AmbiPack: A Systematic Algorithm for Packing of Macromolecular Structures With Ambiguous Distance Constraints

Cheuk-san Edward Wang,[1] **Tomás Lozano-Pérez,**[1] **and Bruce Tidor**[2]*
[1]*Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts*
[2]*Department of Chemistry, Massachusetts Institute of Technology, Cambridge, Massachusetts*

ABSTRACT **The determination of structures of multimers presents interesting new challenges. The structure(s) of the individual monomers must be found and the transformations to produce the packing interfaces must be described. A substantial difficulty results from ambiguities in assigning intermolecular distance measurements (from nuclear magnetic resonance, for example) to particular intermolecular interfaces in the structure. Here we present a rapid and efficient method to solve the packing and the assignment problems simultaneously given rigid monomer structures and (potentially ambiguous) intermolecular distance measurements. A promising application of this algorithm is to couple it with a monomer searching protocol such that each monomer structure consistent with *intramolecular* constraints can be subsequently input to the current algorithm to check whether it is consistent with (potentially ambiguous) *intermolecular* constraints. The algorithm AmbiPack uses a hierarchical division of the search space and the branch-and-bound algorithm to eliminate infeasible regions of the space. Local search methods are then focused on the remaining space. The algorithm generally runs faster as more constraints are included because more regions of the search space can be eliminated. This is not the case for other methods, for which additional constraints increase the complexity of the search space. The algorithm presented is guaranteed to find all solutions to a predetermined resolution. This resolution can be chosen arbitrarily to produce outputs at various level of detail. Illustrative applications are presented for the P22 tailspike protein (a trimer) and portions of β-amyloid (an ordered aggregate). Proteins 32:26–42, 1998.**
© 1998 Wiley-Liss, Inc.

Key words: intermolecular restraints; solid-state NMR; symmetric multimer; branch and bound; amyloid

INTRODUCTION

The determination of atomic-level structures is of growing importance in modern chemistry and biology. Advances in biophysical techniques result in data on more ambitious structures being generated at an ever increasing rate. However, finding structures that are consistent with these data presents a number of formidable computational problems. This paper gives an efficient, systematic algorithm for one such problem: finding packings of rigid predetermined subunit structures that are consistent with ambiguous intermolecular distance measurements from nuclear magnetic resonance (NMR) experiments.

Whereas X-ray crystallography essentially provides atomic-level information in *absolute* coordinates, NMR spectroscopy typically provides *relative* distance and orientation information through chemical shifts, coupling constants, and especially distances estimated from magnetization transfer experiments. In NMR spectroscopy, the identity of an atomic nucleus is indexed by its chemical shift (in 2D experiments) and also that of its neighbors (in higher dimensional experiments). Thus, two atoms that occupy exactly the same environment (e.g., symmetry-mates in a symmetric dimer) cannot generally be distinguished, and distances measured to them can be ambiguous. For instance, in the symmetric dimer case, intra- and intermolecular distances are ambiguous. This type of ambiguity can generally be removed through isotopic labeling schemes. However, for higher order multimers, in which different types of intermolecular relationships exist, each intermolecular distance remains ambiguous. Furthermore, in solid-state NMR experiments,[1] one can obtain unambiguous intramolecular distances but generally only ambiguous intermolecular distances. This kind of problem is evident with symmetric coiled coils,[2] the trimeric P22 tailspike protein,[3] and the fibrils formed from fragments of the Alzheimer precursor protein.[1]

Grant sponsor: National Institutes of Health; Grant numbers: NS33366 and AG14366.

*Correspondence to: Bruce Tidor, Department of Chemistry, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: tidor@mit.edu

Received 23 September 1997; Accepted 5 March 1998

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

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

A different approach is to design a potential function that has the effect of performing a logical "OR" over the possible solutions for the ambiguous constraints. For example, this function can be a sum of terms reflecting a penalty for unsatisfied distance measurements. Each term can contribute zero when the corresponding distance is satisfied in any way consistent with its labeling ambiguity. The penalty function may increase monotonically with the magnitude of the distance violation so that global optimization techniques, such as simulated annealing, may be utilized to search for solutions. If multiple packed structures exist that are consistent with the measurements, there would be many minima with zero penalty. Nilges' *dynamic assignment strategy*[4,5] uses a smooth function with these properties for ambiguous inter- and intramolecular distances. Dynamic assignment has the significant advantage of not assuming a rigid monomer. Instead, the monomer is assumed to be flexible and restrained by intramolecular distances. O'Donoghue et al.[6] successfully applied this technique to the leucine zipper homodimers, where the monomer structure is known. Unfortunately, this approach must contend with the multiple local minima problem; there are many placements of the structures that satisfy only a subset of the distances but such that all displacements cause an increase in the potential. As the number of ambiguous distances increases, the minimization takes longer to find valid solutions, due to increasing ruggedness of the potential landscape. Furthermore, since this approach is a randomized one, it is not guaranteed to generate all packings satisfying the constraints. Likewise, if no structure can possibly match all distances, this method will not be able to prove that conclusively.

Yet another approach is to sample rigid transformations systematically,[7,8] apply them to the subunit, and then test whether the resulting structures match all distances. Since a rigid transformation has six parameters (three translations and three rotations), one needs to test n^6 transforms where n is the number of samples for each transformation parameter. This will take a great deal of computer time even for a moderate size n, such as 30, since $30^6 = 729{,}000{,}000$). Furthermore, this approach may miss solutions that are "between" the sampled transformations. So, to have a fair degree of confidence that no solutions have been missed requires very fine sampling, that is, a large value of n (generally much greater than 30).

We have developed a new algorithm, AmbiPack, that generates packed structures from ambiguous (and unambiguous) intermolecular distances. AmbiPack is both exhaustive and efficient. It can find all possible packings, at a specified resolution, that can satisfy all the distance constraints. This resolution can be chosen by the user to produce packings at any level of detail. It gives a null answer *if and only if* there is no solution to the constraints. In our implementation, AmbiPack takes minutes to run on a problem with more than 40 ambiguous constraints. (All runs were on a Sun Ultra 1 workstation.) Its running time does not depend significantly on the size of the subunits. Furthermore, while most other techniques run slower when more constraints are added, AmbiPack generally runs *faster* with more constraints because this allows earlier pruning of a greater number of solutions and requires detailed exploration of a smaller number of solutions. Therefore, it is quite practical to apply AmbiPack to a family of NMR-derived subunit structures to obtain a corresponding family of packed structures. Moreover, it can be used in tandem with a subunit generating procedure (which satisfies *intrasubunit* distances) to filter out those subunit models incompatible with the set of *intersubunit* distances.

PROBLEM DEFINITION

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

Two Structures

The inputs to the AmbiPack algorithm are:

1. Two rigid structures (S and S') that are to be packed. Without loss of generality, we assume that S' is fixed in the input configuration and S has an unknown rigid transformation relative to S'. S is the same as S' for identical structures, which is frequently the case.
2. A set of constraints on the intermolecular distances. These constraints specify the allowed ranges of distances between atoms, e.g., $3\ \text{Å} < PQ' < 6\ \text{Å}$ where P and Q' are atoms on S and S', respectively. The constraints can be specified *ambiguously,* i.e., only one of several bounds needs to be satisfied. Suppose P and Q are atoms on S while P' and Q' are correspondingly atoms on S'. One ambiguous constraint may be

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

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

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

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

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

More Than Two Structures

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

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

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

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

The more complex case is when the positive distance constraints are not all satisfied between every pair of structures. Figure 2 shows a structure of a C-terminal peptide (β34–42) of the β amyloid protein (β1–42).[1] This infinitely repeating structure forms an ordered aggregate. There are two types of interfaces in this structure. Solid-state ^{13}C NMR experiments have produced 8 positive and 12 negative constraints. Either interface satisfies all negative constraints but only a subset of the positive ones. Together the interfaces satisfy all positive constraints. A direct approach to this type of problem is to enumerate subsets of the constraints that may hold between different pairs of structures. AmbiPack can be used to find solutions for each of these subsets of constraints. A valid multimer can be constructed from combinations of output transformations, applied singly or successively, such that each constraint is satisfied at least once in the multimer. This is the strategy illustrated in the section C-Terminal Peptide of β-Amyloid Protein, below. This strategy is only feasible when the number of ambiguous constraints is relatively small since the number of

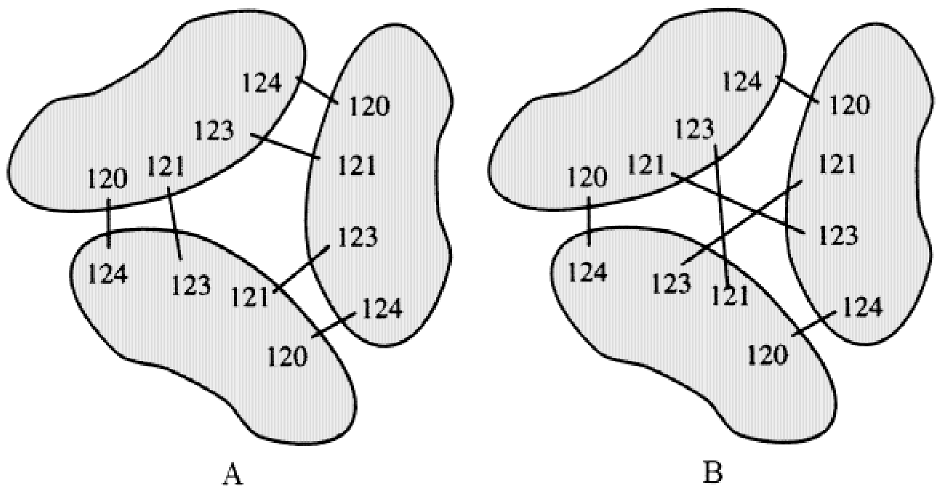

Fig. 1. Two ambiguous intermolecular distances can have two different interpretations. The P22 tailspike protein is shown schematically with two different interpretations (**A**,**B**) for the proximity pairs 120–124 and 121–123.

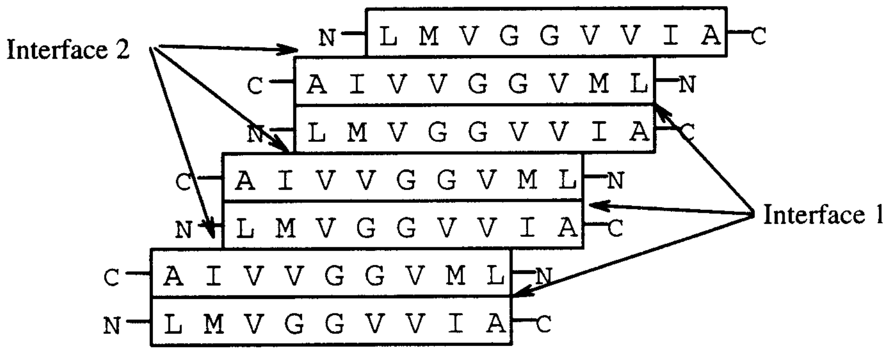

Fig. 2. Structure of β-amyloid fibril proposed by Lansbury et al.[1]

constraint subsets also grows exponentially with the number of ambiguous constraints. A more efficient variant of this strategy that exploits the details of the AmbiPack algorithm is discussed in the Algorithm Extensions section, below.

THE AmbiPack ALGORITHM

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

Algorithm Overview

AmbiPack is based on two key observations:

1. Suppose there is some constraint of the form $PQ' < x$ Å, where P and Q' are atoms on S and S', respectively. This constraint specifies the approximate location of P. Specifically, it describes a sphere of radius x Å around Q' in which P must be found.
2. If we fix the positions of three non-collinear atoms on S, we have specified a unique rigid transformation.

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

1. Select three (unambiguous) constraints ($P_iQ'_i < x_i$ Å, $i = 1, 2, 3$) from the input set.
2. For each P_i, uniformly sample its possible positions inside the sphere with radius x_i Å centered on Q'_i.
3. Calculate rigid transformations based on the positions of P_is. Test whether these transformations satisfy all the input constraints.

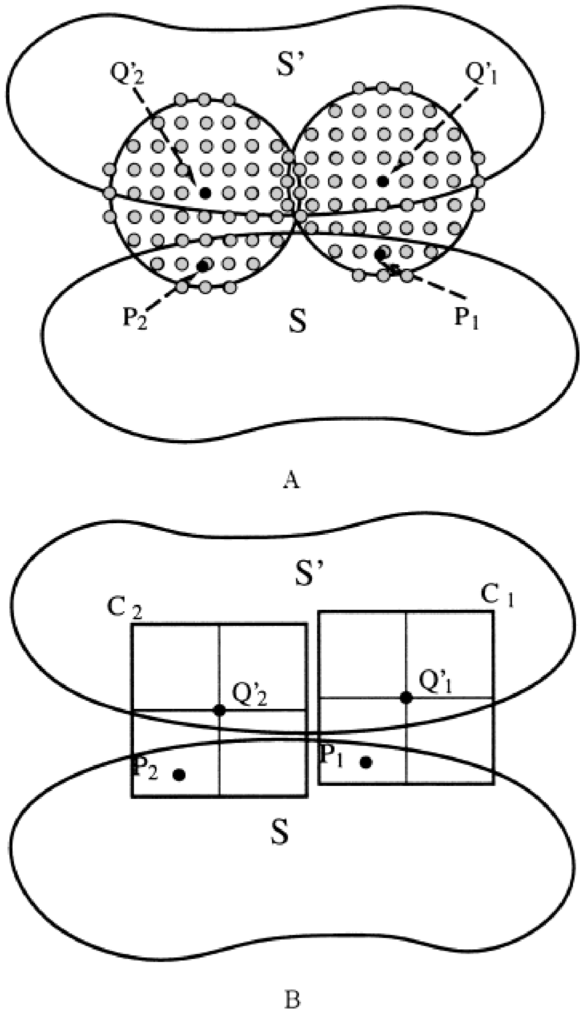

Fig. 3. Two approaches to generating transforms (illustrated here in two dimensions, where two points are sufficient to place a structure): (**A**) matching points P_i from S to sampled points in spheres centered on Q_i', or (**B**) placing points P_i from S somewhere within cubes contained in spheres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

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

Note, however, that this approach is *not* guaranteed to find a legal packing whenever one exists. In particular, it misses solutions that would require placing any of the P_i away from the sampled points. Of course, by sampling very finely we can reduce the chances of such a failure, but this remedy would exacerbate the two other problems with this approach as it stands. One problem is that the method needs to test m^3 transformations where m is the number of sampled points in each of the three spheres. Typically we would sample hundreds of points in each sphere, and thus millions of transformations are to be generated and tested. The other, related, problem is that the finer the sampling, the greater the number of transformations, many nearly identical, that will be produced, since the constraints seldom define a single solution exactly. To alleviate these latter problems, we want a relatively coarse sampling.

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

The key to the efficiency of the algorithm, obtained without sacrificing exhaustiveness, is the ability to disprove that a solution exists when the three P_i are placed anywhere within the three given cubes, C_i. Since the P_is are restricted to the cubes, the possible locations of other S atoms are also limited. If one can conservatively bound the locations of other atoms, one can use the input constraints to disprove that a solution can exist. AmbiPack uses *error spheres* to perform this bounding. For each atom on S, its error sphere includes all of its possible positions given that the P_is lie in C_is (Fig. 4). The details of the error sphere computation are given in the sections Centers of Error Spheres; and Radii of Error Spheres, below.

Up to this point we have not dealt with ambiguous constraints. However, we only need to modify the algorithm slightly to deal with them. Note that once we have a candidate transformation, checking whether ambiguous constraints are satisfied is no more difficult than checking unambiguous constraints; it simply requires dealing with constraints including ORs as well as ANDs. So, the only potential difficulty is if we cannot select an initial set of three unambiguous constraints in Step 1 of the algorithm. If the constraints are ambiguous, we cannot tell whether the atoms referred to in the

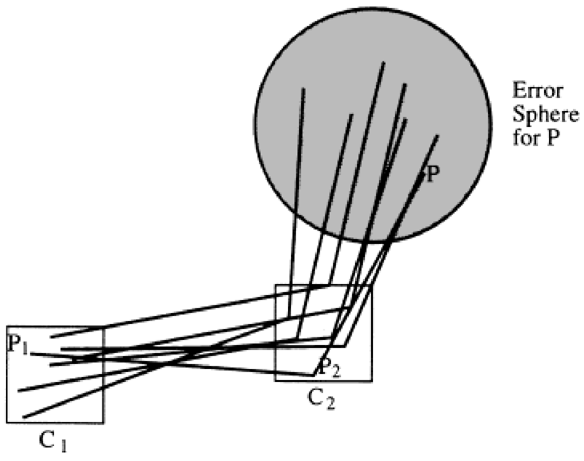

Fig. 4. The error spheres for points in S when the P_i are constrained to be somewhere within cubes contained in spheres centered on Q'_i. Illustrated here in two dimensions.

constraints are drawn from S or S'. In that case, however, we can enumerate the possible interpretations of the ambiguous constraints and find the solutions for each one. Assuming that all the distance constraints hold between the given structures and since we are dealing with at most three ambiguous measurements, this generally involves a small number of iterations of the algorithm.

Algorithm Details

AmbiPack is an example of a branch-and-bound tree-search algorithm.[9] During the search, it prunes away branches that are ruled out by the bound function. Figure 5 illustrates the algorithm. Initially, AmbiPack selects three constraints, $P_iQ'_i < x_i$ Å, $i = 1, 2, 3$. Each node in the search tree corresponds to three cubes in space—C_1, C_2, and C_3; the position of P_i is limited to be inside C_i. At the root of the tree, each C_i is centered at Q'_i and has length $2x_i$ in each dimension. Thus, all possible positions of P_is satisfying the constraints are covered. At every internal node, each C_i is subdivided into eight cubes with half the length on each side. Each child has three cubes ⅛ the volume of its parent. Each parent has 512 children because there are $8^3 = 512$ combinations of the smaller cubes. At each level further down the search tree, the positions of P_is are specified at progressively finer resolution. If one calculates transformations from all nodes at a certain level of the tree, one systematically samples all possible solutions to the packing problem at that level's resolution. The method of computing a transformation for a node is described below.

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

The first technique is to exploit the known distances between the P_i, since the monomers are predetermined structures. Between any pair of atoms in S, the distance is fixed because S is a rigid structure. Suppose C_1, C_2, and C_3 are the cubes corresponding to a tree node. Let max(C_1, C_2) and min(C_1, C_2) be the maximum and minimum separation, respectively, between any point in C_1 and any point in C_2. A necessary condition for P_1 to be in C_1 and P_2 to be in C_2 is

$$\min(C_1, C_2) \leq P_1P_2 \leq \max(C_1, C_2).$$

Similarly, for the other two pairs of atoms, we require $\min(C_2, C_3) \leq P_2P_3 \leq \max(C_2, C_3)$ and $\min(C_1, C_3) \leq P_1P_3 \leq \max(C_1, C_3)$. If any of the three conditions are violated, the node can be rejected; since the P_is cannot be simultaneously placed in these cubes.

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

The key remaining problem is efficiently finding the centers and radii of error spheres for the specified P_is and C_is.

Centers of error spheres

We want to make the error spheres as small as possible, since this will give us the tightest constraints and best pruning. We can think of the center

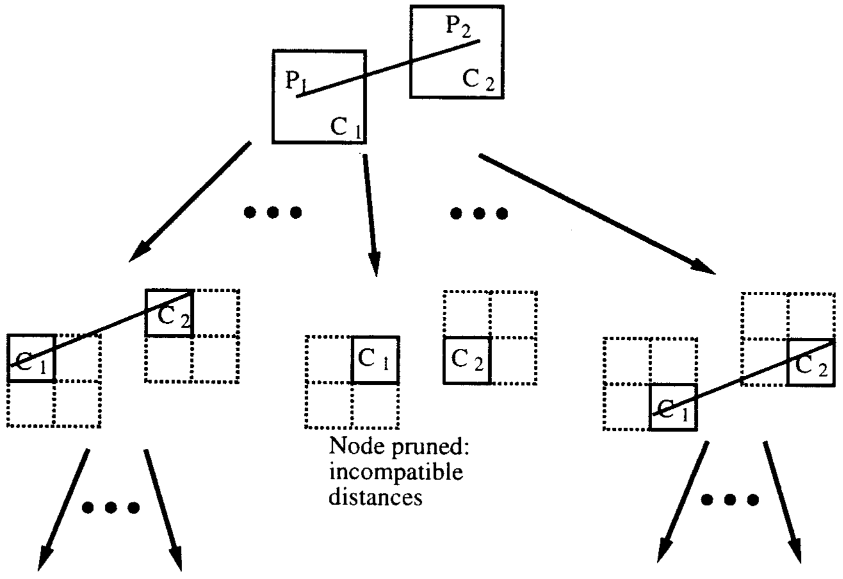

Fig. 5. The AmbiPack algorithm explores a tree of assignments of the three P_i to three cubes C_i. Illustrated here in two dimensions.

of the error sphere as defined by some "nominal" alignment of the P_i to points with the C_i. The points within the error sphere are swept out as the P_i are displaced to reach every point within the C_i. The magnitude of the displacement from the error sphere center depends on the magnitude of the displacement from the "nominal" alignment. This suggests that we can keep the error sphere small by choosing a "nominal" alignment that keeps the displacement of the P_i needed to reach every point in C_i as small as possible, that is, we want the "nominal" alignment to place the P_i close to the centers of the C_i.

We find the centers of the error spheres by calculating a transformation, T, that places the P_is as close as possible to the centers of the C_is. For every atom P in S, $T(P)$ is taken as the center of its error sphere. There are many well-known iterative algorithms for computing transformations that minimize the sum of distances squared between two sets of points, e.g., Ferro and Hermans.[10] However, in our case, since we are dealing with only three pairs of points, we can use a more efficient analytic solution.

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

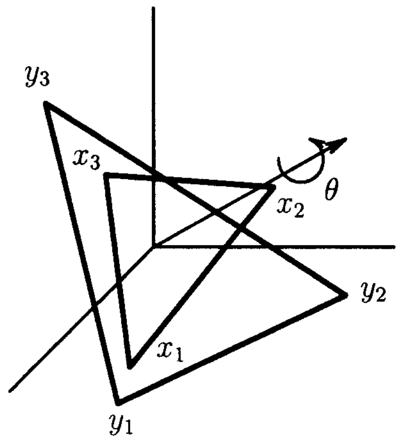

Fig. 6. When two triangles are coplanar and their centroids are coincident, there is only one angle of rotation, θ, to determine.

with respect to θ. The condition for this derivative being zero is:

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

With θ found, the required transformation that matches P_is to the centers of C_is is

$$T = T_4 R_3 R_2 T_1$$

where

- T_1 translates the centroid of P_is to the origin;
- R_2 rotates the P_is about the origin to a plane parallel to that of the centers of C_is;

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

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

Fig. 7. The AmbiPack algorithm. *Search_depth* is the limit on the depth of the tree.

- R_3 rotates the P_is about their centroid by the optimal θ;
- T_4 translates the P_is centroid from the origin to the centroid of the C_is centers.

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

Radii of error spheres

The radii of error spheres are harder to find than the centers. For each sphere, its radius must be larger than the maximum displacement of an atom from the error sphere center. The sizes of the spheres depend not only on the C_is, but also the locations of atoms on S relative to the P_i. The range of motion of an atom increases with the dimension of the C_is, as well as its separation from the P_is. For efficiency, AmbiPack calculates a set of radii for atoms of S depending on the sizes of the C_is, but not on the C_is exact locations. Thus all nodes at the same level of the search tree share the same set of error sphere radii; it is not necessary to recalculate them at every node.

Suppose the largest C_i has length d on each side. A sphere of radius $\sqrt{3}d$ centered at any point in the cube will contain it, regardless of the cube's orientation. Therefore we can restate the problem of finding error sphere radii as: Given S and the P_is where each P_i may have a maximum displacement of $\sqrt{3}d$, find the maximum displacements of all other atoms in S. These displacements will be used as the radii of the error spheres. There are two possible approaches to this problem—analytical and numerical. One can calculate an analytical upper bound of the displacement of each atom, but it is quite difficult to derive a tight bound. A loose bound will result in excessively large error spheres and ineffective pruning. We choose a simple randomized numerical technique to find the maximum displacements. A large number (1,000) of random transformations, which displace the P_is by $\sqrt{3}d$ or less, are generated. These transformations are applied to all atoms on S. For each atom, we simply record its maximum displacement among all transformations. Empirically, this technique converges very quickly on reliable bounds. The performance data in the next section also show that the resulting radii are very effective in pruning the search tree.

Algorithm Summary

Figure 7 summarizes the basic AmbiPack algorithm. Note that for a problem where some solutions

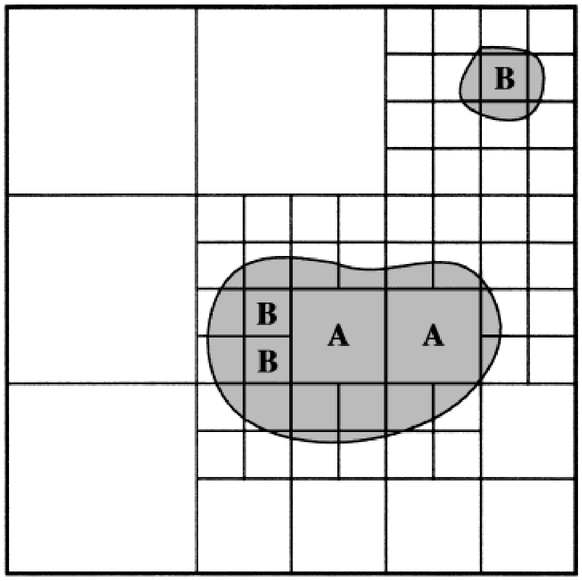

Fig. 8. Solutions will generally be clustered into regions (shown shaded) in a high-dimensional space, characterized by the placements of the P_i. Each leaf of the search tree maps into some rectangular region in this space whose size is determined by the resolution. AmbiPack samples one point (or at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

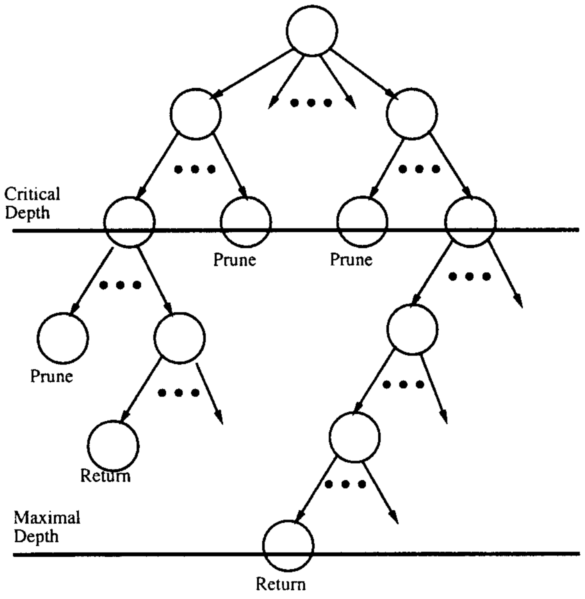

Fig. 9. Illustration of the search using a critical depth and maximal depth.

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

The simplest strategy for using the AmbiPack algorithm is:

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

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

However, if stronger guarantees are required, then a more sophisticated variant of this strategy can be followed. As we discussed in the section Algorithm Overview, above, there are two meaningful, and conflicting, resolution limits in this type of problem. One arises from the goal of finding solutions if they exist. There is a minimum resolution determined by the accuracy of the measurements below which it makes no sense to continue partitioning space in search of a solution. Therefore, there is a *maximal depth* in the tree beyond which we never want to proceed. However, we do not want to expand all the leaves of the tree to this maximal depth. The other search limit stems from a desire to avoid generating too many nearly identical solutions. This defines a *critical depth* in the tree beyond which we want to return at most a single transform if one exists, rather than returning all the transforms corresponding to leaves. Therefore, we can modify the AmbiPack algorithm so that, below the critical depth and above the maximal depth, it attempts to find a solution (using minimization) for every node that is not pruned (Fig. 9). If a solution is found, it is stored, and search resumes with the next subtree at the critical depth. In this way, at most one solution is stored per subtree at the critical resolution, but the subtree is searched to the maximal resolution.

Algorithm Extensions

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

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

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

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

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

TABLE I. Results of AmbiPack Running on the P22 Tailspike Trimer With Different Values of the Maximal Resolution (res.) for the Search

Critical res.	Maximal res.	No. of solutions	Avg. RMSD	Time (sec)
2.0	2.0	47	0.6078	279
2.0	0.5	743	1.073	3,801
2.0	0.25	827	1.033	11,497

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

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

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

RESULTS AND DISCUSSION

We have carried out two detailed studies using the AmbiPack algorithm to explore its performance and

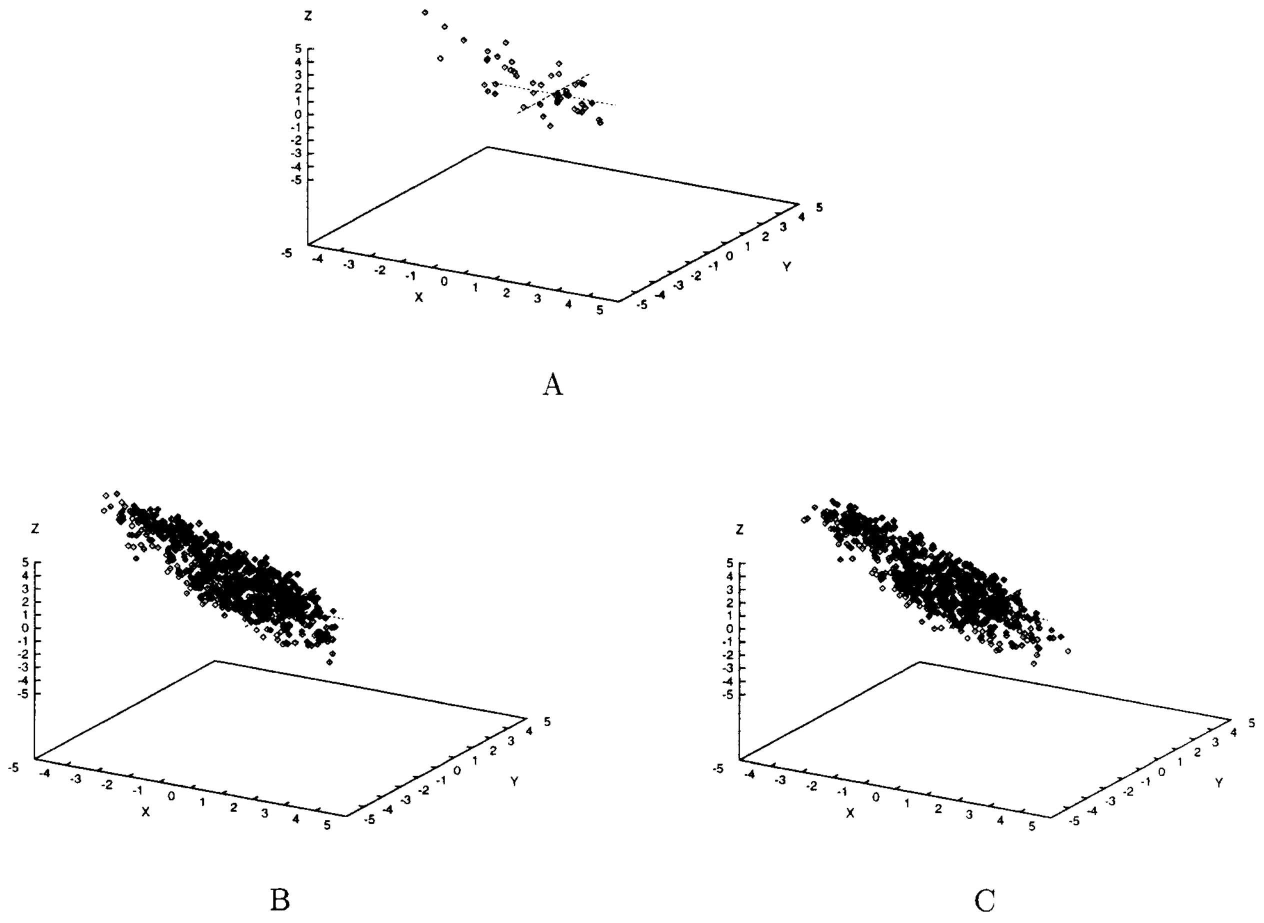

Fig. 10. The translation components of solutions from AmbiPack running on the P22 tailspike trimer, ignoring steric clashes, with critical resolution of 2.0 Å and maximal resolution of (**A**) 2.0 Å, (**B**) 0.5 Å, and (**C**) 0.25 Å. The solution from the crystal structure, (0,0,0), is marked by a cross.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

illustrate its range of applicability. The first study involves a large protein (the P22 tailspike protein—a homotrimer involving 544 residues), a large number (43) of (simulated) ambiguous measurements, and a single type of interface between three subunits. The second study involves a nine-residue peptide from β-amyloid, a small number (eight) of ambiguous measurements, and two different interfaces between an indefinitely repeating group of subunits.

In all the tests discussed below, the AmbiPack algorithm is implemented in Common Lisp,[11] running on a Sun Ultra 1 workstation. All constraints used involved carbon-carbon distances because they are commonly measured in solid-state NMR experiments, rather than distances to hydrogen, which are currently more common in solution NMR spectroscopy.

P22 Tailspike Protein

The first test of the AmbiPack algorithm is the P22 tailspike protein[3] (PDB[12] code 1TSP). In its crystal structure, the positions of 544 residues are determined. The protein forms a symmetric homotrimer. Each subunit of the homotrimer contains a large parallel β-helix. We use AmbiPack to find the relative orientation of two subunits; the third subunit can be placed by applying the solution transformation twice, as discussed in the section Problem Definition, above.

First, we measured the intermolecular distances between C_α carbons at the interface of the subunits. There were 43 C_α–C_α distances less than 5.5 Å (a typical upper bound for distance measurements in some NMR experiments), giving 43 positive constraints. To be conservative, we specified each constraint with an upper bound of 6 Å. For example, one of the constraints was $C^{120}_\alpha C'^{124}_\alpha < 6.0$ Å OR $C^{124}_\alpha C'^{120}_\alpha < 6.0$ Å. The 43 ambiguous constraints and the two identical subunit structures were given to the algorithm. The constraints have a total of 2^{42} possible interpretations. Our program solved this problem in 279 seconds to a maximal resolution where the C_is are 2 Å on each side. Most of the computer time was spent in the recursive search procedure.

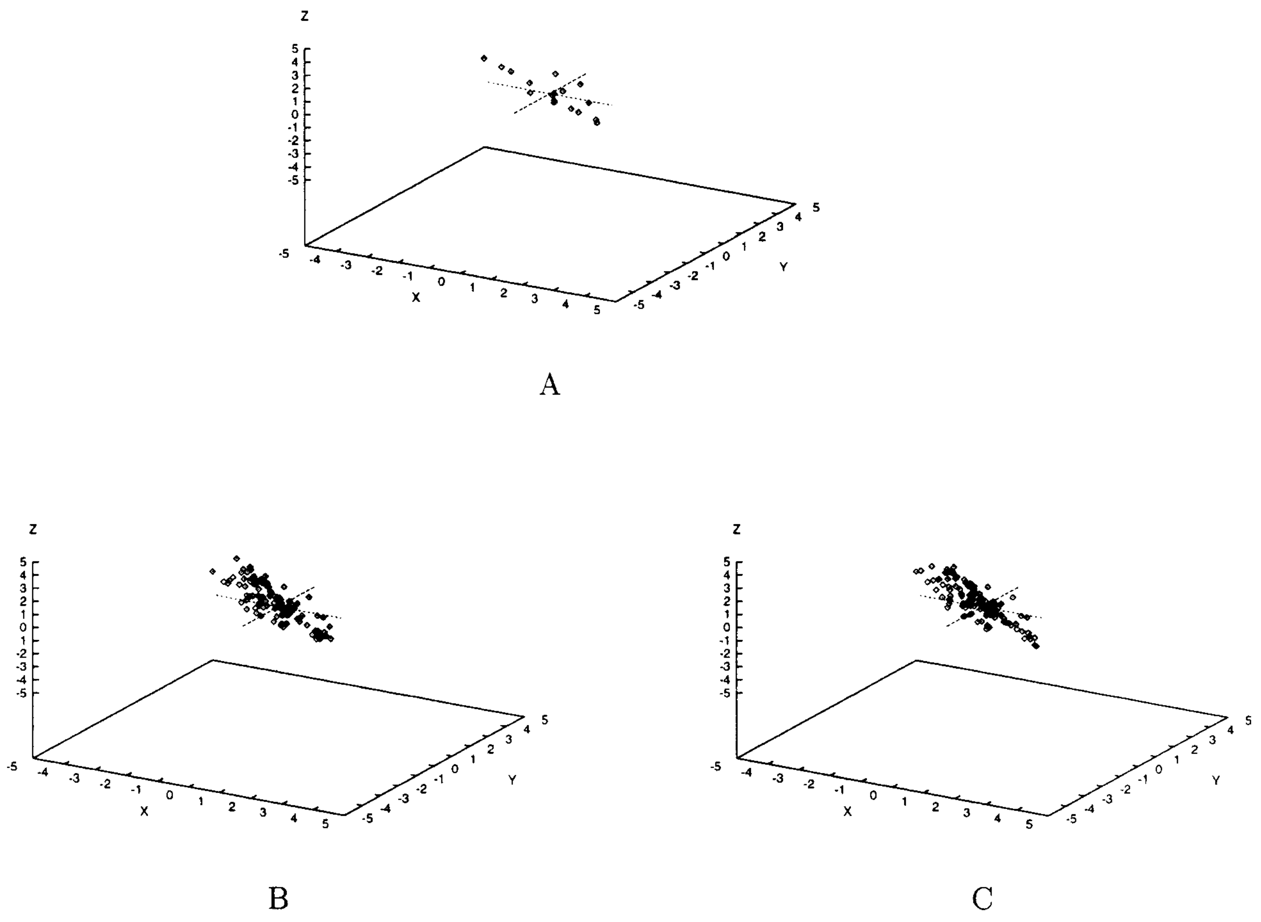

Fig. 11. The translation components of solutions from AmbiPack running on the P22 tailspike trimer, with five or fewer severe steric clashes. The critical resolution is 2.0 Å and the maximal resolutions are (**A**) 2.0 Å, (**B**) 0.5 Å, and (**C**) 0.25 Å. The solution from the crystal structure, (0,0,0),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

When the 47 solution transformations were applied to a subunit, the results had an average root mean square deviation (RMSD) of 0.6078 Å from the X-ray structure. The worst solution had an RMSD of 2.853 Å. This error is much smaller than the ranges of input constraints. The four search trees (arising from exploring the ambiguous assignments of the first three constraints chosen, given identical subunits) had a total size of 57,425 nodes. The effective branching factor is 24.3 instead of the theoretical worst case of 512. The branching factor becomes smaller as we search deeper because the pruning techniques become more powerful.

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

These solutions are obtained ignoring steric clashes. Following the suggestion of an anonymous reviewer, we filtered these structures by steric constraints. Figure 11 shows solutions that contain five or fewer severe steric clashes. We define a severe steric clash as having two atoms' van der Waal's spheres overlapping by 1 Å or more. These selected solutions are confined to a small region around the packing of the crystal structure. This shows that steric constraints are effective as a filter of packing solutions. The user would have the choice of provisionally accepting solutions that violate steric and using refinement methods to relax the structures while satisfying all constraints, or, alternatively, filtering out such solutions and requiring the monomer-generating procedure to provide more accurate structural models. Presumably data-rich problems

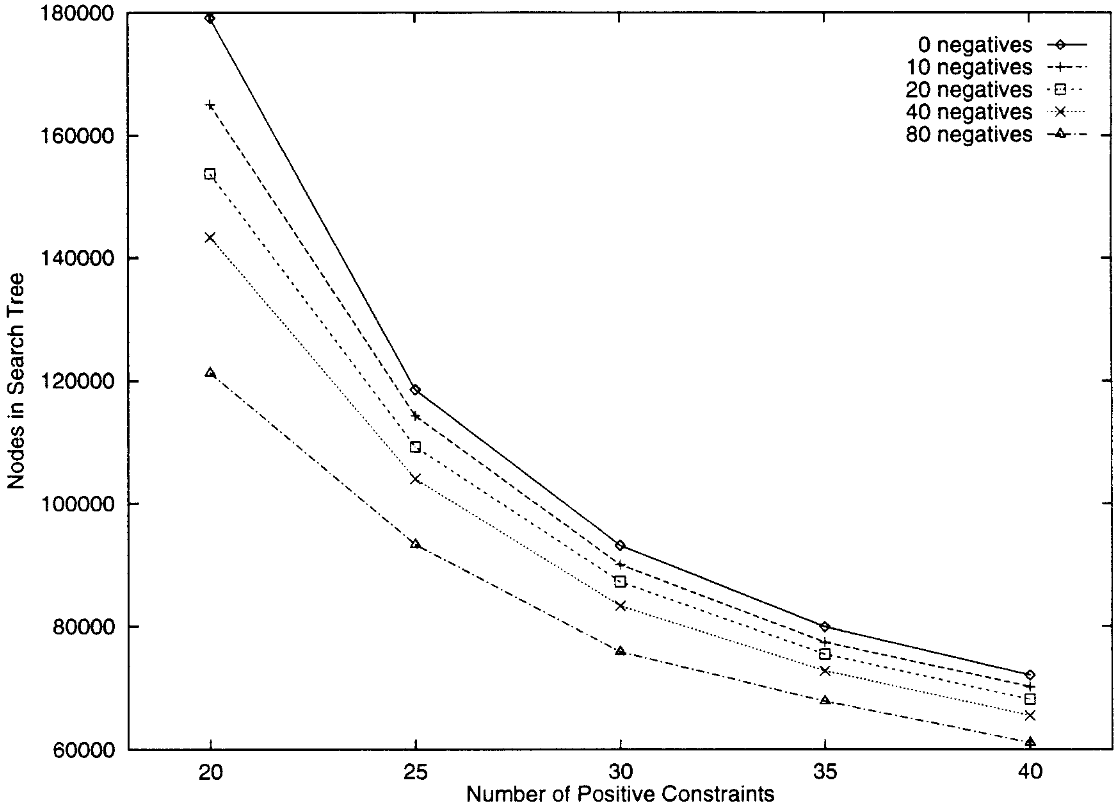

Fig. 12. Computational resources used by various combinations of positive and "near-miss" negative constraints.

would be amenable to the latter solution and data-poor problems more efficiently solved with the former.

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

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

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

To explore the potential impact of well-chosen negative results, in the later runs, we used only "near-miss" negative constraints from C_α pairs whose actual distances are just a little above the lower bound. In the P22 tailspike protein, we found 589 "near-misses" from measuring the crystal structure with an upper bound of 10 Å. These constraints did affect the computer time and solution accuracy. The

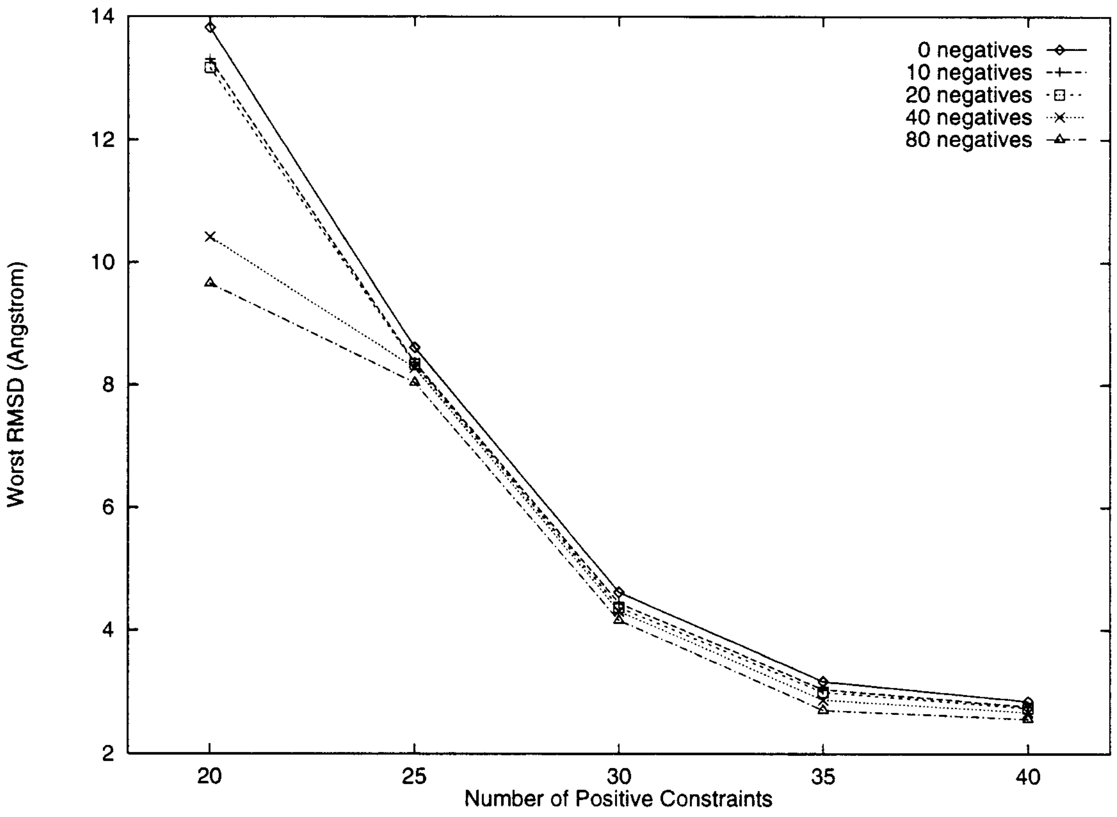

Fig. 13. Quality of structures produced by various combinations of positive and "near-miss" negative constraints. The RMSD is measured by applying the computed transforms to one of the P22 tailspike monomers (from the X-ray structure) and measuring the displacement from this monomer to the nearby monomer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

results of runs with "near-misses" are also shown in Figures 12 and 13. Computational efficiency and solution quality improved as more "near-miss" negative constraints were added, although their effect is not as significant as the same number of positive constraints. In these simulations, positive constraints contain ambiguous information, but they are more valuable for structure determination than the unambiguous negative constraints. These results also suggest that experiments should be designed to obtain negative data close to the boundary of detection, thus maximizing information on the structure. For example, if a small set of constraints is known, we can use the triangle inequality to establish upper bounds on other distances.[13] Further experiments can be directed toward measurements with small upper bounds. It should be noted that in certain circumstances, strategically chosen negative constraints may be especially useful; the results here suggest that randomly selected negative constraints are unlikely to be as useful as randomly selected positive constraints.

C-Terminal Peptide of β-Amyloid Protein

The second test of the AmbiPack algorithm is a nine-residue peptide. This peptide (β34–42) is from

TABLE II. Results of AmbiPack Running With Multiple Subsets of Positive Constraints

Constraint set size	No. of sets	Sets with solutions	Running time/set (sec)
8	1	0	63
7	8	1	90
6	28	9	92
5	56	29	106
4	70	50	135
3	56	50	201

TABLE III. The Complete Positive Constraints and Three Subsets

All positive constraints	*A*	*B*	*C*
C_{α}^{37}, C^{38}	C_{α}^{37}, C^{38}	C_{α}^{37}, C^{38}	C_{α}^{37}, C^{38}
C^{37}, C_{α}^{39}	C^{37}, C_{α}^{39}		
C^{36}, C_{α}^{39}	C^{36}, C_{α}^{39}	C^{36}, C_{α}^{39}	C^{36}, C_{α}^{39}
C^{36}, C_{α}^{40}	C^{36}, C_{α}^{40}		C^{36}, C_{α}^{40}
C^{34}, C_{α}^{39}		C^{34}, C_{α}^{39}	C^{34}, C_{α}^{39}
C^{34}, C_{α}^{40}	C^{34}, C_{α}^{40}	C^{34}, C_{α}^{40}	C^{34}, C_{α}^{40}
C_{α}^{36}, C^{38}	C_{α}^{36}, C^{38}	C_{α}^{36}, C^{38}	C_{α}^{36}, C^{38}
C_{α}^{36}, C^{39}	C_{α}^{36}, C^{39}	C_{α}^{36}, C^{39}	

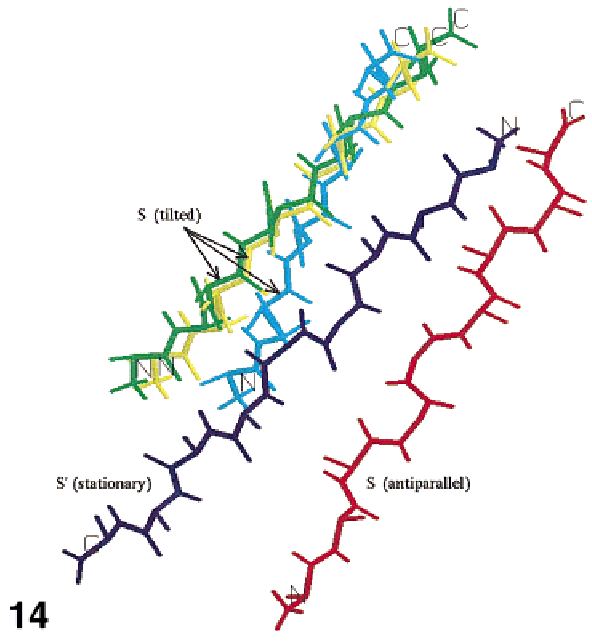

Fig. 14. 4 packing solutions to constraint set *A*.

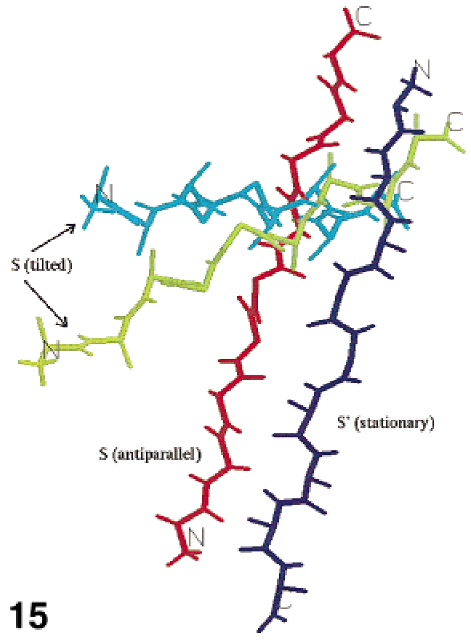

Fig. 15. 3 packing solutions to constraint set *B*.

the C-terminus of the β-amyloid protein (β1–42). Lansbury et al.[1] have applied solid-state ^{13}C NMR to this peptide and measured intra- and intermolecular ^{13}C–^{13}C distances. Their experiments produced 8 positive and 12 negative intermolecular constraints. A pleated antiparallel β-sheet was proposed as the structure that satisfies all constraints, although the data can only define the structure to a relatively low resolution. There are two types of interfaces in their proposed structures. They alternate among the β-strands in the sheet (Fig. 2). Either interface satisfies all negative constraints but only a subset of the positive ones. Together the interfaces satisfy all positive constraints.

A nine-residue poly-alanine idealized β-strand was energy minimized subject to the intramolecular backbone carbon-carbon distance restraints and used as the monomer. (Initial experiments had suggested that the monomer was a bent β-strand due to a *cis* peptide bond, but more recent evidence is consistent with an all-*trans* structure.) AmbiPack was given the 20 measured constraints and additional constraints that eliminate steric clashes. The positive constraints have very lenient upper bounds of 6.5 Å and the negatives also have lenient lower bounds of 5.5 Å. Still, AmbiPack could not find any solution to this problem, meaning that there is no single packing that satisfies all the constraints; two or more different packings are required. This result is consistent with Lansbury et al.'s[1] two-interface structures.

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

To find the full structure of the peptide, we need to combine the solutions from *A* with those from *B* or *C*. $A \cup B$ or $A \cup C$ gives the complete set of positive constraints. Lansbury et al.[1] have shown that this peptide forms a noncrystalline, yet ordered aggregate. The most plausible structure consists of an infinite number of subunits that "tile" the space in a regular fashion. Therefore, we try to calculate a continuous structure with alternating interfaces from *A* and *B* or *A* and *C*. First, we focus on *A* and *B*. Suppose T_1 and T_2, two rigid transformations, are

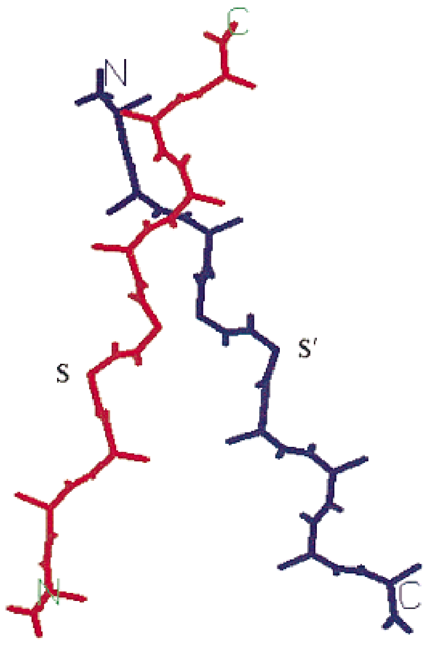

Fig. 16. A single solution to constraint set *C*.

solutions to A and B, respectively. Let S be the subunit. Then

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

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

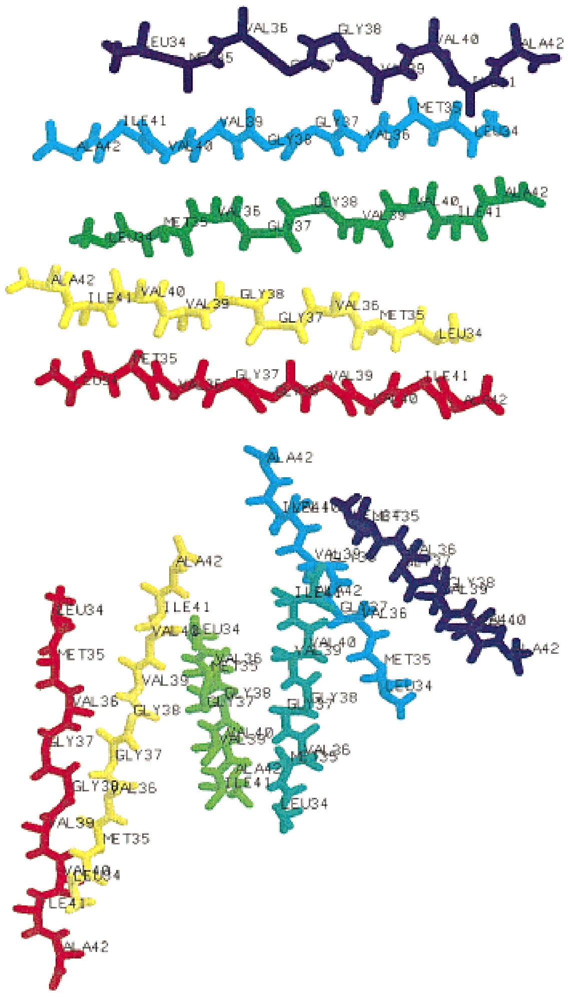

Fig. 17. Two continuous structures with alternating interfaces satisfying *A* and *B*, but without steric clashes.

CONCLUSION

The AmbiPack algorithm has been developed to pack preconformed monomer structures into multimers using interatomic distance constraints. A novel feature of the approach taken here is that it deals efficiently and accurately with the labeling ambiguity inherent in symmetric multimers due to a lack of knowledge about which atom in an intermolecular distance constraint comes from which monomer. The branch-and-bound method is applied to a search tree defined for progressively finer levels of resolution in the placement of three points on one of the monomers. Efficient pruning dramatically reduces the branching factor for this tree from a theoretical value of 512 to values typically 20-fold lower. Improved pruning causes the algorithm to run faster when more constraints are present. While the algorithm is exhaustive at any desired level of resolution, we have found that it is generally sufficient to stop the search at relatively coarse resolution of 2 Å. In our tests, resolutions down to 0.25 Å did not generate distinct new solutions. Methods based on this algorithm could be especially useful in instances where it is important to establish the uniqueness of a packing solution or to find all possible solutions for a given set of constraint data.

ACKNOWLEDGMENT

The authors thank Peter T. Lansbury, Jr. and Robert G. Griffin for many fruitful discussions.

REFERENCES

1. Lansbury, P.T. Jr., Costa, P.R., Griffiths, J.M., et al. Structural model for the β amyloid fibril: Interstrand alignment of an antiparallel β sheet comprising a C-terminal peptide. Nature Struct. Biol. 2:990–998, 1995.
2. Harbury, P.B., Zhang, T., Kim, P.S., Alber, T. A switch between two-, three-, and four-stranded coiled coils in GCN4 leucine zipper mutants. Science 262:1401–1407, 1993.
3. Steinbacher, S., Seckler, R., Miller, S., Steipe, B., Huber, R., Reinemer, P. Crystal structure of P22 tailspike protein: Interdigitated subunits in a thermostable trimer. Science 265:383–386, 1994.
4. Nilges, M. A calculation strategy for the structure determination of symmetric dimer by ^{1}H NMR. Proteins 17:297–309, 1993.
5. Nilges, M. Calculation of protein structures with ambiguous distance restraints, automated assignment of ambiguous NOE crosspeaks and disulphide connectivities. J. Mol. Biol. 245:645–660, 1995.
6. O'Donoghue, S.I., King, G.F., Nilges, M. Calculation of symmetric multimer structures from NMR data using a priori knowledge of the monomer structure, co-monomer restraints and interface mapping: The case of leucine zippers. J Biomol. NMR 8:193–206, 1996.
7. Lipton, M., Still, W.C. The multiple minimum problem in molecular modeling. Tree searching internal coordinate conformational space. J. Comp. Chem. 9:343–355, 1988.
8. Bruccoleri, R.E., Karplus, M. Prediction of the folding of short polypeptide segments by uniform conformational sampling. Biopolymers 26:137–168, 1987.
9. Aho, A., Hopcroft, J.E., Ullman, J.D. "Data Structures and Algorithms." Reading, MA: Addison-Wesley, 1982.
10. Ferro, D.R., Hermans, J. A different best rigid-body molecular fit routine. Acta Crystallogr. A33:345–347, 1977.
11. Steele, G.L. Jr. "Common Lisp," 2nd ed. Bedford, MA: Digital Press, 1990.
12. Bernstein, F.C., Koetzle, T.F., Williams, G.J.B., et al. The protein data bank: A computer-based archival file for macromolecular structures. J. Mol. Biol. 112:535–542, 1977.
13. Crippen, G.M., Havel, T.F. "Distance Geometry and Molecular Conformation." Chemometrics series. New York: John Wiley & Sons, 1988.