Optimization of van der Waals Energy for Protein Side-Chain Placement and Design

Amr Fahmy and Gerhard Wagner*
Biological Chemistry and Molecular Pharmacology, Harvard Medical School, Boston, Massachusetts

ABSTRACT Computational determination of optimal side-chain conformations in protein structures has been a long-standing and challenging problem. Solving this problem is important for many applications including homology modeling, protein docking, and for placing small molecule ligands on protein-binding sites. Programs available as of this writing are very fast and reasonably accurate, as measured by deviations of side-chain dihedral angles; however, often due to multiple atomic clashes, they produce structures with high positive energies. This is problematic in applications where the energy values are important, for example when placing small molecules in docking applications; the relatively small binding energy of the small molecule is drowned by the large energy due to atomic clashes that hampers finding the lowest energy state of the docked ligand. To address this we have developed an algorithm for generating a set of side-chain conformations that is dense enough that at least one of its members would have a root mean-square deviation of no more than $R \text{ Å}$ from any possible side-chain conformation of the amino acid. We call such a set a side-chain cover set of order $R$ for the amino acid. The size of the set is constrained by the energy of the interaction of the side chain to the backbone atoms. Then, side-chain cover sets are used to optimize the conformation of the side chains given the coordinates of the backbone of a protein. The method we use is based on a variety of dead-end elimination methods and the recently discovered dynamic programming algorithm for this problem. This was implemented in a computer program called Octopus where we use side-chain cover sets with very small values for $R$, such as 0.1 Å, which ensures that for each amino-acid side chain the set contains a conformation with a root mean-square deviation of, at most, $R$ from the optimal conformation. The side-chain dihedral-angle accuracy of the program is comparable to other implementations; however, it has the important advantage that the structures produced by the program have negative energies that are very close to the energies of the crystal structure for all tested proteins.

INTRODUCTION

Side-chain conformation prediction, or protein side-chain placement, is an important problem in computational structural biology. Its accurate and efficient solution would have many applications, among them being ab initio protein structure prediction, protein design, and protein complex structure prediction—all of which are vital problems leading toward advances in understanding fundamental questions in structural biology.

The problem can be stated as the prediction of the conformation of the amino-acid side chains, given the coordinates of the protein backbone. The problem is easier than the full protein folding problem; however, it is still computationally hard in the sense that a polynomial time algorithm for side-chain placement can be transformed to solve other computationally hard problems in polynomial time, such as the “traveling-salesman problem” (1,2), for which, as of this writing, no polynomial time algorithm is known.

Despite this, and because of the importance of finding methods to solve the problem of protein side-chain placement, it has been addressed several times in the literature. Many different algorithmic techniques have been used to attempt solving the problem. Some methods are exact, based on the dead-end-elimination (DEE) algorithms (3–6) and on graph theory methods (7,8). Other methods are approximate, such as simulated annealing and Monte Carlo methods (9). Most of the above methods use rotamer libraries to model amino-acid side-chain conformations. The idea is to limit the number of side-chain conformations by gathering conformations that have already appeared in experimentally determined protein structures (see, for example, (10–12)).

Starting with the assignments of all possible rotamers to each residue of the protein from a rotamer library, the program TreePack (http://ttic.uchicago.edu/~jinbo/TreePack.htm) (8) first performs the DEE algorithms by removing rotamers that cannot be part of the optimized energy state. After this step, a small number of rotamers per residue will be left. To optimize the energy of the side-chain conformations, a dynamic programming (13)-based algorithm has been discovered whose runtime is short enough for handling many interesting proteins; in a few seconds of computer time, it is able to select from the remaining set a single rotamer for each residue that will minimize the protein energy. Based on the work of Xu and Berger (8), Krivov et al. (14) developed SCWRL4 (http://dunbrack.fccc.edu/scwrl4/), which also uses the dynamic programming method for side-chain placement and the rotamer library developed by Dunbrack and Karplus (11).

The tree decomposition algorithm used in these programs is certainly correct, which can be proven in that they indeed identify the assignment that minimizes the energy within the
set of conformers used. Thus, we hypothesize that the large positive energies of the resulting structures (see Table 1) are due to the limited choices of rotameric states found in the rotamer libraries and, due to the use of an artificial energy function (15), a coarse approximation of the Lennard-Jones potential.

Here we test this hypothesis by using the dynamic programming algorithm while generating, on the fly and at any desired resolution, all possible side-chain conformations for all residues of a protein, and by using the exact van der Waals energy function. The side-chain conformations are generated by uniformly rotating all dihedral angles, starting from any initial side-chain state, by small-enough angular increments so as to increase the number of available side-chain conformations, thus avoiding atomic clashes, and guaranteeing that a conformation close to the optimal conformation is included in the generated set. The effect of increasing the resolution of rotamer libraries on the energy of the protein is discussed in Xiang and Honig (16). The side-chain states that are generated will be used in the dynamic programming algorithm to predict the conformation of the side chains in proteins. The approach presented here creates near ideally placed side chains without clashes. It minimizes the side-chain energies with a correct van der Waals energy and will allow flexible docking of small molecule ligands or other proteins.

**THEORY AND METHODS**

We first provide an overview of the method followed by an explicit description. The basis of the method is to generate a set of side-chain conformations that we term side-chain cover sets (SCCS), each set of which is dense enough so that at least one of its members would be close to the conformation in the native state of the protein. The generation method is exhaustive up to a resolution $R$, which controls the dihedral angle increment in a manner that will be discussed shortly. The smaller the value of $R$, the more the number of valid side-chain conformations is increased, thus making available many more choices to the optimization algorithm. Next, we discuss the energy-resolution step and the DEE methods used in our implementation. The dynamic programming energy optimization algorithm is summarized after that.

**TABLE 1  vDW side-chain energy of several proteins from the PDB compared to the energy of the predicted structures of TreePack and SCWRL4**

<table>
<thead>
<tr>
<th>PDB ID</th>
<th>Residues</th>
<th>TreePack vDW energy Kcal/mol</th>
<th>TreePack vDW energy Kcal/mol</th>
<th>TreePack vDW energy Kcal/mol</th>
</tr>
</thead>
<tbody>
<tr>
<td>16pk.pdb</td>
<td>415</td>
<td>−658.3</td>
<td>$&gt;10^9$</td>
<td>109,087.6</td>
</tr>
<tr>
<td>1a8d.pdb</td>
<td>452</td>
<td>−890.2</td>
<td>$&gt;10^9$</td>
<td>45,003.1</td>
</tr>
<tr>
<td>1a8i.pdb</td>
<td>812</td>
<td>−1650.6</td>
<td>$&gt;10^9$</td>
<td>485,521.7</td>
</tr>
<tr>
<td>1b6a.pdb</td>
<td>355</td>
<td>−656.9</td>
<td>$&gt;10^9$</td>
<td>10,680.5</td>
</tr>
<tr>
<td>1bfg.pdb</td>
<td>126</td>
<td>−236.8</td>
<td>$&gt;10^{12}$</td>
<td>−103.9</td>
</tr>
<tr>
<td>1bg6.pdb</td>
<td>349</td>
<td>−506.4</td>
<td>$&gt;10^9$</td>
<td>−364.3</td>
</tr>
</tbody>
</table>

Value of the vDW energy is often very large due to atomic clashes as compared to the energy of the structure found in the PDB.

**Side-chain cover sets**

Rotamer libraries store amino-acid side-chain conformations based on experimentally determined protein structures. There is small variation in bond-length and angle-bending values of the side chains—the major variables that determine the conformation of an amino-acid side chain are the dihedral angles.

In this section, we explain how to generate a side-chain conformation set that is sufficiently dense that the root mean-square deviation (RMSD) to an arbitrary side-chain conformation is, at most, $R$ A for any given $R$. The generated set will be called a side-chain cover set of order $R$ (SCCS-$R$). For example, we will show that for $R=1$ A—given an arbitrary side-chain conformation $t$ (not necessarily a member of the SCCS)—the set will contain a member $s$ such that the RMSD between $s$ and $t$ is, at most, 1 A.

**Definition**

Let $t$ be an arbitrary side-chain conformation of some amino acid and let $S$ be a set of side-chain conformations of the amino acid. $S$ is called a side-chain cover set of order $R$ for the amino acid if there exists at least one member, $s \in S$, such that the RMSD between $s$ and $t$ is at most $R$.

The SCCS-$R$ provides a better alternative than using rotamer libraries alone because of several reasons. First, the SCCS-$R$ is complete because it uniformly covers the entire $2\pi$ range of each of the dihedral angles of the side chain. This is contrasted with rotamer libraries where some valid conformations may not have appeared in any experimentally determined protein structure. A second reason is that the resolution, $R$, can always be made smaller for any one amino acid or individual residue, to obtain more conformations that will allow for more accommodation among the atoms of the side chains of the protein so as to minimize its energy. Rotamer libraries are fixed and it is inevitable that side-chain placement programs perform a search near the stored rotamers to minimize energy (16). Finally, the resolution $R$ is known a-priori, which allows more research into other factors that contribute to the accuracy of the solutions to the problem of side-chain placement.

For reasonable values of $R$, the size of the SCCS will be large for use in the optimization algorithm. After the generation of a SCCS, conformations with energies possessing too high a backbone are deleted from further consideration.

**Generation of side-chain cover sets**

In the problem we are addressing, the backbone will be held fixed and the dihedral angles of the side chain will be rotated. Within this picture, dihedral angles are nested, meaning that rotating about the first dihedral bond rotates the entire side chain and fixes only the first group. Rotating about the second dihedral bond leaves the first group unchanged, then rotates the rest of the side chain and fixes the second group, and so on.

To generate a SCCS-$R$, for each increment of each dihedral, the $2\pi$ range of all of the remaining dihedral bonds will be exhausted. The initial state of the side chain used in the generation algorithm is not important because we are going to rotate all dihedral angles over the $2\pi$ range. We can arbitrarily choose the initial state to be the state with all dihedral angles set at 0°. As an example of the generation algorithm presented here, for a side chain with two dihedrals, the first dihedral will be rotated once by its increment and the second dihedral will be rotated by its increment over its entire $2\pi$ range, then the first dihedral will be rotated by another increment, etc. This process continues until the $2\pi$ range of the first dihedral has been exhausted.

In Appendix A, we give the details of deriving the angular increments for side chains of up to four dihedral angles that are necessary to meet the required RMSD $R$. Table 2 shows sizes of SCCS-$R$, for various values of $R$ for all amino acids with rotatable bonds. These values were computed when the amino acid is isolated and are smaller in the presence of a protein backbone. The table also gives the average smallest RMSD of 1000 random conformations of each amino acid to the SCCS-$R$ for various values of $R$. 

Biophysical Journal 101(7) 1690–1698
These values are smaller than \( R \) for each case, which shows that the set covers all the randomly generated side chains. As the number of dihedral bonds increases, the difference between \( R \) and the smallest RMSD to the random conformations becomes larger. This is due to the accumulated effect of computing upper bounds on the atomic deviations.

Steric hindrance does not permit all possible combinations of the dihedral angles. However, there are occasions in which strained side-chain conformations have been found in experimentally determined protein structures (17). We have restrained the size of the SCCS by the value of the self-energy of the side chain but we also allow strained conformations to be part of the set. The high energy of the strained conformation may be compensated for by low energy of the surrounding structure. Discarding or keeping strained side-chain conformations becomes part of the energy optimization problem.

### Protein side-chain optimization algorithm

The optimization algorithm used here is based on the dynamic programming method in Xu and Berger (8) over the tree decomposition of the residue interaction graph of the protein. The major difference is that we generate the SCCS for each residue in the protein instead of using rotamer libraries. An outline of the steps of our algorithm is:

1. For each residue, generate the SCCS-\( R \) for the residue.
2. For each residue, compute the energy of all side-chain conformations to each of the neighboring residues.
3. For each residue, perform the energy-resolution step.
4. Perform the DEE step.
5. Build the residue interaction graph.
6. Compute the tree-decomposition of the graph.
7. Traverse the tree from the leaves to the root to compute the value of the minimum energy.
8. Traverse the tree from the root to the leaves to extract the minimum energy side-chain assignment.

After these steps are carried out, each residue will have one side-chain conformation left that minimizes the energy of the protein up to \( R \).

### A precise energy function

The energy function that is optimized in the programs TreePack and SCWRL (7,14,15) is an approximation of the repulsive component of the Lennard-Jones potential. It is designed so that it does not have very large positive values when atoms clash. Furthermore, atomic radii of terminal atoms in longer residues are reduced so as “to make the steric term more forgiving…” (Bower et al. (15)). In studying energy functions for protein side-chain placement and design, Pokala and Handel also used a linearized version of the Lennard-Jones potential function “to relieve clashes due to the use of discrete rotamers” (18).

Here we optimize the exact Lennard-Jones potential of the van der Waal’s energy (19). The Lennard-Jones potential between a pair of atoms \( a \) and \( b \) is given by

\[
E(a, b) = \frac{A}{r^{12}} - \frac{B}{r^6},
\]

where \( r \) is the distance between the atoms and where \( A \) and \( B \) are constants that depend on the atomic species.

The self-energy, \( E_{self}(s_i) \), of side-chain conformation \( s_i \), is the energy between atoms of the side chain, and the backbone atoms of the protein including those of the side chain itself.

The energy of a side-chain assignment \( P \) (i.e., a choice of a side-chain conformation for each residue of the protein) can be written as a sum of pairs of interacting side chains. The energy, \( E(P) \), of a side-chain assignment \( P \) for the protein can thus be written as

\[
E(P) = \sum_{i=1}^{N} E_{self}(s_i) + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} E_{pair}(s_i, s_j),
\]

where \( s_i \) is the side-chain conformation of residue \( i \), and \( E_{pair}(s_i, s_j) \) is the energy of side chains \( s_i \) and \( s_j \). The aim of this work is to assign a conformation for each residue so that \( E(P) \) is minimized. Parameters for the Lennard-Jones potential and the torsion angle potential are the same ones used in the program CHARMM (19).

### Side-chain energy-resolution method

Consider an arginine or a lysine on the surface of the protein with a small number of neighboring residues. The algorithm for generating the side-chain conformations for such residues will produce many conformations that are very close to each other, yet their energy to the rest of the structure is the same or very close. Here we resolve the energy of these conformations by calculating the RMSD between side-chain conformations of the residue as well as their minimum energy to the rest of the structure. Any pair of conformations within 0.1 Å and energy difference of <0.1 Kcal/mol are merged into a single conformation. This method has minimal impact on the global minimum energy conformation yet reduces the number of side-chain conformations whose energies are indistinguishable. This energy resolution step is carried out for each residue and its computational time is very small.

### Dead-end elimination step

After generating and energy-resolving the side-chain conformations for each residue, we use the DEE method to remove side-chain conformations that cannot be part of the minimum energy state. The first method we use is the Goldstein criteria (4), which states that, for residue \( i \), a conformation \( s_i \) can be eliminated if there is another conformation, \( r_i \), such that...
Residue interaction graph and its tree decomposition

To model the dependence of residues on each other in the structure of a protein, Canutescu et al. (7) introduced the idea of building a graph. Each node in the graph represents a residue of the protein. There is an edge between nodes \( i \) and \( j \) in the graph if the interaction energy between the residues in the protein is significant.

In their program, called SCWRL3.0, Canutescu et al. (7) decompose the residue interaction graph into biconnected components where any two biconnected components share, at most, one residue. For each possible rotamer assignment to the shared residue, each biconnected component is optimized and the global minimum assignment is computed. The algorithm is deterministic and will identify the minimum energy state given the rotamer library. SCWRL3.0 can take a long time for large proteins and for proteins where the biconnected components are large. This motivated Xu and Berger (8) to discover a new algorithm for the problem, which uses the tree-decomposition of the residue interaction graph in a program called TreePack. TreePack is up to five-times faster than SCWRL3.0. Later these methods were used in SCWRL4.0. (Here, we give only an outline of these methods; for a more complete discussion, including proof of correctness of the method, see Xu and Berger (8).)

The tree decomposition of a graph was introduced by Robertson and Seymour (20). It is a data structure that is used to compute solutions of instances of many NP-hard problems if the graph used to model the problem is sparse. This is the case with the residue interaction graph of a protein because the energy between pairs of residues beyond a certain distance is very small. The tree decomposition of the residue interaction graph describes the interdependence of subsets of residues in the protein in the form of a tree. The tree is important because subsets of residues that are represented by different children of a tree-node can be optimized independently given an enumeration of the side-chain conformations of residues of their parent node. See Appendix B for the definition of a tree decomposition.

Dynamic programming equation

Given a tree decomposition of the residue interaction graph and a set of side-chain conformations for each residue, Xu and Berger (8) describe a procedure to compute the assignment of a side-chain conformation to each residue that will minimize the energy of the structure. It is the tree data structure that will allow fast calculation of the minimum energy. We follow the notation and methods given in Xu and Berger (8) for the minimization algorithm.

The calculation of the value of the minimum energy starts at the leaves of the tree and proceeds upwards. Computation of the energy at a node can start only when the energy has been computed for all of its children. At the root of the tree, all possible assignments to residues in the root node are enumerated and the energy of each of its children is added. The energy at the root node is the energy of the entire protein.

In the top-down stage, starting at the root node, the assignments to the residues at the root node that minimizes the energy can be found. Recursively, this assignment is passed down to the children and the assignments to the residues at the children that minimize the energy can also be found. See Appendix C for the dynamic programming equation.
acids with three dihedrals, and the axis using the program Octopus and the energy is measured again (axis). The correlation coefficient is 0.92. The resolution $R$ for the SCCS was as described in the text, $R = 0.1$ for all amino acids with a single dihedral, $R = 0.3$ for all amino acids with two dihedrals, $R = 0.6$ for all amino acids with three dihedrals, and $R = 1.0$ for arginine and lysine.

3. Octopus produces structures with correct side-chain dihedral angles similar to other methods (see Fig. 2, A and B). Historically, performance of side-chain placement programs has been measured by the angular deviation of the side-chain dihedral angles from those found in the PDB structures. Energy of predicted structure as compared to the experimentally determined structure was not used due to the atomic clashes that causes the van der Waals energy to have very large values.

4. Octopus produces structures without side-chain clashes. As an example, Fig. 3 shows the location of TRP-33 in the PDB structure 1BG6 as predicted by the program TreePack on the right with the side chain clashing with many of its neighbors including the protein backbone. The prediction made by Octopus on the left is nearly identical to the crystal structure with no atomic clashes.

5. In general, Octopus is slower than both TreePack and SCWRL4. The program Octopus takes $\sim$1–2 min of computer time on a 3-GHz Linux workstation for a 500-residue protein. TreePack takes $\sim$0.5 s whereas SCWRL4 takes $\sim$3 s for the same protein and same workstation. This is due to the large number of side-chain conformations that are considered by Octopus.

DISCUSSION

To our knowledge, we have developed a new approach for side-chain placement in proteins with a given backbone structure. This problem has been tackled by a number of groups who developed tools to advance this issue, such as dead-end-elimination, graph theory methods, simulated annealing, and Monte Carlo methods. Unfortunately, these sophisticated approaches produce structures with high energies and numerous side-chain clashes. We argue that the reason for these difficulties is the use of rotamer libraries upon which most of these approaches rely. Rotamer libraries create somewhat artificial boundaries for side-chain conformations and have a predefined resolution that cannot easily be modified to eliminate clashes.

Thus, we decided to use a new, to our knowledge, concept of side-chain cover sets that contain all possible side-chain conformations within a resolution $R$. Importantly, $R$ can be made as low as desired during the process of side-chain placement. We also utilized the sophisticated tools of DEE and graph theory as well as similar methods developed by others.

Being able to decrease the value of $R$ in particular cases turned out to be crucial for eliminating clashes. In a first attempt to compare energies of crystal structures with those obtained with the Octopus program as shown in Fig. 1, we found a few drastic outliers when working with a suboptimal resolution; however, lowering the resolution resulted in perfect agreement of energies between the selected PDB entries and the structures calculated with Octopus.

Overall, the approach presented here can place side chains correctly within predefined backbone structures, with low negative energies, nearly identical to the experimental crystal structures. To our knowledge, this is not achievable with other available approaches, such as TreePack or SCWRL4.0.

The approach presented here and the implementation in the Octopus program has numerous potential applications, as enumerated here:

1. We anticipate that it will become an invaluable tool in homology modeling for replacing nonconserved amino acids.

2. The Octopus program will be able to place side-chain conformations in NMR structures of large proteins, and in particular of membrane proteins that have been determined from backbone constraints only, such as residual dipolar couplings, paramagnetic relaxation enhancements, and a few nuclear Overhauser effect (NOE) restraints. An example is the recently determined structure of the mitochondrial uncoupling protein UCP2 (21).

3. This approach will also be suitable for placing side chains correctly in other NMR structures that are based on a restricted number of NOEs, such as between the methyl groups of isoleucines, leucines, and valines, and amide protons. Such constraints are typically obtained for large proteins that are deuterated except for these methyl groups to eliminate dipolar broadening. Examples are the 37-kDa structure of the T-TE didomain of the nonribosomal peptide synthetase EntF (22), or the 50-kDa structure of the C-domain of the same system (23).
4. Another application will be the placement of side chains in protein interfaces that are obtained based on NMR chemical shift mapping, cross-saturation experiments, mutation data, and molecular docking. The short computation times will enable the evaluation of binding energies for multiple docking orientations combined with side-chain rearrangements with Octopus.

5. The Octopus program is ideally suited for placing small molecules into binding sites on proteins identified with chemical shift mapping or NOE measurements. It will allow flexible docking with side-chain rearrangements within a rigid backbone. This will be suitable for many small molecules or drug leads that bind weakly to target proteins without causing backbone rearrangements.

6. The side-chain placement will facilitate assignments of NOE cross peaks in NMR structures in advanced stages of structure determination when the backbone fold is close to being defined.

There are certainly other applications in protein design and recognition of cofactors as well.

CONCLUSION

In this work we present a new, to our knowledge, approach that is able to place side chains in a given backbone structure without clashes and at negative energies essentially identical to those of the parent crystal structures. This is achieved by using side-chain cover sets, which allow placing side chains at any desired fine resolution, and on the fly. The approach does not rely on predefined rotamer libraries where the resolution is limited by the rotamer classes. We obtain low and
A dihedral angle is specified by four atoms, \((a_1, a_2, a_3, a_4)\). Numbering atoms such that an atom is further from the backbone with increasing index (for example, for valine, if \(a_1\) is the backbone nitrogen, \(a_2\) is the \(\alpha\)-carbon, \(a_3\) is the \(\beta\)-carbon, and \(a_4\) is the \(\gamma_1\)-carbon) if we fix \(a_1\), \(a_2\), and \(a_3\), and rotate \(a_4\) about the \(a_2->a_1\) bond, the dihedral angle will change. We can generate many different conformations for the side chain of valine by changing its dihedral angle.

To meet the density requirement for the SCCS-R, we move \(a_4\) over the entire \(2\pi\) range by \(2R\) \(\AA\) increments in successive rotations. The angular increment must therefore be \(2R/d_1\) radians where \(d_1\) is the length of the perpendicular from \(a_4\) to the \(a_2->a_3\) bond (the rotation axis). To see that the set generated using an angular increment of \(2R/d_1\) radians is an SCCS-R, consider an arbitrary side-chain conformation; the position of its \(a_4\) atom will be, in general, between the positions of the \(a_3\) atoms of two successive conformations in the set. In the worst case, it will be exactly between the positions of the \(a_3\) atom of two successive conformations. Thus there is at least one member of the set where the deviation of the \(a_4\) atom is at most \(R\) \(\AA\) from the arbitrary conformation of the side chain and the RMSD of \(a_4\) is, at most, \(R\).

Using an angular increment of \(2R/d_1\) generates a SCCS-R of minimal size. Using a larger increment makes the set lose its covering property, and a smaller increment is not necessary.

### Calculation of angular increments for two dihedral side chains

To calculate the angular increments for the two nested dihedral angles \((a_1, a_2, a_3, a_4)\) and \((a_2, a_3, a_4, a_5)\), i.e., \(\theta_1\) and \(\theta_2\), which ensure that we generate a SCCS-R, we write an expression for the maximum RMSD to an arbitrary conformation

\[
R^2 = \frac{1}{2}(\delta^2(a_4) + \delta^2(a_5)),
\]

where \(\delta(a_4)\) and \(\delta(a_5)\) are the maximum deviations of \(a_4\) and \(a_5\), respectively. From the case of a single dihedral side chain, \(\delta(a_k) = d_1^2\theta_k^2/4\).

To calculate \(\delta^2(a_5)\), consider two successive positions for \(a_4\), i.e., \(p_1\) and \(p_2\), from each of which there will be successive rotations by \(\theta_2\) increments that generate two circles centered at \(p_1\) and \(p_2\). The position of \(a_5\) that would have the largest possible deviation from any of the points on the circles is when it lies exactly between the two circles and also exactly between two pairs of points on the circles, i.e., in the middle of a rectangle whose sides are bounded by \(d_4\theta_1\) and \(d_1\theta_2\) (if that is not the case, i.e., \(a_4\) is closer to one circle over the other, its deviation would be smaller than what is computed here). Thus, we have

\[
\delta^2(a_5) = \frac{d_4^2\theta_1^2}{4} \left[ 1 + \frac{\theta_2}{\theta_1} \right].
\]

Letting \(\theta_2 = k_1\theta_1\), the maximum RMSD of the two atoms can now be computed from

\[
R^2 = \frac{\theta_1^2}{8}(d_4^2 + d_2^2 + d_3^2 k_1^2).
\]

The number of possible positions for \(a_4\) and \(a_5\) is given by \(4\pi^2/k_1\theta_1^2\).

For a fixed \(R\), we find the optimal \(k_1\) that will result in the smallest size SCCS-R given the upper bounds on the deviations of \(a_4\) and \(a_5\). This can be obtained by solving for \(k_1\) in

\[
\frac{dN}{dk_1} = 0.
\]

Using Eq. 3, we thus obtain

\[
k_1^2 = \frac{d_2^2 + d_3^2}{d_4^2}.
\]

Substituting in Eq. 3, the values of \(\theta_1\) and \(\theta_2\), in radians, that will ensure that the smallest size SCCS-R will be

\[
\theta_1 = \frac{2R}{\sqrt{d_4^2 + d_2^2}} \text{ and } \theta_2 = \frac{2R}{d_3}.
\]
Generation of cover sets for three-dihedral side chains

Rotating about the three dihedrals bonds by \( \theta_1 \), \( \theta_2 \), and \( \theta_3 \), respectively, we calculate the maximum RMSD of an arbitrary conformation

\[
R^2 = \frac{1}{3} \left( \delta^2(a_4) + \delta^2(a_5) + \delta^2(a_6) \right),
\]

where \( a_6 \) is the atom at the end of the third dihedral and \( \delta(a_6) \) is its maximum deviation. Let \( d_4 \) be the maximum length (over all possible conformations) of the perpendicular from \( a_6 \) to the \( a_2-a_3 \) bond (the first dihedral bond), \( d_5 \) be the maximum length (over all possible conformations) of the perpendicular from \( a_6 \) to the \( a_5-a_6 \) bond, and \( d_6 \) be the length of the perpendicular from \( a_6 \) to the \( a_4-a_5 \) bond. The maximum deviation of \( a_6 \) is calculated in the same manner as the deviation of \( a_2 \) in the previous section, and we have

\[
R^2 = \frac{\theta_3^2}{12} \left( d_4^2 + d_5^2 + d_6^2 + (d_4^2 + d_5^2) k_1^2 + (d_4^2 + d_6^2) k_2^2 \right),
\]

where \( \theta_3 = k_2 \theta_1 \). The number of possible positions for \( a_6 \) is given by \( 2\pi \theta_3 \) and the size of the SCCS is

\[
N = \frac{8\pi^3}{k_1 k_2 \theta_1}. \tag{10}
\]

To obtain a SCCS-R with the smallest size given the atom deviations, we solve for \( k_1 \) and \( k_2 \) in

\[
\frac{\partial N}{\partial k_1} = 0 \quad \text{and} \quad \frac{\partial N}{\partial k_2} = 0 \tag{11}
\]

to get

\[
k_1 = \frac{d_4^2 + d_5^2 + d_6^2}{d_5^2 + d_6^2} \quad \text{and} \quad k_2 = \frac{d_4^2 + d_5^2 + d_6^2}{d_4^2 + d_6^2}. \tag{12}
\]

The values of \( \theta_1 \), \( \theta_2 \), and \( \theta_3 \), in radians, that will ensure the smallest size SCCS-R, will be

\[
\theta_1 = \frac{2R}{\sqrt{d_4^2 + d_5^2 + d_6^2}}, \quad \theta_2 = \frac{2R}{\sqrt{d_5^2 + d_6^2}} \quad \text{and} \quad \theta_3 = \frac{2R}{d_6}. \tag{13}
\]

For four-dihedral side chains, the calculation of their angular increments that ensures a SCCS-R with the smallest size, given the atom deviations, proceeds in the same manner and produces increments with the same pattern.

APPENDIX B: TREE DECOMPOSITION OF A GRAPH

Definition

Let \( G = (V, E) \) be a graph, a tree decomposition of \( G \) is a pair \( (T, X) \) satisfying the conditions:

1. \( T = (I, F) \) is a tree where \( I \) is the set of nodes of the tree and \( F \) is its set of edges.
2. \( X = \{X_i \mid i \in I, X_i \subseteq V \} \) and \( \bigcup_{i \in I} X_i = V \). Each node in the tree \( T \) is a subset of \( V \) and the union of the subsets is \( V \).

3. For every edge \( e = \{v, w\} \in E \), there is at least one \( i \in I \) such that both \( v \) and \( w \) are in \( X_i \).
4. For all \( i, j, k \in I \). If \( j \) is a node on the path from \( i \) to \( k \) in \( T \), then \( X_i \cap X_k \subseteq X_j \).

The tree decomposition of a graph is not unique; for example, the trivial decomposition consisting of a tree with a single node will result in a very inefficient optimization algorithm. It is thus desirable to have a tree decomposition where the size of the largest subset of residues per tree node is small. The notion of tree-width was also introduced by Robertson and Seymour (20) to differentiate between tree decompositions for the same graph. The tree width of a tree decomposition is \( \max_{i \in I} |X_i| - 1 \), i.e., the node with the maximum number of vertices of the graph minus 1. The tree width of a graph \( G \) is the minimum width over all possible tree decompositions of \( G \). Computing the tree width of a graph is \( \mathcal{NP} \)-hard (23); however, many heuristics are known (24) for computing tree decompositions of a graph with small, but not necessarily smallest, tree width. One such heuristic is the minimum degree heuristic (24), which we use here.

APPENDIX C: DYNAMIC PROGRAMMING EQUATION

Let \( D(i) \) denote the possible side-chain conformations for residue \( i \) and \( D(X) \) denote the possible side-chain conformations of the subset of residues \( X \). Also let \( A(X) \) be an assignment of conformations to residues of subset \( X \) from \( D(X) \). Let \( X_i \) be the parent of node \( X_i \) in the tree and \( C(j) \) be the set of children of the node \( X_j \). If all the residues in \( X_i \cap C(j) \) are removed from the tree, the tree splits into two separate subtrees. Assume that the side-chain conformations of residues in \( X \), \( X_i \), and \( C(j) \) have been assigned conformations \( A(X_i) \cap C(j) \). Define \( F(X_i, A(X_i \cap C(j))) \) to be the minimum energy of the subtree rooted at \( X_i \) given that the side-chain assignment for \( X_i \cap C(j) \) is given by \( A(X_i \cap C(j)) \). Because the energy at a tree node depends only on the assignment to the residues in common with its parent (and not on the assignment of its siblings in the tree), the minimum energy of the tree root at \( X_j \) can be written as the recursive equation

\[
F(X_i, A(X_i \cap C(j))) = \min_{A \in D(X_i - X_i \cap C(j))} \left\{ \sum_{i \in C(j)} F(X_i, A(X_i \cap C(j))) + E(A(X_i)) \right\}.
\]

The calculation of the value of the minimum energy starts at the leaves of the tree and proceeds upwards. Computation of the energy at a node can start only when the energy has been computed for all of its children.

For node \( X_i \), let \( X_j \) be its parent. For each possible assignment \( A(X_i \cap X_j) \), from \( D(X_i \cap X_j) \), enumerate the assignments of residues in \( X_i - X_j \), enumerate the assignment that will minimize the energy of the subgraph induced by \( X_i \). This energy is recorded in a table (as the energy of the residues in \( X_i \cap X_j \)) because it will be used later to minimize the energy at the parent node. In addition, the assignment from \( X_i - X_j \cap X_i \) that minimized the energy is recorded so that in the next stage, the assignments that minimize the energy for the protein can be found. At the root of the tree, all possible assignments to residues in the root node are enumerated and the energy of each of its children is added. The energy at the root node is the energy of the entire protein.

In the top-down stage, starting at the root node, the assignments to the residues at the root node that minimizes the energy can be found. Recursively, this assignment is passed down to the children and the assignments to the residues at the children that minimize the energy can also be found.

A.F. thanks Deani Cooper, Sebastian Hiller, and John Myers for valuable discussions.

This research was supported by National Institutes of Health grant No. GM 47467.
REFERENCES