Fast Protein Folding in the Hydrophobic-hydrophilic Model Within Three-eights of Optimal (Extended Abstract)

[*]

William E. Hart[†] Sorin Istrail[‡]

Abstract

We present performance-guaranteed approximation algorithms for the protein folding problem in the hydrophobic-hydrophilic model, Dill (1985). To our knowledge, our algorithms are the first approximation algorithms in the literature with guaranteed performance for this model, Dill (1994). The hydrophobic-hydrophilic model abstracts the dominant force of protein folding: the hydrophobic interaction. The protein is modeled as a chain of amino acids of length n which are of two types: H (hydrophobic, i.e., nonpolar) and P (hydrophilic, i.e., polar). Although this model is a simplification of more complex protein folding models, the protein folding structure prediction problem is notoriously difficult for this model. Our algorithms have linear ($3n$) time and achieve a three-dimensional protein conformation that has a guaranteed free energy within 3/8 of optimal. By achieving speed and near-optimality simultaneously, our algorithms are consistent with the recently proposed framework of protein folding by Sali, Shakhnovich and Karplus (1994). Equally important, the folding pathway and final conformations of our algorithms are biologically plausible. The algorithms define folding pathways that fit within the framework of diffusion-collision protein folding proposed by Karplus and Weaver (1979), and final conformations generated by the algorithms have significant secondary structure (anti-parallel sheets, beta sheets, hydrophobic core). Previous algorithms have employed exhaustive search of protein sequences and conformation for sequences of length 11 or less. For longer sequences (length ≤ 30), previous algorithms have performed random sampling of sequences for which exhaustive search of conformations was performed. Our result answers the open problem of Ngo, Marks and Karplus (1994) about the possible existence of an approximation algorithm for protein structure prediction in any well-studied model of protein folding.

*This work was supported by the Applied Mathematical Sciences program, U.S. Department of Energy, Office of Energy Research, and was performed at Sandia National Laboratories, operated for the U.S. Department of Energy under contract No. DE-AC04-76DP00789.

†wehart@cs.sandia.gov

‡scistra@cs.sandia.gov

1 Introduction

A protein is a chain of amino acids residues. Although linear, under specific conditions it folds into a unique *native* three-dimensional structure. Experiments show that proteins unfold when folding conditions provided by the environment are disrupted and spontaneously refold to their native structures when conditions are restored. This is the basis for the belief that prediction of the native structure of a protein can be done *computationally* from the information contained in the amino acid sequence. Various protein folds (or conformations) are analyzed in terms of their free energy. The so called Thermodynamical Hypothesis states that the native structure of a protein is the one for which the free energy achieves the global minimum.

In a recent comprehensive review of the computational aspects of protein folding prediction, Karplus et al. [15] articulate one of the basic problems of the field. "The central question addressed in this review is this: Is there some clever algorithm, yet to be invented, that can find the global minimum of a protein's free-energy function reliably and reasonably quickly? Or is there something intrinsic to the problem that prevents such a solution from existing?" The review analyzes recent NP-hardness results for the complexity of simple protein folding models [14, 19]. These results "lend intellectual rigor to the existing pessimism" concerning the existence of algorithms for solving global free-energy minimization problems. Karplus et al. note that approximation algorithms with provable performance guarantees may be an effective way to cope with these NP-hardness results. They observe that

> Such an approximation algorithm might be of significant practical use in protein-structure prediction, because exactness is not such a central issue. If the guaranteed error bound were sufficiently small, an approximation algorithm might be useful for generating crude structures. If not, merely knowing that the energy of the optimal structure is below a certain threshold could still be of use as part of a larger scheme ... approximation algorithms can be joined with an existing stochastic algorithm to form a hybrid that is better than either of its components alone.

However, they note that approximation algorithms are not possible for all NP-hard problems, and pose the following question

> Is there an approximation algorithm for the global free-energy minimization? ... To our knowl-

edge, the possible existence of an approximation algorithm for protein structure prediction has not been addressed either on a specific, ad hoc basis, ... or using the more general techniques introduced by Arora et al. [1].

The results in this paper answer this problem by describing approximation algorithms for the two- and three-dimensional hydrophobic-hydrophilic models [5]. In Section 3, we describe the complexity measures that we will use to analyze these approximation algorithms. In Section 2, we describe the hydrophobic-hydrophilic model and review previous algorithms used to find low-energy conformations within it. In Section 4, we analyze the structure of protein instances. This analysis decomposes protein instances in a way that naturally leads to a normal form for conformations of subsequences of a protein. We utilize this normal form in Section 5 to describe approximation algorithms for the two-dimensional model. Then we describe approximation algorithms for the three-dimensional model in Section 6, and conclude with a discussion of our results.

2 The Hydrophobic-hydrophilic Model

The hydrophobic-hydrophilic model was introduced by Dill [5] and has been intensively studied [2, 3, 10, 11, 13, 20, 19, 21]. It models a protein as a linear chain of amino acid residues. Each amino acid can be either of two types: H (hydrophobic, i.e., nonpolar) or P (hydrophilic, i.e., polar). For simplicity, we denote H by "1" (black) and P by "0" (white).

Conformations of proteins are embedded in either a two-dimensional or three-dimensional square lattice. The lattice simply serves as a tool to discretize the conformational space. It accounts for the soft degrees of freedom which generate different backbone conformations and is consistent with the so called "excluded volume" condition, namely, no two residues can occupy the same lattice site. A chain conformation is represented as a self-avoiding walk on the two-dimensional or three-dimensional square lattice. Each amino acid in the chain is represented by occupying one lattice site, connected to its chain neighbor(s) on adjacent lattice sites.

In the two-dimensional lattice, every lattice site has four neighbors, and the number of bond orientations within the chain for every internal residue is $z - 1 = 3$. In the three-dimensional lattice, every lattice site has six neighbors, and the number of bond orientations within the chain for every internal residue is $z - 1 = 5$. The value z is called the *lattice coordination number.* The conformation space is the set of all possible internal conformations of a molecule, due to all the different bond orientations. The sequence space is the set of all possible sequences of H and P residues. It is convenient to distinguish between pairs of amino acids that are *connected neighbors* in the chain, i.e., occurring in positions j and $j+1$ along the chain, and *topological neighbors* that are adjacent in space (in contact) but not adjacent in the chain.

The model assumes that the free energy is computed between the topological neighbors in a conformation. Every HH contact between topological neighbors has a contact free energy of $\epsilon (< 0)$. Any other pairwise combination of H and P has free energy equal to 0. This is a very simple model of the free energy of protein conformations. However, the model used for prediction in conjunction with mean-field approximation predicts well the experimentally measured temperature and solvent dependencies of protein stability [10]. Following the Thermodynamical Hypothesis, the native structure of a given sequence is the conformation that achieves minimum free energy.

Dill and others have extensively analyzed the biological properties of the hydrophobic-hydrophilic model [2, 3, 10, 11]. For sequences of length 11 or less, their analyses employed algorithms that exhaustively search both the protein sequences and conformations. For longer sequences (length ≤ 30), they used algorithms that performed random sampling of sequences for which exhaustive search of conformations was performed. Unger and Moult [20, 21] have applied approximation algorithms to search for minimal free-energy conformations. Their approximation algorithms are genetic algorithms, and they do not provide performance guarantees for their final solutions. Using these algorithms, they successfully folded a sequence of length 60 on a two-dimensional lattice after approximately 200,000 energy evaluations [19]. They also folded sequences of length 64 on a three-dimensional lattice [21], though the optimal conformation was not know for the protein instances they considered. Stoloz [18] has developed recursively-based computational methods that can be used to identify a large number of low energy conformations, instead of simply identifying the globally minimal conformation. These methods are useful, for example, for analyzing thermodynamic barriers in the model. Finally, Kleitman and Linial [12] have analyzed lower bounds for the two-dimensional model (work in progress).

3 Guaranteed Performance for Approximation Algorithms

Our research examines approximation algorithms for predicting the structure of folded proteins that are fast and have guaranteed performance. The efficiency of an approximation algorithm insures that it will terminate with a solution quickly. The guaranteed performance insures that for every protein instance the energy of the solution will be below a given threshold in a mathematically provable way.

To our knowledge, our approximation algorithms are the first to insure both efficiency and provable guaranteed performance for the protein folding problem. Thus they may be distinguished from previous work with ***heuristic*** approximation algorithms that do not have these properties (e.g. Unger and Moult [20, 21]). Algorithms with guaranteed performance provide solutions with the given performance for every problem instance. In contrast, heuristic approximation algorithms typically perform well for certain problem instances, while performing less well for other problem instances. Consequently, an analysis of heuristic approximation algorithms relies on empirical tests to determine whether they are appropriate in practical settings. In the context of protein folding, approximation algorithms with guaranteed performance are very important because they can generate solutions for large (long) protein instances, for which current heuristic methods cannot generate good solutions within acceptable time limits.

We use computational definitions of efficiency and performance to analyze approximation algorithms. An algorithm is *efficient* if it terminates with a solution after executing for a length of time that is polynomial in the length of the protein instance. For example, if $L(s)$ is the length of a protein instance s, then the number of steps executed by a polynomial-time algorithm is proportional to $L(s)^m$ for some

m greater than zero. This definition places no restrictions on the amount of memory (space) used by the algorithm, but the approximation algorithms that we describe use an amount of space that is polynomial in the length of the protein instance.

Two types of performance guarantees are often described for approximation algorithms [8]: the *absolute performance ratio* and the *asymptotic performance ratio.* Let $\mathcal{A}(s)$ be the energy of the conformation generated for protein instance s by algorithm $\mathcal{A}$, and let $OPT(s)$ be the energy of the optimal conformation. The *absolute performance ratio* $R_{\mathcal{A}}$ of algorithm $\mathcal{A}$ is given by

$$R_{\mathcal{A}} = \sup\{r \geq 1 \mid R_{\mathcal{A}}(s) \geq r, \forall s\},$$

where $R_{\mathcal{A}}(s) = \mathcal{A}(s)/OPT(s)$. Given $N \in \mathbf{Z}$, let $S_N = \{s \mid OPT(s) \leq N\}$, and let $R_{\mathcal{A}}^N = \inf\{R_{\mathcal{A}}(s) \mid s \in S_N\}$. The *asymptotic performance ratio* $R_{\mathcal{A}}^{\infty}$ is given by

$$R_{\mathcal{A}}^{\infty} = \sup\{r \mid R_{\mathcal{A}}^N \geq r, N \in \mathbf{Z}\} = \sup_{N \in \mathbf{Z}} \inf_{s \in S_N} R_{\mathcal{A}}(s).$$

If $R_{\mathcal{A}} = r$, then the value of solutions generated by algorithm $\mathcal{A}$ are within a factor of r of the optimum. If $R_{\mathcal{A}}^{\infty} = r$, then as $\mathcal{A}$ is applied to larger protein instances, the value of solutions generated by $\mathcal{A}$ approaches a factor of r of the optimum. Since $\mathcal{A}(s) \leq 0$ and $OPT(s) \leq 0$, both of these ratios are scaled between 0 and 1 such that a ratio closer to 1 indicates better performance.

4 Structure Analysis

We represent a protein instance s as a sequence of elements $s_1, s_2, \ldots, s_L$, where $s_i \in \{0,1\}$. Let $L(s)$ equal the length of the sequence s. Let $M(s)$ equal the length of the longest sequence of zeros in s. Finally, let $E(s)$ is the number of connected neighbors in the sequence, s_j and s_{j+1} for which $s_j = 1$ and $s_{j+1} = 1$.

Perhaps the most salient feature of this discrete protein folding problem is that two elements s_i and s_j can be topological neighbors in a conformation only if $|i-j|$ is odd. This has two important implications. First, we can decompose a sequence s into a sequence of blocks such that 1's within each block cannot be topological neighbors in any conformation, but 1's between alternating blocks can be topological neighbors. This description can be used to define a *normal form* for sequences that is useful when reasoning about approximation algorithms. Second, we can bound the optimal free energy of a protein instance by counting the number of 1's that can be topological neighbors.

4.1 Decomposition

A protein instance s can be decomposed into a sequence of subsequences that contain either 0's or 1's. Thus we can decompose s into $Z_0 I_1 Z_1 \ldots Z_{k-1} I_k Z_k$, where Z_i are sequences of 0's and I_i are sequences of 1's. The lengths of Z_i satisfy the constraint

$$L(Z_i) \geq \begin{cases} 0 & i = 0, k \\ 1 & \text{otherwise} \end{cases},$$

while the lengths of I_i satisfy $L(I_i) \geq 1$. For example, the sequence 010101111010100001010101 can be decomposed such that

$$\begin{aligned} L(Z) &= (L(Z_0), \ldots, L(Z_9)) \\ &= (1,1,1,1,1,4,1,1,1,0) \end{aligned}$$

and

$$\begin{aligned} L(I) &= (L(I_1), \ldots, L(I_9)) \\ &= (1,1,4,1,1,1,1,1,1). \end{aligned}$$

An instance s can also be decomposed into a sequence of *blocks*. A block b_i has the form $b_i = 1$ or $b_i = 1Z_{i_1}1 \ldots Z_{i_h}1$, where the Z_{i_j} are odd-length sequences of 0's and $h \geq 1$. A *block separator* z_i is a sequence of 0's that separates two consecutive blocks, where $L(z_i) \geq 0$ and $L(z_i)$ is even for $i = 1, \ldots, h-1$. Thus s is decomposed into $z_0 b_1 z_1 \ldots b_h z_h$. Since $L(z_i) \geq 0$, this decomposition treats consecutive 1's as a sequence of blocks separated by zero-length block separators. Let $N(b_i)$ equal the number 1's in b_i. Thus the sequence

$$0\,\underbrace{10101}\,\underbrace{1}\,\underbrace{1}\,\underbrace{10101}\,0000\,\underbrace{1010101}$$

can be represented as

$$\begin{aligned} L(z) &= (1,0,0,0,4,0) \\ N(b) &= (3,1,1,3,4). \end{aligned}$$

Recall that two 1's can be topological neighbors only if there is an even number of elements between them. It follows from our definition of blocks that two 1's within a block cannot be topological neighbors. Further, any pair of 1's take from blocks b_k and b_j may be topological neighbors only when $|k-j|$ is odd. To see this, observe that the length of each block is odd. If $|k-j|$ is even, then there are an odd number of blocks between b_j and b_k. Since block separators between blocks have even length, the total distance between blocks b_j and b_k is odd. Consequently, the distance between any pair of 1's taken from blocks b_k and b_j is odd. If $|k-j|$ is odd, then there are an even number of blocks between b_j and b_k, so the total distance between blocks b_j and b_k is even. Consequently, the distance between any pair of 1's taken from b_k and b_j is even.

Since 1's from a block can only be topological neighbors of 1's from every other block, it is useful to divide blocks into two categories: x-blocks and y-blocks. For example, let $x_i = b_{2i}$ and let $y_i = b_{2i-1}$. This makes it clear that 1's from an x-block can only be topological neighbors to 1's from an y-block. Let B_x and B_y be the number of x-blocks and y-blocks respectively. Further, let $X = X(s) = \sum_{i=1}^{B_x} N(x_i)$ and $Y = Y(s) = \sum_{i=1}^{B_y} N(y_i)$. We assume that the division into x- and y-blocks is such that $X \leq Y$. For example, the sequence

$$0\,\underbrace{10101}_{y_0}\,\underbrace{1}_{x_0}\,\underbrace{1}_{y_1}\,\underbrace{10101}_{x_1}\,0000\,\underbrace{1010101}_{y_2}$$

can be represented as $z_0 y_0 z_1 x_0 z_2 y_1 z_3 x_1 z_4 y_2 z_5$, where

$$\begin{aligned} L(z) &= (1,0,0,0,4,0) \\ N(x) &= (1,3) \\ N(y) &= (3,1,4). \end{aligned}$$

A *superblock* B_i is comprised of sequences of blocks as follows: $B_i = b_{i_1} z_{i_1} \ldots z_{i_{k-1}} b_{i_k}$. Let $N_x(B_i)$ equal the sum of $N(b_j)$, where b_j are x-blocks in B_i. Let $N_y(B_i)$ equal the sum of $N(b_j)$, where b_j are y-blocks in B_i. Finally, let $N(B_i) = N_x(B_i) + N_y(B_i)$.

4.2 The Normal Form

Given a decomposition of a protein instance into blocks and superblocks, we can configure the instance into a normal form that is useful when describing approximation algorithms. The normal form specifies the conformation of blocks and superblocks. With only mild restrictions, each block and superblock can be independently configured into this normal form.

Figure 1a illustrates the normal form for the block 10001010001000001. The normal form for a block b_i specifies that the 1's in b_i be placed on a two-dimensional grid such that the 1's lie in along a line, separated by single 0's. The single 0 comes from the odd-length sequence of 0's between consecutive 1's, and the remaining even-length sequence of 0's is folded in the remaining dimension. The normal form specifies that all of these *zero-loops* lie on the same "side" of the 1's in the block. For example, if the 1's are placed in a vertical line, then all zero-loops lie either to the left or right of the vertical line. The configuration of a block in this normal form is called the *block structure* of b_i. The line through which the 1's of a block are configured is called the *face* of the block structure.

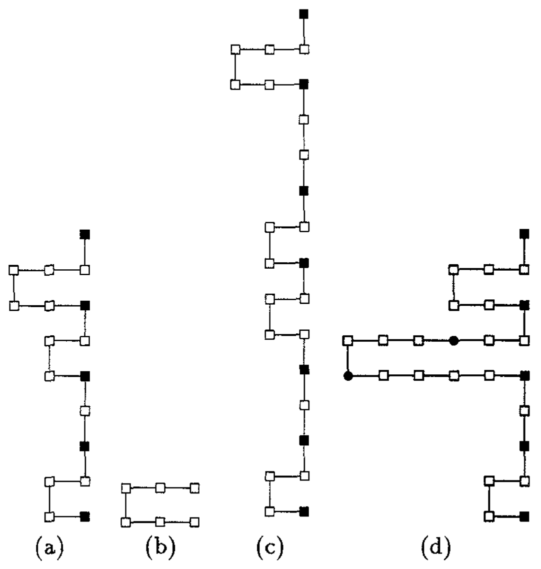


Figure 1: Illustrations of normal forms, where left to right along the sequence is mapped from bottom to top in the normal form. A black box represents a 1 in a y-block, a black circle represents a 1 in a x-block, and a white block represents a 0. Figures are (a) normal form block structure for block 10001010001000001, (b) normal form block separator structure for 000000, (c) normal form superblock structure for block 1000101000010001001000001, and (d) normal form x-superblock structure for block 1000101000010001001000001.

Figure 1b illustrates the normal form for the block separator 000000. If the length of a block separator is greater than two, then we can define a normal form for the *block separator structure*. Block separators are configured such that the two 0's at its endpoints are vertically or horizontally adjacent. The remaining 0's are looped in the same manner as the zero-loops in blocks. The *face* of the block separator structure is the two 0's at its endpoints.

Figure 1c illustrates the normal form for the superblock 1000101000010001001000001. The normal form for superblocks uses the block structures of its constituent blocks and block separators. The block structures are constructed on a two-dimensional grid such that the face of each block structure lies along the same line on the grid. Further, all of the zero-loops from these blocks are placed on the same side of the faces of the blocks. The structures for block separators are constructed such that the face of the block separator is aligned with the faces of the block structures. The configuration of a superblock in this normal form is called the *superblock structure* of B_i. The line through which the 1's of a superblock are configured is called the *face* of the superblock structure.

It is often of interest to construct a superblock structure that contains 1's from either x-blocks or y-blocks. We call such a superblock structure an *x-superblock structure* if only 1's from x-blocks are in the face, and an *y-superblock structure* if only 1's from y-blocks are in the face. To construct an x-superblock structure, the block structure is constructed for each x-block in the superblock. The block structures are constructed on a two-dimensional grid such that the face of each block structure lies along the same line on the grid. Consecutive x-blocks are separated by an odd-length sequence that contains an y-block (the sequence contains an y-block, which is odd-length, along with two block separators, which are even-length, so the total length is odd). This sequence is folded like the odd-length sequences of 0's in the x-block, which fills in the space between x-blocks and loops on the same side of the block faces as the zero-loops. The y-superblock structures are constructed analogously. Figure 1d shows the x-block superstructure for the sequence

$$\underbrace{1000101}_{y_1}\,0000\,\underbrace{10001}_{x_1}\,00\,\underbrace{1000001}_{y_2},$$

which is decomposed into blocks y_1, x_1 and y_2.

5 Folding Sequences in the Two-Dimensional Model

This section describes approximation methods for the two-dimensional hydrophobic-hydrophilic model. We begin by describing bounds on the optimum in two-dimensions. We then describe two approximation algorithms that generate solutions that asymptotically are within 1/4 of optimal. These algorithms are distinguished by differences in their absolute performance guarantees.

5.1 Bounds on the Optimum

Before describing approximation algorithms for the two-dimensional hydrophobic-hydrophilic model, we consider a lower bound on the energy of the optimal conformation. Consider an instance s as an alternating sequence of x-blocks x_i and y-blocks y_i. Let $T_x(s)$ be the number of endpoints that are 1's in an x-block, and let $T_y(s)$ be the number of endpoints that are 1's in an y-block. We assume that if $X = Y$ then $T_x(s) \geq T_y(s)$.

Every 1 in each x-block can be a topological neighbor of at most two other 1's, except when the 1 is an endpoint of s. In that case, the 1 may be next to three 1's. Thus the optimal energy is at most

$$OPT(s) \geq -2X - T_x(s). \qquad (1)$$

5.2 Basic Approximation Algorithm

Our first approximation algorithm for the two-dimensional hydrophobic-hydrophilic model is Algorithm $\mathcal{A}$. Given a protein instance, Algorithm $\mathcal{A}$ selects a single folding point (turning point) that divides the protein instance into a y-superblock B' and an x-superblock B'', such that $N_y(B')$ is balanced with $N_x(B'')$ (i.e. the folding point is between blocks). Superblock structures are generated for the two superblocks, which "eliminates" the x-blocks from the face of the superblock structure for B' and "eliminates" the y-blocks from the face of the superblock structure for B''. Finally, the two superblock structures are folded together to constitute the "hydrophobic core" [10].

Figure 2 describes the subroutine used to select the folding point. This subroutine is described separately since it is used by all of our approximation algorithms. Lemma 1 describes a property of the resulting fold which is used to provide performance bounds for the approximation algorithms.

Lemma 1 The folding point selected by Subroutine 1 partitions a protein instance s into two superblocks B' and B'' such that either

$$N_y(B') \geq \left\lceil \frac{Y+1}{2} \right\rceil \quad \text{and} \quad N_x(B'') \geq \left\lceil \frac{X}{2} \right\rceil, \qquad (2)$$

or

$$N_y(B') \geq \left\lceil \frac{Y}{2} \right\rceil \quad \text{and} \quad N_x(B'') \geq \left\lceil \frac{X+1}{2} \right\rceil. \qquad (3)$$

Figure 3 formally defines Algorithm $\mathcal{A}$. Figure 7a illustrates the application of Algorithm $\mathcal{A}$.

1. Label the blocks of s with x_i and y_i such that $X \leq Y$ and if $X = Y$ then $T_x(s) \geq T_y(s)$. Apply Subroutine 1 to select a folding point.
2. Construct the x-superblock structure for B'' and construct the y-superblock structure for B'.
3. Fold the two superblock structures together face-to-face.

Figure 3: **Algorithm $\mathcal{A}$**

Note that each step of Algorithm $\mathcal{A}$ is linear. Decomposition into x- and y-blocks requires a single pass through the protein instance. Subroutine 1 requires a single pass through the sequence of blocks, which is no longer than the length of the protein instance. Finally, the superblock structures for B' and B'' and final conformation can be constructed with a single pass through the protein instance. Thus there are on the order of $3L(s)$ computations required by Algorithm $\mathcal{A}$ which is linear in $L(s)$.

Let $\mathcal{A}(s)$ represent the energy of the final conformation generated by Algorithm $\mathcal{A}$. The performance of Algorithm $\mathcal{A}$ can be bounded as follows.

Lemma 2

$$\mathcal{A}(s) \leq \begin{cases} -\lceil X/2 \rceil & ,E(s) = 0 \\ -\lceil X/2 \rceil + 1 & ,E(s) > 0 \end{cases}.$$

Proposition 1 uses Lemma 2 to prove asymptotic and absolute performance ratios for Algorithm $\mathcal{A}$.

Proposition 1 $R^{\infty}_{\mathcal{A}} = 1/4$ and $R_{\mathcal{A}} = 0$.

Note that $R_{\mathcal{A}}(s) = 0$ for all s such that $X(s) = 1$ or $X(s) = 2$. We expect that these sequences occur quite infrequently in practice, so the empirical performance of Algorithm $\mathcal{A}$ should be much better than the absolute performance ratio suggests.

5.3 Improved Two-Dimensional Approximation Algorithm

Figure 4 describes Algorithm $\mathcal{B}$, an approximation algorithm for the two-dimensional model which has a better absolute performance ratio than Algorithm $\mathcal{A}$. Algorithm $\mathcal{B}$ modifies the final conformation generated by Algorithm $\mathcal{A}$ to create additional topological neighbors between hydrophobic amino acids, which enables tighter performance guarantees. Figure 6 illustrates the application of Algorithm $\mathcal{B}$ on four sequences which show the different modified folds performed by Algorithm $\mathcal{B}$. Figure 7b illustrates the application of Algorithm $\mathcal{B}$.

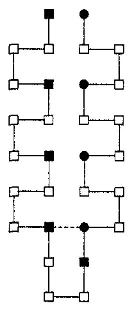
Each step of Algorithm $\mathcal{B}$ is linear, so Algorithm $\mathcal{B}$ requires linear time. Let $\mathcal{B}(s)$ represent the energy of the final conformation generated by Algorithm $\mathcal{B}$. The performance of Algorithm $\mathcal{B}$ can be bounded as follows.

$$\underbrace{10001000100010001}_{B''}\,00\,\underbrace{1000100010001}_{B'}$$

(a)

$$\underbrace{1000100010001}_{B''}\,000100\,\underbrace{1000100010001}_{B'}$$

(b)



(c)

Figure 5: Illustration of Step 2 of Algorithm $\mathcal{B}$: (a) initial decomposition into B' and B'', (b) modified decomposition, and (c) conformation using new folding point, with a dashed line indicating the folding point.

Lemma 3

$$\mathcal{B}(s) \leq -\lceil (X+1)/2 \rceil.$$

Proposition 2 uses Lemma 3 to prove asymptotic and absolute performance ratios for Algorithm $\mathcal{B}$.

Proposition 2 $R^{\infty}_{\mathcal{B}} = 1/4$ and $R_{\mathcal{B}} = 1/4$.

6 Folding Sequences in the Three-Dimensional Model

This section describes approximation methods for the three-dimensional hydrophobic-hydrophilic model. We begin by describing bounds on the optimum in three-dimensions. Next we show how performance guarantees for algorithms

Let $cond(a,b,A,B) \equiv (a < b)\ OR\ ((a = b)\ AND\ (A < B))$. For a partition (B_1, B_2), let $m_{xy} = \min(N_x(B_1), N_y(B_2))$, $m_{yx} = \min(N_y(B_1), N_x(B_2))$, $M_{xy} = \max(N_x(B_1), N_y(B_2))$, and $M_{yx} = \max(N_y(B_1), N_x(B_2))$.

Decompose s into blocks, such that $s = z_0 b_1 z_1 b_2 \ldots b_k z_k$, and do the following.

```
$B_1 = z_0 b_1 z_1$    $B_2 = b_2 z_2 \ldots b_k z_k$
if ($m_{xy} > m_{yx}$)
  $B' = B_2$   $B'' = B_1$   $e = m_{xy}$   $E = M_{xy}$
else
  $B' = B_1$   $B'' = B_2$   $e = m_{yx}$   $E = M_{yx}$
endif
for $i$ in 3:$k$
  $B_1 = z_0 b_1 \ldots b_{i-1} z_{i-1}$    $B_2 = b_i z_i \ldots b_k z_k$
  if ( $cond(e, m_{xy}, E, M_{xy})$ )
    $B' = B_2$   $B'' = B_1$   $e = m_{xy}$   $E = M_{xy}$
  endif
  if ( $cond(e, m_{yx}, E, M_{yx})$ )
    $B' = B_1$   $B'' = B_2$   $e = m_{yx}$   $E = M_{yx}$
  endif
endfor
```

Figure 2: **Subroutine 1**

1. Label the blocks of s with x_i and y_i such that $X \leq Y$ and if $X = Y$ then $T_x(s) \geq T_y(s)$. Apply Subroutine 1 to select a folding point.
2. Let z_i be the block separator at the selected folding point. Construct the folding point as follows.
   (a) If $N_y(B') = N_x(B'')$ or $L(z_i) > 0$, then construct the standard folding point with z_i.
   (b) If $N_y(B') > N_x(B'')$ and $L(z_i) = 0$, then redefine B' to exclude the 1 that is adjacent to z_i and construct the standard folding point with the sequence between ends of B' and B'' (see Figure 5).
   (c) If $N_y(B') < N_x(B'')$ and $L(z_i) = 0$, then redefine B'' to exclude the 1 that is adjacent to z_i and construct the standard folding point with the sequence between ends of B' and B''.
3. Construct the x-superblock structure for B'' and construct the y-superblock structure for B'.
4. Do one of the following:
   (a) If $N_y(B') = N_x(B'')$ then refold the last 1 from the face of the two superblocks as illustrated in Figure 6a. Fold the last 1 from the x-superblock onto the end of the y-superblock, and fold the last 1 from the y-superblock around the last 1 from the x-superblock.
   (b) If $|N_y(B') - N_x(B'')| > 1$ and the superblock with the shorter face ends with a 1 on the face, then wrap the superblock with the longer face so it wraps around the end of the shorter superblock (see Figure 6b).
   (c) If $|N_y(B') - N_x(B'')| = 1$, or $|N_y(B') - N_x(B'')| > 1$ and the superblock with the shorter face does not end with a 1 on the face, then modify the fold the of the superblock with the longer face so it wraps on top of the superblock with the shorter face (see Figure 6c).

Figure 4: **Algorithm $\mathcal{B}$**

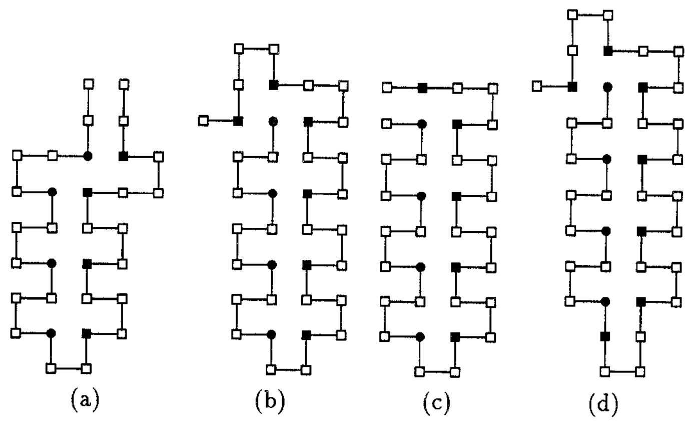


Figure 6: Applications of Algorithm $\mathcal{B}$ to examples that illustrate the different processing done in the last step of the algorithm: (a) Step 4a, (b) Step 4b and (c) Step 4c. Figure (d) illustrates the application of Algorithm $\mathcal{B}$ when a modified folding point is selected.

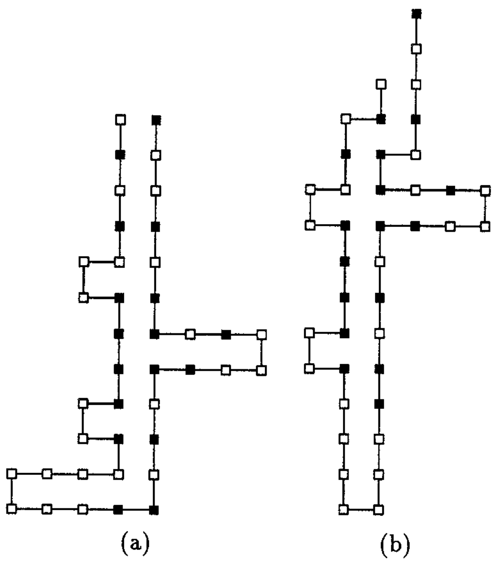


Figure 7: Examples of the application of (a) Algorithm $\mathcal{A}$ and (b) Algorithm $\mathcal{B}$.

applied for the two-dimensional model can be used to provide performance guarantees for the three-dimensional model. Finally, we describe an approximation algorithm that generates solutions that asymptotically are within 3/8 of optimal.

6.1 Bounds on the Optimum

In the three-dimensional model, each 1 in an x-block can be a topological neighbor of at most four other 1's, except when the 1 is at an endpoint of s. In that case, the 1 can be a topological neighbor of five other 1's. Thus the free energy can be bounded as follows

$$OPT(s) \geq -4X - T_x(s). \tag{4}$$

Let $OPT_{2D}(s)$ be the value of the optimal conformation of s in the two-dimensional model, and let $OPT_{3D}(s)$ be the value of the optimal conformation of s in the three-dimensional model. The following theorem relates these values, showing that $OPT_{3D}(s)$ is within a constant factor of $OPT_{2D}(s)$.

Theorem 1 $OPT_{2D}(s) \geq OPT_{3D}(s) \geq 8 OPT_{2D}(s)$.

The upper bound on $OPT_{3D}(s)$ is tight (e.g. consider the sequence 1001). It is not clear whether the lower bound is tight, since the proof of the lower bound uses the absolute performance guarantee for Algorithm $\mathcal{B}$. Consequently, improved approximation methods for the 2D HP model would improve this bound.

6.2 Extension of Results for Two-dimensional Approximation Algorithms

The approximation algorithms for the two-dimensional model can clearly be used to generate conformations for the three-dimensional model. Using the bounds on $OPT_{3D}(s)$ provided by Theorem 1, we can bound the values of an algorithm's absolute and asymptotic performance guarantees for the three-dimensional model, given its performance guarantees for the two-dimensional model. Let $\mathcal{Z}_{2D}$ ($\mathcal{Z}_{3D}$) refer to the application of a generic Algorithm $\mathcal{Z}$ in the two-dimensional (three-dimensional) model, and let $S_N^{2D} = \{s \mid$

$OPT_{2D}(s) \leq N\}$ ($S_N^{3D} = \{s \mid OPT_{3D}(s) \leq N\}$). The following two propositions provide bounds on $R_{\mathcal{Z}_{3D}}$ and $R^{\infty}_{\mathcal{Z}_{3D}}$.

Proposition 3 If $\kappa_1 \leq R_{\mathcal{Z}_{2D}} \leq \kappa_2$ then $\kappa_1/8 \leq R_{\mathcal{Z}_{3D}} \leq \kappa_2$.

Proposition 4 If $\kappa_1 \leq R^{\infty}_{\mathcal{Z}_{2D}} \leq \kappa_2$ then $\kappa_1/8 \leq R^{\infty}_{\mathcal{Z}_{3D}} \leq \kappa/2$.

For specific algorithms, these bounds can be tightened. For example, the following proposition shows that Algorithm $\mathcal{B}$ has a one-eigth bound in the three-dimensional model.

Proposition 5 $R^{\infty}_{\mathcal{B}_{3D}} = 1/8$ and $R_{\mathcal{B}_{3D}} = 1/8$.

6.3 Iterative Approximation Algorithm

Our approximation algorithm for the three-dimensional model, Algorithm $\mathcal{C}$, constructs a conformation in an "iterative" fashion by dividing the x- and y-superblocks into K subpieces each. The faces of the $2K$ new superblocks are configured in x-y planes such that the tops of the faces form a rectangular array of width two in the x-z plane. This configuration places the face of each superblock adjacent to three other superblocks, except for the superblocks at the corners of the rectangular array, which are adjacent to two other superblocks.

Figures 8a and 8b illustrate the two different types of planes used to construct a final configuration. These differ only in the way they connect the current plane to the next plane. The n-th plane connects to the next plane by connecting to either the top or bottom of the next plane. Figure 8c illustrates a final conformation after K planes have been connected together. Figure 8d shows the relative locations of 1's in the protein instance with dashed lines between topological neighbors.

Algorithm $\mathcal{C}$ is formally defined in Figure 9. Algorithm $\mathcal{C}$ partitions the x- and y-superblocks into K superblocks of length J that are seperated by blocks of length 1 or 3 alternatively. If $L = \min(N_x(B''), N_y(B'))$, then $L \geq KJ + 2K - 1$. This bound can be tight, so this is the greatest lower bound on L possible. Now given K, we can let J be

$$J = \left\lfloor \frac{L - 2K + 1}{K} \right\rfloor.$$

To allow both K and J to grow as X increases, we let $K = \lfloor f(L) \rfloor$, where f is a monotonically increasing function such that $f(x) < x$. Figure 10 illustrates the application of Algorithm $\mathcal{C}$.

Each step of Algorithm $\mathcal{C}$ is linear, so Algorithm $\mathcal{C}$ is requires linear time. Let $\mathcal{C}(s)$ represent the energy of the final conformation generated by Algorithm $\mathcal{C}$. The performance of Algorithm $\mathcal{C}$ can be bounded as follows.

Lemma 4 Let $\bar{X} = \lceil X/2 \rceil$, Let $\bar{K} = \lfloor f(\bar{X}) \rfloor$ and $\bar{J} = \bar{X}/\bar{K} - 3$. If $\bar{X} \geq 5$ then

$$\mathcal{C}(s) \leq -3\bar{K}\bar{J} + 2\bar{J} + 1.$$

Theorem 2 uses Lemma 4 to prove the asymptotic performance ratio for Algorithm $\mathcal{C}$.

Theorem 2 If $f(x) = \sqrt{x}$ then $R^{\infty}_{\mathcal{C}} = 3/8$.

The choice of $f(x) = \sqrt{x}$ makes the asymptotic convergence of Algorithm $\mathcal{C}$ nearly as fast as possible. The optimal selection for f is approximately $11\sqrt{x}/18$.

7 Discussion

For lack of space, we have omitted a discussion of several other algorithms. For example, we have developed dynamic programming methods for the two-dimensional model. This algorithm has a poor worst-case behavior, but in practice it works much better than the simple single-fold algorithms in two-dimensions.

Garey and Johnson [8] note that the relative importance of the asymptotic and absolute performance ratios typically depends on the problem being considered. For protein folding, we expect that the asymptotic performance ratios will be more important since the most difficult protein instances to analyze are precisely those which have a native state with very low free energy, and the actual performance ratio approaches the asymptotic performance ratio for these protein instances. The values of solutions generated by most of our approximation algorithms differ from the asymptotic performance ratio by an additive constant to the numerator of the ratio, so they asymptotic performance ratio should accurately estimate the true performance ratio even for even protein instances with moderately low free energies. The exception is Algorithm $\mathcal{C}$, which has an additive square-root factor.

Karplus et al. [15] observe that approximation algorithms with provable guarantees are one way to cope with the NP-hardness of protein folding problems. This observation responds to recent NP-hardness results for a number of models of protein folding [7, 19, 14]. While the hydrophobic-hydrophilic model is a special case of the lattice model considered by Unger and Moult [19], to our knowledge there is no proof that folding proteins in the hydrophobic-hydrophilic model is NP-hard. While it is possible that a polynomial time algorithm exists for this problem, we conjecture that it is in fact NP-hard.

A related problem which we also conjecture to be NP-hard, is the Hydrophobic-Core Compactness Problem. This objective of the problem is to maximize the number of 1's in a conformation that are topological neighbors to two or more 1's. Approximation algorithms for this problem could also be of interest, since solutions to this problem have a dense hydrophobic core.

We conclude by discussing the biologically plausibility of our algorithms. Figure 11 symbolically illustrates the folding pathways that Algorithm $\mathcal{A}$ and Algorithm $\mathcal{C}$ use to fold a protein instance. The final conformations generated by the algorithms have significant secondary structure. The block structures reminiscent of helixes. The face-to-face foldings of the superblock structures are similar to anti-parallel sheets since the directionality of one half of the fold is opposite that of the other. A hydrophobic core arises in both the two- and three-dimensional algorithms from the adjacent faces of the superblock structures, which result in an aggregation of 1's. Finally, the three-dimensional algorithms fold the superblock structures in a zigzag fashion (in the x-z plane) is similar to the packing that occurs in beta sheets.

The approximation algorithms suggest pathways described in phases that may happen during protein folding. Figure 11 schematically illustrates these phases. Figure 11a presents a protein in an unfolded state. Figure 11b describes the phase in which the sequence is decomposed into blocks that are functionally self-contained, and which separated by block-separators which are even-length sequences of 0's. These are the helix-like structures. Implicit in this decomposition is the alternation of such blocks which divides them into x- and y-blocks according to our algorithms. Note that this

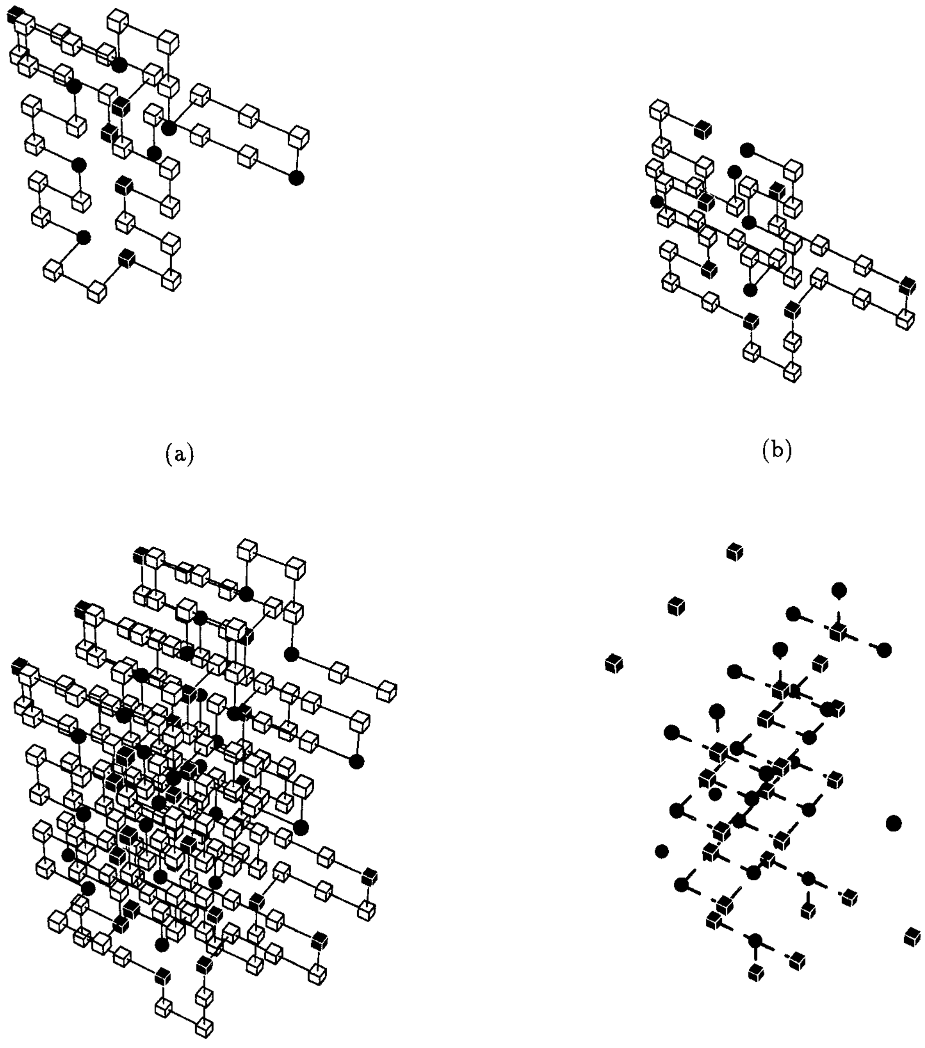


Figure 8: Example of an application of Algorithm $\mathcal{C}$ to a sequence of length 202, showing (a) the configuration of the x- and y-superblocks moving *up*, with connections to the next x-y plane at the top of the superblocks, (b) the configuration of the x- and y-superblocks moving *down*, with connections to the next x-y plane at the bottom of the superblocks, (c) the final conformation of the sequence and (d) the relative positions of 1's in final conformation, with dashed lines between topological neighbors highlighting the hydrophobic core.

1. Label the blocks of s with y_t and x_t such that $X \leq Y$ and if $X = Y$ then $T_x(s) \geq T_y(s)$. Apply Subroutine 1 to select a folding point (B', B'').

2. Let $f(x)$ be a monotonically increasing function such that $f(x) < x$. Let $K = \lfloor f(\min(N_x(B''), N_y(B')))\rfloor$, and let $J = \left\lfloor \frac{\min(N_x(B''),N_y(B'))-2K+1}{K} \right\rfloor$. If $K = 1$ or $J < 2$, then apply Algorithm $\mathcal{B}$. Otherwise, continue.

3. Configure the folding point and perform the following iteration

```
For $t = 1, 2, \ldots, K$
  If $t$ is odd
    Construct an $x$-superblock of length $J$ from $B''$ and
       an $y$-superblock of length $J$ from $B'$
    Configure the two superblock face-to-face in the current $x$-$y$ plane,
       moving *up*
    If $t < K$ then construct the connections to the next $x$-$y$ plane
       as illustrated in Figure 8a
  If $t$ is even
    Construct an $x$-superblock of length $J$ from $B''$ and
       an $y$-superblock of length $J$ from $B'$
    Configure the two superblock face-to-face in the current $x$-$y$ plane,
       moving *down*
    If $t < K$ then construct the connections to the next $x$-$y$ plane
       as illustrated in Figure 8b
```

Figure 9: **Algorithm $\mathcal{C}$**

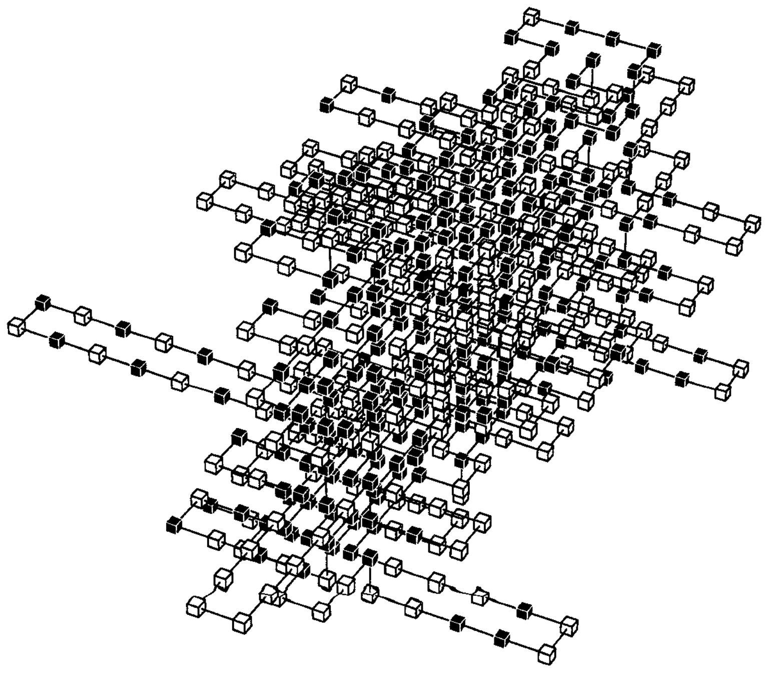


Figure 10: Application of Algorithm $\mathcal{C}$ to a sequence of length 500.

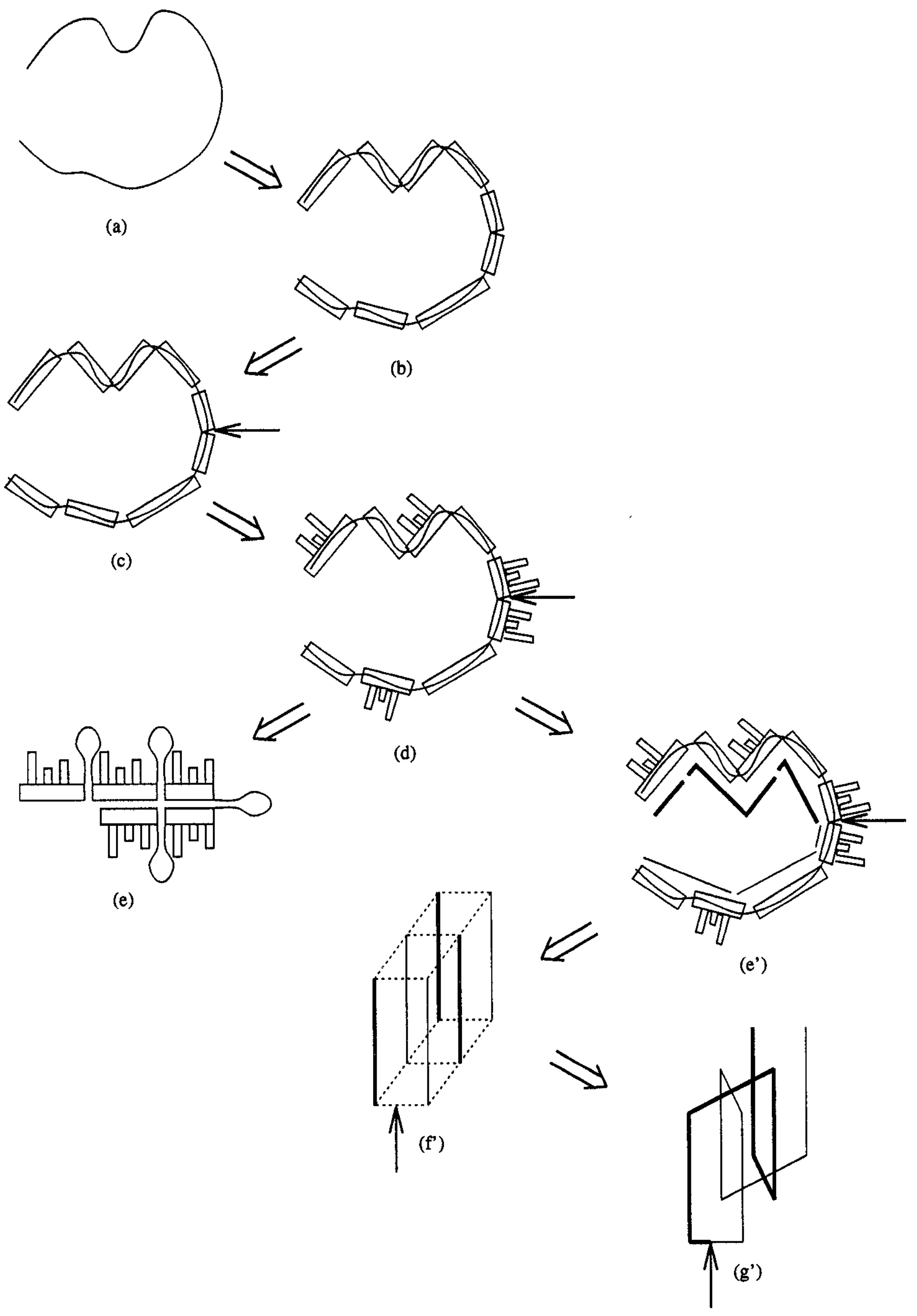


Figure 11: Folding pathway: Symbolic illustration of folding pathways consistent with Algorithms $\mathcal{A}$ and $\mathcal{C}$ (a) initial (random) conformation, (b) initial conformation decomposed into blocks (rectangular blocks), (c) conformation with identified folding point (at arrow), (d) conformation after block structures constructed for x-blocks and y-blocks (zero-loops represented by rectangles attached to blocks). For Algorithm $\mathcal{A}$, (e) final conformation of Algorithm $\mathcal{A}$ after block superstructures constructed for each half of the fold (loops represent block-separators and blocks that are removed from the superblocks' faces). For Algorithm $\mathcal{C}$, (e') blocks on both halves of the folding point are split into K superblocks (represented by the thin and thick dashed lines on the inside of the sequence), (f') superblock structures are aligned vertically on a rectangle in the x-z grid (dotted lines illustrate the three-dimensional configuration of the superblock structures), and (g') superblock structures are connected to form the final conformation.

step uses only short-range interactions. Figure 11c describes long-range interactions. This step balances the bondable hydrophobicity by finding a position in a sequence, called the folding point, which is a position between two consecutive blocks. The bondable hydrophobicity is measured by matching the number of 1's in the y-blocks on one half of the folding point with the number of 1's in the x-blocks on the other half of the folding point. Note that on each half of the folding point, only x-blocks or y-blocks are selected for bonding. In Figure 11d is shown how the block structure is constructed for each block that was selected.

Figure 11e finishes the fold for Algorithm $\mathcal{A}$ by aligning the block structures on each half of the folding point, creating a common x-face and, similarly, a common y-face. The unselected blocks are looped opposite the face to connect consecutive x-blocks and y-blocks respectively. Finally, the common faces are placed face-to-face and the block-separator between the x-face and y-face is folded in the obvious manner.

Figure 11e' illustrates how the two halves of the fold are split into superblocks by Algorithm $\mathcal{C}$. Figure 11f' shows the three-dimensional configuration of the faces of the superblocks such that the faces form a rectangular array on the x-z plane. Finally, figure 11g' shows how the faces of the superblocks are connected with connections that alternate between the top of the faces and the bottoms of the faces. The final conformation of each pair of superblocks in the rectangular array is analogous to the conformation illustrated in Figure 11e.

Acknowledgements

We thank Ron Unger for bringing this protein folding problem to our attention. We also thank David Lipman for encouragement and pointers to the literature.

References

[1] S. Arora, C. Lund, R. Motwani, and M. Sudan. Proof verification and intractability of approximation problems, 1994. Preliminary version.

[2] H. S. Chan and K. A. Dill. Origins of structure in globular proteins. In *Proc. Natl. Acad. Sci. USA*, volume 87, pages 6388–6392, August 1990.

[3] H. S. Chan and K. A. Dill. Polymer principles in protein structure and stability. *Annual Reviews of Biophysics and Biophysical Chemistry*, 20:447–490, 1991.

[4] T. E. Creighton, editor. *Protein Folding.* 1993.

[5] K. A. Dill. *Biochemistry*, 24:1501, 1985.

[6] K. A. Dill, November 1994. Personal communication.

[7] A. S. Fraenkel. Complexity of protein folding. *Bulletin of Mathematical Biology*, 55(6):1199–1210, 1993.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability - A guide to the theory of NP-completeness.* W.H. Freeman and Co., 1979.

[9] M. Karplus and D. L. Weaver. Diffusion-collision model for protein folding. *Biopolymers*, 18:1421, 1979.

[10] K. F. Lau and K. A. Dill. A lattice statistical mechanics model of the conformation and sequence spaces of proteins. *Macromolecules*, 22:3986–3997, 1989.

[11] K. F. Lau and K. A. Dill. Theory for protein mutability and biogenesis. In *Proc. Natl. Acad. Sci. USA*, volume 87, pages 638–642, January 1990.

[12] N. Linial, November 1994. Personal communication.

[13] D. Lipman and J. Wilber. *Proc. Royal Society of London*, 245(8), 1991.

[14] J. T. Ngo and J. Marks. Computational complexity of a problem in molecular structure prediction. *Protein Engineering*, 5(4):313–321, 1992.

[15] J. T. Ngo, J. Marks, and M. Karplus. *Computational Complexity: Protein structure prediction and the Levinthal paradox.* Birkhauser, 1994.

[16] R. H. Pain, editor. *Mechanisms of Protein Folding.* Oxford University Press, 1994.

[17] A. Sali, E. Shakhnovich, and M. Karplus. How does a protein fold? *Nature*, 369:248–251, 1994.

[18] P. Stolotz. Recursive approaches to the statistical physics of lattice proteins. In L. Hunter, editor, *Proc. of the 27th Hawaii Intl. Conf. on System Sciences*, pages 316–325. IEEE Computer Society Press, 1994. Vol. 5.

[19] R. Unger and J. Moult. Finding the lowest free energy conformation of a protein is a NP-hard problem: Proof and implications. *Bulletin of Mathematical Biology*, 55(6):1183–1198, 1993.

[20] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231(1):75–81, 1993.

[21] R. Unger and J. Moult. A genetic algorithms for three dimensional protein folding simulations. In *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA-93)*, pages 581–588. Morgan Kaufmann, 1993.