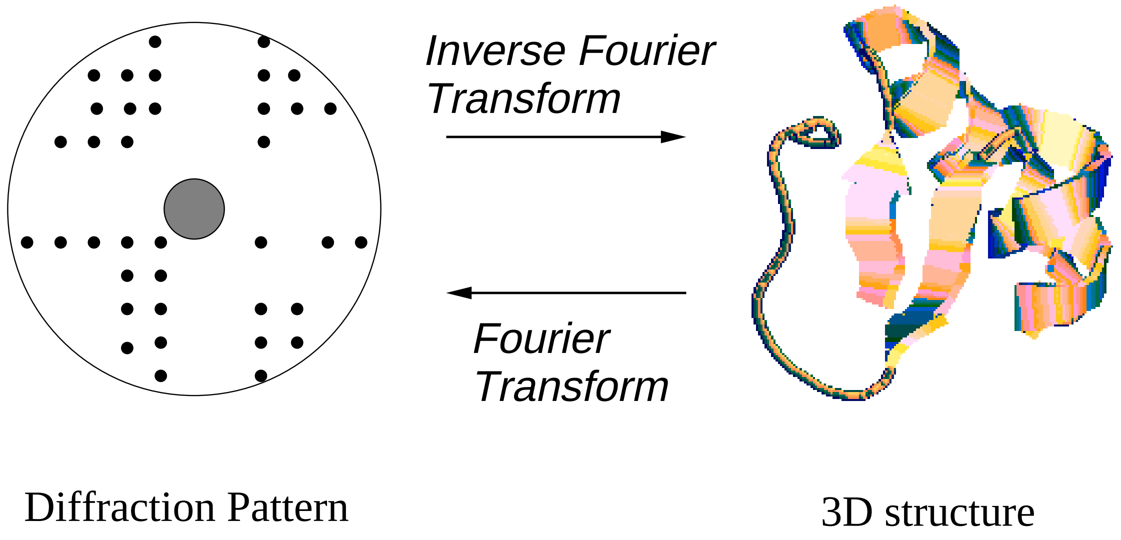


Figure A-5: Solving crystal structure is an inverse problem from Fourier transform magnitudes to the 3-dimensional molecular structure.

and therefore they cannot solve structures with more than 2-300 atoms [15].

Solving the phase problem for larger proteins requires additional experimental information. In protein crystallography, the phase problem is currently solved by three methods [11]: multiple isomorphous replacement (MIR), multiple wavelength anomalous dispersion (MAD), and molecular replacement (MR). MIR requires binding an atom of high atomic number to each protein molecule within the crystal and measuring a set of $|F(h,k,l)|$s for the modified crystal. This derivative crystal produces a slightly different set of magnitudes because of the heavy atoms. By combining the $|F(h,k,l)|$s from several modified crystals with the original $|F(h,k,l)|$s, the rough phase of each reflection, hence the electron density, can be determined. MAD also requires placing one or more anomalously scattering atoms (usually Selenium) into the protein. Without anomalous scattering, $|F(h,k,l)| = |F(-h,-k,-l)|$, the magnitude of a diffraction is identical to that of its Friedel mate. Anomalous scattering breaks this equivalence. Phases can be derived from the small difference between $|F(h,k,l)|$ and $|F(-h,-k,-l)|$. Due to experimental uncertainties, MIR and MAD both generate phases with large errors. If one uses these phases to compute an electron density distribution, the resulting density map has large discrepancies with the actual density. Months of tedious labor may be required to fit and refine a molecular structure to the experimental density map. On the other hand, MR is a purely computational method. It however requires knowledge of the protein's approximate structure, which is usually not available. Any computational progress in any approach to the phase problem can make a big difference for protein crystallography.

Often the structure of a protein is solved in stages of increasing *resolution*. The resolution of the structure is the minimum separation of two groups in the electron-density plot that can be distinguished from one another. Thus, as the fineness of resolution increases, finer detail can be seen in the electron-density plot. At 3Å resolution the path of the polypeptide backbone of a protein usually can be traced in an electron-density map. At 2Å resolution most amino acid side chains can be positioned accurately. And at 1.5Å resolution (achieved so far for only a few proteins) many resolved atoms can be seen. It is the available $|F(h,k,l)|$s and $\alpha(h,k,l)$s that determines the resolution of the electron density: as terms of with increasing values of h, k, and l are added, the resolution increases. In the crystal structure analysis of the protein myoglobin, which has a molecular weight of 17,800, about 400 reflections exist in the reciprocal lattice out to 6Å resolution, 9,600 out to 2Å resolution, and more than 25,000 out to 1.4Å resolution.

The correctness of a solution to a particular crystal is usually measured by the R-factor, which compares the experimentally observed magnitude, $|F(h,k,l)|$, with the magnitude calculated from the structure, $|F(h,k,l)^{\text{calc}}|$:

$$R\text{−factor} = \frac{\sum_{h,k,l} ||F(h,k,l)| - k|F(h,k,l)^{\text{calc}}||}{\sum_{h,k,l} |F(h,k,l)|}$$

where k is a scaling constant which minimizes the R-factor by normalizing computed intensities. The smaller the R-factor, the closer the agreement between the relative intensities computed from the structural model and measured from the crystal; this closer agreement of intensities is generally accepted as representing closer agreement between the model and actual structures.

Appendix B

Direct Methods to the Phase Problem

As many other inverse problems, the phase problem may be solved by a generate-and-test approach. Exhaustive search of the conformational space and testing by R-factor can identify the correct structure if the sampling is sufficiently fine. However, assuming rigid bond lengths and angles, a typical macromolecule still has hundreds or thousands of free dihedral angles. It is computationally impractical to systematically search that many degrees of freedom.

B.1 The Low-Resolution Envelope Approach

Recently, several new techniques for solving the phase problem have been proposed. The first technique attempts to find a rough outline of the structure by trial and error. The target structure is either represented as a collection of points [9] or a few spheres with uniform or Gaussian-distributed density [29]. Finding the structure then becomes a global optimization problem—calculate the placement of points or spheres that maximizes the correlation of the computed structure factors with the very low resolution portion of experimental data. Subbiah's point representation ignores the intricate primary structure of a protein. All points are free to move while avoiding collisions. This simple representation allows the use of simulated annealing as the randomized search method. Similarly, Lunin et al.'s "few atoms" representation allows the use of systematic or randomized search methods in placing these identical, unrelated spheres. Unfortunately, these types of low-resolution techniques are unable to produce a high-quality envelope of the structure. Their results still cannot be used for solving high-resolution structures.

Other new techniques for solving the phase problem are based on the classical direct method for small molecules [15]. The following section is a brief summary of the classical method.

B.2 Classical Direct Method

The classical direct method (which works successfully for small molecules) has three fundamental assumptions:

1. The electron density is always positive in the unit cell.
2. The unit cell consists of discrete atoms.
3. The atomic positions are random variables with uniform distribution throughout the unit cell.

The method attempts to calculate the phases directly from the magnitudes of normalized structure factors, $|E|$. The normalized structure factors idealize the atoms as point scatters in order to make the phase problem mathematically tractable. In an actual unit cell, the electron density is distributed around atom centers. If an atom's electron density were concentrated at the center of the atom, a diffraction pattern with the normalized structure factors would result. The magnitudes of the normalized structure factors, $|E_{\mathbf{h}}|$, can be calculated from the measured diffraction magnitude, $|F_{\mathbf{h}}|$, by the following equation:

$$|E_{\mathbf{h}}| = \frac{|F_{\mathbf{h}}|}{\sqrt{< |F|^2 >}}$$

where $< |F|^2 >$ is the average magnitude over a narrow range of resolution around $F_{\mathbf{h}}$.

Another important mathematical tool of the classical direct method is the structure invariants. Let $\phi_{\mathbf{h}_i} \in [0, 2\pi)$ be the phase of a structure factor $F_{\mathbf{h}_i}$. While $|F_{\mathbf{h}_i}|$ depends on the content of the crystal alone, $\phi_{\mathbf{h}_i}$ depends on both the content and the choice of origin of the unit cell. However, it can be proven that if $\sum_{i=0}^{m} \mathbf{h}_i = \mathbf{0}$, $\sum_{i=0}^{m} \phi_{\mathbf{h}_i} = \Phi$ does not vary with respect to the choice of origin and depends on the content alone. Therefore Φ is a structure invariant. When $m = 3$, the invariants, called triplet invariants, are especially important for direct methods.

Most atoms in a protein have similar numbers of electrons. It is customary to treat all atoms as the same in direct methods. Let **h** and **k** be two arbitrary indices of the reciprocal lattice. $\Phi_{\mathbf{hk}} = \phi_{\mathbf{h}} + \phi_{-\mathbf{k}} + \phi_{-\mathbf{h}+\mathbf{k}}$ is a triplet invariant. Assuming a random distribution of equal atoms, the probability distribution of the invariant, $P(\Phi_{\mathbf{hk}})$, is given by

$$P(\Phi_{\mathbf{hk}}) = (1/L) \exp(G_{\mathbf{hk}} \cos \Phi_{\mathbf{hk}})$$

where L is a normalization constant;

$$G_{\mathbf{hk}} = (2/\sqrt{N})|E_{\mathbf{h}} E_{\mathbf{k}} E_{\mathbf{h}-\mathbf{k}}|$$

where N is the number of atoms in the unit cell. ($|E_{\mathbf{h}} E_{\mathbf{k}} E_{\mathbf{h}-\mathbf{k}}| = |E_{\mathbf{h}}||E_{\mathbf{k}}||E_{\mathbf{h}-\mathbf{k}}|$) $P(\Phi_{\mathbf{hk}})$ is a so-called von Mises distribution. It is maximal when $\Phi = 0$ and decreases as Φ deviates further from 0. $P(\Phi_{\mathbf{hk}} = 0)$ increases as $|E_{\mathbf{h}} E_{\mathbf{k}} E_{\mathbf{h}-\mathbf{k}}|$ increases and N decreases. Each reflection is involved in multiple invariants and hence multiple probability distributions. The tangent formula

$$\tan \phi_{\mathbf{h}} = \frac{\sum_{j=1}^{r} G_{\mathbf{hk}_j} \sin(\phi_{\mathbf{k}_j} + \phi_{\mathbf{h}-\mathbf{k}_j})}{\sum_{j=1}^{r} G_{\mathbf{hk}_j} \cos(\phi_{\mathbf{k}_j} + \phi_{\mathbf{h}-\mathbf{k}_j})}$$

combines these distributions and assigns the most probable phase of **h** given other phase angles. Most classical direct methods use the probability distributions or the tangent formula to generate phases with high overall probability. For small molecules, these phases can usually produce a good density map as the basis of structure refinement.

As the number of atoms, N, increases, the probability distribution flattens. One of the reasons that the classical direct method fails for molecules with more than 200 atoms is because it becomes much harder to predict the phases when $P(\Phi_{\mathbf{hk}})$ is almost the same for all values of $\Phi_{\mathbf{hk}}$. Additionally, there are many fewer reflections per atom for macromolecules than for small molecules. The next two approaches attempt to overcome these difficulties and extend the classical method to proteins.

B.3 The Maximum Entropy Approach

A new approach to the phase problem is based on information entropy maximization and likelihood ranking [4, 5]. It differs from the traditional method in not always assuming a uniform distribution of atoms. Instead, the distributions of random atomic positions are updated whenever phase assumptions are made so as to retain maximum entropy under the constraints embodied in these assumptions. One of the results is that the distribution of $\Phi_{\mathbf{hk}}$ depends on the phase assumptions of reflections other than **h**, **k**, and **h-k**. This modification leads to a search tree where the space of hypothetical phase sets is explored hierarchically. Each trial phase set is ranked according to the log-likelihood gain or the Bayesian score which acts as a heuristic function in guiding the growth of the tree. Unfortunately, attempts based on the maximum entropy principle have not yet been successful on macromolecules.

B.4 The Real/Reciprocal Space Approach

The most recent approach to the phase problem combines optimization in both real and reciprocal space. ConfMatch is directly applicable to this approach. Starting from a random structure or a random set of phases, refinements in real and reciprocal space are applied alternatively (Figure B-1). The solution structure is found if a low

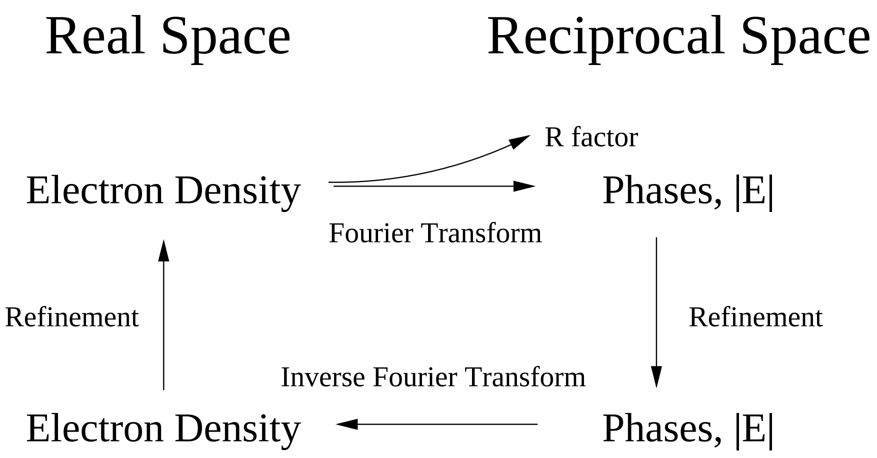


Figure B-1: The real/reciprocal space approach integrates refinement in both domains.

R-factor is obtained in the forward Fourier transform[1]. In real space, a molecular structure is subject to constraints like atomicity, positivity, bond lengths and angles, etc. By applying these constraints, an electron density map can be improved. In reciprocal space, the phases are also subject to constraints like the distribution of triplet and quartet invariant phase sums. Phases can be improved by these constraints as well. In Sheldrick and Gould's method [38], phase refinement is achieved by applying the tangent formula repeatedly. Density refinement is achieved by a "peaklist optimization" scheme where the highest $1.3N$ peaks in the density map are selected as potential atoms. A peak is eliminated if doing so improves the correlation of the structure factor with the experimental data, otherwise it is retained. The elimination is applied repeatedly until convergence. Their method is reported to have solved structures with 4-500 atoms. An important limitation of their method is that the resolution of the data must be 1.2Å or better, so atoms will have distinct density peaks.

Detitta et al. [10] proposed a similar method, called "Shake-and-bake," in which density refinement involves simply picking the N highest peaks in the density map. These peaks are assumed to be the location of atoms. Phase refinement is achieved

[1]For convenience, $|E|$'s are used instead of $|F|$'s in calculating the R-factor.

by optimizing the minimal function

$$R(\psi) = \frac{1}{\sum_{\mathbf{h},\mathbf{k}} G_{\mathbf{hk}}} \sum_{\mathbf{h},\mathbf{k}} G_{\mathbf{hk}} (\cos \Phi_{\mathbf{hk}} - t_{\mathbf{hk}})^2$$

where $t_{\mathbf{hk}}$ is the expected value of $\cos \Phi_{\mathbf{hk}}$. $R(\psi)$ is minimized when the cosines of the invariant phases match their expected values. $G_{\mathbf{hk}}$ weights the error terms higher if $\cos \Phi_{\mathbf{hk}}$ has a smaller standard deviation, and vice versa. Therefore minimizing $R(\psi)$ generally improves the consistency of the phase set. Detitta et al. have found that $R(\psi)$ is relatively small when the phases are correct. Their technique is reported to have solved structures with more than 300 atoms. However, it also requires diffraction data with at least 1.2Å resolution.

Very few crystals diffract to 1.2Å. Most macromolecules of interest diffract to 2.0Å or worse. Therefore the applicability of the methods above is limited. ConfMatch may be able to extend direct methods to data at more realistic resolutions. We can improve the real/reciprocal space approach by applying chemical constraints on the density refinement process. In the past, phase refinement has been the focus of most research. Its techniques are much more developed than density refinement. New algorithms for real-space refinement are likely to advance the state of the art of direct methods.

Previous methods failed on low resolution data because peak picking was ineffective with lower than 1.2Å data. At low resolution, atoms rarely locate at the peaks of electron density map. Peak picking ignores all chemical constraints on a molecule except atomicity and positivity. Applying all commonly-accepted chemical constraints such as bond lengths, angles, chirality, etc., is equivalent to limiting the molecule to its chemically feasible conformations. The possible distribution of atoms is much more restricted if they must obey the basic rules of chemistry. Thus density refinement can be turned into a conformational matching problem where one finds a conformation that matches the density best. When the peaks of a density map do not correspond to atom positions, this method is more likely to identify the solution structure than peak picking. The solution conformation will in turn produce a more chemically reasonable density map for the next stage of phase refinement. By applying more chemical constraints on density refinement, we may overcome the shortcoming of earlier methods on low resolution data.

Appendix C

Correctness of Bounds Table Memoization

This appendix explains the correctness of the updates to the bounds table. It shows that the entries are always upper bounds of valid solutions.

We know that a structural solution is accepted only if its density d is greater than or equal to f_{lim}. f_{lim} is raised to $d + \epsilon$ after finding the solution. During a single depth-first search, f_{lim} is monotonically increasing. All entries in the bounds table are updated by Equations 3.4 and 3.5. The bounds are monotonically decreasing. After a depth-first traversal of a sub-search-tree, if we perform the search again with the updated f_{lim} and bounds, Equation 3.3 will be satisfied less often. The new search would explore only a subspace of the earlier one. Since we have raised f_{lim} to be above any solution in the sub-search-tree, we can safely say no solution would be found in the new search.

Invariant 1 *For all $n, \mathbf{j}$, before and after n's sub-fragment-tree is searched,*

$$E^l_{n,\mathbf{j}} \leq \max_{i=1}^{s} S^i_{n,\mathbf{j}}.$$

We prove this invariant by induction:

Base case Initially, $E^l_{n,\mathbf{j}} = E_{n,\mathbf{j}}$ for all $n, \mathbf{j}$. Because the bounds table was built by Equations 3.1 and 3.2. The invariant holds.

Inductive case After searching n's sub-fragment-tree, the bounds table is updated by Equations 3.4 and 3.5. The invariant holds because both equations contain the term $\max_{i=1}^{s} S^i_{n,\mathbf{j}}$.

A sub-conformational search is a search involving only a portion of the fragment tree. Suppose we start a sub-conformational search with just the sub-fragment-tree of node n whose in-bond locates at $\mathbf{j}$. We define $f^l_{n,\mathbf{j}}(t)$ to be the f-value $(g(t) + h(t))$ of a state t in the sub-conformational search. At the initial state t_{init}, $f^l_{n,\mathbf{j}}(t_{\text{init}}) = E^l_{n,\mathbf{j}}$. In the search tree, there is a "greedy" path which selects $\arg\max_{i=1}^{s} S^i_{n,\mathbf{j}}$ at every branch. Because of invariant 1, $f^l_{n,\mathbf{j}}(t)$ is increasing on this path. Therefore, on the "greedy" path of the sub-conformational search, $f^l_{n,\mathbf{j}}(t) \geq E^l_{n,\mathbf{j}}$ for all states t.

Invariant 2 *For all $n, \mathbf{j}$, $E^l_{n,\mathbf{j}}$ is always a valid upper bound.*

This invariant means that all entries in the bounds table are always upper bounds. If n's inbond is located at $\mathbf{j}$, there is no structural solution (without any violations of the distance matrices) for the sub-fragment-tree of n with density sum above $E^l_{n,\mathbf{j}}$. Again, we prove this invariant by induction:

Base case Initially, $E^l_{n,\mathbf{j}} = E_{n,\mathbf{j}}$ for all $n, \mathbf{j}$. Because $E_{n,\mathbf{j}}$ is an upper bound of all possible structures, the invariant holds.

Inductive case Assuming all $E^l_{n,\mathbf{j}}$ are valid at state t, we need to prove that the update rules preserve the upper bound property of $E^l_{n_1,\mathbf{j}_1}$. In other words, we need to show that all terms in Equations 3.4 and 3.5 are valid upper bounds.

1. By the inductive assumption, the original value $E^l_{n_1,\mathbf{j}_1}$ is an upper bound.
2. By the inductive assumption and the construction of Equation 3.2, $\max_{i=1}^{s} S^i_{n_1,\mathbf{j}_1}$ is an upper bound.
3. We now prove by contradiction that $f_{\lim} - g(t) - \sum_{i=2}^{b} E^l_{n_i,\mathbf{j}_i}$ is a valid upper bound if all violations of the distance matrices occur within the

sub-fragment-tree of n_1: Suppose there is a valid structure x for the sub-fragment-tree of n_1 with $g(x) > f_{\lim} - g(t) - \sum_{i=2}^{b} E^{l}_{n_i,\mathbf{j}_i}$. As we have explained earlier, no solution would be found if we perform a new search from t with $f_{\lim}$ (without changing $f_{\lim}$ or any E^l). Let t' be a descendant state of t. t' can be partitioned into the partial structure of t, and the partial structures from the sub-conformational search of $n_1, \ldots, n_b$. Therefore,

$$f(t') = g(t) + \sum_{i=1}^{b} f^{l}_{n_i,\mathbf{j}_i}(t').$$

Consider a particular dead end state t^* which follows x, as well as the "greedy" paths of $n_2, \ldots, n_b$ (Figure C-1). t^* has no violations because x is a valid structure. It must be terminated by condition 3.3. Thus $f(t^*) < f_{\lim}$. Since t^* is in the subtree of t,

$$f(t^*) = g(t) + \sum_{i=1}^{b} f^{l}_{n_i,\mathbf{j}_i}(t^*).$$

We know that for $i \in [2, \ldots, b]$, $f^{l}_{n_i,\mathbf{j}_i}(t') \geq E_{n_i,\mathbf{j}_i}$ because of the "greedy" paths. From the inductive assumption, $f^{l}_{n_1,\mathbf{j}_i}(t^*) \geq g(x) > f_{\lim} - g(t) - \sum_{i=2}^{b} E^{l}_{n_i,\mathbf{j}_i}$. Therefore,

$$f(t^*) > g(t) + \sum_{i=2}^{b} E^{l}_{n_i,\mathbf{j}_i} + f_{\lim} - g(t) - \sum_{i=2}^{b} E^{l}_{n_i,\mathbf{j}_i}$$

which means

$$f(t^*) > f_{\lim}.$$

We have reached a contradiction.

Since all terms of the update rules are correct, they must preserve the upper bound property.

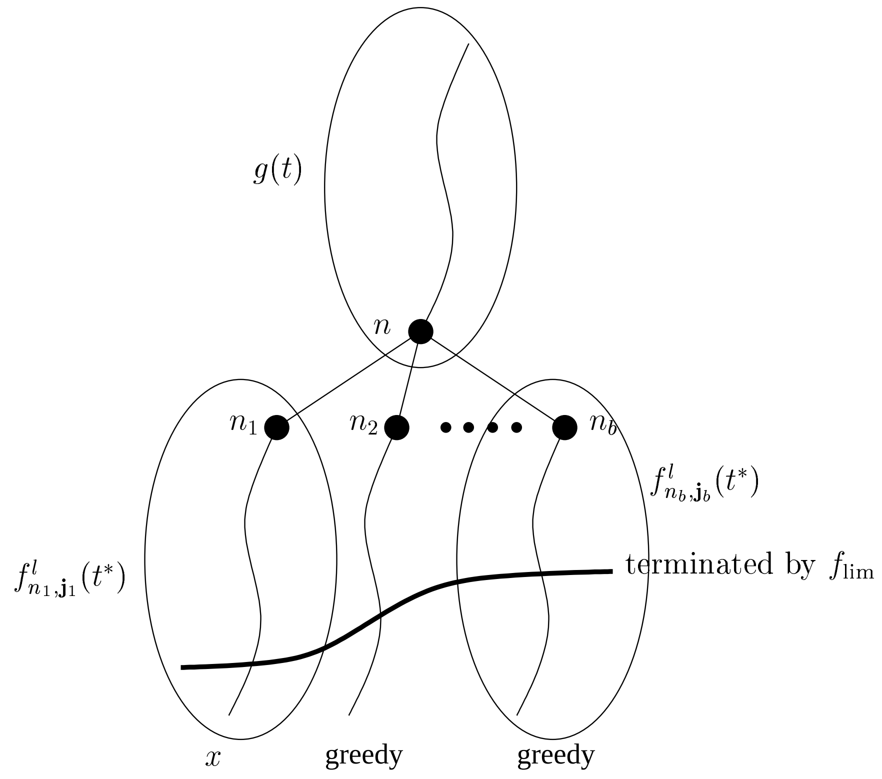

$g(t)$
n
n_1
n_2
n_b
$f^l_{n_b,\mathbf{j}_b}(t^*)$
$f^l_{n_1,\mathbf{j}_1}(t^*)$
terminated by f_{lim}
x
greedy
greedy

Figure C-1: An impossible dead end state whose f-value is greater than f_{lim} and that does not have any violations.

Bibliography

[1] Alfred Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1982.

[2] S. K. Arora. Correlation of structure and activity in ansamycins, molecular structure of sodium rifamycin SV. *Molecular Pharmacology*, 23:133–140, 1983.

[3] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer, Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The protein data bank: A computer-based archival file for macromolecular structures. *J. Mol. Biol.*, 112:535–542, 1977.

[4] G. Bricogne. Maximum entropy and the foundation of direct methods. *Acta Crystallographica*, A40:410–445, 1984.

[5] G. Bricogne. Direct phase determination by entropy maximization and likelihood ranking: Status report and perspectives. *Acta Crystallographica*, D49:37–60, 1993.

[6] Robert E. Bruccoleri and Martin Karplus. Prediction of the folding of short polypeptide segments by uniform conformational sampling. *Biopolymers*, 26:137–168, 1987.

[7] John J. Craig. *Introduction to Robotics*, chapter 2. Addison Wesley, 1986.

[8] G. M. Crippen and Timothy F. Havel. *Distance geometry and molecular conformation*. Chemometrics series. Wiley, 1988.

[9] P. R. David and S. Subbiah. Low-resolution real-space envelopes: the application of the condensing-protocal approach to the *ab initio* macromolecular phase problem of a variety of examples. *Acta Crystallographica*, D50:132–138, 1994.

[10] George T. Detitta, Charles M. Weeks, Pamela Thuman, Russ Miller, and Herbert A. Hauptman. Structure solution by minimal-function phase refinement and fourier filtering. I. theoretical basis. *Acta Crystallographica*, A50:203–210, 1994.

[11] Jan Drenth. *Principles of Protein X-ray Crystallography.* Springer-Verlag, 1994.

[12] David Eisenberg and Donald Crothers. *Physical Chemistry with Applications to the Life Sciences*, chapter 17. Benjamin Cummings, 1979.

[13] Dino R. Ferro and Jan Hermans. A different best rigid-body molecular fit routine. *Acta Crystallographica*, A33:345–347, 1977.

[14] Michael R. Garey and David S. Johnson. *Computers and intractability. A guid to the theory of NP-completeness.* W. H. Freeman, New York, 1979.

[15] C. Giacovazzo, editor. *Fundamentals of Crystallography*, chapter 5. Oxford University Press, 1992.

[16] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[17] J. Greer. Three-dimensional pattern recognition: an approach to automated interpretation of electron density maps of proteins. *Journal of Molecular Biology*, 82:279–301, 1974.

[18] Pehr B. Harbury, Tao Zhang, Peter S. Kim, and Tom Alber. A switch between two-, three-, and four-stranded coiled coils in GCN4 leucine zipper mutants. *Science*, 262:1401–1407, November 1993.

[19] T. Alwyn Jones, Jin yu Zou, S. W. Cowan, and M. Kjeldgaard. Improved methods for building protein models in electron-density maps and the location of errors in these models. *Acta Crystallographica*, A47:110–119, 1991.

[20] Guy L. Steele Jr. *Common Lisp.* Digital Press, Bedford, MA, second edition, 1990.

[21] Peter T. Lansbury Jr., Philip R. Costa, Janet M. Griffiths, Eric J. Simon, Michele Auger, Kurt J. Halverson, David A. Kocisko, Zachary S. Hendsch, Ted T. Ashburn, Richard G. S. Spencer, Bruce Tidor, and Robert G. Griffin. Structural model for the β amyloid fibril: interstrand alignment of an antiparallel β sheet comprising a C-terminal peptide. *Nature Structural Biology*, 2(11):990–998, 1995.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[23] Gerard J. Kleywegt and T. Alwyn Jones. Template convolution to enhance or detect structural features in macromolecular electron-density maps. *Acta Crystallographica*, D53:179–185, 1997.

[24] Eaton E. Lattman. Optimal sampling of the rotation function. *Acta Crystallographica*, B28:1065–1068, 1972.

[25] L.-P. Lee and Bruce Tidor. Optimization of electrostatic binding free energy. *Journal of Chemical Physics*, 106:8681–8690, 1997.

[26] L. Leherte, S. Fortier, J. Glasgow, and F. H. Allen. Molecular scene analysis: A topological approach to the automated interpretation of protein electron density maps. *Acta Crystallographica*, D50:155–166, 1994.

[27] Jae S. Lim. *Two-dimensional signal and image processing.* Prentice-Hall signal processing series. Prentice Hall, Englewood Cliffs, N.J., 1990.

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

[29] V. Yu. Lunin, N. L. Lunina, T. E. Petrova, E. A. Vernoslova, A. G. Urzhumtsev, and A. D. Podjarny. On the *ab initio* solution of the phase problem for

macromolecules at very low resolution: The few atoms model method. *Acta Crystallographica*, D51:896–903, 1995.

[30] Duncan E. McRee. *Practical Protein Crystallography.* Academic Press, 1993.

[31] Michael Nilges. A calculation strategy for the structure determination of symmetric dimer by ^{1}H NMR. *Proteins: Structure, Function, and Genetics*, 17:297–309, 1993.

[32] Michael Nilges. Calculation of protein structures with ambiguous distance restraints. automated assignment of ambiguous NOE crosspeaks and disulphide connectivities. *Journal of Molecular Biology*, 245:645–660, 1995.

[33] Sean I. O'Donoghue, Glenn F. King, and Michael Nilges. Calculation of symmetric multimer structures from NMR data using a priori knowledge of the monomer structure, co-monomer restraints and interface mapping: The case of leucine zippers. *Journal of Biomolecular NMR*, 8:193–206, 1996.

[34] B. G. Patrick, M. Almulla, and M. M. Newborn. An upper bound on the time complexity of iterative-deepening-A*. *Annals of Mathematics and Artificial Intelligence*, 5:265–278, 1992.

[35] G. G. Prive, D. H. Anderson, L. Wesson, D. Cascio, and D. Eisenberg. Packed protein bilayers in the 0.9Å resolution structure of a designed alpha helical bundle. *Protein Science*, 8(7):1400–1409, 1999.

[36] S. Narasinga Rao, Jyh-Hwang Jih, and Jean Ann Hartsuck. Rotation-function space groups. *Acta Crystallographica*, A36:878–884, 1980.

[37] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*, chapter 4. Prentice Hall, 1995.

[38] George M. Sheldrick and Robert O. Gould. Structure solution by iterative peaklist optimization and tangent expansion in space group P1. *Acta Crystallographica*, B51:423–431, 1995.

[39] Stefan Steinbacher, Robert Seckler, Stefan Miller, Boris Steipe, Robert Huber, and Peter Reinemer. Crystal structure of P22 tailspike protein: Interdigitated subunits in a thermostable trimer. *Science*, 265:383–386, 1994.

[40] M. M. Teeter, S. M. Roe, and N. H. Heo. Atomic resolution (0.83 Å) crystal structure of the hydrophobic protein crambin at 130 K. *Journal of Molecular Biology*, 230:292, 1993.

[41] Jin yu Zou and T. Alwyn Jones. Towards the automatic interpretation of macromolecular electron-density maps: Qualitative and quantitative matching of protein sequnce to map. *Acta Crystallographica*, D52:833–841, 1996.