PROTEINS: Structure, Function, and Genetics 39:112–131 (2000)

RESEARCH ARTICLES

A Fast Method to Sample Real Protein Conformational Space

Howard J. Feldman[1,2] **and Christopher W.V. Hogue**[1,2]*
[1]*Department of Biochemistry, University of Toronto, Toronto, Ontario, Canada*
[2]*Samuel Lunenfeld Research Institute, Mount Sinai Hospital, Toronto, Ontario, Canada*

ABSTRACT **A fast computer program, FOLDTRAJ, to generate plausible random protein structures is reported. All-atom proteins are made directly in continuous three-dimensional space starting from primary sequence with an N to C directed build-up method. The method uses a novel pipelined residue addition approach in which the leading edge of the protein is constructed three residues at a time for optimal protein geometry, including the placement of *cis* proline. Build-up methods represent a classic N-body problem, expected to scale as N^2. When proteins become more collapsed, build-up methods are susceptible to backtracking problems which can scale exponentially with the number of residues required to back out of a trapped walk. We have provided solutions to both these problems, using a multiway binary tree that makes the N-body problem of bump-checking scale as NlogN, and speeding up backtracking by varying the number of tries before backtracking based on available conformational space. FOLDTRAJ is independent of energy potentials, other than that implicit in the geometrical properties derived by statistical studies of known structures, and in atomic Van der Waals radii. WHAT_CHECK shows that the program generates chirally and physically valid proteins with all bond lengths, angles and dihedrals within allowable tolerances. Random structures built using sequences from PDB files 1SEM, 2HPR, and 1RTP typically have 5–15% α-helical content (according to DSSP) and on the order of 20% β-strand/extended content. Ensembles of random structures are compared with polymer theory and with experimentally determined fluorescence resonance energy transfer distances. Reasonably sized structure ensembles do sample most of the conformational space available to proteins. The method is also capable of protein reconstruction using Cα—Cα direction vectors, and it compares favorably with methods that reconstruct protein backbones based on alpha-carbon coordinates, having an average backbone and Cβ root mean square deviation of 0.63 Å for nine different protein folds. Proteins 2000;39:112–131.**

Key words: prediction; trajectory distribution; conformer; structure; random; polypeptide

INTRODUCTION

Evidence exists to suggest that the protein folding problem is intractable in its crudest form[1]—exploring all of conformational space. Fortunately, this may not be necessary to solve the problem, as the scientific community's view of the problem has begun to change.[2] Just as "all roads lead to Rome," numerous folding pathways may lead ultimately to the same folded structure.[3] We are ultimately interested in methods that allow us to work with ensembles of proteins. We wish to explore methods of ensemble optimization in order to predict folded protein conformations. However, just as one must walk before one can run, we found it necessary to be able to make a single, plausible unfolded protein conformer before we can produce a folded one. We are interested in a fast method based on geometric parameters in order to generate protein structure ensembles independent of any specific potential function.

Various methods exist for generating protein structures at both the backbone and side chain levels of detail, having been developed to solve homology modeling problems and in the fitting of experimental data for NMR and X-ray structures. Few of these tools can be optimally adapted to generate unfolded random protein conformations. For

Abbreviations: ASN.1, abstract syntax notation 1; BPTI, bovine pancreatic trypsin inhibitor; CDF, cumulative distribution function; CPU, central processing unit; FRET, fluorescence resonance energy transfer; FWHM, full width at half-maximum; MMDB-API, molecular modelling database applications programming interface; NCBI, national center for biotechnology information; NMR, nuclear magnetic resonance; PDB, brookhaven protein data bank; PDF, probability distribution function; RMS, root mean square; RMSD, root mean square deviation; SCWRL, sidechain placement with rotamer library.

*Correspondence to: Christopher W.V. Hogue, Samuel Lunenfeld Research Institute, Mt. Sinai Hospital, 600 University Avenue, Toronto, Ontario, Canada M5G 1X5. E-mail: hogue@mshri.on.ca

Received 23 July 1999; Accepted 15 November 1999

example, CNS [Brünger, A. T. v 0.5 (1998). Yale University, New Haven, CT] is capable of doing so in an indirect manner, by first generating an extended structure with ideal geometry and then subjecting this to simulated annealing, but with no restraints (such as NOEs or dihedral restraints). The quality of the generated structures is acceptable, but the structures tend to be very loose and often have just a single region of tertiary contact, since no restraints are used. It may be possible to produce more compact structures with CNS by adding restraints to the annealing, but it is unclear how these could be chosen so as not to introduce any bias towards a particular fold or shape, i.e., so that the restraints do not dominate the outcome of the simulation.

While many methods involve molecular dynamics refinements, some are based strictly on protein structure geometry. Rey and Skolnick[4] provide a method for the reconstruction of an entire backbone, including β-carbons, from only the α-carbon trace. Their method is based on a lookup table of director cosines and requires only the $i - 1$, i and $i + 1$ α-carbon coordinates to place the ith β-carbon. A different approach to the same problem[5] matches fragments of the α-carbon trace to similar fragments in the protein database and uses this information to reconstruct the backbone. Others perform a refinement along with the reconstruction as well[6] to help remove bias from the existing database, or use a conformational search.[7] We seek a method that does not involve a refinement pass.

In this work we describe a complete, detailed method to construct all-atom protein conformers in $O(NlogN)$ time. The method is based on pipelined residue addition and probabilistic geometry sampling. We have set out to optimize this method both in speed and in the quality of the protein conformers it generates. The method uses no explicit potential functions to perform its work, yet makes high quality protein conformers. Proteins generated by this method from sequence alone are sterically plausible, but nonetheless are random, unfolded conformers. The method is a collection of new and old algorithmic approaches to protein chain build-up, and it allows us and others to rapidly generate ensembles of protein conformers. Since the conformers are all-atom models, one can apply any potential or scoring functions to members of an ensemble, and objectively score them, in a manner independent from the build-up method.

What is most important about this method is that the input is a map of conformational space that we call a *trajectory distribution.* The input can be derived from the output, that is to say that an ensemble of protein conformers can be scored with any potential function, and mapped back onto the trajectory distribution surface. This can then be used as input for the generation of the next ensemble. Thus, the trajectory distribution can be used as a memory[8] of the conformational space that has already been explored by an ensemble and scored by the potential. By performing ensemble scoring with an appropriate potential, an iterative system for protein folding may be developed starting with the building block we describe here, the *trajectory directed protein conformer generator,* FOLDTRAJ. The concept is similar to that of a genetic algorithm, where the "fittest" structural elements in one pool of structures are kept to help build the next generation of structures.[9]

The method we use builds a polypeptide chain in a "pipelined" process from N to C terminus working on three residues at a time, to make use of as much information as possible in order to construct accurate, plausible protein geometry. The first step is the generation of an extending ith α-carbon. The placement of this alpha carbon is selected by sampling a probability distribution function of the conformational space available to it, as described on the surface of a sphere with center at the previous α-carbon. We have corrected this surface for the inherent spherical sampling bias, forming the *trajectory distribution,* which is described in detail in the Methods section. Having placed the α-carbon of the ith residue, the $i - 1$ residue β-carbon is placed using the table-driven placement method of Rey and Skolnick,[4] which has been modified and extended for placing *cis* residues. Next, the peptide atoms between the $i - 1$ and the $i - 2$ residues are placed simultaneously using a new binary search and optimization function we have developed. The amino acid side chain rotamer placement is possible since now the $i - 2$ α-carbon has a complete backbone on either side. We tested two different approaches to placing rotamers, and selected one based on Dunbrack's backbone-dependent rotamer library used by SCWRL (Dunbrack, R. L., Jr. v 2.0 (1997)). Finally the proton of the $i - 2$ α-carbon is placed, then the process is repeated, with approximations required for polypeptide N and C termini.

At each stage in the build-up, atoms are tested for collisions with other atoms on-the-fly using our own $O(NlogN)$ technique, described in the Methods section. Backtracking is performed whenever the chain cannot proceed after a pre-set number of collisions. We also report a "smart" approach to backtracking, in which the number of attempts before backtracking is varied with the conformational space available at each residue. This smart backtracking affords a significant speed-up to the overall method.

METHODS AND MATERIALS

Software and Languages

FOLDTRAJ and INITTRAJ (which generates the input trajectory distribution file for FOLDTRAJ) were programmed completely in ANSI C using the Molecular Modeling Data Base Applications Programming Interface (MMDB-API)[10] and the National Center for Biotechnology Information (NCBI) toolkit (Ostell, J. v 6.0 (1997), available at ftp://ncbi.nlm.nih.gov/toolbox/ncbi_tools). The CodeBase (Sequiter Software, Edmonton, Alberta, Canada) C database library was used, as well as bzip2 compression (Seward, J. v 0.9.0 (1998)) in order to handle the very large files that are used as the input probability density functions, described below. The software is cross-compiled under Win32, Linux, IRIX, and Solaris platforms, and is available in executable form at: ftp://bioinfo.mshri.on.ca/pub/TraDES/.

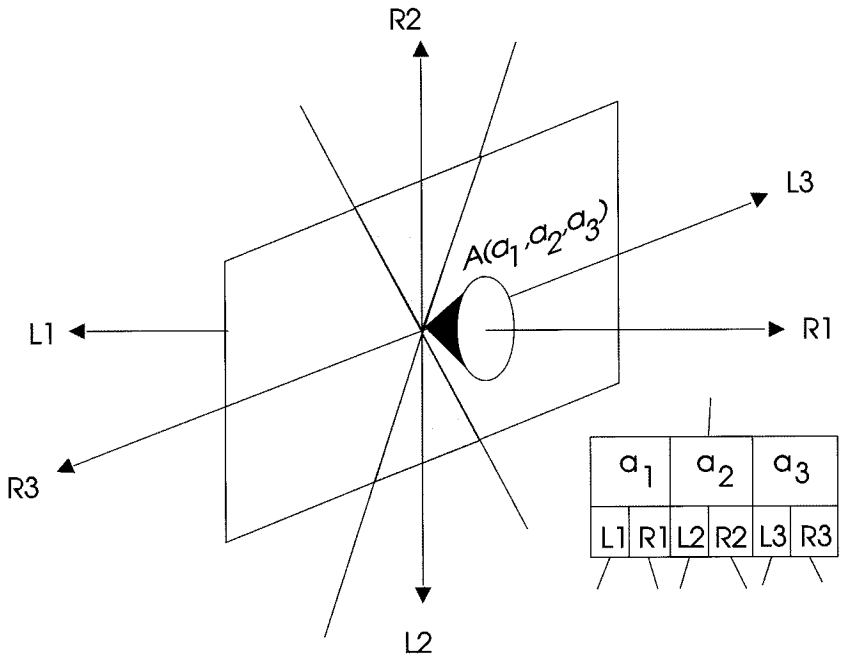

Fig. 1. Node structure of the binary-d tree. Note that the left and right child pointer pairs {L1, R1}, {L2, R2} and {L3, R3} are simply binary trees that descend down 3, 2 and 1 dimensions.

Non-Redundant Set of PDB

The majority of statistical data from the Brookhaven Protein Data Bank (PDB)[11] used in this work was obtained from a non-redundant set of the PDB. We utilized the nonredundant set (http://www.ncbi.nlm.nih.gov/Structure/VAST/nrpdb.html) provided by NCBI. A representative structure (or chain, where appropriate) was selected manually at NCBI for each of the 1,255 resultant groups using a BLAST p-value cutoff of 10^{-7}. This list was further culled by removing entries less than 40 residues in length, chains containing "unknown" residues and those with X-ray resolution greater than 3.0 Å. Chains of less than 200 residues were limited to 2.5 Å resolution. Finally, those missing residues or co-ordinates were removed, resulting in a final non-redundant set of 834 protein chains. The complete list can be found on our web page: http://bioinfo.mshri.on.ca/trades/filtmmdblist/.

O(*NlogN*) Collision Detection

Nearest-neighbor measurements to perform bump-checking represents a classic N-body problem. A system of N entities forms an N-body problem when each contributes a force or property that can influence all other N − 1 entities, and the total number of interactions is N(N − 1)/2. The computational time dependence of such a system increases as the square of the number of entities, or $O(N^2)$, and such computation can grow to dominate CPU time.[12] Very few protein structure generation methods we have found report analyses of their time-complexity with respect to chain length N. Since the overall time complexity of a method is largely dependent on the time complexity of the worst-behaved underlying algorithm, we wanted an optimal solution for collision detection, which is *O*(*NlogN*).

Many different algorithms to improve the computational efficiency of N-body systems have been proposed.[12–17] Hierarchical structures have been identified as the most efficient of these[15,18,19] but are not routinely used in molecular simulations. We were interested in implementing an *O*(*NlogN*) algorithm for bump-checking van der Waals collisions that (a) did not require imposition of global volume constraints and (b) was easy to maintain in memory as atoms are added and removed during the simulation through chain elongation and backtracking.

The algorithm, based on a hierarchical data structure, partitions atoms in a molecular scene into a tree comprised of nodes shown in Figure 1. This tree represents points in space relative to one another in a multidimensional format. This is quite different from other *O*(*NlogN*) approaches, for example the oct-tree which encloses points in nested volume elements. Unlike the oct-tree, this tree does not need to pre-declare a fixed volume in which the protein structure is to be made. We have not found this particular tree described in the computing science literature, and thus refer to the tree as a *binary-d tree,* since it is a binary tree in the first dimension, extended into d binary trees at each node, as shown in Figure 2. Our binary-d tree has nodes pointing to atomic coordinate locations, together with a set of three pairs of left/right child pointers. The difference between this tree and the oct-tree or k-d tree[15] is that it is comprised of points or vectors, rather than partitioned volumes or hypercubes. It is also memory efficient, requiring only two pointers for each dimension (2d), as opposed to 2^d pointers for oct-trees. In practice, a small one-time overhead is paid when adding each new atom's coordinates to the tree. Importantly, coordinates in this tree can be removed with only minor adjustments to the child nodes. This is useful when backtracking, as atoms can be removed without having to rebuild the entire tree.

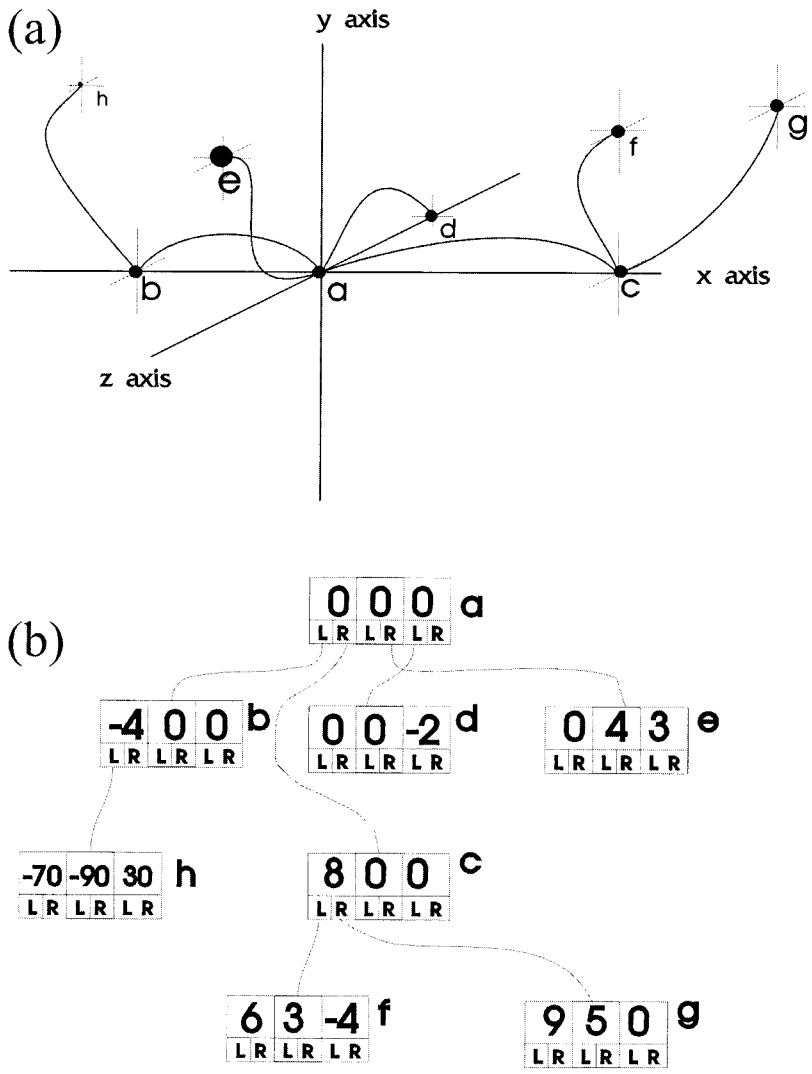

Fig. 2. Inserting the points a(0, 0, 0), b(−4, 0, 0), c(8, 0, 0), d(0, 0, 2), e(0, 4, 3), f(6, 3, −4), g(9, 5, 0) and h(−70, −90, 30) in sequence forms the binary-d tree depicted in **(a)** as a three-dimensional representation of the points in space connected by their pointers, and **(b)** as a tree showing the node structure and how the pointers interconnect. Node c may be easily removed and its children reattached by simply inserting them back into the tree.

Nearest neighbors are obtained by probing the tree with a multidimensional search using boundary coordinates that describe a three-dimensional rectangular probe volume, shown in Figure 3. The search returns a list of the atomic coordinates found in the probe volume. The method primarily distinguishes coordinates based on the first dimension, i.e., as a binary tree. The second and third set of left/right child pointers are rarely used. These higher dimensional left/right child pointers provide the key to higher-dimensional discrimination in the case where the first dimension is equivalent, as illustrated in the simple tree in Figure 2, and add little or no overhead to the implementation or search method. Given the three decimal place precision found in the PDB, it is extremely likely (>99.99%) that any given 100-residue or more structure will have at least two atoms with identical x co-ordinates. The time-complexity of this algorithm was determined to be $O(NlogN)$ with simple α-carbon kinetic random walks, prior to incorporation in the overall method (data not shown).

Hard/Soft Atoms and H-Bonds

We experimented with two models for describing atoms, a hard sphere and a soft sphere approach. If any two atoms are closer than allowed by van der Waals radii[20] a hard-sphere collision is deemed to occur. The most recently added residue would be removed and a new location chosen for it. In order to model van der Waals interactions in a slightly more realistic manner, atoms can be made "soft" by allowing collisions to occur under certain conditions. When a new atom is placed close to an existing one, the sum of their van der Waals radii, R, is compared with the actual distance between them, r. With a hard atom model, the placement is rejected if $r < R$ and backtracking occurs. However, the van der Waals radius is simply an energy minimum for atomic contacts, and is by no means an absolute minimum. Atoms closer together than this energy minimum will only remain so if they cannot push apart because of tight packing in the vicinity, and so there is a smaller chance of occurrence as interatomic distances get smaller. Following the Monte Carlo approach to soft

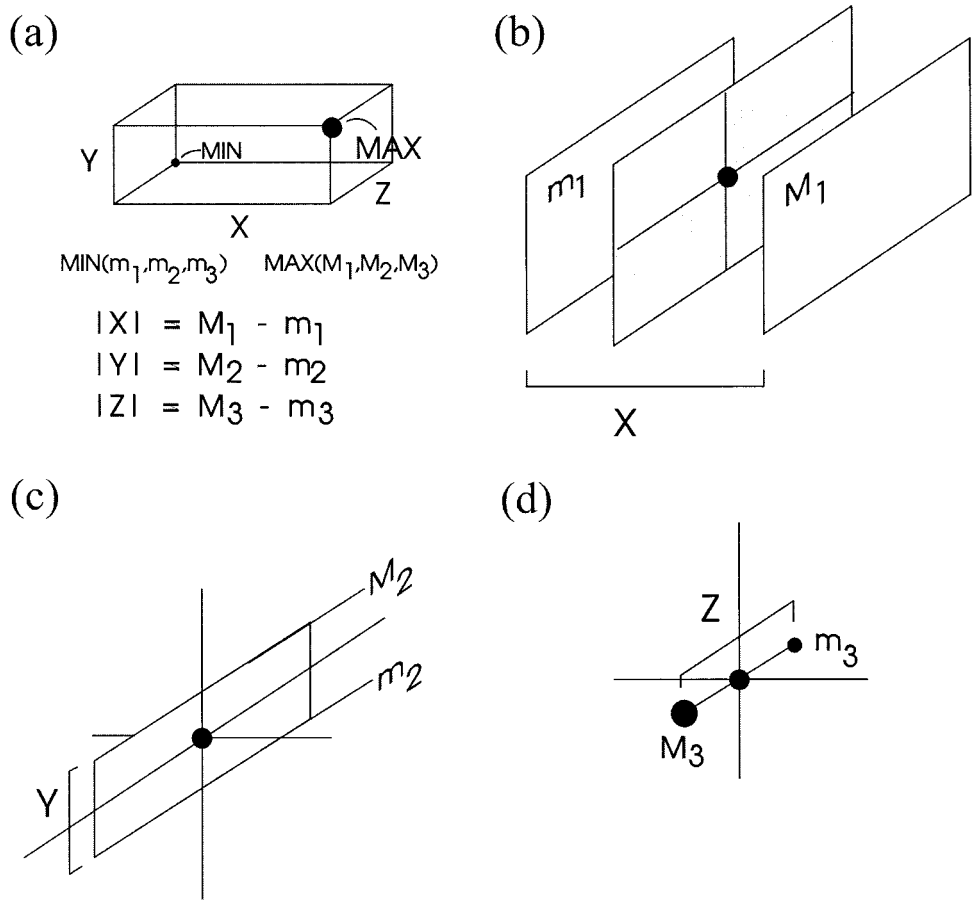

Fig. 3. The three-dimensional branch and bound search process. A search volume is designated by two points MIN(m_1, m_2, m_3) and MAX(M_1, M_2, M_3). These form a rectangular prism **(a)** with sides parallel to the xy, yz and xz planes. (b–d) shows the elimination of: **(b)** planes orthogonal to x containing points in the tree that are not in the range $m_1 \ldots M_1$; **(c)** lines on each plane containing points perpendicular to y that are not in the range $m_2 \ldots M_2$; and **(d)** points on each line that are not in the range $m_3 \ldots M_3$.

atoms,[21] whenever $r < R$, a random number is used to determine whether the collision is permitted, otherwise, a crash occurs just as in the hard atom model. The further r is from R, the greater the chance of a collision.

Whenever one of the two colliding atoms is a hydrogen, and the potential to form a H-bond exists (the colliding atoms and the atom bonded to the hydrogen are either O or N), the angle formed by the two colliding atoms plus the atom bonded to the H is calculated. For a successful H-bond to form, this angle must be between 150° and 180°, and the van der Waals radius of the hydrogen is then reduced by 0.5 Å to account for the special bond. If either criteria is not met, the collision is treated as a normal soft atom collision as above.

Placement of α-Carbons Using the Trajectory Distribution

Our goal was to find a method for generating complete chirally-plausible protein structures by choosing from as few variables as possible per residue, combining and enhancing methods conceived earlier for both chain elongation,[21] and for positioning β-carbons.[4] Our peptide backbone atom placement algorithm is a new approach using a binary-search optimization. A geometrical build-up procedure is based on the assumption that by specifying the positions of only the α-carbons in a protein backbone, the remaining coordinates can be optimized in a deterministic manner.

Given any three consecutive α-carbon coordinates, the next one can be specified by its distance R from the previous atom, and its location on the sphere of radius R surrounding the previous atom. The exact position can be specified using spherical coordinates ϕ and θ, where ϕ corresponds to the azuthimal angle ranging from 0 to π, while θ is longitudinal and runs from 0 to 2π. In this case θ would also be the dihedral angle between the four Cαs and ϕ, the supplement of the angle between three consecutive Cαs. The distance between consecutive backbone α-carbons is approximately 3.81 Å (or 2.9 Å for a pair where the second residue is *cis*-proline[4]) which gives R, so that the position of the next residue can be completely determined by choosing only two angles.

In order to randomly generate a complete α-carbon backbone of length N then, $N - 3$ points had to be chosen

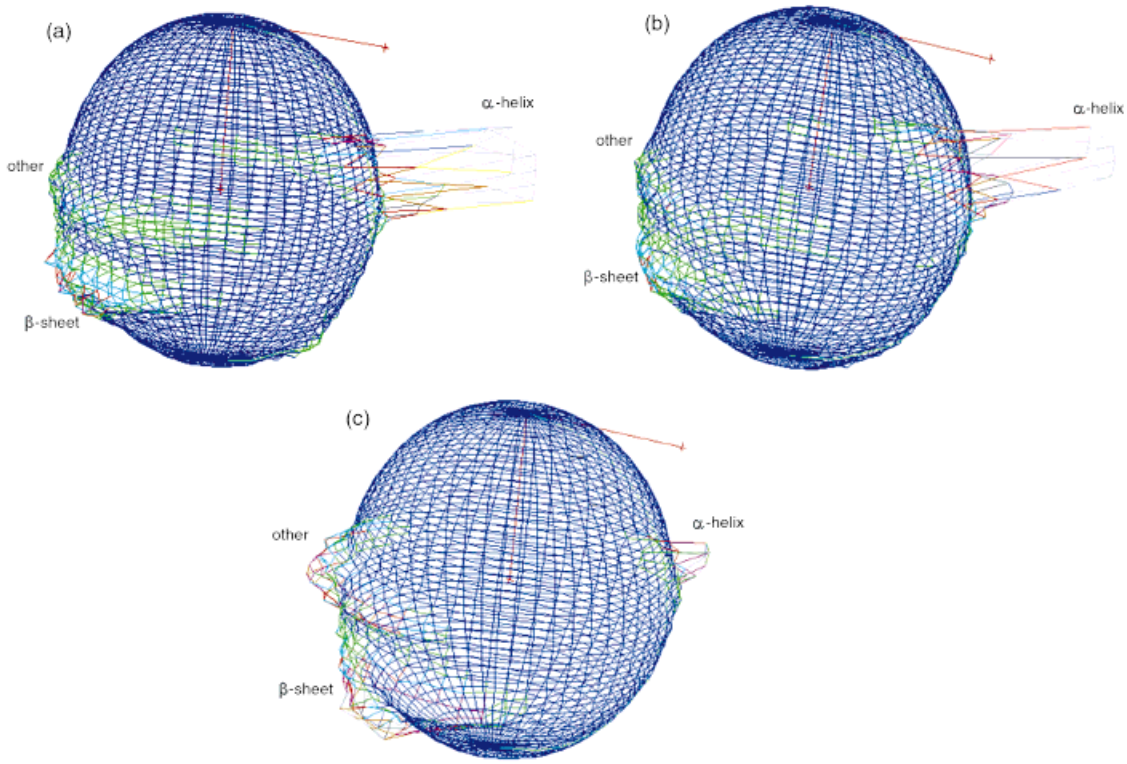

Fig. 4. Trajectory distributions as they appear on the surface of a sphere for **(a)** Ala, **(b)** Lys, **(c)** Pro. Each grid square corresponds to a patch of equal area on the surface of the sphere. Peaks for helix, sheet and other are as indicated. Note that while one of the peaks is labeled 'other,' residues in random coil or turn conformations can actually fall anywhere on the trajectory distribution, including on top of the α-helix or β-sheet peaks. The coloring scheme is intended to give only an approximate indication as to the relative heights of each peak. Note that for Ala and Lys, the helix peak has been truncated for clarity, and as well, only 50 divisions were used in each direction rather than the usual 400. The prime meridian in each figure (i.e. $\theta = 0$) is on the right-hand edge of the sphere and the three red crosses represent the backbone. $C\alpha_{i-2}$ lies outside the sphere, in the direction of the prime meridian. $C\alpha_{i-1}$ is at the north pole and $C\alpha_i$ is at the center of the sphere. The height of the landscape at any point on the surface of the sphere is hence proportional to the probability that $C\alpha_{i+1}$ will be found there.

at random in ϕ–θ space (the first three atoms are defined uniquely only by the angle between them). The approach used is thus similar to that of Gregoret and Cohen.[21] However, we realized that in order to allow the sampling of this space, it had to be discretized evenly. The area of a small patch on a unit sphere of size $\Delta\phi$ by $\Delta\theta$ is:

$$\int_{\theta}^{\theta+\Delta\theta}\int_{\phi}^{\phi+\Delta\phi} 1^2 \sin\phi \, d\phi \, d\theta = \Delta\theta(\cos\phi - \cos(\phi+\Delta\phi))$$

$$= -\Delta\theta\Delta(\cos\phi) \quad (1)$$

Thus to ensure that each discrete unit on the surface of the sphere had the same area, it was required that θ be divided evenly (in its range from $-180°$ to $180°$) and $\cos\phi$ be divided into equal increments between -1 and 1. In other words, the value of $\Delta\phi$ must become larger near the poles to compensate for the narrowing in the longitudinal direction and provide a more unbiased sampling of the spherical surface than an even angular distribution, which tends to cluster samples near the poles, where there are simply more patches per unit surface area. This is similar to the problem encountered by geographers when trying to represent the earth on a flat piece of paper. Their solution was the Mercator projection, invented by Gerhardus Mercator in 1568, which projected a sphere onto the inside of a cylinder, which was subsequently unrolled.

By processing the non-redundant set of known protein structures and recording the populations in this angular 2-D space for each residue type, an initial "trajectory distribution," or frequency distribution, in this vector space was obtained for each amino acid. This could then serve as a starting point for the random backbone generator, using the observed frequency distribution function to approximate a discretized probability distribution function (PDF). While processing the nonredundant database, one distribution was produced per residue type, for a total of twenty. When actually beginning the random walk however, one trajectory distribution is maintained for each residue, with its starting value based on residue type alone. To optimize the speed, a binary search on the domain of the cumulative distribution function (CDF), F(x) (the integral of the PDF), is used to choose random ϕ–θ pairs.

The trajectory distributions as they appear in their native two-dimensional vector space, the surface of a sphere, are shown in Figure 4. It should be noted that

TABLE I. Information Contained in Each Trajectory Distribution Record of the Database

Field name	Purpose
Residue number	Primary search key—index into database
Amino acid	One-letter amino acid code
Dimension	Size of trajectory distribution in both dimensions, normally 400; each residue can potentially have different resolutions of discretization
Endianness	Keeps track of what type of CPU the binary data was generated on to maintain platform independence of the file
Compression type	Indicates whether bzip2, RLE, or some other compression method was used on the trajectory distribution data
Buffer size[a]	Size of the compressed trajectory information (needed for decompression)
Trajectory distribution data[a]	Actual compressed binary data
Integral[a]	Total area under the trajectory distribution; units are arbitrary
Peak value[a]	Largest single value in the trajectory distribution—allows comparison of relative sizes of different trajectory distributions
First non-zero row[b]	First row in the trajectory distribution array which contains a non-zero number; the first row is row 1
Number of non-zero rows[b]	The number of rows from the first non-zero row to the last one; rows of zeroes may appear amongst the non-zero ones, but this counts up to the last row with any non-zero data in it
Number of elements ≤ 0[c]	Number of entries in the entire trajectory distribution array which are zero; gives an indication of shallowness
Number of elements $\leq 5\%$ peak[c]	Same as above, but includes up to 5% of the peak value
Number of elements $\leq 10\%$ peak[c]	Same as above, but includes up to 10% of the peak value
Number of elements $\leq 15\%$ peak[c]	Same as above, but includes up to 15% of the peak value
Timeout	Number of tries at each residue before backtracking
Omega mean	Average peptide dihedral angle between residue $i - 1$ and i
Omega standard deviation	Standard deviation to use when choosing omega randomly
Cis probability	% probability that the bond between residue i and $i + 1$ will be *cis* (0 unless $i + 1$ is Pro)
S-S probability	% probability that residue is involved in a disulfide bridge (0 unless Cys) (for future use)
Rotamer info	Unused—possibly for storing favored rotamer index

[a]Has a corresponding field for when the residue is chosen to be *cis,* provided Cis probability is not zero. All other fields apply to both sets of trajectory distributions (*cis* and *trans*) where appropriate, unless otherwise noted; *cis* trajectory distributions are used to place residue $i + 1$ when residue i is a *cis*-proline.
[b]Always applies to *trans* trajectory distribution only; non-zero rows of *cis* trajectory distributions are stored explicitly.
[c]Always applies to most probable trajectory distribution, i.e., *trans* unless Cis probability > 0.5.

there is no easy way to choose N equally distributed points on a sphere, when N is any integer other than the number of vertices found in the five Platonic solids.[22] The best that we can do to get a fair discretization of the sphere is to ensure that each patch has the same area, since, all else being equal, the probability that a patch is chosen is proportional to its area.

Trajectory Distribution Implementation

It was important that the method could accurately represent the α-carbon trace of real proteins since our goal was to represent proteins in continuous space. It was determined that 400×400 discretization of cosϕ–θ space provided an optimal compromise between backbone precision (measured by Cα—Cα RMSD between actual and "discretized" structure) and memory and disk usage for this sampling system, with 500×500 showing little improvement in quality and 200×200 showing a marked decrease (data not shown). This requires a rather large amount of data to use as input for this program, especially if all $N \times 400 \times 400$ values are loaded into memory at once for a large protein! To deal with this problem we built a database system that contains compressed trajectory distributions as database records, which can be loaded as required for each amino acid being sampled. This database approach affords the extra advantage of buffering, so if the algorithm should backtrack and require a recently loaded trajectory distribution, it may still be in memory and saves the time of loading from disk again.

Using compression, the amount of space required to store each amino-acid based trajectory distribution was reduced by over a factor of 70. These compressed trajectory distributions were stored as CodeBase binary database records. Additional data about the trajectory distribution was added to database fields, summarized in Table I. The overall scheme is depicted in Figure 5. While compression time can be lengthy, it need only occur once for each sequence, during the initial creation of the database.

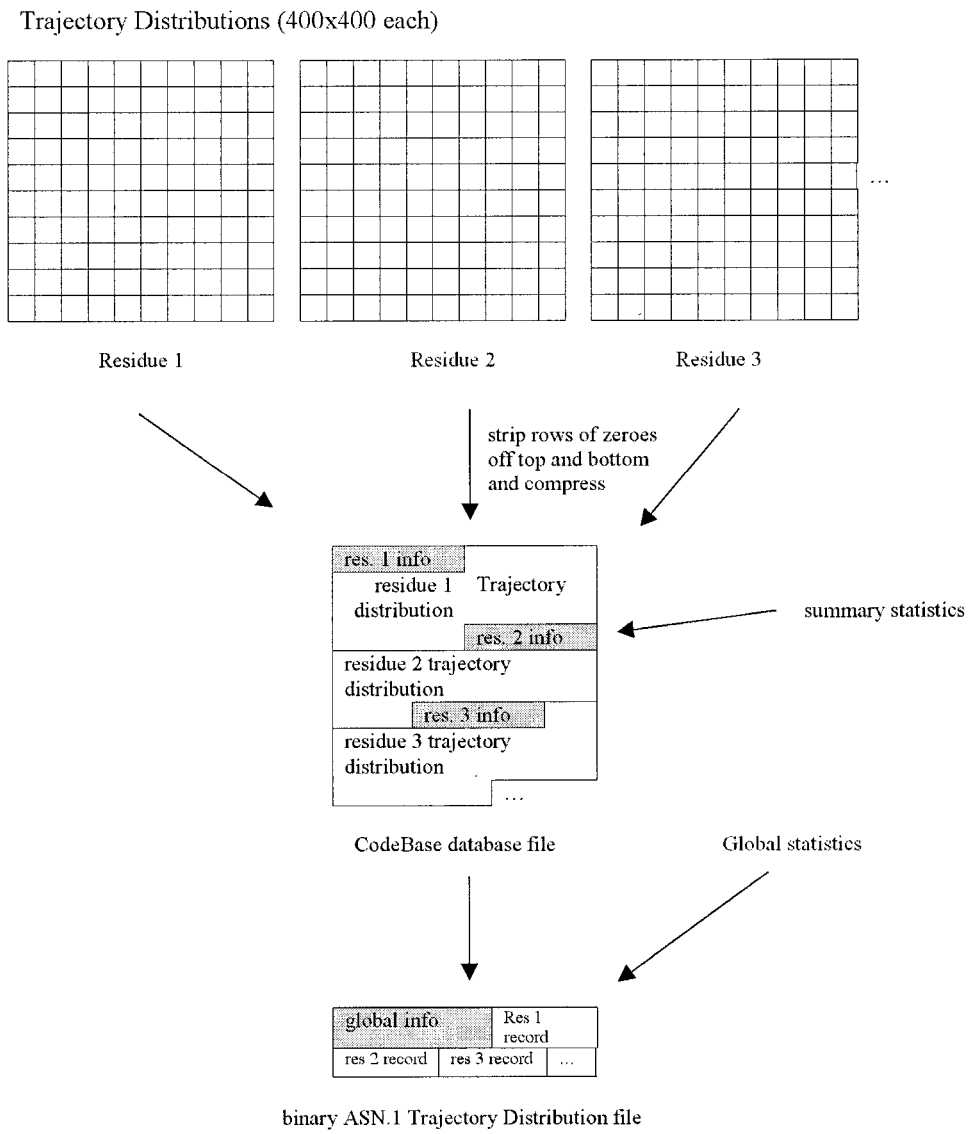

Fig. 5. Trajectory distributions are represented in memory as large 2-dimensional arrays, one per residue. Rows of zeroes at the beginning and end of each trajectory distribution are stripped off, and the remaining data compressed with bzip2 and stored as a CodeBase database record along with other information about the residue (Table I). The entire database, with one record for each residue in the protein it represents, is then packed into a binary ASN.1 file along with some global information such as sequence length and amino acid sequence, and written to disk. It is this "trajectory file" which is used as input to FOLDTRAJ.

Decompression, on the other hand, is nearly instantaneous. The same trajectory distribution binary database file can be used on almost any 32-bit machine (Mac, PC, or UNIX) without modification. Finally, to reduce the number of files from three (for a Codebase database) to just one, the files were packed into an Abstract Syntax Notation (ASN.1) binary format file, along with some global information about the protein. Thus the input of the main conformer generator program, FOLDTRAJ, is a single, packaged data file created by a separate program, INITTRAJ.

Placement of β-carbons and Consideration of *cis* Residues

Rey and Skolnick[4] found that the precise position of the β-carbon (or hydrogen in the case of Gly) was almost completely determined by the positions of the two adjacent α-carbons relative to the one in question and provide a lookup table to locate them. We found this method to work well in most cases, however, at certain locations, especially in short loops between helices or sheets, some (approx. 5–10% of all residues) very poor placements occur that are greater than 0.5 Å from the correct location. Once the Cβ location is known, the placement of the remainder of the side chain (except for hydrogens) is completely determined by the torsional angles χ_1, χ_2, χ_3, χ_4 of the residue.

We found it necessary to improve on the treatment of *cis* residues, and our method accurately accounts for the possible addition of *cis*-prolines. A *cis* bond greatly affects the virtual backbone angles in the two residues joined by the peptide bond, and not just in the Pro residue, as

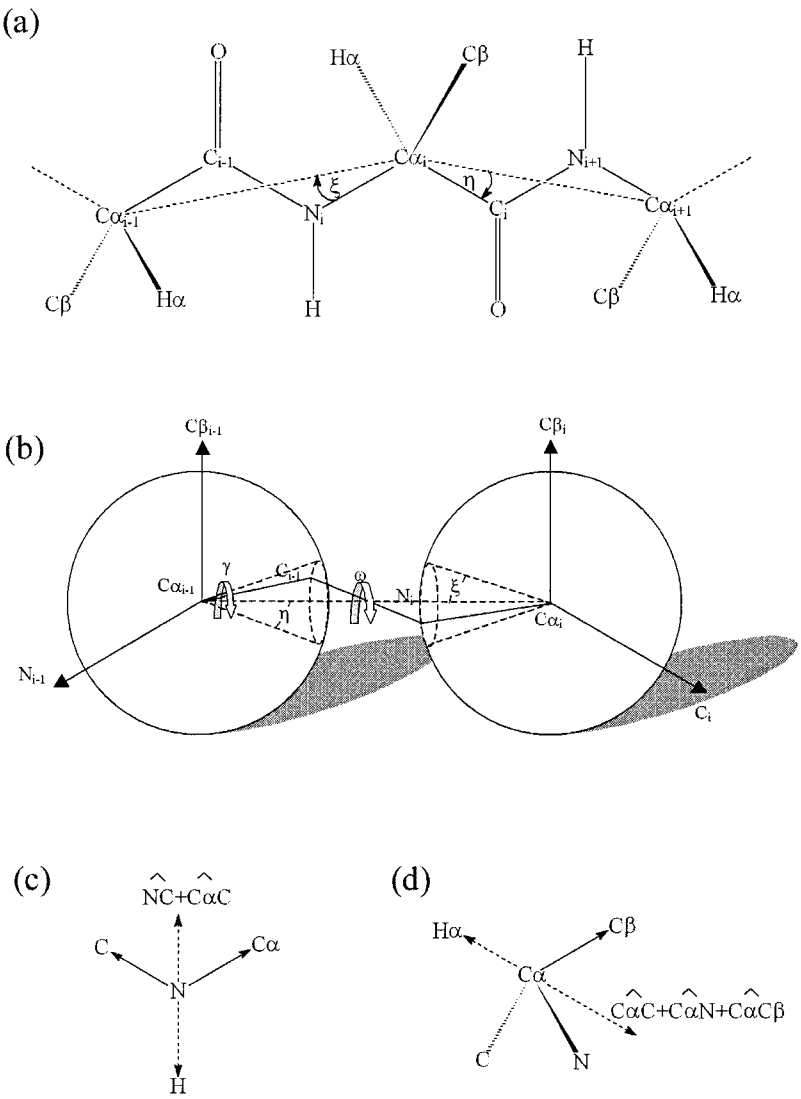

Fig. 6. **(a)** Definitions of the backbone angles ξ and η. **(b)** Placement of the peptide C and N atoms. As C_{i-1} is rotated by angle γ about the $C\alpha_i$—$C\alpha_{i-1}$ axis, it traces out the base of a cone of angle η lying on the surface of a sphere, with radius equal to the fixed CαC bond length. N_i is then fixed by ω, the dihedral between C_{i-1}, $C\alpha_{i-1}$, $C\alpha_i$ and N_i, and the angle ξ, and bond length NCα. The peptide bond length, as well as other backbone angles are not explicitly used, and hence are minimized for changing angle of rotation γ. **(c)** and **(d)** Calculation of the H (c) and Hα (d) co-ordinates by taking the direction opposite the sum of the unit vectors for each of the other atoms attached at the same node, as indicated.

implied by the tables of Rey and Skolnick. As a result, for a *cis* bond between residues $i - 1$ and i, different values had to be used for η_{i-1}, ξ_i, angle O_i—$C\alpha_i$—$C\alpha_{i-1}$ and length O—$C\alpha_{i-1}$ (see Fig. 6a), all of which play a crucial role in our placement of peptide atoms (see next section). As well, the Cβ direction vector at residue $i - 1$ tends to be quite different.

Using the non-redundant set of PDB files, these values were recorded at each occurrence of a *cis*-Pro, and sorted by amino acid type at the $i - 1$ position. Due to scarcity of the data, Cβ directions were averaged over all $C\alpha_{i-1}$—$C\alpha_{i+1}$ distance bins (i.e., that variable was integrated out in this case) and global averages (independent of amino acid type at residue $i - 1$) of 57.81° were used for η_{i-1}, 84.46° for O_i—$C\alpha_i$—$C\alpha_{i-1}$ and 2.388 Å for O—Cα. A value of 56.56° was used for ξ_i at *cis*-prolines. These all had relatively small variation so a global average was a meaningful approximation. Since Cα's are closer across a *cis* bond as well, a value of 2.955 ± 0.097 Å was derived for the Cα—Cα distance. The Cβ values thus derived are given in Table II.

As the random walk progresses, there is a 4.63% chance at each Pro residue that it will be placed *cis,* the observed frequency of *cis*-Pro in the non-redundant set from the PDB. If a given residue is chosen to be *cis,* but upon attempt to fill in the atoms, a collision or other error occurs, resulting in backtracking, then upon returning to the Pro, it will again have the same chance to be *cis.*

Finally, a separate trajectory distribution has been recorded and used for *cis*-Pro. While presently non-Pro residues are always *trans,* there is no reason why these could not also be added *cis* with a certain probability, provided enough data were available to obtain reliable estimates of parameters. We found that approximately 0.04% of all 175,000 non-Pro residues surveyed in the PDB are in the *cis* conformation. Similar fractions have been reported elsewhere.[23,24] This value may be artificially low since residues may be biased towards the *trans* conforma-

TABLE II. Direction Cosines for Cβ at Residue n-1 When There is a *cis* Peptide Bond Between Residues n-1 and n[†]

Residue	cosδ1	cosδ2	cosδ3
ALA	0.1595	0.9518	0.03410
ARG	0.2739	0.9351	0.1052
ASN	0.2285	0.9381	−0.06389
ASP	0.09837	0.9303	−0.06508
CYS	0.2181	0.9433	−0.01515
GLN	0.2697	0.9333	0.02162
GLU	0.1843	0.9451	0.09224
GLY	−0.3340	0.1663	0.1887
HIS	0.1298	0.9566	0.04983
ILE	0.2001	0.8726	0.1227
LEU	0.1266	0.9035	0.1235
LYS	0.1661	0.9563	0.07943
MET	0.1235	0.9174	0.1855
PHE	0.1477	0.9611	0.06610
PRO	−0.01185	0.9670	0.1570
SER	0.2224	0.9337	0.02720
THR	0.3036	0.9305	0.05830
TRP	0.2032	0.9384	0.006974
TYR	0.1853	0.9596	0.07812
VAL	0.1942	0.8873	0.09114

[†]They are relative to the reference frame described by Rey and Skolnick.[4] Values are global averages over all 414 *cis* bonds in the non-redundant set of proteins used in this study. Standard deviations are on the order of 0.1–0.2 depending on frequency of the amino acid prior to a *cis* bond.

tion during crystal structure refinement by software, or manually.[25,26]

Placement of Peptide Atoms

The remaining backbone atoms were filled in as the random walk progressed from one Cα to the next. In order to locate the peptide C and N atoms, the Cα—C and N—Cα bond lengths[27] were held fixed, and virutal bond angles η ($\angle C_i$—$C\alpha_i$—$C\alpha_{i+1}$) and ξ ($\angle N_i$—$C\alpha_i$—$C\alpha_{i-1}$) (see Fig. 6a) were chosen to be approximately normally distributed, with means depending on residue type[4] and standard deviation 1.5°, since these values vary significantly in real proteins. The peptide dihedral angle ω was fixed at 179.8° on average, with a standard deviation of 1.5°. Under these conditions, the positions of N_i and C_{i-1} were then chosen to minimize the total degrees squared error in $\angle N_i$—$C\alpha_i$—$C\beta_i$, $\angle C_{i-1}$—$C\alpha_{i-1}$—$C\beta_{i-1}$, $\angle N_{i-1}$—$C\alpha_{i-1}$—C_{i-1} and peptide bond length (for Gly, the errors in the angles involving Cβ are taken to be zero). Here error is defined to be the deviation from ideal bond lengths and angles. To account for their different units, the error in bond length (in Å) was multiplied by 150, an approximate ratio between standard deviations in the bond angles in question (in degrees) and those in peptide bond lengths (in Å).[27]

The C_{i-1} atom was rotated about the $C\alpha_i$—$C\alpha_{i-1}$ axis, and the other angles and lengths then fixed the position of N_i, so that the position yielding the minimum total squared error could be found. The process could be imagined as an axle, the peptide bond, rotating about the bases of two cones, as shown in Figure 6b. Because the error surface (i.e., backbone error as a function of angle of rotation of C_{i-1}) was found to be smooth, with one or two local minima at most, a binary search (on rotation angle γ in Figure 6b) was used to find the absolute minimum.[28] This is a much more efficient and precise process compared to trying all possible angles in increments of ten or so degrees.[29] 360 tests would be required to find the best angle with 1° accuracy stepping one degree at a time, as opposed to about 15 with the present technique. This search process is effectively a local energy minimization on the two peptide atoms with one degree-of-freedom.

If no solution could be found within a maximum backbone error tolerance (specified by the user), the current step was aborted and the program would back up one residue, choosing new random co-ordinates to try. If a "good" solution was found, O was then placed using the planarity of the peptide bond, virtual bond angle $\angle O_i$—$C\alpha_i$—$C\alpha_{i+1}$ and length O—Cα.[4] The amino hydrogen was calculated only approximately, using a typical N—H bond length (Brünger, A.T. X-PLOR v 3.851 [1992]. Yale University, New Haven, CT) and the direction opposite the sum of unit vectors in the directions N—C and N—Cα. Similarly, Hα was approximated in the direction opposite the sum of Cα—Cβ, Cα—N and Cα—C (Fig. 6c, d). Additional atoms appearing at the N- and C-termini were appended in a similar fashion.

Placement of Sidechains—Static Rotamer Dictionary

The last part of protein construction is the addition of the side chains. In our method, we have the complete backbone in place at the time when the side chain is to be added. Methods for side chain placement on protein backbones have been published, for example using backbone-dependent rotamer dictionaries such as that used by SCWRL, or by self-consistent ensemble optimization,[30,31] which optimizes a large collection of side chain conformations simultaneously, iteratively, beginning with an initial guess.

We wrote functions to generate amino acid side chains based on standard atom and bond parameters, and generated a precomputed set of standard rotamers—a static rotamer dictionary. Rotamers are built using all side chain atoms, including hydrogens, as well as the backbone N, Cα and C. These latter three atoms allow for a canonical alignment of the side chains into a rotamer dictionary to allow for easier placement later on. The static rotamers were constructed using discrete χ angles, varying in quantity and value with residue type. Hydrogens are typically added on equally spaced, in the staggered conformation where applicable.

We attached a randomly-selected standard rotamer to the growing backbone once the β-carbon position and backbone N and C are determined at that residue. We thought that this process would remove the need to perform detailed calculations at each residue to determine the positions of side chain atoms. We expected it to be efficient, using a sort of look-up table, and requiring only a simple rotation and translation for each atom from the dictionary coordinate frame.

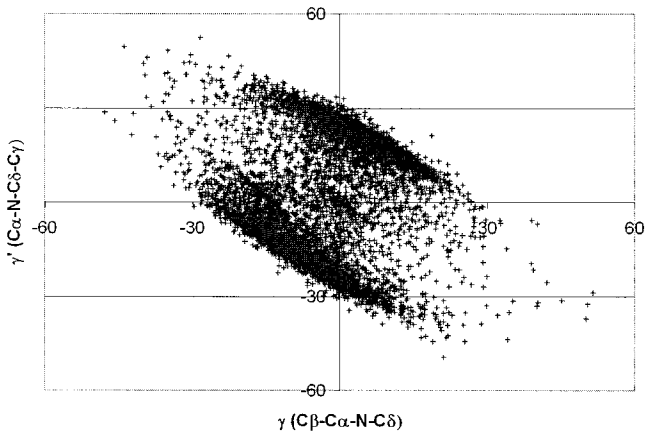

Fig. 7. Two of the five possible dihedral angles in the proline ring, enough to uniquely determine its conformation, are plotted against each other for all Pro residues in the non-redundant set of the PDB. For a given value of one angle, a valid range for the second can be easily picked out as described in the text.

Placement of Sidechains—Backbone Dependent Rotamers

Using the same rotamer generation techniques used to make our static library, we could also compute rotamers and place them on-the-fly, atom by atom, with arbitrary χ angles. We did so together with a sampling approach. The backbone-dependent rotamer library/database of Dunbrack et al.[32,33] is derived through a statistical analysis of protein side chain rotamers. We decided to use this as a sampling distribution function for the placement of rotamers on-the-fly. Once the location of Cα, N, and C at a given residue are known, along with the direction of Cβ, the remainder of the side chain can be completely defined by up to four dihedral angles (χ_1, χ_2, χ_3, χ_4) (excluding hydrogens). First the ϕ and Ψ values at the residue are computed, and used along with amino acid type to index into the backbone-dependent rotamer database, which is discretized at every 10° × 10° region of Ramachandran space. This database provides a series of rotamers, expressed as χ angles, each with an associated probability. The sum of probabilities over all rotamers in a given bin is always 1.0, so we have employed the database as a discrete probability distribution function for rotamer χ angles. Hydrogens are placed in most cases simply using standard dihedral angles, such as 180°, −60°, 60° for methyl hydrogens.

Placement of Proline Side Chains

For the special case of proline, a completely different construction method was used, as follows. First Cδ is placed using the approximate planarity of proline's imino nitrogen and a standard bond length and angle. The location of Cβ is fixed by the backbone position, so this leaves only Cγ to place. It was found empirically that given γ = dihedral(Cβ—Cα—N—Cδ) (which is now fixed), then $-0.8*\gamma < \gamma' < -0.8*\gamma + 32$ or $-0.8*\gamma - 48 < \gamma' < -0.8*\gamma - 16$, where γ' = dihedral(Cα—N—Cδ—Cγ) (see Fig. 7). Hence one of these two ranges was chosen, with 50% chance each, for γ', and the optimal value in that range found by a binary search, to minimize errors in bond lengths and interior ring angles. If a total error of more than 12.5 degrees squared was found, the other γ' range would be checked, or backtracking would occur. This method results in good proline models, except that they sometimes have unusual puckering phases.[34,35]

RESULTS

Input to the INITTRAJ program takes the form of a simple text or FASTA-formatted file containing the sequence to be folded, using the one-letter amino acid abbreviations. INITTRAJ generates the trajectory distribution database and packages it into the ASN.1 input file used by FOLDTRAJ. The running time of INITTRAJ scales linearly with sequence length and so is simply $O(N)$. Default configuration parameters are read in each time the program is executed as well, overridden by parameters passed in an optional configuration file.

As the method proceeds, coordinate values are added to the chemical graph data structure. Output from the program is a protein structure in NCBI's MMDB ASN.1 format, or a PDB file. In testing, the method works with even the largest of eukaryotic proteins, for example a random conformer of BRCA2 (> 3,000 amino acids) completes in about an hour and uses only 40MB of RAM on a 400 MHz Pentium II processor running the Linux operating system.

Our system has a few parameters that may be adjusted, and whose values will affect both the speed of the program, and the resulting standard deviations in bond angles, lengths, and improper dihedrals in the structure. These parameters include the average number of tries to place the (i + 1)th residue before backtracking, the maximum squared error tolerance permitted without rejecting a solution for the backbone atoms (as described in Methods) as poor, and a parameter for "atom bounciness" with zero corresponding to hard atoms and positive values soft atoms. Increasing the error tolerance will speed up the program but generate poorer structures, evenly distributed over all of Ramachandran space. Decreasing the error tolerance may prevent any structure from being found, and will result in an unrealistically "perfect" protein. A value intermediate to these extremes, 50 degrees squared cutoff, was used. A value of 100 retries on average before backtracking at each residue was chosen, generating structures quickly while rarely backtracking more than 2–3 residues at a time. For soft atom collisions, an "atom bounciness" of 0.25 was used as a very crude approximation to a Lennard-Jones type potential function.

Testing With Myoglobin

To see more clearly the effect of collision testing and optimizing backbone angles, the database sequence 111M (sperm whale myoglobin, 154 amino acids) was randomly generated 250 times.

With checking for hard-sphere collisions only between Cαs and no backbone atom optimization, each run took an average of 2 seconds (on a Pentium II-300). An average of 179 residue trial placements per run were needed (a trial

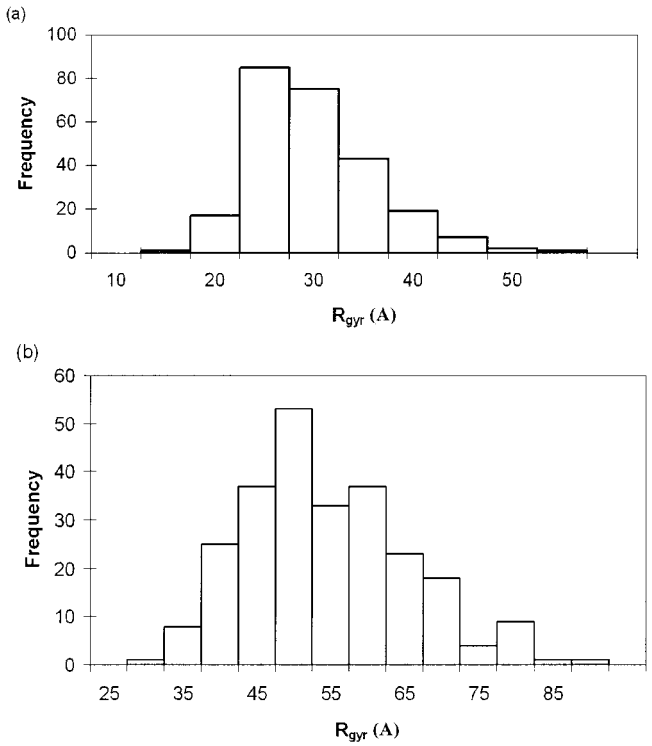

Fig. 8. Distribution of radius of gyration (in Å) for 250 randomly generated myoglobin structures, without **(a)** and with **(b)** full collision testing and backbone optimization. The distributions appear to be skewed to the right, so that the peak value is slightly less than the mean.

placement or "try" is defined as an attempt to place an amino acid, whether successful or not). In this test case, the only way a placement could fail was by Cα collision. The mean radius of gyration (R_{gyr}) was 27.3 ± 6.2 Å (compared to 15.19 Å in the folded holo-protein) and end-to-end distance (R_{NC}) was 66 ± 26 Å (19.45 Å in the native holo-protein). With complete collision testing and backbone checking, each protein took an average of 155 seconds, or about 51,000 tries, and resulted in a radius of gyration of 52.0 ± 11.1 Å and end-to-end distance of 123 ± 47 Å. A histogram of R_{gyr} in both cases (Fig. 8) shows a narrow spread, skewed to the right for the Cα-only walk, and a broader right-skewed shape for the all-atom walk. As expected, collision testing and backbone optimization swell the protein to roughly double its size from when non-Cα atoms are placed arbitrarily (or equivalently, atoms are assumed to have zero radius except Cα, which was given a diameter of 3.81 Å, the distance between consecutive Cαs). Also, many more tries are required until valid atom placements can be found. Steric hindrance greatly reduces the area of conformational space open to the protein.

Further tests with myoglobin were used to compare the effects of other parameters on the performance of the method, with results summarized in Table III. From this we can see, for example, that the use of soft atoms, as opposed to hard atoms, reduces the average time by about 40 seconds, results in many less crashes and backbone errors occuring, and produces significantly more compact structures, as may be expected. The use of flat rather than amino-acid based trajectory distributions slows down the execution time a little since more time is wasted trying to place atoms in invalid locations, but in the end results in structures of similar compactness, except when combined with variable backtracking in which case they are found to be slightly more compact. Other effects are discussed in the following sections.

Rotamer Placement

We used two different methods for the placement of side chain rotamers. The first method involved picking a set of rotamer coordinates for the side chain from a pre-computed set of static rotamers, selected at random in a backbone-independent manner. We expected this method to be more efficient. The second method involved sampling rotamer angle values from Dunbrack's backbone-dependent rotamer library,[33] then using these χ angles to generate the selected rotamer on-the-fly. Testing showed that the latter method was similar in speed, only 4 seconds slower on average, and produced slightly more compact structures ($p < 0.05$), for the myoglobin structure population test (Table III). It was superior in the quality of side chain placement, as determined by WHAT_CHECK (WHATIF, version 19970704-1848).[35] While the latter method may involve slightly more floating point computations, the time complexity was equalized primarily because the backbone-dependence information biases sampling to avoid obvious backbone-side-chain clashes. The static rotamer placement method ended up discarding many rotamers that clashed with the backbone, apparently because there was no information in that method that prevented the placement of a rotamer that clashed with the backbone. The on-the-fly placement method is of course far more powerful than using a static library, allowing placement of rotamers in arbitrary conformations to allow for better packing when trying to construct compact, folded structures.

Backtracking

Backtracking in a build-up method adds yet another dimension to the time complexity of the algorithm. We used the setting of 100 attempts to place a residue before backtracking, which seemed to work well in most cases. However, in some topologies, a combinatorial problem can arise if the only recourse for the build-up method is to backtrack many residues (e.g., backtracking N residues could require up to 100^N failed placements!). The kind of topology that causes this problem is a region of low conformational freedom (e.g., a helix) building up along a tube or channel with a dead-end, not uncommon in a folded structure. The only way out is to backtrack to the most recent turn.

We know that backtracking one step only allows us to resample the same trajectory distribution. If that trajectory distribution is shallow (e.g., Gly), chances are we will get a very different residue placement, but if the trajectory distribution is deep and spiked (e.g., Lys), we may keep sampling the same amino acid placement again and again. We used this information, the depth of the trajectory

TABLE III. Effects of Various Simulation Conditions on Randomly Generated Structures[†]

AA-based	Variable back-tracking	Bbdep. rotamers	Soft atoms	Time (s)[a]	Tries[b]	Bad backbone[c]	Crashes[d]	R^{e}_{gyr}	R^{f}_{NC}	R^{g}_{n}	C^{h}_{n}	C_n/R_n	N
●	●	●	●	94 ± 42	44830 ± 21257	30400 ± 10352	13928 ± 12115	39.2 ± 8.3	90.4 ± 36.1	0.72 ± 0.31	4.2 ± 3.1	5.5 ± 2.8	300
●	○	●	●	122 ± 51	35637 ± 14064	25141 ± 7581	10012 ± 7109	39.4 ± 8.8	91.2 ± 36.9	0.73 ± 0.33	4.3 ± 3.5	5.5 ± 2.8	300
●	○	○	●	118 ± 62	31889 ± 15434	22351 ± 7021	9093 ± 9207	41.1 ± 9.8	97.6 ± 41.6	0.80 ± 0.40	5.0 ± 4.1	5.9 ± 3.0	300
●	○	●	○	166 ± 59	59762 ± 18603	41853 ± 11133	17187 ± 7935	47.5 ± 11.0	107.3 ± 45.9	1.07 ± 0.50	6.1 ± 4.8	5.3 ± 2.8	300
●	○	○	○	155 ± 56	51140 ± 17355	34331 ± 8936	16175 ± 8830	52.0 ± 11.1	122.9 ± 46.9	1.27 ± 0.55	7.8 ± 5.8	5.8 ± 2.9	250
○	●	●	●	120 ± 28	115061 ± 26531	72447 ± 12517	42032 ± 15421	36.5 ± 8.1	84.6 ± 34.7	0.63 ± 0.29	3.7 ± 2.9	5.6 ± 2.9	300
○	○	●	●	128 ± 39	62712 ± 11092	41722 ± 6856	20252 ± 4533	39.9 ± 8.8	92.2 ± 36.5	0.75 ± 0.34	4.4 ± 3.3	5.5 ± 2.8	300

[†]All values given are mean ± s.d. over N structures. As indicated, either AA-based (●) or completely flat (○) trajectory distributions were used, and the number of attempts before backtracking was either dependent on the shallowness of each trajectory distribution (●), or not (○) (and fixed at 100). Rotamers were built on-the-fly using the backbone dependent rotamer library (●) or placed statically from a pre-built rotamer dictionary (○) allowing only three possible values for each χ angle. Either soft (●) or hard (○) atom collisions were simulated. 400 × 400 trajectory distributions were used in all of the above runs.

[a]From placement of first atom to placement of final atom.

[b]One try is an attempt to place a residue, whether successful or not, and is the sum of Bad Backbone errors, Crashes, and successful placements; number of successful placements is usually greater than the length of the protein due to backtracking.

[c]Number of tries that failed because no placement of the backbone N and C could be found within the specified backbone error tolerance.

[d]Number of tries which failed due to van der Waals collisions.

[e]Radius of gyration (Å).

[f]Distance from Cα of first residue to Cα of last residue (Å).

[g]R^2_{gyr}/rl^2, where l = 3.81 Å and r = # residues.

[h]R^2_{NC}/rl^2, where l = 3.81 Å and r = # residues.

distribution, to adjust the number of backtracking attempts at each residue. Specifically, the following formula seemed to work well empirically for expressing the amount of "information" in the trajectory distribution:

$$i(t) = 400*(6\varphi_t(0) + 2\varphi_t(5) - \varphi_t(10) - 3\varphi_t(15)) \qquad (2)$$

where $i(t)$ is the number of tries to make at a given residue with trajectory distribution t before backtracking, and $\varphi_t(x)$ is the fraction of trajectory distribution space t which lies above x% of the peak value. For example, $\varphi_t(100) = 0$ always, since nothing lies above the peak, and $\varphi_t(0)$ gives the fraction of t which is non-zero. If Eq. 2 results in a value outside the range 4–250 it is truncated to the appropriate boundary value, to ensure the $i(t)$ stays reasonable regardless of trajectory distribution complexity. For sharply peaked distributions, Eq. 2 results in an $i(t)$ close to zero, meaning minimal tries should be made at that residue, while for broader distributions with "hills and valleys," larger numbers on the order of 100 result. This change was successful at speeding up the overall algorithm (Table III), and afforded a speed-up of 28 seconds to the mean times of computation for a random myoglobin with no change in R_{gyr}. This kind of "smart backtracking" avoids futile residue placements without affecting the characteristics of the structure.

Trajectory Distributions From Database Frequencies

A trajectory distribution grid size of 400 × 400 provides 0.9° resolution in θ and resolution in ϕ between 0.29° (near the equator) and 4.1° (near the poles, but 0.5° when $\phi > 16°$, as is the case in all real proteins). Changing the grid resolution of the trajectory distributions affects speed and memory usage of the program. The grid resolution must be fine enough to provide the desired angular resolution between consecutive Cαs.

The "trajectory" from one α-carbon to the next completely determines secondary structure, and it was found that helices corresponded approximately to $\theta = 51°$, $\phi = 90°$ while β-sheets had $\theta = -165°$, $\phi = 60°$. This compares well with Gregoret and Cohen[21] who found $\theta = 50°$, $\phi = 88°$ for helix and $\theta = -160°$, $\phi = 60°$ for sheet. The improvement in our implementation comes from choosing from a non-uniform, discrete PDF in cosϕ–θ space. We are able to bias the selection of conformations at each residue, without introducing significant biases due to the curvature of the spherical surface (which would be the case had ϕ–θ space been used).

Quality Validation

Sequences from actual proteins of various folds and lengths under 250 residues were generated, with coordinates being calculated for all atoms, and the resulting structures submitted to WHAT_CHECK for analysis. All test proteins passed the nomenclature, chirality/planarity, improper dihedral, peptide bond planarity, carbonyl O placement, bond length and bond angle tests. This confirms that the coordinates of the protein atoms are physically and chemically valid, aside from proline occasionally having an unusual puckering phase.[34] The number of interatomic clashes (using the hard atom model) was not significantly larger than that for any crystal structure, the majority occuring between the carbonyl O of one residue and the carbonyl C of the next, and due to the difference in our choice of Van der Waals radii from that of WHAT_CHECK. Packing tests were not very good, as would be expected for a random protein conformer. Many buried unsatisfied hydrogen bond donors were located as well, which is not surprising since only steric clashes have been considered to this point.

Secondary Structure Content

The biases introduced by the use of amino-acid-based trajectory distributions were investigated by quantifying amounts of secondary structure in the generated random structures. To this end, 1,000 structures were generated for each of 1SEM (an all-β SH3 domain), 2HPR (phosphocarrier protein, an α/β open sandwich) and 1RTP (all α, α-parvalbumin), using both amino acid-based trajectory distributions and uniform (equiprobable) distributions. DSSP[36] was used to detect helical content (at least four consecutive residues with at least one hydrogen bond) but is not capable of detecting extended structure per se. Hydrogen-bonded parallel and anti-parallel sheets were virtually non-existent (< 0.1% according to DSSP) due to the lack of tight tertiary structure in most of the random structures, so β-strand was detected simply by looking for windows of five consecutive α-carbons spanning 13.25 Å or more.

The secondary structure content is shown in Figure 9, which shows that the main effect of the amino-acid-based bias is to add helix to the structures, with the largest amount (13.3% helix, 20.1% extended) in the natively helical protein (1RTP) and the least (6.6% helix, 22.0% extended) in the SH3 domain (1SEM). The mixed structure (2HPR) was intermediate to these two (9.8% helix, 22.3% extended). Helices are essentially non-existent when uniform trajectory distributions are used (< 0.2% in each case). An increase in the amount of strand produced is evident as well for the amino-acid-based structures. 1SEM, 2HPR, and 1RTP generated from uniform trajectory distributions had 14.6%, 15.1%, and 16.4% extended structure respectively.

Time Complexity of the Algorithm

Random sequences with lengths varying from 10 to 1,000 amino acids, in increments of 5, were generated to estimate the time complexity of the algorithm used for constructing random proteins. Amino-acid based trajectory distributions were used, along with the hard atom model, a fixed backtracking count of 100 tries and static rotamer placement. Five sequences of each length were created as follows. A count was made of the frequency of amino acid occurence in the entire *E. coli* genome (a "typical" assortment of proteins) (A: 9.49%; C: 1.17%; D: 5.14%; E: 5.74%; F: 3.90%; G: 7.37%; H: 2.27%; I: 6.00%; K: 4.41%; L: 10.64%; M: 2.85%; N: 3.95%; P: 4.43%; Q: 4.43%; R: 5.54%; S: 5.82%; T: 5.41%; V: 7.06%; W: 1.53%; Y: 2.85%), and for each sequence, each amino acid was chosen randomly according to this distribution. The overall distri-

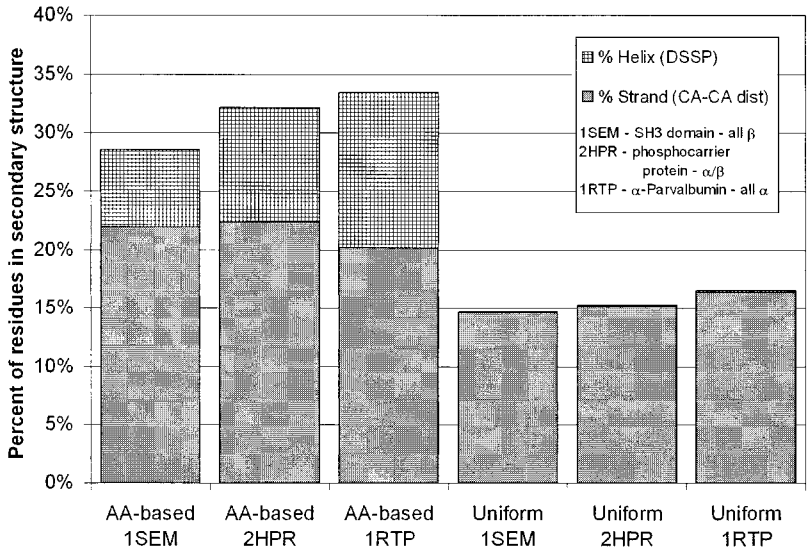

Fig. 9. Average secondary structure content in 1000 random conformers of the proteins 1SEM, 2HPR and 1RTP generated using both uniform (i.e. flat) trajectory distributions and AA-based ones (as indicated). Helices were all four or greater residues, and identified with DSSP, while β-strand was identified as any stretch of five or more α-carbons in which the separation from $C\alpha_i$ to $C\alpha_{i+4}$ was greater than 13.25 Å for all possible (i, i + 4) windows in the interval. This cutoff was chosen to agree well with PDB secondary structure assignments for real proteins with significant β-sheet content.

bution of residues in the resulting 995 proteins was then calculated (in much the same way as it was for *E. coli*) and the composition of each residue was within 0.1% of that in the sampling distribution. The time taken to generate each structure, R_{gyr} and R_{NC} (the N to C terminus separation) were all recorded as well. Time taken was then plotted against both N and $NlogN$ (Fig. 10), where N is the sequence length in residues, and was fitted to an expression of the form $y = ax^b$, with t in seconds on the y axis and N or $NlogN$ on the x axis. In the former case, $a = 0.38$, $b = 1.18$ gave the best fit ($R^2 = 0.95$), while in the latter, $a = 0.49$, $b = 0.99$ did ($R^2 = 0.95$). This indicates that the overall structure generation is $O(NlogN)$ and more specifically, to generate a random protein of length N will take approximately $0.49Nlog_{10}N$ seconds on a PentiumII-300 running Linux. Very similar results were observed on a Silicon Graphics Origin 200 system using a single 180MHz R10000 CPU.

The number of tries, T (attempts to place a residue, whether successful or not) was plotted against N. The resulting fit to $T = aN^b$ gave $a = 530$, $b = 0.90$ ($R^2 = 0.99$) indicating that roughly 530 tries were made before placement of each residue was successful. Backtracking in the random walk occurred after 100 failed attempts to move forward. Thus on average, about five different $C\alpha_{i-1}$ placements were tried at each residue, before a solution for the backbone co-ordinates between it and $C\alpha_i$ could be found (i.e., a solution with no clashes, and valid angles). The large number of failed tries is due to either collisions with existing residues (about 25–30%) or, more often, due to the lack of a solution for the peptide atoms within the desired backbone tolerance level (about 70–75%).

R_{gyr} and R_{NC} were also plotted vs. N (Fig. 11). The radius of gyration was found to fit well to $2.84N^{0.57}$ ($R^2 = 0.82$). Similarly, R_{NC} fit well to $7.72N^{0.54}$ ($R^2 = 0.54$). The mean value of C_n/R_n, the characteristic ratio and radius of gyration ratio, $C_n = R_{\mathrm{NC}}/Nl^2$ and $R_n = R_{\mathrm{gyr}}/Nl^2$ respectively, was found to be 6.03, close to the theoretical value of 6.0 for large polymers.[37]

Reconstruction of PDB Structures

If the present algorithm is successful in reconstructing known proteins, then the random proteins it generated would also most likely be valid, possible structures for given sequences. In order to test the ability of the algorithm described above to reconstruct the backbone atoms (namely C, N, Cβ, and O), several known proteins were reconstructed by recording their "trajectory distributions" (see Methods) from the following PDB files: 111M, 1OMD, 1SEM, 3TS1, 4PTI, 2PCY, 2MHR, 2LYM, and 1TIM. This is equivalent to recording the Cα co-ordinates except without Cα—Cα distances; only the direction from each to the next is known. All atom positions were generated completely randomly as before, the only change being that the trajectory distributions used were delta functions, with their peaks at the precise orientation of a given residue in the crystal or NMR structure. No reconstruction took over one minute, and most less than 10 seconds.

Structures were generated with a backtracking retry value of 10 rather than 100, since very little could change between these retries. Also, bump checking was turned off since side chains were being placed (but not used in calculations) and were expected to collide since they were being placed randomly. Bump checking was not expected to improve the quality of the backbone since the whole shape of the protein was approximately held rigid, but some backbone-backbone clashes occurred. The simplifications the algorithm makes, for example constant NCα and CCα bond lengths, would not allow certain residue back-

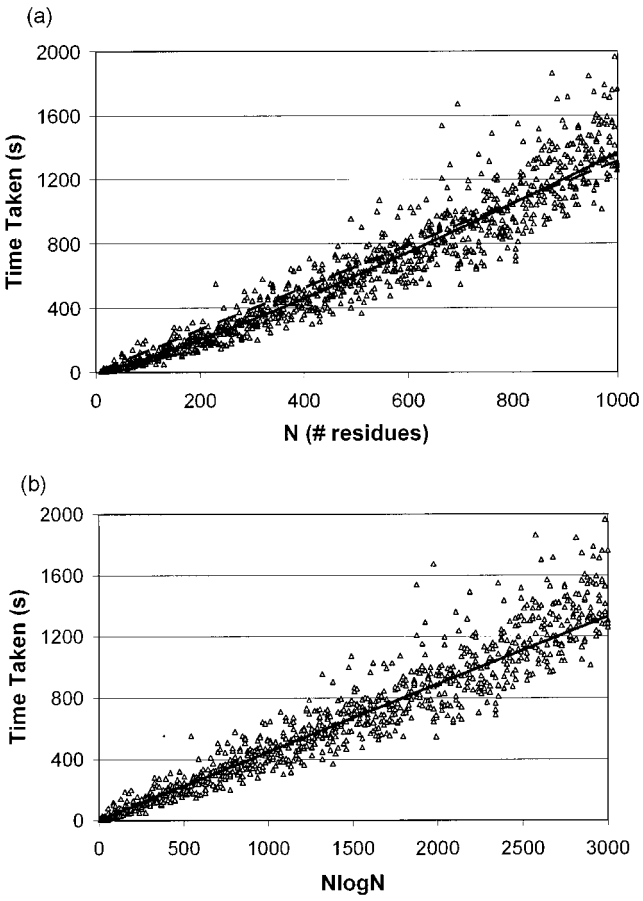

Fig. 10. Time taken to generate a series of 995 random structures as a function of sequence length, N **(a)** and NlogN **(b)**. Each triangle corresponds to a single randomly generated protein and the dark solid line is the line of best fit, in each case of the form $t = aN^b$ with a and b the adjustable parameters. The dark dashed line in (a) is a linear best fit, whereas in (b) the solid line of best fit is already linear.

bones in real proteins to be reconstructed without a few backbone-backbone clashes.

Depending on the structure, the backbone error tolerance had to be increased to various levels to allow each protein to be built. Usually one or two residues per structure required relatively large errors in some of their angles in order to find any valid solution at that residue. This was most often due to an incorrect Cβ placement at that point. These occurred at turns coming out of helices or sheets and leading into new ones. Some of the proteins were chosen with different types of ligands attached (the heme ring in 111M, calcium bound to 1OMD, tyrosinyl-5′-adenylate in 3TS1 and a small peptide in 1SEM) to observe the quality of reconstruction in such instances. Some of the observed Cβ distortions occur at or near the sites of ligand attachment since liganding can alter the residue conformation. A slightly better structure may have been obtained if different tolerances were used at different locations, but the "worst case" error was used as the cutoff for all residues.

To compare the reconstructed proteins with their crystal structures, the coordinate RMSD of various atoms and atom groups were determined and summarized in Table IV. There is no obvious relationship between length or resolution and quality of the reconstruction. All were fairly well reconstructed, with the worst case being 2PCY, having poor N and C placement. This is at least partially due to the fact that every ten or so residues, a Cβ is very poorly placed (> 0.5 Å in space away) in this structure. The latter five proteins of Table IV were reconstructed by Rey and Skolnick[4] using their algorithm and their final values for Cβ, C, N, and O RMSDs are shown as well for comparison.

The square root of the backbone tolerance mentioned earlier is the cutoff of the backbone error for each residue, and is shown in Table IV. Also shown is the resulting average backbone error for the reconstructed protein, and that for the PDB structure. These latter values can be thought of as an angular backbone RMS error (RMSbb error) of the recorded structure from a structure with "ideal" bond lengths and bond angles at each residue, even if such angles could not all simultaneously exist in space. This should not be confused with atomic coordinate RMSD.

Generally, the closer the reconstructed RMSbb error was to the crystal structure RMSbb error, the better the reconstruction. Note that in all cases, the reconstructed RMSbb error was significantly less than the cutoff, indicating that only a few residues actually exhibited deviations as large as this cutoff value and most were much smaller.

Each reconstructed protein was then run through WHAT_CHECK for analysis. Results were generally poorer than those from *de novo* generated peptides, due to the added constraints and increased backbone error cutoff (which is normally set to about 7 degrees). Nevertheless, all except 2PCY and 2MHR had normal bond length variability and bond angle variability, despite a few extreme values. In most cases, these occurred at or near locations where a Cβ had been grossly misplaced. Several structures had high improper dihedral variability, usually at or near the same sites where bond angles or lengths were poor. Generally the backbone at 80% or more of the residues was quite good. Two structures, with areas of poor reconstruction highlighted, are shown in Figure 12.

Comparison to Experiment

Earlier studies by Haas' group[38,39] looked at the distance distributions from N-terminus to fluorescently labelled lysine residues of bovine pancreatic trypsin inhibitor (BPTI), using fluorescence resonance energy transfer (FRET). The experiments were done under denaturing and reducing conditions, first in a mixture of guanadine hydrochloride and glycerol at −30°C, and later without the glycerol at 20°C and 35°C. Their distributions were fit to skewed Gaussian functions, of the form $f(x) \propto x^2 * e^{-\alpha(x-b)^2}$ with x the distance in question and $f(x)$ the frequency of occurrence. To compare, we have generated 5,000 structures of BPTI using each of the trajectory distribution methods as described above: uniform and amino-acid based. Distances between the α-carbon of the N-terminal residue and the Nζ of the lysine sidechain on each of residues 15, 26, 41, and 46 were recorded, the same places fluorophores were attached in the described experiments. All distance distributions fit well to the skewed Gaussian form, with $R^2 > 0.98$.

The results are summarized in Table V where they are

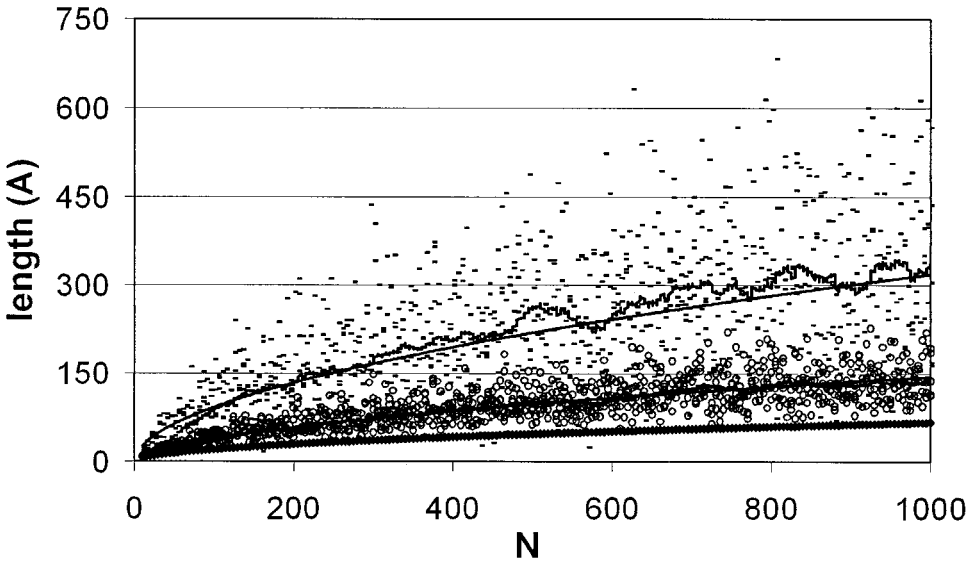

Fig. 11. Plots of radius of gyration (○) and end-to-end distance (−) against protein length for the same 995 structures in figure 10. A moving average for each (window size 50) is shown as a solid, jagged curve, as well as the curve of best fit (smooth solid curve). The thick line below all the points is the theoretical minimum radius of gyration, $2.106N^{1/2}$ (see Discussion).

TABLE IV. Using Only the Cα Trajectories (i.e. φ, θ at each residue) of Nine Proteins With Known Folds, the Remaining Backbone Atoms Were Reconstructed†

Protein	Number of residues	Resolution (Å)	Structural class	RMSbb error cutoff	Reconstructed RMSbb error[a]	Crystal structure RMSbb error[a]	Cα RMSD (Å)	C, N RMSD (Å)	Cβ[b] RMSD (Å)	O RMSD (Å)	Cβ, C, N, O RMSD (Å)	R & S Total RMSD (Å)
111M	154	1.88	α	13.4°	7.2°	2.6°	0.10	0.310	0.242	0.834	0.486	—
1OMD	107	1.85	α	15.5°	8.3°	6.0°	0.09	0.249	0.274	0.735	0.411	—
1SEM	58	2.0	β	22.6°	9.9°	3.9°	0.18	0.409	0.375	1.089	0.644	—
3TS1[c]	211	2.7	α/β	13.8°	8.2°	5.6°	0.17	0.347	0.266	1.012	0.578	—
4PTI	58	1.5	α + β	13.8°	7.7°	5.0°	0.12	0.383	0.360	0.915	0.561	0.626
2PCY	99	1.8	β	30.8°	11.2°	4.7°	0.59	0.707	0.404	1.342	0.861	0.725
2MHR	118	1.7	α	22.6°	9.9°	5.0°	0.44	0.584	0.350	1.226	0.760	0.711
2LYM	129	2.0	α + β	14.5°	8.3°	4.5°	0.24	0.407	0.275	1.204	0.681	0.765
1TIM	247	2.5	α/β	28.5°	10.1°	5.0°	0.23	0.410	0.403	1.213	0.702	0.626

†The RMSD co-ordinate deviations of various atom groups between the crystal and reconstructed structures are summarized here, and compare well with Rey and Skolnick's reconstruction technique[4] (last column). For the last four proteins, high backbone error tolerances were required in order to successfully build the protein with no atomic clashes occuring, most likely due to the occasional incorrect placement of Cβ atom.
[a]Excludes contributions at Gly residues.
[b]Estimated by comparing crystal structure to "ideal" look-up table Cβ locations.
[c]Only the first 211 residues were modelled since certain residues are missing beyond this point.

compared with those found by Haas and co-workers. The agreement is good, when the variation in their own data at different temperatures and different conditions is taken into account. The amino-acid based structures have distances about 4 Å greater than the uniform ones, and although the peak for the (1–46) distance appears much greater (55 Å vs. 45 Å for uniform) the distribution is very flat here, making it difficult to choose an exact peak value. Both methods explore similar amounts of conformational space. Our peak values are slightly higher than those found by Gottfried and Haas using a one-component model, though within one standard deviation, except for (1–26). However, they also fit their data to a two-component model, with the greater peaks being 45, 50 or 55, and 45 for (1–26), (1–41), and (1–46) respectively, very close to our peak values in Table V. The full width at half maximum (FWHM) is comparable to Amir et al. for (1–15) meaning a similar amount of conformational space is explored, but it is difficult to compare for (1–26) since they only fit the fluorescence lifetime data to a two-component discrete system and not a continuous model. Their data is thus constricted to appear as two distinct sub-populations. Our observed distribution appeared more like a single Gaussian with a "hump" on it (not shown), indicative of overlap of two or more Gaussian-like functions. Our results indicate a complex superposition of Gaussians for the (1–41) and (1–46) distance distributions, not representable by a simple one- or two-component model. Gottfried and Haas mention that "the model used is not a perfect representation of the real system," so any comparisons made here should be taken lightly.

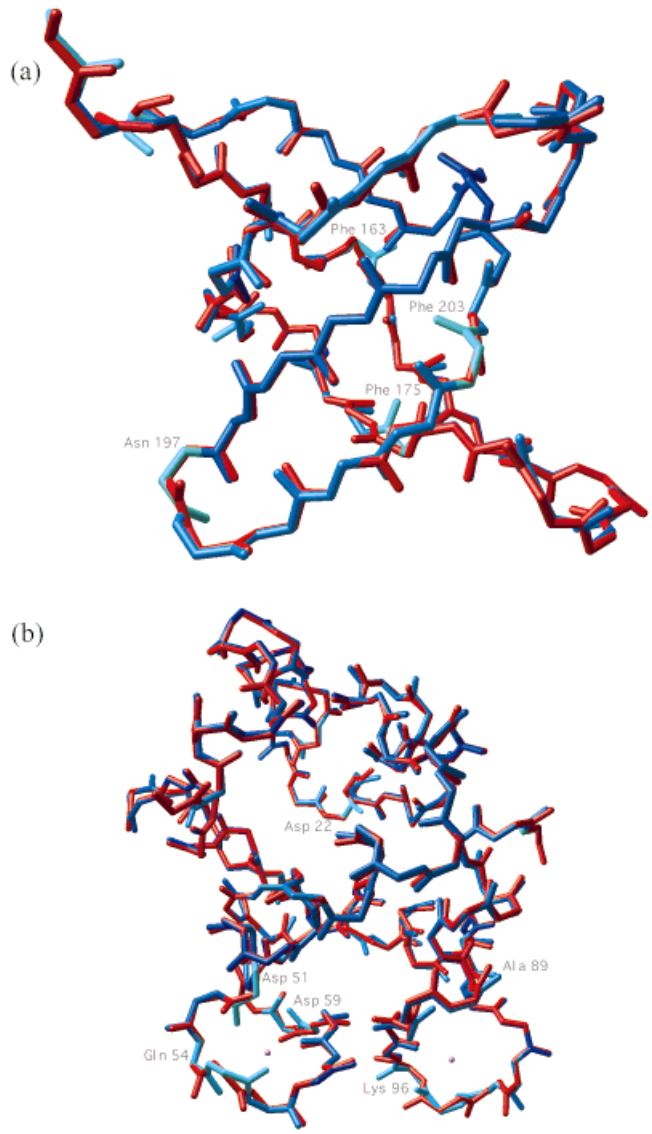

Fig. 12. Reconstructions of PDB structures **(a)** 1SEM, an SH3 domain and **(b)** 1OMD, the calcium-binding protein oncomodulin. The crystal structure backbones are shown in red, and the reconstructed ones in blue. Areas in which the reconstruction was particularly poor are highlighted in green, and in some cases, the offending residues labeled. 3-D alignment was obtained solely based on Cα co-ordinates. The reconstructions appear quite good except at a few residues in each protein, either at sharp turns or where ligands are normally attached. A poor Cβ placement (> 0.5 Å error) occurs at or adjacent to each labeled residue.

DISCUSSION AND CONCLUSIONS

A method was developed that is able to generate full-atom instantiations of protein structures of arbitrary sequence and length, in $O(NlogN)$ time, owing primarily to the efficient collision-testing provided by the binary-d tree, but also to the binary search methods used in refining the backbone atoms. These structures are for the most part physically and chemically valid, and could represent possible conformations of proteins in the unfolded or denatured state, for example in 8M urea. The proteins were found to agree well with Flory polymer theory, namely that the radius of gyration varied with the square root of length, as did the end to end distance. In contrast to other random walk protein generators, no attempt was made to discretize space on a lattice, and the growing chain was free to wander wherever it would fit. The quality of the protein reconstructions testify to the method's soundness.

Our method has been shown to work well at both ends of the hypothetical folding funnel: at the "tip," reconstruction of known, folded proteins; and at the "mouth," generation of random, denatured protein structures. A large body of theory on general random walks and polymers[40,41] has investigated the relationships between polymer length and 3-D shape, and is useful for comparison to these randomly generated structures. For a polymer generated via random walk, with N atoms, bond angle θ and bond length 1,

$$\frac{R_{gyr}^2}{Nl^2} = \frac{1 - \cos\theta}{6(1 + \cos\theta)} \tag{3}$$

so that for a protein with $l = 3.81$ Å and $\theta \approx 107°$, $R_{gyr}^2 = 4.419$ N is expected. However, since a kinetic self-avoiding walk was used in this study, the walk is statistically biased, increasing the expected size. Also, since entire backbones and side chains are being added on and tested for collisions, rather than just testing for collisions between α-carbons (as theory would have), even larger values are expected. Using $N = 154$ this gives an expected R_{gyr} of 26.1 Å for random myoglobin. Hence the observed average of 27.3 ± 6.2 Å when collisions and backbone optimization was ignored agrees with this. The distributions of R_{gyr} shown in Figure 8 illustrate the swelling of the distribution, as well as the mean, when full bump checking is used compared with the relatively narrow distribution achieved when only Cα collisions can take place and all residues are treated as spheres of constant radius. It is also interesting to note that in both cases, $(R_{NC}/R_{gyr})^2$ is close to the expected limit of 6 as N approaches infinity. The theoretical equation for R_{gyr} served as a lower bound for the observed values in the 995 randomly generated proteins as well, as can be seen in Figure 11. The fitted exponent for the R_{gyr} dependence on N of 0.57 is close to the value of 0.6 predicted by theory for a kinetic self-avoiding walk in 3 dimensions.[42]

By using uniformly flat trajectory distributions, avoiding collisions, and ensuring accurate bond lengths and bond angles, all residues in the resulting randomly generated proteins except Gly fall within the allowable regions of Ramachandran space (not shown).[20] The observed discrimination is a result of steric clashes and possibly Cβ placement (which was used to place C and N) since the lookup table used contains some inherent secondary structure bias. Only certain areas of Φ–Ψ space are populated because the bond length and bond angle restraints, and Van der Waals repulsions physically forbid certain conformations of the backbone. It has been known for some time[37] that by simply applying steric restraints, and implicitly fixing bond angles and bond lengths, the acceptable areas of Ramachandran space correspond well to areas of low rotational potential energy.

As the random walk proceeds, the points chosen will

TABLE V. Distance Distributions in BPTI in Randomly Generated and Experimentally Denatured Protein

Distance reported[a]	RMS distance[b]/peak (uniform) (Å)	RMS distance[b]/peak (AA-based) (Å)	Experimental[d] RMS distance (Å)	Experimental[e] peak at 20°C (Å)	Experimental[e] peak at 35°C (Å)	FWHM[b] (uniform) (Å)	FWHM[b] (AA-based) (Å)	Experimental FWHM[d] (Å)
(1–15)	27.0	30.2	32 ± 4	26.0 ± 6.0	24.5 ± 5.0	18	20	25 ± 6
	30	32						
(1–26)	37.3	42.2	23 ± 3[c]	34.8 ± 7.3	29.5 ± 7.8	33	33	10 ± 4[c]
	44	45	55 ± 15					40 ± 20
(1–41)	47.1	51.9		39.8 ± 13.4	40.0 ± 8.7			
	46	47						
(1–46)	50.1	55.2		39.5 − 43.6	33.3 ± 10.9			
	45	55		± 12.3 − 20.0				

[a]We report the distance from N-terminal α-carbon to Nζ of Lys-15 or Lys-26, while experimental values are the distance from donor to acceptor probes attached respectively to the N-terminus and ε-amino group on the Lys residue in question.
[b]Results over 5,000 structures generated with FOLDTRAJ starting with Uniform or AA-based trajectory distributions as indicated. RMS value of distribution is on top, and value at which peak occurs is given below this.
[c]Amir et al.[38] fit the (1–26) data as two distinct skewed Gaussians, so the RMS and FWHM of each is given.
[d]from Amir et al.[38] in 26% (v/v) glycerol, 6M guanadine hydrochloride, reducing conditions, −30°C.
[e]From Gottfried and Haas[39] in 6 M guanadine hydrochloride, reducing conditions. Values are given ± 1 standard deviation; the authors do not provide an exact value for (1–46) but rather a range, as duplicated above in the table.

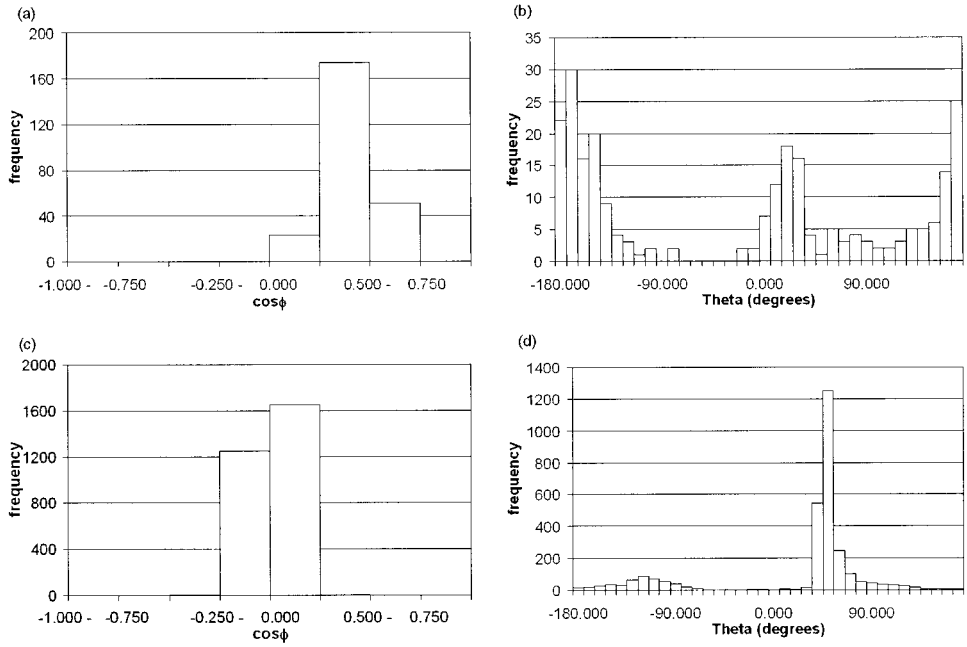

Fig. 13. Relation between trajectory space and Ramachandran space. **(a)** and **(b)** show the distribution for Ala of cosϕ and θ, respectively in the region around Φ = −150°, Ψ = 90° which is predominantly β-sheet. cosϕ is very sharply peaked at about 0.4, while θ has two peaks, one near 25° and another, larger one near −165°. **(c)** and **(d)** show the area near Φ = −30°, Ψ = −30°, the α-helical region of Ramachandran space. The cosϕ peak has shifted to about 0 and only one sharp peak occurs for θ at about 55°, with a second broad, small peak at −105°. Similar results were observed for other residues (not shown), and each region of Ramachandran space had a corresponding set of peaks in trajectory space (cosϕ − θ space).

have some effect on the conformation of that section of the backbone (i.e., helical, sheet, turn, etc.) and so a relation between trajectory co-ordinates (ϕ, θ) at a given residue, and those of Ramachandran space, (Φ, Ψ), was tested for and confirmed. The relations are complex, but could be expressed empirically in a large transformation table. In fact since θ involves four consecutive residues, two (Φ, Ψ) pairs are needed to completely specify it. Generally, in a given region of Ramachandran space, the distribution of cosϕ (and hence ϕ) was sharply peaked while θ had one or two preferred values (Fig. 13), which indicates up to two favored locations for $C\alpha_{i+1}$ related by a rotation about the $C\alpha_{i-1}C\alpha_i$ vector. Each point in Ramachandran space corresponded to a very narrow range of values of ϕ and θ. The existence of such a mapping means that the trajectory distributions implicitly contain information about each residue's preferences for the various forms of secondary structure.

There is clearly enough information in the amino-acid based trajectory graphs to spontaneously form properly hydrogen-bonded α-helices, and the secondary structure content attained is comparable to that of Gregoret and Cohen[21] for their native-like random walks (they found about 13% in helices and 22% strand, comparable to what

we found for AA-based structures in Figure 9). The intrinsic propensity of the residues in the protein to form helices results in twice as much helix on average in 1RTP, the protein which folds into an all-helical structure, compared with 1SEM, which has no helix in its native state. This observed bias must be a result of the bias towards helix of certain residues in the non-redundant set of proteins used to generate the trajectory distributions. Greater secondary structure content, on the order of that found in native proteins, can be generated by further biasing input trajectory distributions based on secondary structure predictions, such as that of the GOR method,[43,44] and will be reported in future work.

We have shown that using amino-acid based trajectory distributions, we are able to rapidly generate ensembles of structures which may serve as a plausible model for the denatured state of the protein, although two distances in one protein is certainly not enough to fully substatiate this claim; more comparisons with NMR data are underway. We can conclude that similar amounts of conformational space are explored in our ensembles as are by denatured BPTI in solution, as evidenced by the similar FWHMs and curve forms obtained.

Trajectory distributions serve as an extremely flexible input for the protein conformer generator, acting as a map of conformational space. This allows for future expansion of the method by simply changing the way in which trajectory distributions are generated. Information about adjacent residues, or even experimental data revealing the conformation of certain residues could easily be applied to bias the trajectory distributions, reducing the conformational space available at certain residues.

ACKNOWLEDGMENT

H.J. Feldman is supported by a graduate scholarship from NSERC.

REFERENCES

1. Ngo JT, Marks J. Computational complexity of a problem in molecular structure prediction. Protein Eng 1992;5:313–321.
2. Karplus M. The Levinthal paradox: yesterday and today. Fold Des 1997;2:S69–S75.
3. Onuchic JN, Wolynes PG, Luthey-Schulten Z, Socci ND. Toward an outline of the topology of a realistic protein-folding funnel. Proc Natl Acad Sci USA 1995;92:3626–3630.
4. Rey A, Skolnick J. Efficient algorithm for the reconstruction of a protein backbone from the α-carbon coordinates. J Comp Chem 1992;13:443–456.
5. Holm L, Sander C. Database algorithm for generating protein backbone and side-chain coordinates from a Cα trace application to model building and detection of coordinate errors. J Mol Biol 1991;218:183–194.
6. Correa PE. The building of protein structures from α-carbon coordinates. Proteins 1990;7:366–377.
7. Bassolino-Klimas D, Bruccoleri RE. Application of a directed conformational search for generating 3-D coordinates for protein structures from α-carbon coordinates. Proteins 1992;14:465–474.
8. Abagyan RA. Towards protein folding by global energy optimization. FEBS Lett 1993;325:17–22.
9. Forrest S. Genetic algorithms: principles of natural selection applied to computation. Science 1993;261:872–878.
10. Hogue CWV. Cn3D: a new generation of three-dimensional molecular structure viewer. Trends Biochem Sci 1997;22:314–316.
11. Bernstein FC, Koetzle TF, Williams GJ, et al. The Protein data bank: a computer-based archival file for macromolecular structures. J Mol Biol 1977;112:535–542.
12. Van Gunsteren WV, Berendsen HJC, Colonna F, Perahia P, Hollenberg JP, Lellouch P. On searching neighbors in computer simulations of macromolecular systems. J Comp Chem 1984;5:272–279.
13. Boris J. A vectorized "nearest-neighbors" algorithm of order N using a monotonic Lagrangian grid. J Comput Phys 1986;66:1–20.
14. Appel AW. An efficient program for many-body simulation. SIAM J Sci Stat Comput 1985;6:85–103.
15. Omohundro SM. Efficient algorithms with neural network behavior. Complex Systems 1987;1:273–347.
16. Barnes J, Hut P. A hierarchical O(N log N) force-calculation algorithm. Nature 1986;324:446–449.
17. Karfunkle HR, Eyraud V. An algorithm for the representation and computation of supermolecular surfaces and volumes. J Comput Chem 1989;10:628–634.
18. Hernquist L. Hierarchical N-body methods. Comput Phys Comm 1988;48:107–115.
19. Katzenelson J. Computational structure of the N-body problem. SIAM J Sci Stat Comput 1989;10:787–815.
20. Ramachandran GN, Sasisekharan V. Conformation of polypeptides and proteins. Adv Protein Chem 1968;23:283–438.
21. Gregoret LM, Cohen FE. Protein folding. Effect of packing density on chain conformation. J Mol Biol 1991;219:109–122.
22. Lesk M. Protein architecture. A practical approach. New York: Oxford University Press; 1991. p 57–58.
23. Jabs A, Weiss MS, Hilgenfeld R. Non-proline cis peptide bonds in proteins. J Mol Biol 1999;286:291–304.
24. Stewart DE, Sarkar A, Wampler JE. Occurrence and role of cis peptide bonds in protein structures. J Mol Biol 1990;214:253–260.
25. Brunger AT. Crystallographic refinement by simulated annealing. In: Isaacs NW, Taylor MR, editors. Crystallographic computing 4. New techniques and new technologies. Chester: Oxford University Press; 1988. p 126–140.
26. Isaacs N. Refinement techniques: use of the FFT. In: Sayre D, editor. Computational crystallography. Oxford: Oxford University Press; 1982. p 398–408.
27. Engh RA, Huber R. Accurate bond and angle parameters for X-ray protein structure refinement. Acta Crystallogr 1991;A47:392–400.
28. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Minimization or maximization of functions. In: Numerical recipes in C. Cambridge: Cambridge University Press; 1992. p 394–455.
29. Wang Y, Huq HI, de la Cruz, XF, Lee B. A new procedure for constructing peptides into a given Cα chain. Fold Des 1998;3:1–10.
30. Lee C, Subbiah S. Prediction of protein side-chain conformation by packing optimization. J Mol Biol 1991;217:373–388.
31. Lee C. Predicting protein mutant energetics by self-consistent ensemble optimization. J Mol Biol 1994;236:918–939.
32. Dunbrack RLJ, Karplus M. Backbone-dependent rotamer library for proteins. Application to side-chain prediction. J Mol Biol 1993;230:543–574.
33. Dunbrack RLJ, Cohen FE. Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Sci 1997;6:1661–1681.
34. Cremer D, Pople JA. A general definition of ring puckering coordinates. J Am Chem Soc 1975;97:1354–1358.
35. Hooft RW, Vriend G, Sander C, Abola EE. Errors in protein structures. Nature 1996;381:272–272.
36. Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 1983;22:2577–2637.
37. Brant DA, Flory PJ. The configuration of random polypeptide chains. II. Theory. J Amer Chem Soc 1965;87:2791–2800.
38. Amir D, Krausz S, Haas E. Detection of local structures in reduced unfolded bovine pancreatic trypsin inhibitor. Proteins 1992;13:162–173.
39. Gottfried DS, Haas E. Nonlocal interactions stabilize compact folding intermediates in reduced unfolded bovine pancreatic trypsin inhibitor. Biochemistry 1992;31:12353–12362.
40. Majid I, Jan N, Coniglio A, Stanley HE. Kinetic growth walk: a new model for linear polymers. Phys Rev Lett 1984;52:1257–1260.
41. Flory PJ. Statistical mechanics of chain molecules. Oxford: Oxford University Press; 1989.
42. Pietronero L. Survival probability for kinetic self-avoiding walks. Phys Rev Lett 1985;55:2025–2027.
43. Garnier J, Osguthorpe DJ, Robson B. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. J Mol Biol 1978;120:97–120.
44. Garnier J, Gibrat J-F, Robson B. GOR method for predicting protein secondary structure from amino acid sequence. Methods Enzymol 1996;266:540–553.