Acta Crystallographica Section D
Biological Crystallography
ISSN 0907-4449

ConfMatch: automating electron-density map interpretation by matching conformations

Cheuksan Edward Wang

Artificial Intelligence Lab, MIT, 545 Technology Square, Cambridge, MA 02139-3539, USA

Correspondence e-mail: wang@ai.mit.edu

Received 28 February 2000
Accepted 29 August 2000

Building a protein model from the initial three-dimensional electron-density distribution (density map) is an important task in X-ray crystallography. This problem is computationally challenging because proteins are extremely flexible. The algorithm *ConfMatch* is a global real-space fitting procedure in torsion-angle space. It 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* is a practical systematic algorithm based on a branch-and-bound search. 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, 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. Experiments have shown that *ConfMatch* may be able to automate the interpretation of density maps of small proteins.

1. Introduction

Determining the structures of proteins is essential to understanding the molecular biology of cells. X-ray crystallography is the 'gold standard' for protein-structure determination. This paper 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. *ConfMatch* performs a global conformational search of all dihedral, translational and rotational degrees of freedom. 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 detailed and has reasonable geometry and stereochemistry. It includes all non-H atoms of the target molecule. The conformation satisfies various commonly accepted chemical constraints such as bond lengths, angles, chirality *etc.*

ConfMatch has two important potential applications.

(i) 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.

(ii) 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 (§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 (Lee & Tidor, 1997). 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.

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 declares them the atom positions. The identities of different peaks are usually labelled manually. Peak picking has long been used in small-molecule direct methods, as 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 data of resolution worse than 1.2 Å. At lower resolution, atoms rarely locate at the peaks of the electron-density map. Most macromolecules of interest diffract to 2.0 Å 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 (Greer, 1974) is an early 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. The skeleton is an unlabelled structure and usually has many errors and ambiguities. Adding the atom labels, as well as correcting errors in the skeleton, must be performed manually in order to build an initial model.

Based on skeletonization, Jones & Thirup (1986) developed a widely used method for building protein crystal structures. They observed that a protein structure can be constructed from fragments of other proteins. Thus, the known protein structures can be treated as a knowledge base from which one extracts information to be used in molecular modelling. Their method first calculates a skeleton from the electron-density map. The errors in the skeleton are located and corrected interactively. Fragments of 5–7 residues from known structures are matched to the skeleton. After the best-matching fragments are selected, the fit of each residue can be further improved to generate an initial model.

Terry's *CRYSALIS* system (Terry, 1988) is another map-interpretation method based on skeletonization. *CRYSALIS* is an expert system which labels the peaks and ridges of a skeleton. Its knowledge base consists of many heuristics concerning protein crystal structures. Terry developed a hierarchical production system in which control proceeds through many levels of strategy heuristics until one specific action at the problem-solving bottommost level is selected. Unfortunately, *CRYSALIS* cannot overcome the many errors usually present in skeletons.

Molecular-scene analysis (Leherte *et al.*, 1994) is a new approach to map interpretation. At medium (∼3 Å) 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 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 loca-

tions 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 (Zou & Jones, 1996). 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 (Kleywegt & Jones, 1997) 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.

Recently, Perrakis and coworkers developed the *warpNtrace* method (Perrakis *et al.*, 1999) to automate protein model building. It attempts to automatically construct a protein model starting from electron-density maps without user intervention. This method is based on an iterative procedure that describes the electron-density map as a set of unconnected atoms and then searches for protein-like patterns. For many proteins, *warpNtrace* can construct a large part of the backbone and some of the side chains. An important requirement is that the diffraction data extends to 2.3 Å or better.

The goal of *ConfMatch*, similar to that of *warpNtrace*, is to fully automate map interpretation. No human guidance is required in constructing a detailed high-quality molecular structure. *ConfMatch* generates its output conformation directly from the density map.

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. Thus, it is currently practical to use *ConfMatch* to solve only small proteins or small parts of large proteins. 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 or lower resolution diffraction data than other algorithms. In addition, *ConfMatch*'s result is a complete structure. The output includes every backbone and side-chain atom specified by the user. This completeness property cannot be achieved by most other methods.

3. The conformational matching problem

This section describes an approach to interpreting an electron-density map by solving the conformational matching problem, *i.e.* 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 as follows: *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-H) 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 (§4). The objective function, the sum of the density at non-H atom positions, ignores the different identities of atoms. A C atom occupying a position is valued the same as if an N atom occupies it. If all non-H 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-H atoms have much higher atomic numbers than others, the objective function may need to be modified. One possible solution (§5.2.1) 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.

Table 1
These symbols are defined in this paper and listed in the order of their appearance.

Notation	Description	Section
p	Number of free dihedral angles	§3
n	A node in the fragment tree	§4.1
j	Bond placement	§4.1
$E_{n,/bfj}$	An upper bound of the density sum of n's sub-fragment tree, where n's in-bond is placed at **j**	§4.1
s	Number of torsional samples	§4.1
$e^i_{n,\mathbf{j}}$	The density sum of the ith sample of fragment n	§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.	§4.1
R	A rotational transformation in Cartesian coordinates	§4.2.2
θ_L	Sampling interval of torsional angles (pseudo-uniform)	§4.2.2
t	A state in conformational search	§4.3
$f(t)$	An upper bound of the density sum of the entire structure given the current partial structure at t	§4.3
$g(t)$	Density sum of the partial structures at t	§4.3
$h(t)$	An upper bound of the density sum of the remaining structure	§4.3
$f_{\rm lim}$	f-value limit for a depth-first search	§4.3
M	An upper bound of the density sum of the entire structure	§4.3
d	The density sum of a solution structure	§4.4.4
ε	The minimal improvement in solution we can accept	§4.4.4
C	A transformation from fractional into Cartesian coordinates	§5.2.2

This is especially true in the presence of errors in the density map. In order to find the global 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 $6 + p$, where p is the number of free dihedral angles. (Table 1 lists all symbols defined in this paper.) The extra six degrees of freedom comes from the rigid displacements. Even for very small proteins, p can run into 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.

4. The *ConfMatch* algorithm

ConfMatch is a systematic algorithm for solving the *discretized* conformational matching problem. The discretization specifies a three-dimensional grid in space. For ease of implementation, we require that the three 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 0.5 Å as the grid spacing. In the rest of this paper, 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 (Lipton & Still, 1998). 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

(i) bond angles,

(ii) bond lengths,

(iii) *cis*/*trans* and planarity of double bonds,[1]

(iv) chirality of all chiral centers,

(v) intramolecular excluded volume constraints, *i.e.* any pair of non-bonded atoms in the same molecule must be further apart than the sum of their hard sphere radii,

(vi) intermolecular excluded volume constraints, *i.e.* any pair of atoms in different molecules (including symmetry mates) must be further apart than the sum of their hard sphere radii (bump checking).

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** ∪ **non-local**). The bound-preprocessing stage satisfies the **local** set:

(i) 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. §4.5 describes a simple extension to the algorithm that handles this case.

(ii) lengths of all bonds except flexible-ring forming ones,

(iii) *cis*/*trans* and planarity of all double bonds except flexible-ring forming ones,

(iv) 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 §4.5. Fig. 1 shows a bond being 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:

(i) angles, lengths and planarity of ring-forming bonds,

(ii) intramolecular excluded volume constraints,

(iii) intermolecular excluded volume constraints.

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 as follows: *without any constraints on the dihedral angles, the optimization of the 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** 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.

(i) Most bonds are included in the calculation. The ring-forming ones constitute a small percentage of the bonds in a typical macromolecule.

(ii) All important local geometries, including angles, lengths, planarity and chirality, are considered in this stage.

Results in §5 will show that the difference between the upper bound and the solution is usually very small for data with a low to medium level of error. This difference is sometimes less than the density of a single atom.

At the first glance, it is not obvious that the bound-preprocessing stage can obtain its results with polynomial time complexity. It may seem that the problem of finding the maximum density sum, under only **local** contraints, still requires an exponential time search: one needs to evaluate all possible combinations of all torsion angles, with the molecule placed at all possible positions inside the asymmetric unit and in all possible orientations. However, the same results can be found in polynomial time by a bottom-up dynamic programming approach (Aho *et al.*, 1982). This approach begins with the observation that all molecules can be viewed as an assembly of small rigid *fragments*. For instance, Fig. 2 shows how a glycine molecule can be formed from two rigid fragments. Note that all H atoms are 'unified' with their heavier neighbors because we do not calculate the positions of H atoms explicitly. H 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

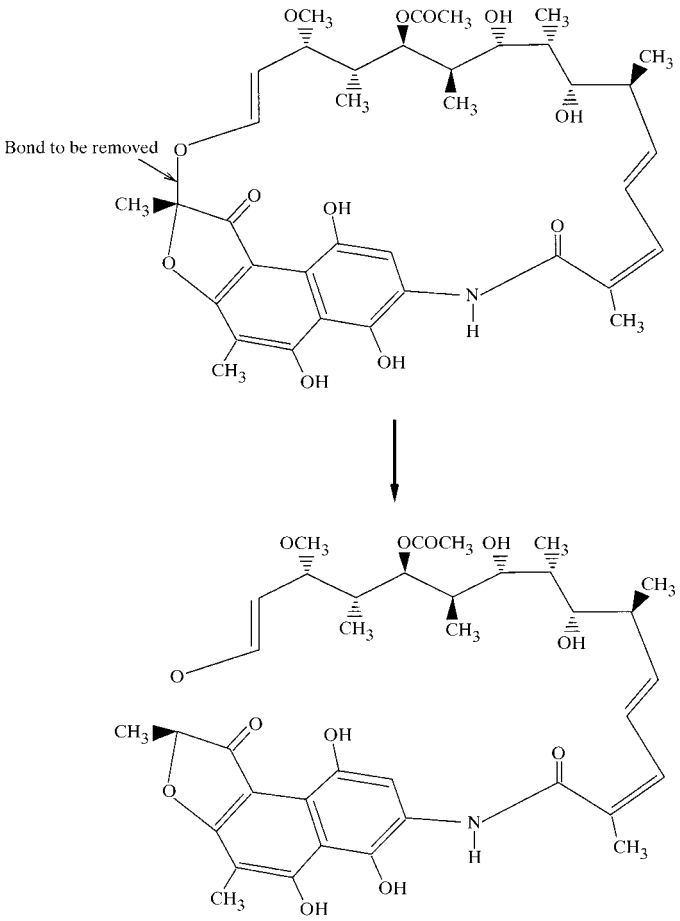

Figure 1
A bond is removed from the flexible ring of rifamycin SV (Arora, 1983), a macrocyclic molecule. The smaller rings are rigid and not broken apart. The local constraint set specifies the local geometries of the reduced molecule.

α-carbon of valine would have two out-bonds: one along the protein backbone and the other along the valine side chain. 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 side chains. 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.

Fig. 3 illustrates the basic idea of the dynamic programming approach by a two-dimensional analogy. In this example, we attempt to find the maximum density sum of a 'glycine' in a rectangular unit cell with a rectangular grid. The concept behind this two-dimensional example can be easily generalized to three dimensions.

Step 1. Separate the flexible glycine molecule into two rigid fragments (fragments 1 and 2).

Step 2. Evaluate the electron-density sum of fragment 2 at all possible positions inside the unit cell and in all possible orientations.

Step 3. Store the density sums in a giant table, indexed by the placements of fragment 2's in-bond; *i.e.* for each possible placement of the in-bond, we store the highest density sum.

Step 4. Evaluate the electron-density sum of fragment 1 at all possible positions inside the unit cell and in all possible orientations.

Step 5. Store (the density sum of fragment 1) + (maximum density sum of compatible placements of fragment 2) in a table indexed by the placements of fragment 1's in-bond. Compatibility here implies that fragment 1's out-bond coincides with fragment 2's in-bond. Fragment 2's sum need not be recalculated. It can be simply looked up from the table in step 3.

One can see that the table from the last step indeed gives the maximum density sum overall. The highest value in this table is identical to that obtained by evaluating all possible values of the torsion angle, with the molecule placed at all possible positions and orientations. This fact can be proven by a mathematical induction. Furthermore, if a molecule has more torsion angles, more fragments will be generated and more tables will be constructed. However, the number of tables and steps are always proportional to the number of fragments. Thus, the running time of the dynamic programming method only grows linearly with the number of torsion angles.

Now we give a more detailed description of the bound-preprocessing stage. The output of this 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 and **j** is the position of a pair of grid points

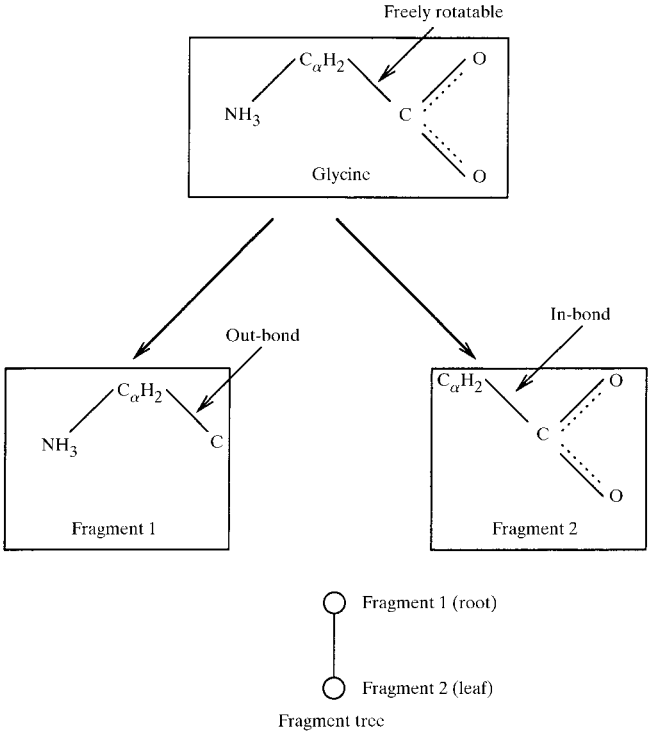

Figure 2
A glycine molecule can be separated into two rigid fragments. Its fragment tree has two nodes.

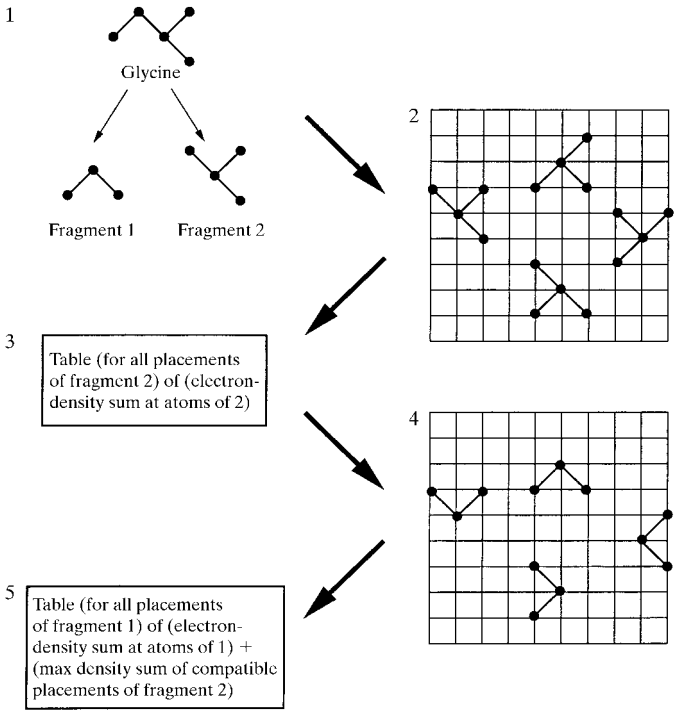

Figure 3
The bound-preprocessing stage executes on a two-dimensional example. Actual executions, in three dimensions, involve similar steps.

$$\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix} \text{ and } \begin{pmatrix} x'_j \\ y'_j \\ z'_j \end{pmatrix}.$$

$E_{n,\mathbf{j}}$, stores the maximum density sum (satisfying only the **local** set) of the sub-fragment tree of n (Fig. 4), where n's in-bond is located at **j**; that is, the first and second atom of the in-bond are located at

$$\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix} \text{ and } \begin{pmatrix} x'_j \\ y'_j \\ z'_j \end{pmatrix},$$

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

The number of fragments in a structure is equal to one plus its torsional degrees of freedom p. A typical bond is between 1 and 2 Å, 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

a small constant $\times (1 + p) \times$ 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 11 000 Å^3 unit cell. §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 **j**, the bond placement, is the inner loop. This calculation is performed 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 perform a uniform torsional sampling about the in-bond and store the maximum density sum. Fig. 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 **j**. The ith sample is generated by a rotation of $\theta_i = 2\pi i/s$ around n's in-bond. This rotation corresponds to the following rigid transform (Craig, 1986):

$$\begin{bmatrix} k_x k_x v\theta_i + c\theta_i & k_x k_y v\theta_i - k_z s\theta_i & k_x k_z v\theta_i + k_y s\theta_i & x_j \\ k_x k_y v\theta_i + k_z s\theta_i & k_y k_y v\theta_i + c\theta_i & k_y k_z v\theta_i - k_x s\theta_i & y_j \\ k_x k_z v\theta_i - k_y s\theta_i & k_y k_z v\theta_i + k_x s\theta_i & k_z k_z v\theta_i + c\theta_i & z_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & -x_j \\ 0 & 1 & 0 & -y_j \\ 0 & 0 & 1 & -z_j \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

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 $v\theta_i = 1 - \cos\theta_i$. §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 ith 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 two. To avoid double counting, we do not include the two atoms of the in-bond. These atoms are included in the density sum of the parent fragment. Then,

$$E_{n,\mathbf{j}} = \max_{i=1}^{s} e^i_{n,\mathbf{j}}. \quad (1)$$

The inner nodes's bounds can be calculated based on their children's values. Suppose n is an inner node whose children

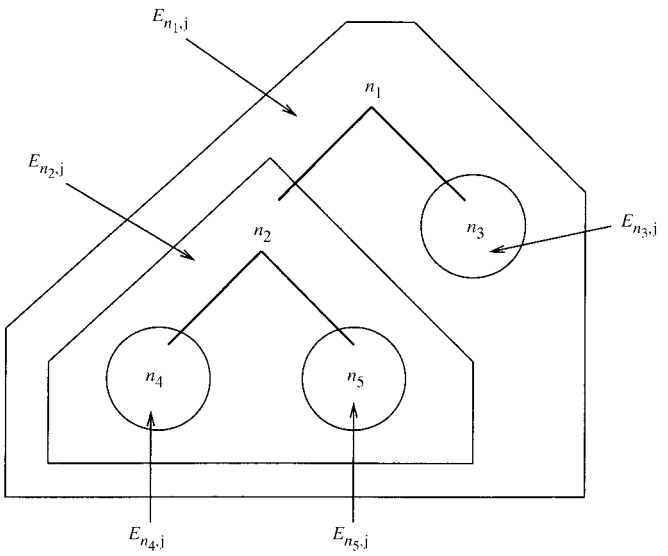

Figure 4
A fragment tree and its entries in the bounds table. Each set of entries stores the upper bounds of a sub-fragment tree.

[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 the $\frac{2}{3}$ power of the number of grid points if we assume uniform grid spacing.

Table 2
Comparison of different optimizations of the bound-preprocessing stage.

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

are nodes $n_1, n_2, \ldots, n_m$. At every grid-point pair, we also perform 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 $\mathbf{j}_l^i$ 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,

$$E_{n,\mathbf{j}} = \max_{i=1}^{s}(e_{n,\mathbf{j}}^i + \sum_{l=1}^{m} E_{n_l,\mathbf{j}_l^i}). \quad (2)$$

If we define $S_{n,\mathbf{j}}^i = e_{n,\mathbf{j}}^i + \sum_{l=1}^m E_{n_l,\mathbf{j}_l^i}$, then

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

$S_{n,\mathbf{j}}^i$ can be considered 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 $E_{n_l,\mathbf{j}_l^i}$ from the table. Let u be the time required to calculate each entry in the table.

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

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

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

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 2 is a comparison of these optimization methods. *ConfMatch* integrates all of them to solve large problems efficiently.

4.2.1. 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

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

for all $\mathbf{j}$. We can avoid redundant calculations if common subtrees can be discovered and merged. Fig. 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 side-chain 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 side-chain bounds is amortized over many residues. Each additional residue usually adds only two nodes to the tree, corresponding to the additional φ and ψ angles on the main chain.

4.2.2. Precomputing torsional sampling. Calculating each entry in the bounds table requires a torsional sampling about an in-bond. With some precomputation, we can avoid the expensive multiplication by a 4×4 transformation matrix at every sample. The precomputation involves a pseudo-uniform

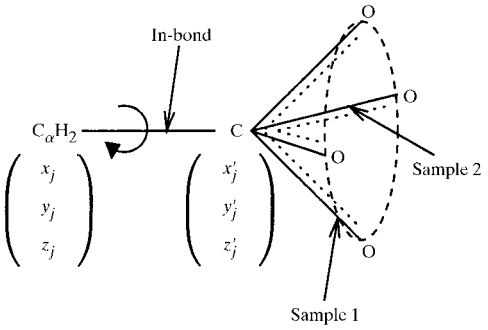

Figure 5
A torsional sampling about the in-bond of the second fragment of glycine.

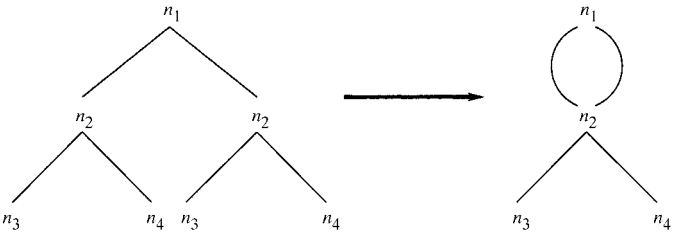

Figure 6
Three nodes of a fragment tree are eliminated by common subtree elimination.

rotational sampling using Lattman's method (Lattman, 1972). This method generates a number of rotational matrices (R_1, R_2, ...) 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 $R_1(N)$, $R_2(N)$, ... These configurations are classified based on the orientations of their in-bonds. Let

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \text{ and } \begin{pmatrix} x_i' \\ y_i' \\ z_i' \end{pmatrix}$$

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

$$\begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix} \text{ and } \begin{pmatrix} x_j' \\ y_j' \\ z_j' \end{pmatrix},$$

$R_1(N)$ would be selected if

$$\begin{pmatrix} x_i' - x_i \\ y_i' - y_i \\ z_i' - z_i \end{pmatrix} = \begin{pmatrix} x_j' - x_j \\ y_j' - y_j \\ z_j' - z_j \end{pmatrix}.$$

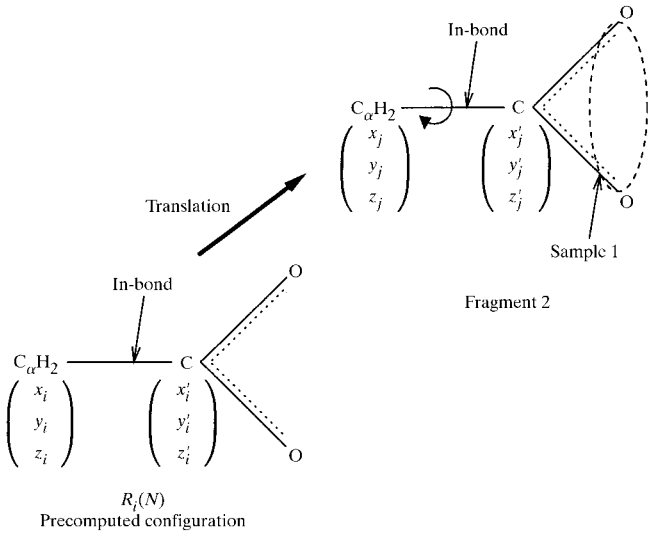

Figure 7
A torsional sample of the second fragment of glycine is generated from a precomputed configuration.

The required configuration is generated by translating $R_i(N)$ by

$$\begin{pmatrix} x_j - x_i \\ y_j - y_i \\ z_j - z_i \end{pmatrix}.$$

Fig. 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° 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 (Fig. 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 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*.

4.2.3. 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.

(i) The electron-density distribution has the same symmetry as the crystal. This is always true with appropriate input data.

(ii) 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 three axes. If the unit cell has two-, three-, four- or sixfold 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.

(iii) 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

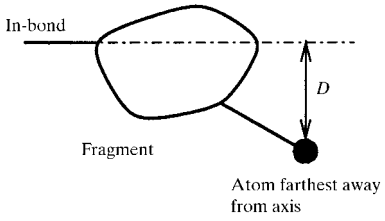

Figure 8
To assure completeness, it is necessary to choose θ_L based on the distance between the in-bond axis and the atom farthest away from it.

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 example, this optimization will result in a twofold reduction in time and space of the bound-preprocessing stage for space group $P2_1$.

4.2.4. Reducing the size of each table entry. Electron-density values are usually stored as floating-point numbers, which usually take 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 losing 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.

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 $n_{\rm root}$ be this root node. $\mathbf{j}^* = \arg\max_{\mathbf{j}} E_{n_{\rm root},\mathbf{j}}$ is the pair of grid points where the in-bond resides. Then one finds $\arg\max_{i=1}^{s} S^i_{n_{\rm root},\mathbf{j}^*}$, which gives the torsion angle of the in-bond. This provides the exact configuration of the root fragment, from which $n_{\rm root}$'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 (Crippen & Havel, 1988), one intramolecular and one intermolecular. Each distance matrix describes a lower and an upper bound of the distance between every pair of atoms. 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. Obviously, the diagonal entries of this matrix are zeroes. 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 three 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 (bump checking). 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.

(i) The initial state is the null structure.

(ii) The successor states of the initial state consist of all **j** pairs which can potentially be the in-bond of the root fragment.

(iii) Every intermediate state is a connected 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. A set of open out-bonds consists of at most one open out-bond on the main chain and any number of open out-bonds on the side chains. The successor states of an intermediate state consist of every torsional sampling of every open out-bond.

(iv) The goal states are structures that satisfy the distance matrices and do not have any open out-bonds. All fragments' positions are specified.

(v) The value of a state is the sum of density of fragments in the partial structure. This 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 (Fig. 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.[3] 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

[3] Intuitively, at any intermediate state one may choose to define the next main-chain torsion angle or one of the several side-chain angles. This choice, the ordering among main chain and side chains, does not affect the end result, where every angle must be defined.

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 §4.4.2.

One possible approach to the problem is to use an informed search method like the A* algorithm (Russell & Norvig, 1995). When A* reaches a state t, it calculates an evaluation function, $f(t)$, 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.[4]

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

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.[5] 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 is available practically.

Iterative deepening A* (IDA*; Russell & Norvig, 1995) 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 ($f_{\rm lim}$) – a guess of the best possible value. Each depth-first search determines whether a solution exists above $f_{\rm lim}$. If a solution is found, IDA* terminates. Otherwise, we reduce the guess $f_{\rm lim}$ and perform another iteration.

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

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

Thus, each iteration expands all nodes inside the contour of the current $f_{\rm lim}$. If $f_{\rm lim}$ 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 new iteration is started using a lower $f_{\rm lim}$ 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 the depth-first search is rather small (Patrick *et al.*, 1992). 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 $f_{\rm lim}$ is close to the best goal value, the depth-first search will search a few more nodes than A*. IDA*'s efficiency will be close to optimal. However, if the last $f_{\rm lim}$ is much smaller than the best goal value, IDA* will search many more nodes than A*.

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

Instead of the best conformation, sometimes one may like to obtain a family of good solutions. This can be achieved by a simple extension to *ConfMatch*: after obtaining the best solution, we set $f_{\rm lim}$ to be slightly lower than the best density sum. We then perform an additional depth-first search and collect a family of solutions whose values are all better than $f_{\rm lim}$.

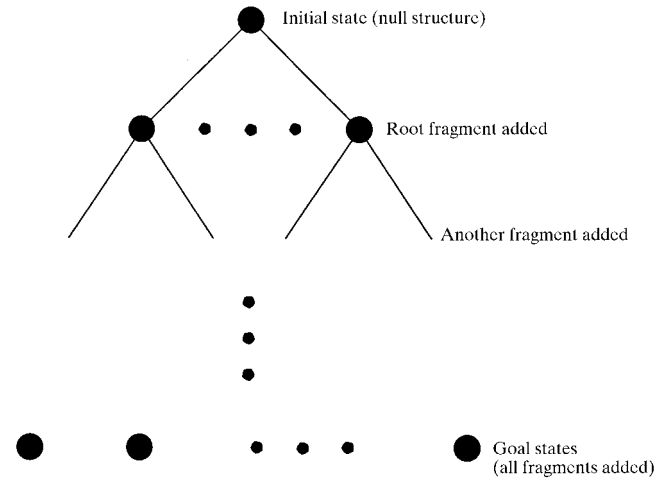

Figure 9
Starting from the initial state, the goal of the search stage is to find the highest value goal state.

[4] 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.

[5] 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.* $|h(t) - h^*(t)| \geq O[h^*(t)]$.

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.

4.4.1. 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.

4.4.2. 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 the search. We found that the most-constrained variable heuristic (Russell & Norvig, 1995) 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

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

$g(t)$ 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 three 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.

4.4.3. 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.

4.4.4. 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 times. Many paths in the search tree may have the same open out-bond placement and hence require the same upper bound.[6] 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, $E_{n,/bfj}$, 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 $E^l_{n,\mathbf{j}} \leq E_{n,/bfj}$ for all n, $\mathbf{j}$.

Initially, $E^l_{n,\mathbf{j}} = E_{n,/bfj}$ 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

$$f_{\lim} \leq g(t) + \sum_{i=1}^{b} E^l_{n_i,\mathbf{j}_i} \qquad (3)$$

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 $d + \varepsilon$, 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 (3) will not be satisfied with the updated $f_{\lim}$. This requires $E_{n_1,\mathbf{j}_1} \leq f_{\lim} - g(t) - \sum_{i=2}^{b} E^l_{n_i,\mathbf{j}_i}$.

[6] 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.

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

(i) 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

$$\min[f_{\lim} - g(t) - \sum_{i=2}^{b} E^l_{n_i,\mathbf{j}_i}, E^l_{n_1,\mathbf{j}_1}, \max_{i=1}^{s} S^i_{n_1,\mathbf{j}_1}]. \quad (4)$$

(ii) 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

$$\min(E^l_{n_1,\mathbf{j}_1}, \max_{i=1}^{s} S^i_{n_1,\mathbf{j}_1}). \quad (5)$$

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 (2). These terms are necessary for the consistency of the bounds table. *Appendix A* gives a rigorous correctness proof of these update rules.

Memoization lowers the upper bounds continually during the search. The better bounds means (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 a speed-up of an order of magnitude.

4.5. Algorithm extension

Sometimes details of a molecule's local covalent structure are not known 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 (Fig. 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 $n_{\rm cis}$ and $n_{\rm trans}$, respectively. Their entries in the bounds table, $E_{n_{\rm cis},\mathbf{j}}$ and $E_{n_{\rm trans},\mathbf{j}}$, are calculated separately. The bounds of their parent fragment, $n_{\rm parent}$, are calculated from the maximum of $n_{\rm cis}$'s and $n_{\rm trans}$'s bounds. Without loss of generality, we assume that n is the only child of $n_{\rm parent}$ in the fragment tree. Following (2), the bounds of $n_{\rm parent}$ are calculated by

$$E_{n_{\rm parent},\mathbf{j}} = \max_{i=1}^{s}[e^i_{n_{\rm parent},\mathbf{j}} + \max(E_{n_{\rm cis},\mathbf{j}^i}, E_{n_{\rm trans},\mathbf{j}^i})].$$

Thus, $E_{n_{\rm parent},\mathbf{j}}$ 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 $n_{\rm cis}$ and $n_{\rm trans}$. 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 $n_{\rm cis}$ samples will be much higher than those of $n_{\rm trans}$. Thus, the $n_{\rm trans}$ 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 side chain as one of several possible amino acids. Furthermore, we can specify only the length of a peptide, but leave all side chains 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 (§1).

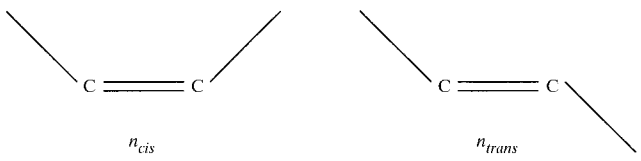

Figure 10
A fragment is ambiguously defined as either a *cis* or a *trans* double bond.

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 α-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 (McRee, 1992). The results given below were run on a 533 MHz Digital Alpha 21164 PC processor with 512 Mb of RAM.

5.1. Alpha-1: a designed α-helical peptide

The first test of the *ConfMatch* algorithm is a designed α-helical peptide, Alpha-1 (Prive *et al.*, 1999; PDB code 1byz; Bernstein *et al.*, 1977). 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 four-helix bundle. The crystal of Alpha-1 diffracts to 0.9 Å resolution with 23 681 structure factors. The space group of the crystal is $P1$. The unit-cell parameters are a = 20.846, b = 20.909, c = 27.057 Å, α = 102.40, β = 95.33,

Table 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).

Resolution (Å)	Number of reflections	RMSD (Å)	Search stage last iteration time (s)	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

$\gamma = 119.62^\circ$. There is a single bundle with four chains in each unit cell.

Since the four 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-H atoms, 55 free dihedral angles and 61 degrees of freedom in total. Alpha-1 has no flexible rings and therefore no ring-forming bonds. We obtain the geometry of our fragments by decomposing a set of ideal residues. We use standard van der Waal's radii for intermolecular 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 0.5 Å 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 1.216 Gb of storage.

We tested *ConfMatch* with diffraction data at 2.0 Å. The density distribution is generated from 2548 structure factors with their published phases. This input is merely 10.8% of the data at 0.9 Å. 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 take 14 700 and 17 s of CPU time, respectively. The overwhelming majority of the running time is spent on the first stage. Fig. 11 shows the solution structure from *ConfMatch* superimposed with the published 0.9 Å structure. *ConfMatch*'s result has an RMSD (root-mean-square deviation) of 0.812 Å 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. 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 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

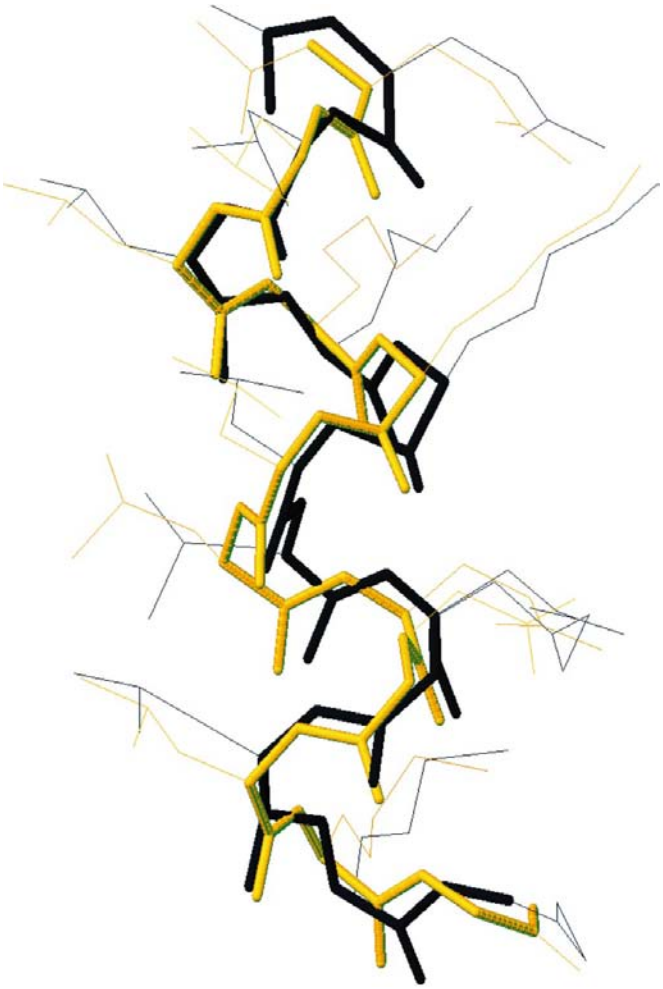

Figure 11
ConfMatch's solution structure (in black) of Alpha-1 from 2.0 Å resolution data and the published 0.9 Å structure (in yellow). The thicker portions are the backbones of the structures.

Table 4
Alpha-1's conformation is matched to phases with various level of error using 2 Å resolution data.

DIFF, difference between the global upper bound M and solution density (equivalent number of atoms).

Phase error (degree standard deviation)	RMSD (Å)	Search stage last iteration	DIFF time (s)
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

deteriorate with worse resolution of the data. There is a large jump in RMSD from 3.3 to 3.4 Å. Fig. 12 shows the 3.4 Å solution structure superimposed with the target structure. The backbones of those two structures are significantly different. For Alpha-1, 3.3 Å seems to be the resolution limit where *ConfMatch* 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 3.3 Å 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.

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.[7] By varying the standard deviation of the Gaussian distribution, we can measure *ConfMatch*'s tolerance. The results from 2 Å resolution data are shown in Table 4. As expected, all performance measures worsen with increasing phase error. The RMSD generally increases with phase error. There is a large increase in both RMSD and search time from 50 to 55°. At 55° 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° may be the limit of *ConfMatch*'s error tolerance of Alpha-1 at 2 Å. We expect this tolerance to improve with higher resolution data and to shrink with worse data.

From Tables 3 and 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° 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*.

5.2. Crambin

The second test of the *ConfMatch* algorithm is crambin (Teeter *et al.*, 1993; PDB code 1ab1), a small 46-residue protein. The crystal of crambin diffracts to 0.89 Å resolution with 19 803 structure factors. The space group of the crystal is $P2_1$. The unit-cell parameters are a = 40.759, b = 18.404, c = 22.273 Å, α = 90.00, β = 90.70, γ = 90.00°. This molecule has 326 non-H atoms, 141 free dihedral angles and 147 degrees of freedom total.

5.2.1. Modifying the objective function of conformational matching. Crambin has six cysteine residues which form three disulfide bonds. Sulfur has a higher electron density than N, C or O because it has a larger atomic number. The overall density distribution has several large peaks corresponding to the sulfur positions. Fig. 13 plots the density of the highest peaks of a typical crambin distribution. There is a significant gap between the sixth highest peak and the seventh one because of the difference between the six S atoms and others. If we use the simple objective function, the sum of density at

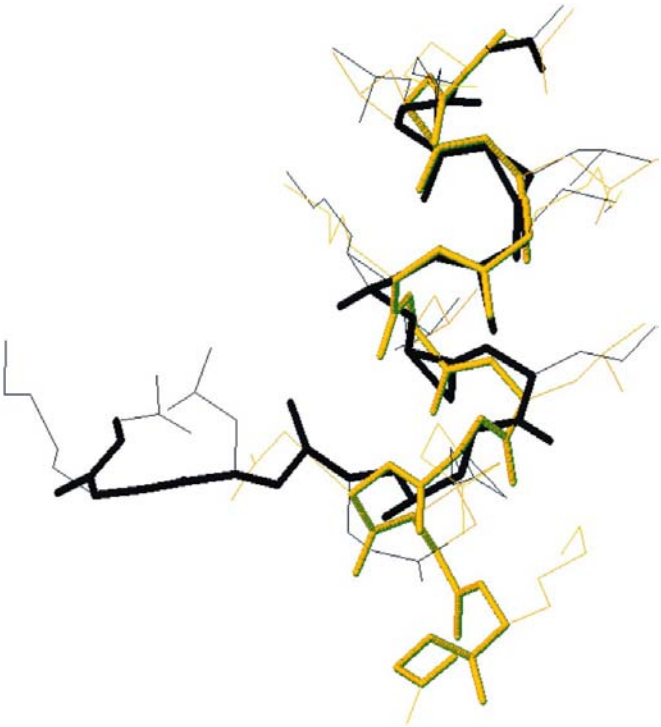

Figure 12
ConfMatch's solution structure (in black) of Alpha-1 from 3.4 Å resolution data and the published 0.9 Å structure (in yellow). The thicker portions are the backbones of the structures.

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

atom locations and ignore the different identities of atoms, the sulfur locations will become strong 'attractors' of all other atoms. Consequently, the bound-preprocessing stage will place multiple atoms at the sulfur 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 two classes: (i) S atoms and (ii) all other non-H atoms. The S atoms, using the original density distribution, will preferably locate at the highest density regions. All atoms other than sulfur 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 seven highest peaks of the original distribution, one more than the number of S atoms. Let $P_1, \ldots, P_7$ be the densities of the seven peaks. The difference between P_i and P_7 is a measure of our confidence that the ith peak shall be occupied by sulfur. We suppress the ith peak by subtracting a typical S-atom density distribution, scaled to height $2(P_i - P_7)$, from the original distribution. After the six highest peaks are suppressed, we have removed much of the influence of the S 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.

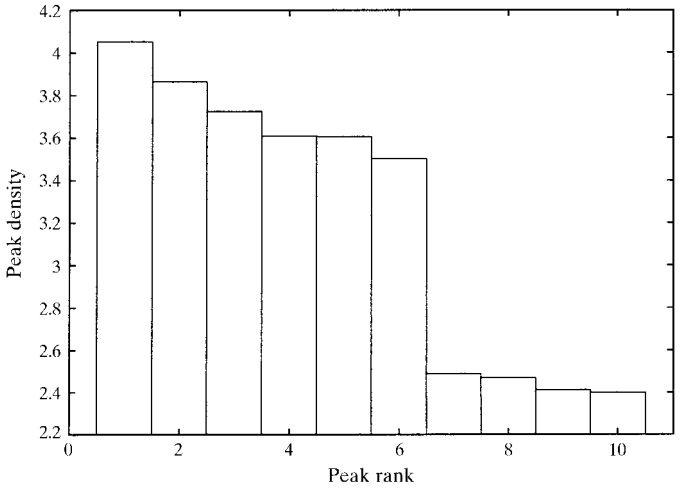

Figure 13
The highest peaks in crambin's 2.0 Å resolution density distribution.

If one has metal ions that one did not know about in advance, the contribution of these ions to the density map cannot be easily removed. The ions' strong peaks may become strong 'attractors' of all other atoms. To avoid this problem, one can incorporate the real-space correlation coefficient into the objective function, as described in §6.

5.2.2. Utilizing crambin's crystal symmetry. The crambin crystal's $P2_1$ symmetry means that there are two 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 §4.2.3 can be satisfied. The $P2_1$ space group has a twofold screw axis parallel to b (Fig. 14). If a particle is located at fractional coordinates

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix},$$

there must be an identical particle symmetrically located at

$$\begin{pmatrix} -u \\ v + \frac{1}{2} \\ -w \end{pmatrix}.$$

The symmetry operation can be represented by the matrix

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

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 $\frac{1}{2}$ in the **b** direction, $\frac{1}{2}$ must be divisible by $1/b$ in order for the grid to be invariant. Therefore, b must be an even number. We selected a

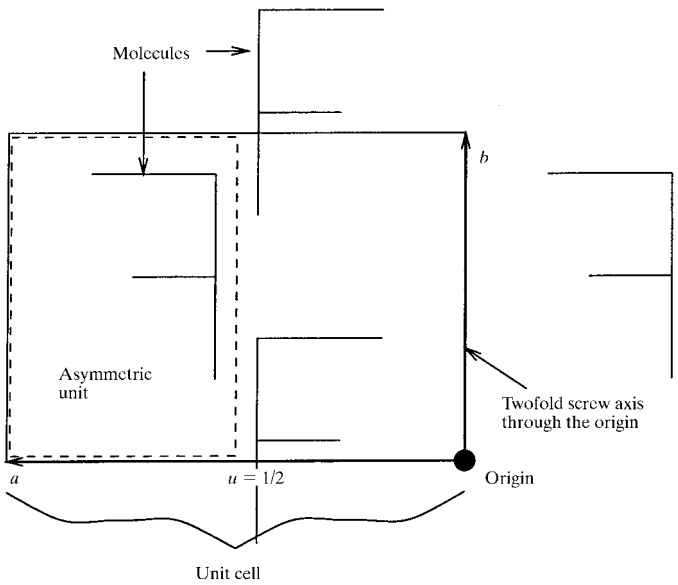

Figure 14
The twofold screw axis of the $P2_1$ space group in two dimensions. The **c** vector projects into the plane. The asymmetric unit is half of the unit cell, divided along $u = \frac{1}{2}$.

grid of 83 × 38 × 45 for the crambin unit cell (b = 38), with grid spacing approximately 0.5 Å in every dimension. The unit cell can be separated into two asymmetric units by dividing along $u = \frac{1}{2}$. We perform all of our calculations on the asymmetric unit where $u \in [0, \frac{1}{2}]$. The grid within this asymmetric unit is 42 × 38 × 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

$$S = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

If we are given a rotational transform R in Cartesian coordinates, its symmetric transform will be $CSC^{-1}R$, where C is the 3 × 3 transform from fractional into Cartesian coordinates. For crambin,

$$C^{-1} = \begin{pmatrix} 0.024534 & 0 & 0.0003 \\ 0 & 0.054336 & 0 \\ 0 & 0 & 0.044901 \end{pmatrix},$$

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 $\theta_L = 20.0^\circ$. (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.* (1980) have described a method which generates the rotational transforms more uniformly for various space groups.

5.2.3. Crambin results. In a typical X-ray experiment, the exact disulfide-bonding residue pairs are not known 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 S 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 two. 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 2.762 Gb of storage.

As in the previous experiment, we tested *ConfMatch* with diffraction data at 2.0 Å. The density distribution is generated from 2360 structure factors with their published phases. This input is merely 11.9% of the data at 0.89 Å. The bound-preprocessing and search stages take 42 900 and 34 s 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 because of the accuracy of the upper bounds.

Fig. 15 shows the solution structure from *ConfMatch* superimposed with the published 0.89 Å structure. *ConfMatch*'s result has an RMSD of 0.588 Å 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 C 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 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 2.6 Å resolution. At 2.7 Å, however, *ConfMatch* could not 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 548 016 CPU s (6.34 CPU d) on the last iteration of IDA*, no solution was found. There were too many structures with steric clashes which had higher density than the best solution. At 2.7 Å, the exponential time search stage requires far more resources than the polynomial time bound-preprocessing stage. Finding a solution at 2.7 Å 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 a varying degree of random Gaussian noise to the perfect phases. The results from 2 Å resolution data are shown in Table 6. *ConfMatch* was able to calculate an accurate conformation with 15° phase error. However, *ConfMatch* could not find a solution with 20° phase error. After spending 521 532 CPU s (6.04 CPU d) on the last iteration of IDA*, no solution was found. Once again, we are limited by the computational resources, not by the chemical constraints.

A 20° phase error is usually not significant. After computing an electron-density map from these phases, we cannot find any obvious problems. This suggests that the space that *ConfMatch* needs to search through may grow rapidly with phase error. In the next section, we propose a different objective function for *ConfMatch* which may ameliorate this problem.

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

Table 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).

Resolution (Å)	Number of reflections	RMSD (Å)	Search stage last iteration time (s)	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	>548016	>1449835314	>1.16	>2.26

includes the grid-based conformations generated from sampling all free dihedral angles, as well as the six rigid degrees of freedom. (To ensure that *ConfMatch* covers all possible conformations on the grid, one may follow the sampling scheme in Fig. 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; Garey & Johnson, 1979), 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.

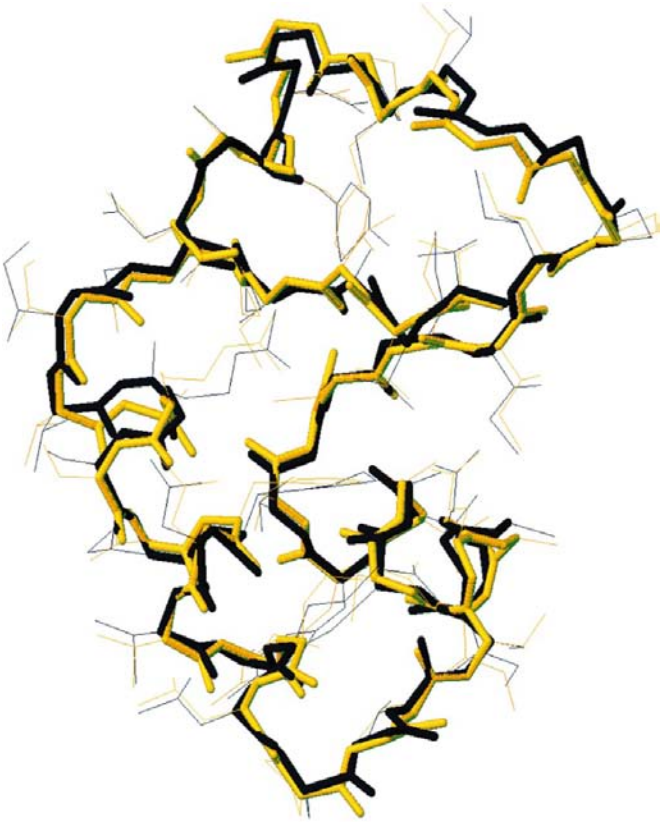

Figure 15
ConfMatch's solution structure (in black) of crambin from 2.0 Å resolution data and the published 0.89 Å structure (in yellow). The thicker portions are the backbones of the structures.

On the practical side, *ConfMatch* is able 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 four 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 side chains such as tryptophan and then gradually filling in the rest of the structure. In MAD-phased maps, the positions of Se atoms serve a similar function. This 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 frag-

Table 6
Crambin's conformation is matched to phases with various levels of error using 2 Å resolution data.

DIFF: difference between the global upper bound M and solution density (equivalent number of atoms).

Phase error (degree standard deviation)	RMSD (Å)	Search stage last iteration time (s)	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	>521532	>1512036839	>1.16	>1.67

ments will have very low values. This modification creates a large 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 large errors is to improve fundamentally the computational efficiency of *ConfMatch*. One possible way to improve the speed of *ConfMatch* is making use of prior knowledge about protein structures. This knowledge includes the allowed values and combinations of values for the main-chain and side-chain torsion angles. Currently, *ConfMatch* samples every torsion angle from $-\pi$ to π as if all values are allowed. However, the Ramachandran plot allows only certain combinations of φ and ψ values. This knowledge can be utilized by consolidating three fragments into a 'super-fragment' (Fig. 16). Since all fragments are required to be rigid, we need to add multiple instances of this 'super-fragment'. Each instance corresponds to a pair of φ and ψ values allowed by the Ramachandran plot. (The disallowed φ and ψ combinations are not generated.) Similar to the extension in §4.5, the best instance of the 'super-fragment' will be selected in the final solution. This modification is a trade-off between the bound-preprocessing stage and the search stage. The bound-preprocessing stage requires more resources because additional bounds tables need to be calculated for the superfragment instances. The search stage requires less resources because of smaller search space and more accurate bounds. (The bounds become more accurate because the conformations with the disallowed φ and ψ combinations are no longer part of the calculation.) The overall effect on *ConfMatch*'s efficiency remains to be determined.

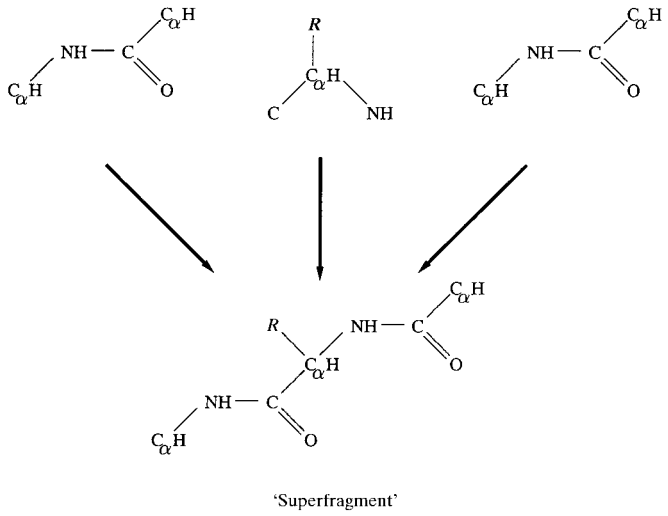

Figure 16
The constraints on φ and ψ from the Ramachandran plot can be utilized by consolidating three fragments into a 'superfragment'. Multiple instances of the superfragment are needed; each corresponds to an allowed combination of φ and ψ.

Another promising technique to improve *ConfMatch* is extracting 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 and coworkers (Jones *et al.*, 1991; Jones & Kjeldgaard, 1996) developed the real-space correlation coefficient (RSCC) which measures the correlation between the expected and the observed density distribution within a region. For *ConfMatch*, every fragment has a unique density distribution. An appropriate envelope can be defined for each fragment to calculate its RSCC. A large RSCC will imply that the fragment is likely to be at the center of the envelope and *vice versa*. *ConfMatch* can be modified to maximize the sum of RSCC over all fragments. If the RSCC 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 larger and more difficult problems.

APPENDIX *A*
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 $f_{\rm lim}$. $f_{\rm lim}$ is raised to $d + \varepsilon$ after finding the solution. During a single depth-first search, $f_{\rm lim}$ is monotonically increasing. All entries in the bounds table are updated by (4) and (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 $f_{\rm lim}$ and bounds, (3) will be satisfied less often. The new search would explore

only a subspace of the earlier one. Since we have raised $f_{\rm lim}$ 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, $E^l_{n,\mathbf{j}} = E_{n,/bfj}$ for all n, $\mathbf{j}$ because the bounds table was built by (1) and (2). The invariant holds.

Inductive case. After searching n's sub-fragment tree, the bounds table is updated by (4) and (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 $t_{\rm init}$, $f^l_{n,\mathbf{j}}(t_{\rm init}) = E^l_{n,\mathbf{j}}$. 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 in-bond 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 $E_{n,/bfj}$ is an upper bound of all possible structures, the invariant holds.

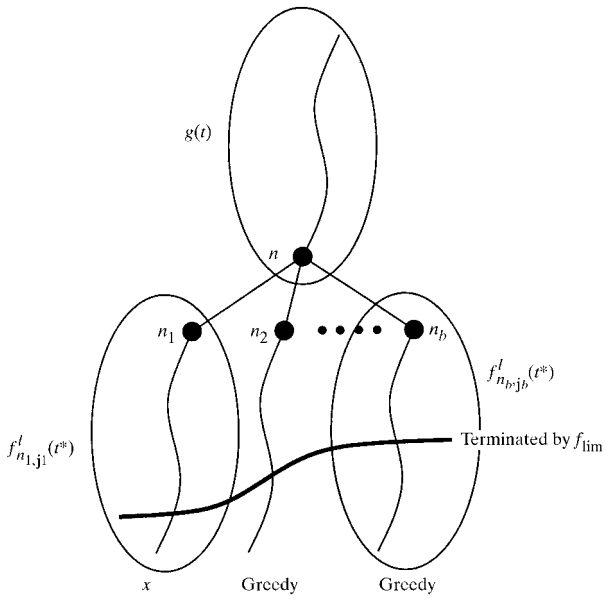

Figure 17
An impossible dead-end state whose f value is greater than $f_{\rm lim}$ and that does not have any violations.

Inductive case. Assuming all $E_{n,/bfj}$ 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 (4) and (5) are valid upper bounds.

(i) By the inductive assumption, the original value $E^l_{n_1,\mathbf{j}_1}$ is an upper bound.

(ii) By the inductive assumption and the construction of (2), $\max_{i=1}^{s} S^i_{n_1,\mathbf{j}_1}$ is an upper bound.

(iii) We now prove by contradiction that $f_{\rm lim} - g(t) - \sum_{i=2}^{b} E^l_{n_i,\mathbf{j}_i}$ 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 $g(x) > f_{\rm lim} - g(t) - \sum_{i=2}^{b} E^l_{n_i,\mathbf{j}_i}$. As we have explained earlier, no solution would be found if we perform a new search from t with $f_{\rm lim}$ (without changing $f_{\rm lim}$ or any E^1). 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-conformationalsearch 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 $n_2, \ldots, n_b$ (Fig. 17). t^* has no violations because x is a valid structure. It must be terminated by condition (iii). Thus, $f(t^*) < f_{\rm lim}$. 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, $f^l_{n_1,\mathbf{j}_i}(t^*) \geq g(x) > f_{\rm lim} - g(t) - \sum_{i=2}^{b} E^l_{n_i,\mathbf{j}_i}$. Therefore,

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

which means

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

We have reached a contradiction.

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

The author wishes to thank Lisa Tucker Kellogg, Tomás Lozano-Pérez, Bruce Tidor, Paul Viola and William M. Wells III for many helpful discussions. The author was partially supported by a Merck/MIT graduate fellowship.

References

Aho, A., Hopcroft, J. E. & Ullman, J. D. (1982). *Data Structures and Algorithms.* Reading, MA: Addison-Wesley.
Arora, S. K. (1983). *Mol. Pharmacol.* **23**, 133–140.
Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F. Jr, Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *J. Mol. Biol.* **112**, 535–542.
Craig, J. J. (1986). *Introduction to Robotics*, ch. 2. Reading, MA: Addison–Wesley.

Crippen, G. M. & Havel, T. F. (1988). *Distance Geometry and Molecular Conformation.* Chemometrics Series. New York: John Wiley.
Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability. A Guide to the Theory of NP-completeness.* New York: W. H. Freeman.
Greer, J. (1974). *J. Mol. Biol.* **82**, 279–301.
Jones, T. A. & Kjeldgaard, M. (1996). *O – The Manual*, 5th ed. University of Uppsala, Sweden.
Jones, T. A. & Thirup, S. (1986). *EMBO J.* **5**(4), 819–822.
Jones, T. A., Zou, J., Cowan, S. W. & Kjeldgaard, M. (1991). *Acta Cryst.* A**47**, 110–119.
Kleywegt, G. J. & Jones, T. A. (1997). *Acta Cryst.* D**53**, 179–185.
Lattman, E. E. (1972). *Acta Cryst.* B**28**, 1065–1068.
Lee, L.-P. & Tidor, B. (1997). *J. Chem. Phys.* **106**, 8681–8690.
Leherte, L., Fortier, S., Glasgow, J. & Allen, F. H. (1994). *Acta Cryst.* D**50**, 155–166.
Lipton, M. & Still, W. C. (1998). *J. Comput. Chem.* **9**(4), 343–355.
McRee, D. E. (1992). *Practical Protein Crystallography*. New York: Academic Press.
Patrick, B. G., Almulla, M. & Newborn, M. M. (1992). *Ann. Math. Artificial Intell.* **5**, 265–278.
Perrakis, A., Morris, R. & Lamzin, V. S. (1999). *Nature Struct. Biol.* **6**(5), 458–463.
Prive, G. G., Anderson, D. H., Wesson, L., Cascio, D. & Eisenberg, D. (1999). *Protein Sci.* **8**(7), 1400–1409.
Rao, S. N., Jih, J.-H. & Hartsuck, J. A. (1980). *Acta Cryst.* A**36**, 878–884.
Russell, S. & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*, ch. 4. Englewood Cliffs, USA: Prentice Hall.
Teeter, M. M., Roe, S. M. & Heo, N. H. (1993). *J. Mol. Biol.* **230**, 292.
Terry, A. (1988). *Blackboard Systems*, ch. 7. Reading, MA: Addison–Wesley.
Zou, J. & Jones, T. A. (1996). *Acta Cryst.* D**52**, 833–841.