Protein Folding in the Hydrophobic-Hydrophilic (HP) Model is NP-Complete

Bonnie Berger* Tom Leighton†

Abstract

One of the simplest and most popular biophysical models of protein folding is the hydrophobic-hydrophilic (HP) model. The HP model abstracts the hydrophobic interaction in protein folding by labeling the amino acids as hydrophobic (H for nonpolar) or hydrophilic (P for polar). Chains of amino acids are configured as self-avoiding walks on the 3D cubic lattice, where an optimal conformation maximizes the number of adjacencies between H's. In this paper, the protein folding problem under the HP model on the cubic lattice is shown to be NP-complete. This means that the protein folding problem belongs to a large set of problems that are believed to be computationally intractable.

1 Introduction

One of the most widely studied models of protein folding is the hydrophobic-hydrophilic (HP) model introduced by Dill [6]. In the HP model, chains of amino acids are configured as self-avoiding walks on the 3D cubic lattice (*i.e.*, adjacent amino acids of each chain lie on adjacent lattice sites, and no site is occupied by more than one amino acid). Based on the assumption that the hydrophobic reactions make an important contribution to the free energy of the embedding, a protein is modeled as a specific sequence of hydrophobic (H for nonpolar) or hydrophilic (P for polar) monomers. An optimal conformation maximizes the number of adjacencies between H's.

The complexity of protein folding has remained open for the major models of biological interest [21, 4, 12]. Foremost among these models is the HP model [7] that was introduced by biophysicists in 1985. The protein folding problem for the HP model has been conjectured to be NP-complete [12], but a proof of NP-completeness has eluded researchers. NP-completeness proofs for the HP model have been sought not only for the purposes of identifying sources of computational complexity in protein structure prediction, but also because NP-completeness in this model would imply the computational intractability of computing the partition function. This is of fundamental importance for a statistical mechanics analysis of protein folding.

There have been many attempts to prove NP-completeness for related lattice models. Paterson and Przytycka [22] proved that the problem is NP-complete when the alphabet size is unbounded (instead of binary) and the goal is to maximize the number of like letters that are adjacent in the lattice. Unger and Moult [23] use a similar model, however the active subsequence is forced to lie on one straight line. Fraenkel [8, 9] proves NP-completeness for protein folding on a lattice with an alphabet of size three, but he embeds a graph rather than just a string. He has recently improved these results so that the graph represents a more realistic model of protein folding [10]. Ngo and Marks [20] show the more general problem of minimizing an energy function related to protein folding is NP-complete. However, their results do not take into account the compactness constraints of protein folding captured by the HP model. Also, their formulation contains a large number of parameters (the number of parameters grows with the length of the sequence) that are not constrained to lie within biologically plausible ranges. Hart and Istrail [13] generalized the results of Unger & Moult and Ngo & Marks to hold for a variety of lattices and energy functions. Proofs of NP-completeness have also been obtained for other related problems, such as protein threading [16, 2, 14]. More recently, Nayak *et al.* [19] have obtained NP-hardness and MAX SNP-hardness results for string folding on a lattice with a large finite alphabet. In independent work, Crescenzi *et al.* [5] have recently shown through different techniques that protein folding in the HP model on a two-dimensional lattice is NP-complete. The authors are currently working towards extending this result to three dimensions.

In this paper, we show for the first time that protein folding in the HP model on a three-dimensional lattice is NP-complete. In fact, we show that even the problem of deciding whether or not a sequence can be folded so

*2-389, Mathematics Dept. and Lab. for Computer Science, MIT, Cambridge, MA 02139; phone: 617-253-4986; email: bab@theory.lcs.mit.edu

†2-377,Mathematics Dept. and Lab. for Computer Science, MIT, Cambridge, MA 02139

RECOMB 98 New York NY USA

that the H's can be packed into a perfect cube (which is the minimum energy configuration for a collection of H's without any string constraints) is NP-complete. The proof uses a transformation from a special form of the BIN PACKING problem [11], which we call MODIFIED BIN PACKING. In particular, given an instance $\mathcal{B}$ of the MODIFIED BIN PACKING problem, we show how to construct a sequence S with n^3 H's so that there is a conformation of S where the H's form an $n \times n \times n$ cube if and only if there is a solution to $\mathcal{B}$.

Our NP-completeness proof relies heavily on the fact that nodes along the edges of a perfect cube have two or three neighbors not in the cube, whereas nodes on a face of a cube (but not on an edge) have just one neighbor not in the cube. The proof does not rely on parity arguments that are derived from the fact that the cubic lattice is bipartite. The NP-completeness proof does not immediately extend to general lattices, although our methods may be helpful in understanding the complexity of the protein folding problem in other models.

The discovery that protein folding in the HP model is NP-complete lends further importance to the study of approximation algorithms for protein folding. Hart and Istrail [12] have achieved approximation algorithms for the HP model on the cubic and square lattices. Their algorithms obtain an approximation factor of 3/8 for the cubic lattice and 1/4 for the square lattice. Recently, Hart and Istrail [15] have presented approximation algorithms for off-lattice and side chain variants of the HP model. Agarwala *et al.* [1] have shown that performance guarantees of roughly 60% can be achieved for the HP model on the hexagonal close packed lattice. This model has the advantage that there are no parity constraints imposed by the lattice.

The journal version of this paper appears in [3].

2 Problem Statement

The protein folding problem consists of embedding a given finite polypeptide sequence S of length N into a given fixed infinite graph G. In this paper, the graph G will primarily be the 3-dimensional cubic lattice $\mathcal{Z}^3$. A fold of S in G is an injective mapping from $[1,\ldots,N]$ to G such that adjacent integers map to adjacent nodes of G. Each node of $\mathcal{Z}^3$ has six neighbors. The energy of a fold of S in G to be minimized is the negation of the number of H–H bonds in the fold, where a bond is a pair of symbols mapped to adjacent nodes.

In the traditional way of looking at this problem, successive pairs of H's in S are not counted as forming a bond when computing energy; thus, a given H has at most four neighboring H's. For simplicity, we will include all H–H bonds in the energy calculation in this paper. Our choice of convention does not make any difference in terms of the NP-completeness result since the number of H–H bonds in the sequence does not depend in any way on the fold.

Another equivalent way of looking at the traditional problem is to minimize the number of non-H's among the four neighbors of each H. In this framework, our main result is that the problem of determining whether or not there is a fold in the HP-model on the 3-d lattice with a total of 8 non-H neighbors (the absolute minimum) is NP-hard. We achieve hardness of approximation results only for this latter problem. The hardness of $(9/8-\epsilon)$-approximation is straightforward. (We believe it is possible to extend our results to hardness of N^ϵ times optimal also, but we do not address this issue in the current paper.)

We define the following decision version of the protein folding problem.

HP STRING-FOLD

Instance: A finite sequence S over the alphabet $\{H,P\}$, an integer m, and a graph G.

Question: Is there a fold (*i.e.*, a self-avoiding walk) of S in G where the number of H–H bonds is at least m?

Our main result is that HP STRING-FOLD is NP-complete when G is $\mathcal{Z}^3$. The proof follows by showing that the following folding problem is NP-hard.

PERFECT HP STRING-FOLD

Instance: An integer n and a finite sequence S over the alphabet $\{H,P\}$ which contains n^3 H's.

Question: Is there a fold of S in $\mathcal{Z}^3$ for which the H's are perfectly packed into an $n \times n \times n$ cube?

The proof that PERFECT HP STRING-FOLD is NP-hard involves a transformation from the (strongly) NP-complete problem of BIN PACKING, which is defined as follows by Garey and Johnson [11].

BIN PACKING

Instance: A finite set U of items, a size $s(u) \in \mathcal{Z}^+$ for each $u \in U$, a positive integer bin capacity B, and a positive integer K.

Question: Is there a partition of U into disjoint sets $U_1, U_2, \ldots, U_K$ such that the sum of the sizes of the items in each U_i is B or less?

To simplify matters, we will use the following variation of BIN PACKING which is easily shown to be strongly NP-complete.

MODIFIED BIN PACKING

Instance: A finite set U of items, a size $s(u)$ that is a positive *even* integer for each $u \in U$, a positive integer bin capacity B, and a positive integer K, where $\sum_{u \in U} s(u) = BK$.

Question: Is there a partition of U into disjoint sets $U_1, U_2, \ldots, U_K$ such that the sum of the sizes of the items in each U_i is precisely B?

3 HP STRING-FOLD is NP-Complete

The proof that HP STRING-FOLD is NP-complete for cubic lattices is comprised of two main parts. In the first part (Section 3.1), we show that PERFECT HP STRING-FOLD is just a special case of HP STRING-FOLD. The proof is not difficult; it quickly follows from the fact that the minimum energy configuration of n^3 unrestricted H's in the cubic lattice is an $n \times n \times n$ cube.

In the second part (Section 3.2), we prove that PERFECT HP STRING-FOLD is NP-complete. This proof comprises the main result of the paper and is somewhat complicated. In order to aid the reader, we have endeavored to supply intuition where it may be helpful.

When the two parts are combined, we find that HP STRING-FOLD in the cubic lattice is NP-complete.[1]

[1] To be precise, we prove that the STRING-FOLD problems are NP-hard. NP-completeness follows from the additional simple fact that the problems are in NP.

3.1 *HP* STRING-FOLD is at Least as Hard as PERFECT *HP* STRING-FOLD

We first show that PERFECT *HP* STRING-FOLD is a special case of *HP* STRING-FOLD. This is accomplished by showing that the optimal configuration for a collection of unrestricted H's in the cubic lattice is uniquely achieved by packing the H's into a perfect cube. This fact may be known in the literature, but we supply a proof for completeness.

Fact 3.1 *Suppose there are n^3 H's in the sequence S for some n. Suppose also that there are no constraints imposed by the sequence; that is, adjacent sequence positions need not map to adjacent nodes in the cubic lattice G. Then the optimal conformation of S in G is uniquely achieved by packing the H's into an $n \times n \times n$ cube.*

Proof. Each of the n^3 H's has six neighbors in the lattice. When the H's are packed into an $n \times n \times n$ cube, all but $6n^2$ of these neighbors are themselves H's. (This is because there are precisely two potential H–H bonds missing on each of the n^2 X, Y, and Z lines of the cube.) Since each H–H bond is counted twice in the preceding formulation, this means that there are $(6n^3 - 6n^2)/2 = 3n^2(n-1)$ H–H bonds in the cubic conformation.

To show that the cubic conformation uniquely maximizes the number of H–H bonds, we will show that any other conformation is missing strictly more than $6n^2$ potential H–H bonds. Assume for the purposes of contradiction that the conformation of H's is not an $n \times n \times n$ cube and that it is missing at most $6n^2$ potential H–H bonds. Let A, B, and C denote the number of X, Y, and Z lines (resp.) that contain an H in the conformation. Then there are at least $2(A+B+C)$ total missing potential H–H bonds. By the assumption, this means that $2(A+B+C) \leq 6n^2$. By a classic result in geometry, the number of grid points p in a configuration with A, B, and C X, Y, and Z lines (resp.) is at most $\sqrt{A \cdot B \cdot C}$. Further, $p = \sqrt{A \cdot B \cdot C}$ only if the points are configured as a rectangular solid with face areas A, B, and C. The maximum of $\sqrt{A \cdot B \cdot C}$ given $A+B+C \leq 3n^2$ occurs when $A = B = C = n^2$. Hence $p \leq n^3$, with equality only if the points are configured as an $n \times n \times n$ grid. Since the H's are assumed to not form a cube, this means that there are fewer than n^3 H's, which is a contradiction. □

Theorem 3.2 *PERFECT HP STRING-FOLD is the special case of HP STRING-FOLD where S has n^3 H's, $m = 3n^2(n-1)$, and $G = \mathcal{Z}^3$.*

Proof. By Fact 3.1, we know that S can be folded with at least $m = 3n^2(n-1)$ H–H bonds in $\mathcal{Z}^3$ if and only if S can be folded so that its H's form a perfect $n \times n \times n$ cube. □

3.2 PERFECT *HP* STRING-FOLD is NP-complete

We next show that PERFECT *HP* STRING-FOLD is NP-complete. The reduction is from MODIFIED BIN PACKING. In particular, given a MODIFIED BIN PACKING problem $\mathcal{B}$, we will construct a sequence S for which its n^3 H's can be packed into the $n \times n \times n$ cube if and only if $\mathcal{B}$ is solvable. Since MODIFIED BIN PACKING is strongly NP-hard, this will prove that PERFECT *HP* STRING-FOLD is NP-hard. NP-completeness follows trivially since we can guess all possible conformations for any string and then check if the H's pack into a cube.

The proof consists of three parts. In the first part (Section 3.2.2), we show how to construct the desired string S from the bin packing problem $\mathcal{B}$. In the second part (Section 3.2.3), we show that if $\mathcal{B}$ has a solution, then S can be packed so that the H's form a perfect $n \times n \times n$ cube. In the third part (Section 3.2.4), we show that if S can be packed so that the H's form an $n \times n \times n$ cube, then $\mathcal{B}$ has a solution. The net effect of the last two parts is to show that any MODIFIED BIN PACKING problem can be transformed into a PERFECT *HP* STRING-FOLDING problem. Since MODIFIED BIN PACKING is known to be strongly NP-complete, then we can conclude that PERFECT *HP* STRING-FOLD is NP-hard.[2]

3.2.1 Intuition

As the reader will soon discover, the details of the reduction are fairly involved. This is because it takes some nontrivial amount of effort to encode a bin packing problem $\mathcal{B}$ into a string S in such a way that $\mathcal{B}$ is solvable if and only if the H's of S fold into a perfect cube. As an aid to the reader in understanding the reduction, we will therefore explain the basic intuition behind the proof before we get into the details.

The crux of the proof relies on the fact that if the H's in a string S are to be perfectly packed into a cube, then any H in S which is adjacent to a P in S must be packed onto the *surface* of the cube. This is because any H that is mapped into an internal node of the cube can only have H's for neighbors, and because adjacent items in S must be mapped to adjacent locations in the cube.

Taking this idea one step further, we can observe that the H's in a substring of the form

$$PHHP\,PHHP \cdots PHHP = (PHHP)^*$$

must be mapped to a connected path on the surface of the cube. This is because:

1. each H is adjacent to a P and so it must be placed on the surface (as argued above),
2. adjacent H's in S must, by definition, be adjacent in the cube, and
3. the H's in $HPPH$ must be adjacent because they are separated by a path of length 3 in the cube, 2 steps of which (HP and PH) must be orthogonal to the surface.[3]

For example, we have illustrated an embedding of $(PHHP)^3$ in Figure 1.[4]

[2] Technically speaking, we also need to be sure that the length of S is polynomial in the unary encoding of $\mathcal{B}$, but this fact is obvious from the construction.

[3] Here, we assume that the P's are not allowed to be on the surface since the cube is supposed to consist solely of H's. If we were allowed to place P's on the surface, then the H's in $HPPH$ would *not* have to be adjacent in the cube.

[4] We should note that Figure 1 is not fully general since the path of H's can move from one face of the cube to another. The path is always connected along the surface of the cube, however.

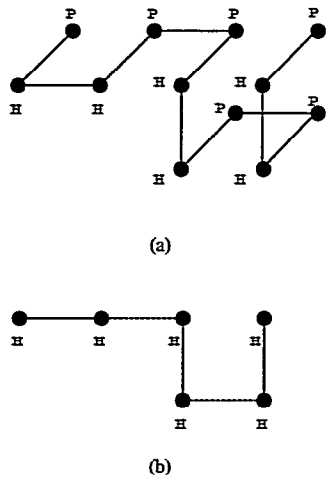

Figure 1: An embedding of $(PHHP)^3$ in the cubic lattice. In (a), the H's are on a surface of the cube and the P's reside distance 1 above the surface. The resulting connected path of H's is shown in (b). In (b), we use a dashed line to denote the existence of a path of two P's above the surface.

Similarly, we can observe that if the H's in a string S are to be perfectly packed in a cube, then the H's in a substring of the form

$$PHP\,PHP\cdots PHP = (PHP)^*$$

must be mapped to a connected path along the *edges* of the cube. (A node of the cube is on an edge if it lies on 2 or more faces.) This is because each H must be adjacent to *two* P's and the only nodes on the cube which are adjacent to two non-cube nodes are the nodes *along the edges*.[5] For example, see Figure 2.

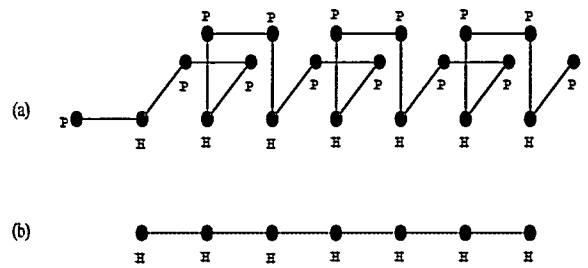

Figure 2: An embedding of $(PHP)^7$ in the cubic lattice. In (a), the H's are on an edge of the cube and the P's reside at distance 1 from the surfaces that define the edge. The resulting connected path of H's is shown in (b), whose dashed lines denote the existence of a path of two P's above the surfaces.

Our goal is to convert a MODIFIED BIN PACKING problem $\mathcal{B}$ into a string folding problem. We accomplish this task by creating a string S which contains substrings corresponding to the bins of $\mathcal{B}$ as well as the items of $\mathcal{B}$. These substrings are formed from specific combinations of $(PHP)^*$, $(PHHP)^*$, and P^*. This means that all the action (or difficulty in folding) will take place on the surface of the cube. In particular, the bins and items of $\mathcal{B}$ will correspond to substrings of S for which the H's are all embedded on a single face of the cube.

The most difficult part of the reduction is constructing substrings of S that correspond to the bins of $\mathcal{B}$. In particular, we need to show that if the H's of S pack perfectly into a cube, then the substrings of S corresponding to the bins must be mapped so as to partition the face of the cube into K "bins" each with B surface nodes. Insuring that no other fold is possible is what makes the proof difficult.

It is easier to create substrings corresponding to items of $\mathcal{B}$. Essentially, we will use a substring $(PHHP)^{s(u)/2}$ for each item u with size $s(u)$ in $\mathcal{B}$. Note that the H's in such a string must occupy precisely $s(u)$ nodes on the surface of the cube. Since these nodes correspond to a connected path on the surface, the path must lie entirely within one bin. This will mean that the "item substrings" can be perfectly packed on the surface if and only if the corresponding bin packing problem is solvable (which is the main goal).

In order to connect all the substrings together in S, we will use long runs of P's. This will give us the flexibility needed to place "item substrings" that are adjacent in S in bins that are far apart on the surface of the cube (as may be necessary in the bin packing solution). Of course, this leaves the problem of how to embed the strings of P's. We solve this problem by using methods developed in the related field of 3D VLSI wire routing.

We are now prepared to present the formal reduction. As anticipated, the string S will consist of many copies of PHP, $PHHP$, long runs of P's, and a single long run of H's. The role of PHP, $PHHP$, and P^* in S has been explained at a high level. The role of H^* is simply to fill up the internal nodes of the cube.

3.2.2 The Reduction

Consider any MODIFIED BIN PACKING problem $\mathcal{B}$. Let

$$\begin{aligned} q &= 2\max(K, \lceil B^{1/3} \rceil), \\ n &= q(2q+1)+2, \text{ and} \\ T &= (2q-1)(n-2), \end{aligned}$$

where B and K are bin packing parameters. Note that q, n, and T are all even integers and that $2(T-B) > T$. Define c to be a sufficiently large constant that will be specified later. The corresponding PERFECT HP STRING-FOLD problem consists of packing the string $S_{\mathcal{B}}$ defined as in Figure 3.

3.2.3 Using a Bin Packing to Fold $S_{\mathcal{B}}$

We first show how to fold the string $S_{\mathcal{B}}$ so that the H's form an $n \times n \times n$ cube provided that we are given a solution to the corresponding MODIFIED BIN-PACKING problem $\mathcal{B}$. The explanation is divided into four parts. First we show how to pack all but the front face of the cube of H's. Next we show how to construct the "walls" for the "bins" on the front face. Then we show how to pack the substrings corresponding to the items into the

[5] The fact that the H's form a connected path follows from the preceding discussion.

$$
\begin{aligned}
S_B \;=\; & (PHP)^{9n-10}\,(PHHP)^{(n-2)^2/2}\,(PHP)^{n-2}P^4 \\
& \left((PHHP)^{(n-2)^2/2}P^7\right)^3\,(PHHP)^{(n-2)^2/2-1}\,PH^{(n-2)^3+2}P^{8+q} \\
& \left((PHP)^q(PHHP)^{(n-2)/2}(PHP)^{2q+1}P^4(PHHP)^{(n-2)/2}(PHP)^qP^{2q}\right)^q\,PH(P^{2q+6}H)^{q-1}P \\
& \left(P^{cn}(PHHP)^{(T-B)/2}\right)^q\,\left(P^{cn}(PHHP)^{B/2}\right)^{q-K}\prod_{u\in U}\left(P^{cn}(PHHP)^{s(u)/2}\right)
\end{aligned}
$$

Figure 3: The construction. We use the notation $\prod_{i=1}^{r} S_i$ to denote the string $S_1|S_2|\cdots|S_r$.

bins. Finally, we show how to pack the long strings of P's that connect the "item" substrings in $\mathcal{S}$.

Packing All but the Front Face of the Cube

We begin by filling in each node in nine of the twelve edges of the cube with an H from the string $(PHP)^{9n-10}$. In particular, we fill nodes in the top face edges starting at position[6] $(n-2, n-1, n-1)$ with H's, move counter clockwise to position $(n-1, n-1, n-1)$, down an edge to position $(n-1, 0, n-1)$, and clockwise around the bottom face to position $(n-2, 0, n-1)$. (See Figure 4). The P's from $(PHP)^{9n-10}$ are embedded in nodes adjacent to the cube so as to keep the string connected. The first P is embedded in node $(n-2, n-1, n)$ and the last one is embedded in node $(n-2, 0, n)$. For example, see Figure 5.

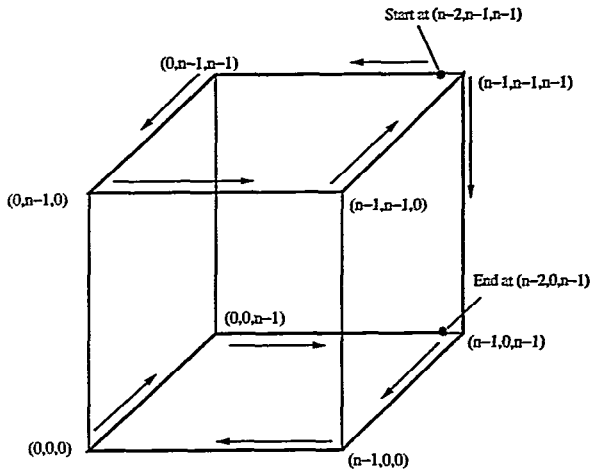

Figure 4: The $n \times n \times n$ cube with each corner position labeled with its coordinate triple (i, j, k). Arrows indicate the order in which the first nine edges are filled. The start and end nodes on this path are also indicated.

We next embed $(PHHP)^{(n-2)^2/2}$ so that the H's fill the back face of the cube in a vertical snake pattern. Note that because each dimension of the face $(n-2)$ is even, we will always end on the same side that we began. See for example Figure 6. We then embed $(PHP)^{n-2}$ so that the H's fill the $((0, 0, n-1), (0, n-1, n-1))$ edge of the cube (skipping the corners which are already filled). We next embed P^4 to get from node $(0, n-2, n)$ to node $(-2, n-2, n-2)$ so that we can enter the left face without hitting any previously filled nodes. We then fill in the left, bottom, right, and top faces sequentially with H's from $((PHHP)^{(n-2)^2/2}P^7)^3(PHHP)^{(n-2)^2/2-1}$, running the snake pattern so as to end up incident to the next face in the sequence as in Figure 7. (The P^7 term is used to get from the end of one face to the start of the next. We need seven P's since nodes along the edges of the cube have their two missing neighbors outside the cube already filled in.) The last two nodes $((n-2, n-1, 1)$ and $(n-3, n-1, 1))$ on the top face are left temporarily open in order to enter and exit the interior.

Figure 5: An example of how the $((0, n-1, n-1), (n-1, n-1, n-1))$ edge of the cube is filled in with $n-1$ PHP's. (The corner $(n-1, n-1, n-1)$ is filled as part of the $((n-1, 0, n-1), (n-1, n-1, n-1))$ edge of the cube.) The first P is in position $(n-2, n-1, n)$, and the first H is in position $(n-2, n-1, n-1)$. The path moves right to left, with only H's labeled in this figure. The last H is in position $(0, n-1, n-1)$ and the last P is in $(-1, n-1, n-1)$.

The interior of the cube is reached through the next to last top face node $(n-3, n-1, 1)$, which is now filled in with an H from PH. The interior of the cube is then filled with $(n-2)^3$ H's in a truncated snake pattern alternating direction each layer, leaving the $((n-2, 0, 1), (n-2, n-1, 1))$ column open until the end when it is used to move back to the node $(n-2, n-1, 1)$, which is embedded with the last H from $H^{(n-2)^3+2}$. Then $8+q$ P's are used to take us to position $(n-1, n-1-q, -2)$. (We use $8+q$ P's so as to route around previously embedded P's.)

Packing the Front Face: Making the Bins

At this point, the entire cube is filled with H's except for the $n \times (n-2)$ region on the front face. We next embed q sets of "bin walls" on this face as follows. The walls for each bin are filled in with the H's from the

[6] Henceforth, we will refer to each position in the $n \times n \times n$ cube by a coordinate triple (i, j, k), where $0 \le i, j, k \le n-1$.

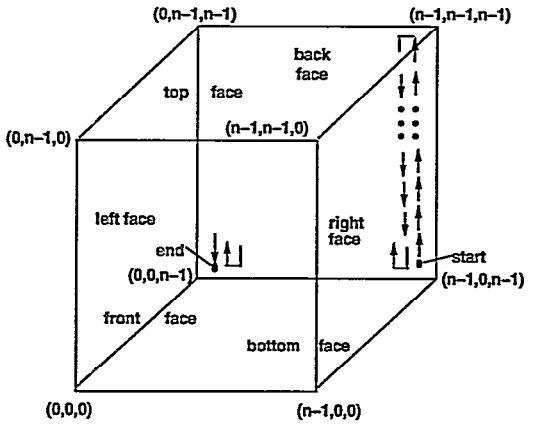

Figure 6: The $n \times n \times n$ cube with the back, left, bottom, right, top, and front faces labeled. The back face is filled with $(n-2)^2/2$ $PHHP$'s. The path indicates the order in which it is filled, starting at node $(n-2,1,n-1)$ and ending at node $(1,1,n-1)$. Notice that the path through the face ends on the same side it began because n is even. The P's are not shown but would appear diagonal to this face. There is an arrow on the $((0,0,n-1),(0,n-1,n-1))$ edge of the cube, which is filled in immediately after the back face, to indicate the order in which it is filled.

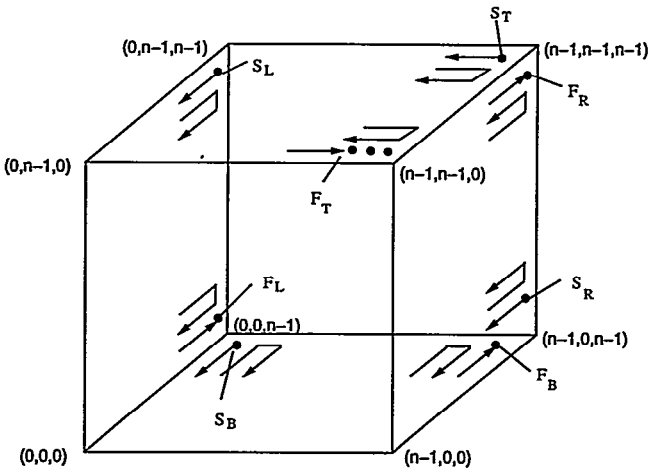

Figure 7: The $n \times n \times n$ cube with paths to indicate the order in which the left, bottom, right, and top faces are filled. The start and end nodes of each face are labeled as follows: left face, $S_L = (0, n-2, n-2)$, and $F_L = (0, 1, n-2)$; bottom face, $S_B = (1, 0, n-2)$, and $F_B = (n-2, 0, n-2)$; right face, $S_R = (n-1, 1, n-2)$, and $F_R = (n-1, n-2, n-2)$; top face, $S_T = (n-2, n-1, n-2)$, and $F_T = (n-4, n-1, 1)$.

string

$$\Big((PHP)^q(PHHP)^{(n-2)/2}(PHP)^{2q+1}P^4(PHHP)^{(n-2)/2}(PHP)^qP^{2q}\Big)^q,$$

as shown in Figure 8. In particular, each bin wall consists of a path of q H's embedded along an edge (called a *q-string*), followed by a path of $n-2$ H's embedded in a straight line across the face, followed by a path of $2q+1$ H's embedded along the other edge (called a $(2q+1)$-*string*), followed by a path of $n-2$ H's embedded back across the face, followed by a path of q H's embedded along the original edge. (Note that since $q+1$ is odd, four P's are needed to connect the $(2q+1)$-string to the second horizontal path across the face. In addition, we need a path of $2q$ P's to connect the end of one bin wall to the beginning of the next bin wall. This path runs at distance 2 from the cube so as not to intersect P's used for the edge segments of the bin walls.)

The bin walls are completed with H's from $PH(P^{2q+6}H)^{q-1}P$ (called a *special q-string*), as shown in Figure 8. There will be one H used to finish off each of the q bin walls. These H's are separated by $2q$ rows in the face, and we use $2q+6$ P's to connect them since we need to route the path at distance 3 from the cube in order to avoid previously embedded P's (from the q-strings and the paths that connect q-strings). (Note that the $2q$ P's between the last q-string and the first H of the special q-string are really not needed, but are included to simplify the expression for S_B. They can be embedded by moving away distance q from the cube and then returning.) The last P in the special q-string is embedded at node $(n-1, n-2-q, -1)$.

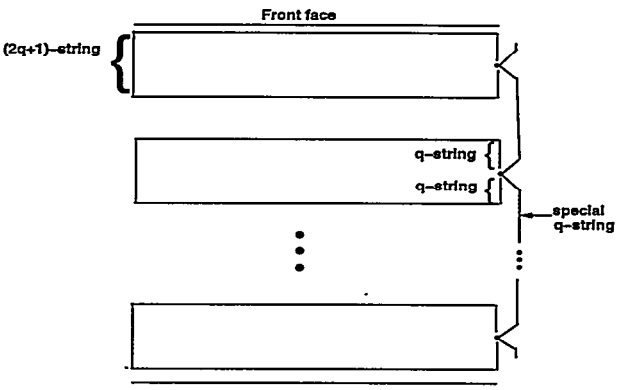

Figure 8: The front face of the cube partitioned into q "bins." The top and bottom edges of the face have previously been filled in.

Packing the Front Face: Placing the Items

At this point, the nodes on the front face that have not been filled have been partitioned into q $(n-2) \times (2q-1)$ rectangular regions that we will refer to as bins $B_1, \ldots, B_q$. We next fill these bins with the remaining H's from S_B.

The remainder of S_B consists of sequences of the form

$$P^{cn}(PHHP)^{x/2}$$

which we call an *x-item-string*. In particular, the remainder of S_B consists of q $(T-B)$-item-strings, $q-k$ B-item strings, and one $s(u)$-item-string for each item $u \in U$ in $\mathcal{B}$.

We will embed one $(T-B)$-item-string in each bin and one B-item-string in each of bins $B_{K+1}, \ldots, B_q$. This will leave us with K bins each with B unfilled nodes. These nodes are filled with the $s(u)$-item-strings using the solution to $\mathcal{B}$ as a guide. In particular, if

$u \in U_i$ in the BIN PACKING solution, then we embed an $s(u)$-item-string in B_i. As a result, every face node will be filled.

The item-strings are embedded so as to form a snake-like pattern within each bin. The exact order or location of the item-strings within the bins does not matter, provided only that the first H of each item-string is embedded in an even node of the cube. (A node (i,j,k) is said to be *even* if and only if $i+j+k \equiv 0 \bmod 2$.) Since each item-string has an even number of H's, we will then be assured that the last H of each item-string is embedded in an odd node of the cube, and that each item-string will consume an equal number of even and odd cube nodes.

Packing the Connecting Strings of P's

The remaining difficulty is to connect the last H in one item-string with the first H in the next item-string in S_B (which might be located in a bin that is far away in the cube). The connection is made by a path of $cn+2$ P's. Note that this path of P's has the correct parity (even) since we are connecting H's on the surface of the cube with differing parities.

The problem of embedding the connecting paths is equivalent to the problem of connecting pairs of endpoints on a 2-dimensional surface with wires that run in a nonintersecting fashion through a 3-dimensional grid that lies above (and beyond) the surface. Moreover, we require that the wires all have the same length $cn+2$ since they correspond to substrings consisting of $cn+2$ P's.

Using methods from 3D VLSI wire routing [18], we can find a way to route the connecting paths for any collection of pairs of endpoints on the surface. In fact, the solution can be considered to be well-known except for the constraint that all the paths have the same length $cn+2$. In this situation, we will route the paths using nodes of the form (i,j,k) where $k < 0$. (Note that no nodes with $k < -3$ have been used thus far nor have any nodes of the form (i,j,k) been used where $k < 0$ and $(i,j,0)$ is used for a string-item.)

In order to route the connecting paths in a non-intersecting fashion, we will use the nodes with $k < -3$ to form a 3-dimensional crossbar switch. The inputs to the switch can be thought of (without loss of generality) as nodes with $0 \le i,j \le n-1$ and $k = -4$. The outputs will be at nodes with $0 \le i,j \le n-1$ and $k = -10n+2$. The switch can connect the inputs to the outputs in any permutation with node-disjoint paths. The construction of the paths for any fixed permutation is provided in the following lemmas, most of which are well known in the VLSI routing literature.

Lemma 3.3 *Given a 2-dimensional $h \times (2h-1)$ array with nodes $\{(i,j) \mid 0 \le i < h,\ 0 \le j < 2h-1\}$, it is possible to route non-intersecting paths of length at most $2(h-1)$ from node $(0,j)$ to node $(h-1,2j)$ for $0 \le j < h$.*

Proof. Immediate from the example shown in Figure 9. □

Lemma 3.4 *Given a 3-dimensional $(2h-1)\times(2h-1)\times(2h-1)$ array, it is possible to route non-intersecting paths of length at most $4(h-1)$ from node $(0,j,k)$ to node $(2h-2,2j,2k)$ for $0 \le j,k < h$.*

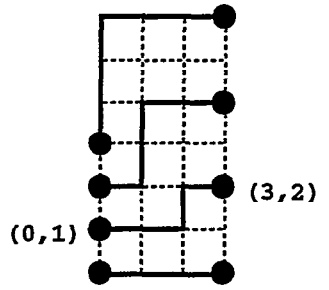

Figure 9: Routing non-intersecting path of length at most $2(h-1)$ from $(0,j)$ to $(h-1,2j)$ for $0 \le j < h$ in an $h \times (2h-1)$ grid for $h = 4$. Note that no vertical segments are used in the last column.

Proof. We first route paths of length at most $2(h-1)$ from $(0,j,k)$ to $(h-1,2j,k)$ using Lemma 3.3, and then of length at most $2(h-1)$ from $(h-1,2j,k)$ to $(2h-2,2j,2k)$ using Lemma 3.3 in the other dimension. □

Lemma 3.5 *Given a $2h \times (3h-2) \times 2$ grid, it is possible to route non-intersecting paths of length at most $6h-5$ from node $(0,2j,0)$ to node $(2h-1,2\pi(j),0)$ for $0 \le j < h$, where π is any permutation on $[0,h-1]$.*

Proof. As illustrated in Figure 10, each path but one uses two columns and one row above row $2h-2$. The remaining path uses just one column and no row above row $2h-2$. □

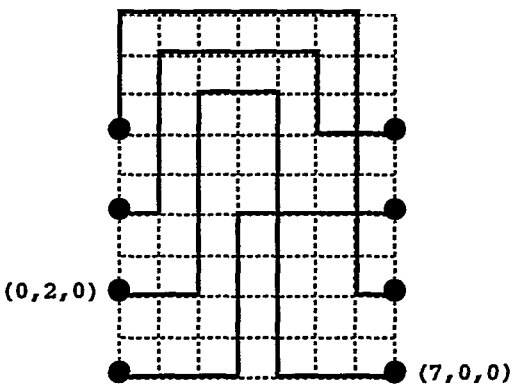

Figure 10: Routing non-intersecting paths of length at most $6h-5$ from $(0,2j,0)$ to $(2h-1,2\pi(j),0)$ for $0 \le j < h$ in a $2h \times (3h-2) \times 2$ grid where $h = 4$ and $\pi = (0\ 2\ 3\ 1)$. Horizontal segments are routed in the bottom layer (i.e., the third coordinate is 0) and vertical segments are routed in the top layer (i.e., the third coordinate is 1). Note that no vertical segments are used in the last column, so two such routings could be placed side by side without conflict.

Lemma 3.6 *The elements of an $h \times h$ matrix can be permuted in an arbitrary fashion by first permuting the elements within each row, then permuting the elements within each column, and then once again permuting the elements within each row.*

Proof. See Theorem 1.16 in Leighton [17]. □

Lemma 3.7 *Given a* $(6h-2)\times(3h-2)\times(3h-2)$ *grid, it is possible to route non-intersecting paths of length at most* $18h-15$ *from node* $(0,2j,2k)$ *to node* $(6h-3,2\pi_1(j,k),2\pi_2(j,k))$ *for* $0\le j,k<h$ *where* $\pi=(\pi_1,\pi_2)$ *is any permutation on* $[0,h-1]\times[0,h-1]$.

Proof. We use Lemma 3.6 to express π as a collection of permutations within the rows, followed by a collection of permutations within the columns, followed by a collection of permutations within the rows. The first collections of row permutations is routed in nodes (i,j,k) with $0\le i\le 2h-1$ using Lemma 3.5 for each row. The column permutations are routed in nodes with $2h-1\le i\le 4h-2$ using Lemma 3.5 for each column. The final set of row permutations is routed in nodes with $4h-2\le i\le 6h-3$, again using Lemma 3.5 for each row. (Note that we have spaced the endpoints of the paths so as to allow each row/column permutation to have dedicated access to a two-layer grid. Note also that we can use nodes with $i=2h-1$ for both row and column permutations since edges on those levels are used only by the column permutations. The nodes with $i=4h-2$ can be reused in a similar fashion.) The length of each path is then $3(6h-5)=18h-15$. □

Lemma 3.8 *Given a* $(10h-6)\times(3h-2)\times(3h-2)$ *grid, it is possible to route non-intersecting paths of length at most* $26h-23$ *from node* $(0,j,k)$ *to node* $(10h-6,\pi_1(j,k),\pi_2(j,k))$ *for* $0\le j,k<h$ *where* $\pi=(\pi_1,\pi_2)$ *is any permutation on* $[0,h-1]\times[0,h-1]$.

Proof. We first use Lemma 3.4 to connect $(0,j,k)$ to $(2h-2,2j,2k)$ using a path of length at most $4(h-1)$. We then use Lemma 3.7 to connect $(2h-2,2j,2k)$ to $(8h-5,2\pi_1(j,k),2\pi_2(j,k))$ using a path of length at most $18h-15$. (Note that the nodes with $i=2h-2$ play the role of nodes with $i=0$ in Lemma 3.7.) Then we use Lemma 3.4 again to connect $(8h-5,2\pi_1(j,k),2\pi_2(j,k))$ to $(10h-7,\pi_1(j,k),\pi_2(j,k))$ using a path of length at most $4(h-1)$. □

We are now ready to embed the paths of length $cn+2$ that connect the item-strings. We first use four P's for each end of each item-string to get to distance four from the front face of the cube (i.e., so $k=-4$). We then use Lemma 3.8 with $h=n$ to permute the ends of the item-strings so that endpoints that are adjacent in S_B will be adjacent on the plane of the lattice with $k=-10n+2$. At this point, we could complete the connections for each path by adding a single edge. Overall, each path would have length at most

$$2(26n-23+4)+1=52n-37.$$

We must insure that each path has length precisely $cn+2$, however. This can be accomplished by setting $c=52$ and routing each path as far as it needs to go away from the cube in a loop in order that it will have length precisely $52n+2$. (Of course, we need to be sure that the parity of the path extension is correct, but this has already been assured by the embedding of the item-strings.)

This concludes the proof that S_B can be embedded in an $O(n)\times O(n)\times O(n)$ cube so that the H's form an $n\times n\times n$ cube, provided that $\mathcal{B}$ is solvable.

3.2.4 Using a Fold of S_B to Solve BIN PACKING

We next show how to solve the MODIFIED BIN PACKING problem $\mathcal{B}$ provided that we are given a solution to the PERFECT HP STRING-FOLD problem for S_B. Throughout, we assume that the sequence S_B is folded so that the H's form a perfect $n\times n\times n$ cube. The following simple facts, which were proved above, are crucial to our analysis.

Fact 3.9 *Any single* H *surrounded by* P*'s in* S_B *(e.g.,* PHP*) must be embedded in a node that is on the edge of the cube.*

Fact 3.10 *Any* H *that is adjacent to a* P *in* S_B *(e.g.,* HP*) must be embedded on a face (including its edges) of the cube.*

Fact 3.11 *If a pair of* H*'s are separated by two* P*'s in* S_B *(e.g.,* $HPPH$*), then the* H*'s are embedded in adjacent positions on a face of the cube.*

The explanation is divided into two parts. First we show how the edges of the cube must be filled with H's from S_B. This is the part of the proof where we argue that certain substrings of S_B must fold so as to form bins of a fixed size on the surface of the cube. In the second part, we show that the "item substrings" in S_B must be packed on the surface of the cube so as to provide a solution to the bin packing problem $\mathcal{B}$.

Forming the Bins

By the construction of S_B, there are $12(n-2)+8=12n-16$ single H's surrounded by P's, which is precisely the number of nodes on the edges of the cube. The manner in which these H's are mapped to nodes on the edges of the cube is highly constrained by S_B, and it is critical to the proof. For example, by Facts 3.9 and 3.11, the $9n-10$ H's in the substring $(PHP)^{9n-10}$ must be mapped to a contiguous path of nodes along the edges of the cube. There are several ways that this can be accomplished, but each such mapping must leave at least one neighbor of at least six corner nodes unfilled. This is because corner nodes have an odd number of neighbors on the edges and a simple path can only fill 0 or 2 of them unless the path ends at a neighbor. Since the path can only have two endpoints and since it is so long, it must fill all eight corner nodes as well as every node in a total of nine edges of the cube. Moreover, the three remaining unfilled edges of the cube (unfilled except for their endpoints, that is) must not share a corner. The nodes along one of the remaining three edges must then be filled with H's from the substring $(PHP)^{n-2}$. This leaves precisely $n-2$ nodes unfilled for each of two nonadjacent edges of the cube.

By the definition of S_B, the remaining two edges of the cube must be filled with H's from substring

$$\Big((PHP)^q(PHHP)^{(n-2)/2}(PHP)^{2q+1}P^4(PHHP)^{(n-2)/2}$$
$$(PHP)^qP^{2q}\Big)^q PH(P^{2q+6}H)^{q-1}P.$$

Note that the single H's in this string can be naturally grouped into three classes as follows: *q-strings* of the

form $(PHP)^q$, $(2q+1)$-*strings* of the form $(PHP)^{2q+1}$, and a *special q-string* of the form $PH(P^{2q+6}H)^{q-1}P$. There are $2q$ q-strings, q $(2q+1)$-strings, and 1 special q-string in S_B. Since the corners of the cube are already filled, it is clear that each q-string and $(2q+1)$-string must be mapped to nodes within a single edge of the cube. In addition, since $2q+6 < n$ and since the unfilled edges are nonadjacent, the points in the special q-string must also be mapped to nodes within a single edge. The following lemma will help us characterize more precisely how the q-strings, $(2q+1)$-strings, and the special q-string are mapped to the two unfilled edges.

Lemma 3.12 *If an edge is perfectly packed with q-strings and $(2q+1)$-strings, then it is perfectly packed with only $(2q+1)$-strings.*

Proof. Suppose for the purposes of contradiction that we could perfectly pack the edge with i q-strings and j $(2q+1)$-strings, where $i > 0$. Then $iq + j(2q+1) = n-2 = q(2q+1)$, which implies that $j \equiv 0 \bmod q$. If $j = 0$, then $i = 2q+1$, which is a contradiction since there are only $2q$ q-strings. If $j \geq q$, then $i = 0$, which is also a contradiction. Hence, the edge is perfectly packed by only $(2q+1)$-strings. □

As a consequence of Lemma 3.12, we know that the $n-2$ nodes in one of the unfilled edges of the cube get filled with H's from the q $(2q+1)$-strings. The $n-2$ nodes in the other unfilled edge get filled with H's from the $2q$ q-strings and the special q-string.

We can also deduce that the two edges are on opposite sides of the same face of the cube. This is because each $(2q+1)$-string is separated from a q-string by $(PHHP)^{(n-2)/2}$, which accounts for distance $n-2$ between the respective edges. Since the edges cannot share a corner, they must be on opposite sides of the same face.

In fact, we can say much more about the manner in which the H's from

$$\Big((PHP)^q(PHHP)^{(n-2)/2}(PHP)^{2q+1}P^4(PHHP)^{(n-2)/2}(PHP)^qP^{2q}\Big)^q$$

are packed. First, the $(2q+1)$-strings form a partition of one edge into q intervals of length $2q+1$. Second, the strings of the form $(PHHP)^{(n-2)/2}$ that surround each $(2q+1)$-string correspond to straight lines through the face containing the opposing edges. This is because the only way to get between two opposite edges with a path of length $n-2$ H's is by a straight line.[7] Hence, the H's in

$$\Big((PHP)^q(PHHP)^{(n-2)/2}(PHP)^{2q+1}P^4(PHHP)^{(n-2)/2}(PHP)^qP^{2q}\Big)^q PH(P^{2q+6}H)^{q-1}P$$

must partition a face of the cube into q rectangular regions (called "bins"), each containing $(n-2)(2q-1)$ unused face nodes. For example, see Figure 8.

Packing the Items into Bins

At this point in our analysis, we have filled all the nodes in the edges of the cube as well as some of the nodes in a face of the cube. As a result, the surface of the cube has been partitioned into five regions (faces) of size $(n-2)^2$ and q regions (bins) of size $T = (n-2)(2q-1)$. Into these regions, we must pack all the H's from four copies of $(PHHP)^{(n-2)^2/2}$, one copy of $(PHHP)^{(n-2)^2/2-1}$, q copies of $(PHHP)^{(T-B)/2}$, $q-K$ copies of $(PHHP)^{B/2}$, as well as one copy of $(PHHP)^{s(u)/2}$ for each item $u \in U$. We also need to pack two H's from $H^{(n-2)^3+2}$ into the remaining face nodes.

Since the H's in any string of the form $(PHHP)^*$ must form a contiguous path on the surface of the cube (Fact 3.11), they must all be packed into the same face or bin. This constraint severely restricts our options for locating the H's. For example, the five empty faces must become filled with H's from the four copies of $(PHHP)^{(n-2)^2/2}$, the single copy of $(PHHP)^{(n-2)^2/2-1}$, and (without loss of generality) two H's from $H^{(n-2)^3+2}$. (These strings are too long to fit in the bins.) Each of the q bins gets one of the q copies of $(PHHP)^{(T-B)/2}$. (This is because the bins are not large enough to contain two copies since $2(T-B) > T$.) In addition, $q-K$ bins get one copy of $(PHHP)^{B/2}$, which completes the filling of those bins.

At this point, we are left with K nonempty bins, each with B available face nodes, and one copy of $(PHHP)^{s(u)/2}$ for each item $u \in U$. Since $\sum_{u \in U} s(u) = BK$, these strings must pack perfectly into the bins and so the packing of the string into the cube provides a solution to the MODIFIED BIN PACKING problem.

4 Discussion

In this paper, we have shown that protein folding in the HP model is NP-complete. We have thus settled the recurring open question about the complexity of protein folding in this model [21].

Our work also complements existing efforts to characterize various hardness aspects of protein folding [20, 21, 13, 9, 23, 22]. In particular, our proof shows that any reasonably fast protein folding algorithm will have to rely on aspects other than just hydrophobicity considerations.

Our methods do not apply to the 2D square lattice, although the 3D cubic model that we study seems to be more relevant to the actual protein folding problem. Nor does our proof directly apply to the 3D tetrahedral lattice, which has been proposed because it avoids

[7] To be very precise, we need to argue that the first H in $P^4(PHHP)^{(n-2)/2}$ must be adjacent to the last H in the preceding $(2q+1)$-string. In fact, the argument is somewhat delicate since it is not generally true that H's which are separated by six P's in S_B need to be adjacent in the embedding. However, in this case, we know that the P's from the $(2q+1)$-strings occupy every node of the form $(0, j, -1)$ where $1 \leq j \leq n-2$. Moreover, the last P for each $(2q+1)$-string is forced to occupy a node of the form $(-1, j, 0)$. In order to get from $(-1, j, 0)$ to the front face without intersecting the cube or the P's in nodes $(0, *, -1)$, we must first go to node $(-1, j, -2)$, and then to $(1, j, -2)$ and then to $(1, j, 0)$, which requires six P's overall. In fact, this is the only non-edge face node that is legally reachable by six or fewer P's from $(0, j, 0)$, given that the first P must reside at $(-1, j, 0)$.

the parity problem of the cubic lattice. Although our methods do not rely on parity arguments, they fail because we cannot utilize as easily the different number of missing neighbors between edge and face nodes in the minimum configuration arrangement.

5 Acknowledgements

Many thanks to Serafim Batzoglou, Sorin Istrail, Esther Jesurum, and Lior Pachter for helpful comments, and to Sorin Istrail for posing this problem to us. Thanks are also due to anonymous referees for their helpful comments. B.B. is partially supported by an NSF Career Award. B.B. and T.L. are partially supported by ARPA contract N00014-95-1-1246 and ARO grant DAAH04-95-1-0607.

References

[1] R. Agarwala, S. Batzoglou, V. Dančík, S. Decatur, M. Farach, S. Hannenhalli, S. Skiena, and S. Muthukrishnan. Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model. *J. Computational Biol.*, 4(3):275–296, Fall 1997.

[2] T. Akutsu and S. Miyano. On the approximation of protein threading. In *Proc. First Annual Int. Conf. on Computational Molecular Biol. (RECOMB)*, pages 3–8, Santa Fe, NM, jan 1997. ACM.

[3] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *J. Computational Biol.*, Spring 1998. In press.

[4] J. D. Bryngelson, J. N. Onuchic, N. D. Socci, and P. G. Wolynes. Funnels, pathways, and the energy landscape of protein folding: a synthesis. *PROTEINS: Structure, Function, and Genetics*, 21:167–195, 1995.

[5] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. In *Proc. 2nd Annual Int. Conf. on Computational Molecular Biology (RECOMB)*, New York City, NY, March 1998. ACM. To appear.

[6] K. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24:1501–1509, 1985.

[7] K. Dill, S. Bromberg, K. Yue, K. Fiebig, D. Yee, P. Thomas, and H. Chan. Principles of protein folding: A perspective from simple exact models. *Protein Science*, 4:561–602, 1995.

[8] A. Fraenkel. Deexponentializing complex computational mathematical problems using physical or biological systems. Technical Report CS90-30, Weizmann Inst. of Science, Dept. of Applied Math and Computer Science, 1990.

[9] A. Fraenkel. Complexity of protein folding. *Bull. Math Biol.*, 55:1199–1210, 1993.

[10] A. S. Fraenkel. Protein folding, spin glass and computational complexity. In *Third Annual DIMACS Workshop on DNA Based Computers*, Philadelphia, PA, June 1997. To appear in proceedings as an invited paper.

[11] M. Garey and D. Johnson. *Computers and Intractability.* Freeman, New York, 1979.

[12] W. Hart and S. Istrail. Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal. *J. Computational Biol.*, 3(1):53–96, Spring 1996.

[13] W. Hart and S. Istrail. Robust proofs of NP-hardness for protein folding: General lattices and energy potentials. *J. Computational Biol.*, 4(1):1–22, Spring 1997.

[14] W. E. Hart. On the computational complexity of sequence design problems. In *Proc. First Annual Int. Conf. on Computational Molecular Biology (RECOMB)*, pages 128–136, Santa Fe, NM, jan 1997. ACM.

[15] W. E. Hart and S. Istrail. Lattice and off-lattice side chain models of protein folding: Linear time structure prediction better than In *Proc. First Annual Int. Conf. on Computational Molecular Biology (RECOMB)*, pages 137–146, Santa Fe, NM, jan 1997. ACM.

[16] R. H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is np-complete. *Protein Engineering*, 7:1059–1068, 1994.

[17] T. Leighton. *Introduction to Parallel Algorithms and Architechtures: Arrays · Trees · Hypercubes*, chapter Arrays and Trees, page 190. Morgan Kaufmann, San Mateo, CA, 1992.

[18] T. Leighton and A. Rosenberg. Three-dimensional circuit layouts. *SIAM J. Computing*, 15(3):793–813, 1986.

[19] A. Nayak, A. Sinclair, and U. Zwick. Spatial codes and the hardness of string folding problems. In *Proceedings of the Ninth Annual ACM–SIAM Symposium on Discrete Alg orithms*, Philadelphia, January 1998. ACM/SIAM. To appear.

[20] J. Ngo and J. Marks. Computatinal complexity of a problem in molecular structure prediction. *Protein Engineering*, 5:313–321, 1992.

[21] J. Ngo, J. Marks, and M. Karplus. *The Protein Folding Problem and Tertiary Structure Prediction*, chapter Computational complexity, protein structure prediction, and the Levinthal paradox. Birkhauser, Basel, 1994. Edited by K.M. Merz and S.M. LeGrand.

[22] M. Paterson and T. Przytycka. On the complexity of string folding. *Discrete Applied Mathematics*, 71:217–230, 1996.

[23] R. Unger and J. Moult. Finding the lowest free energy conformatoin of a protein is an np-hard prblem:proof and implications. *Bull. Math. Biol.*, 55:1183–1198, 1993.