Spatial Codes and the Hardness of String Folding Problems

(Extended Abstract)

Ashwin Nayak * *Alistair Sinclair* † *Uri Zwick* ‡

Abstract

We present the first proof of **NP**-hardness (under randomized polynomial time reductions) for string folding problems over a *finite* alphabet. All previous such intractability results have required an unbounded alphabet size. These problems correspond to the protein folding problem in variants of the hydrophobic-hydrophilic (or HP) model with a fixed number of monomer types. Our proof also establishes the **MAX SNP**-hardness of the problem (again under randomized polynomial time reductions). This means that obtaining even an approximate solution to the protein folding problem, to within some fixed constant, is **NP**-hard. Our results are based on a general technique for replacing unbounded alphabets by finite alphabets in reductions for string folding problems. This technique has two novel aspects. The first is the essential use of the approximation hardness of the source problem in the reduction, even for the proof of **NP**-hardness. The second is the concept of *spatial codes*, a variant of classical error-correcting codes in which different codewords are required to have large "distance" from one another even when they are arbitrarily embedded in three-dimensional space.

1 Introduction

1.1 Synopsis

This paper is concerned with string folding problems of the following type. We are given as input a string or a set of strings over some alphabet. An *embedding* is a mapping of the strings into some given infinite regular lattice (typically the 3-dimensional rectangular grid $\mathcal{Z}^3$) so that adjacent symbols of each string lie on adjacent lattice sites, and no site is occupied by more than one symbol. The *score* of an embedding is the number of pairs of equal symbols that lie at adjacent lattice sites (excluding pairs that are adjacent in the strings themselves). Figure 1 shows an example embedding in $\mathcal{Z}^2$ of a single string over the alphabet $\{0, 1\}$ with a score of four. Our task is to find an embedding of the strings that maximizes the score.

The motivation for these problems comes mainly from computational biology. One of the principal challenges in this field is to infer the 3-dimensional native structure of a protein (or a collection of proteins) from its amino acid sequence. This problem has been investigated under a wide variety of models, each of which attempts to emphasize different aspects of the problem. Perhaps the simplest, and combinatorially most appealing of the widely studied models is the so-called "hydrophobic-hydrophilic" model, or *HP model* of Dill [5, 6]. Here a protein is represented as a string over the two-letter alphabet $\{\mathrm{H}, \mathrm{P}\}$, with the symbol H representing hydrophobic monomers and P hydrophilic (or polar) monomers. Conformations of the protein correspond to embeddings of the string in $\mathcal{Z}^3$ (a discretization of 3-dimensional space). The folded state of the protein is the embedding which maximizes the number of nearest-neighbor H-H contacts; this corresponds to the minimum energy conformation under the assumption that hydrophobic interactions are the dominant contribution to the free energy of the protein. Thus protein folding in the HP model corresponds to our string folding problem over the alphabet $\{\mathrm{H}, \mathrm{P}\}$, in which the symbol P is "neutral" (i.e., does not contribute to the score). The above string folding model is more general in that it allows a larger set of monomer types (i.e., a larger alphabet), with only nearest-neighbor contacts between equal types

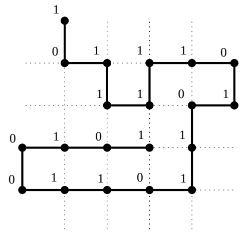

Figure 1: *The embedding of a* 0-1 *string in* $\mathcal{Z}^2$*.*

*Computer Science Division, UC Berkeley. Email: ashwin@cs.berkeley.edu. Supported by NSF grant CCR-9505448.

†Computer Science Division, UC Berkeley. Email: sinclair@cs.berkeley.edu. Supported by NSF grant CCR-9505448 and by the International Computer Science Institute.

‡Computer Science Department, Tel-Aviv University. Email: zwick@math.tau.ac.il. This work was done while this author was visiting ICSI and UC Berkeley.

contributing to the score. This is precisely the model considered by Paterson and Przytycka [17, 18].

Protein folding is notoriously hard in any reasonable model, and it is natural to seek evidence for this using the tools of computational complexity. Ngo, Marks and Karplus [15] present a review of complexity results on the problem, and argue that **NP**-hardness results can be useful in exposing sources of difficulty for algorithm designers (see also [8, 11, 12, 14, 20]). A number of **NP**-hardness results are known for HP-like models (see, e.g., [17, 18, 11]) but all of these have a serious drawback: the alphabet size (i.e., the number of monomer types) is allowed to grow with the length of the input sequence, and hence is unbounded. This is clearly not in the intended spirit of these models. The question of whether the problem remains **NP**-hard for a fixed alphabet has remained a significant open problem for several years — see, e.g., [18, 11].

In this paper, we show that the string folding problem over a suitably large fixed finite alphabet is indeed **NP**-hard (under randomized polynomial time reductions). To the best of our knowledge, this represents the first intractability result for a truly bounded HP-like model.[1]

We believe that our techniques hold at least as much interest as the result itself. In order to explain the techniques in the simplest possible setting, we begin with the problem of folding a *set* of strings. Our starting point here is a reduction similar to one introduced by Paterson and Przytycka [17], in which an unbounded alphabet is used. We reduce from the **MAX SNP**-hard problem MAX-CUT [16], which asks for a cut of maximum cardinality in an undirected graph. Our first innovation is to exploit the *approximation hardness* of MAX-CUT: namely, for suitable constants $0 < \beta < \alpha < 1$, given a graph G with m edges in which the maximum cut is guaranteed to be of size either at least αm or at most βm, it is **NP**-hard to determine whether G has a cut of size at least αm. This strong result is a consequence of the recent dramatic breakthroughs in approximation hardness pioneered by Feige *et al.* [7] and Arora *et al.* [3]. The gap in the source problem MAX-CUT is apparently essential to our reduction. We believe this is the first time that approximation hardness has been a key ingredient in establishing an **NP**-hardness result (as opposed to a stronger approximation hardness result). Not surprisingly, our reduction also immediately yields an approximation hardness result for string folding. A similar route, starting this time from the **MAX SNP**-hard problem MAX-3SAT(b) [16, 2], leads us to the hardness of approximation in the biologically more relevant case of folding a *single* string. This result stands in interesting contrast to the work of Hart and Istrail [9, 12] and Agarwala *et al.* [1] on polynomial time approximation algorithms for protein folding in HP-like models.

The second interesting feature of our technique is the notion of a *spatial code*. In a conventional error-correcting code, different codewords are required to have large distance from one another, where "distance" typically means Hamming distance. In a spatial code, the notion of distance is generalized to take account of the spatial arrangements of the codewords: informally, the "distance" between two codewords of length ℓ (viewed as strings over a finite alphabet) is $b - s$, where b is the *$\mathcal{Z}^3$-bonding capacity* of two strings of length ℓ, i.e., the maximum number of bonds that can be formed between *any* two strings of length ℓ when embedded in $\mathcal{Z}^3$, and s is the maximum score achievable by any embedding of the given pair of words in $\mathcal{Z}^3$. We believe that this idea may be of independent interest.

In order to reduce from an unbounded to a finite alphabet Σ, we replace each symbol of the unbounded alphabet by a codeword of suitable length over Σ. For the reduction to work, we require that these codewords form a good spatial code. We leave the efficient deterministic construction of spatial codes as an intriguing open question. However, we show that *randomly chosen* codewords over a suitably large finite alphabet form a good spatial code with high probability. This fact completes our randomized reduction.

We should emphasize that our approach operates at a high level, and actually provides a general methodology for replacing an unbounded alphabet by a finite one in string folding reductions, provided that these reductions are well-behaved. Since it has consistently proved much simpler to obtain hardness results with an unbounded alphabet, we believe that this methodology is generally useful. The main requirement for good behavior is that the reduction be *approximation-preserving*, i.e., that it should translate a constant factor gap in the objective function of the source problem to a similar gap in the target problem. This property allows us, in principle, to adopt the above approach of replacing symbols by letters from randomly chosen strings. For the problem of folding a *set* of strings, it is very straightforward to come up

[1] Subsequent to this work, Crescenzi *et al.* have informed us of a very interesting **NP**-hardness result for the HP model in $\mathcal{Z}^2$ [4]. While this result is stronger than ours in that their alphabet size is smallest possible, it apparently does not extend to hardness of approximation. Their approach is also quite different from ours: rather than giving a general method for replacing unbounded alphabets by finite ones, as we do, they construct intricate gadgets tailored to the 2-d HP model.

with an approximation-perserving reduction over an unbounded alphabet. In the single string case, the task is a little harder; we show how to accomplish it by modifying an existing reduction of Paterson and Przytycka [18], which is not approximation-preserving. We believe that these two examples suffice to illustrate the generality of our technique. Other variants of the string folding problem (e.g., based on different lattices) can be handled in a similar fashion. We discuss possible extensions of our work in Section 4.

1.2 Statement of results

As described above, we consider the problem of embedding a set of strings in $\mathcal{Z}^3$ so as to maximize the number of nearest-neighbor contacts between equal letters. To study its complexity, we define two versions of the string folding problem. Let A be a fixed alphabet size. The *decision* version, Fold_A, is defined as follows: given a multiset of strings, $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ over the alphabet $\{1, \ldots, A\}$, and an integral *threshold* s, determine whether there is an embedding of the strings in the lattice $\mathcal{Z}^3$ such that its *score*, i.e., the number of pairs of identical letters adjacent to each other in the embedding but not in the strings, is at least s. (We will refer to the adjacencies that contribute to the score as *bonds*.) The problem Max-Fold_A is the optimization version of this problem, namely the problem of finding the maximum score achievable by any embedding of the strings in the 3-dimensional lattice $\mathcal{Z}^3$. The restrictions of these problems in which the input consists of a single string are referred to as 1-Fold_A and Max-1-Fold_A respectively.

Our first result states that there exists a finite alphabet size A for which Fold_A is **NP**-hard under randomized polynomial time reductions. In other words, if there exists a (randomized) polynomial time algorithm for Fold_A, then there is also a randomized polynomial time algorithm for, say, the satisfiability problem. We will in fact prove the stronger result that Max-Fold_A is **MAX SNP**-hard under randomized polynomial time reductions. This means that Max-Fold_A is hard to even *approximate* within a certain constant factor. We can state these results more precisely as follows:

Theorem 1.1 *There is a finite alphabet size A such that the following hold:*

(i) *if there exists a polynomial time algorithm for* Fold_A*, then* $\mathbf{NP} \subseteq \mathbf{co\text{-}RP}$;

(ii) *for some constant $\gamma < 1$, if there exists a polynomial time algorithm that approximates* Max-Fold_A *within a factor of γ, then* $\mathbf{NP} \subseteq \mathbf{co\text{-}RP}$.

Part (i) of Theorem 1.1 is, of course, subsumed by part (ii), the approximation hardness result. To prove part (ii), we present a randomized approximation-preserving reduction from the Max-Cut problem, which is known to be **MAX SNP**-complete [16]. We then extend the technique employed in the proof to get our next (stronger) result, namely, the hardness of approximating Max-1-Fold_A:

Theorem 1.2 *The statements of Theorem 1.1 hold also for the single string folding problems* 1-Fold_A *and* Max-1-Fold_A*, for a suitable finite alphabet size A.*

The proof of this second theorem uses a reduction similar in flavor to that of Theorem 1.1, but starting from the **MAX SNP**-complete problem $\text{Max-3Sat}(b)$, a version of Max-3Sat in which each variable appears in a fixed number b of clauses [16, 2].

The rest of the paper is organized as follows. Section 2 is devoted to the analysis of the problem of folding a set of strings. We begin in Section 2.1 by describing a very simple reduction from Max-Cut to Max-Fold_∞, the version of the string folding problem in which the alphabet size is unbounded. We then describe, in Section 2.2, a way of turning this into a reduction from Max-Cut to Max-Fold_A, for some fixed finite alphabet size A, using a special class of codes that we call *spatial codes*. We know of no efficient deterministic construction of such codes. However, in Section 2.3 we show that a random set of sufficiently long words is, with very high probability, a good spatial code. By using such a random set of words we thus get a randomized reduction from Max-Cut to Max-Fold_A, thereby proving Theorem 1.1. In Section 3, we turn to the problem of folding a *single* string in $\mathcal{Z}^3$. We first extract, in Section 3.1, the essential elements of the multiple-string reduction and then outline how a proof of hardness of Max-1-Fold_A can be synthesized from similar elements. Sections 3.2 and 3.3 fill in the details required to complete the description. We conclude in Section 4 by discussing the limitations and possible extensions of our approach and some directions for future work. Owing to space limitations, most of the proof details are deferred to the full paper [13].

2 Folding a set of strings

2.1 A simple reduction

In this section we describe a very simple reduction from Max-Cut to Max-Fold_∞. The reduction is similar to a reduction from Not-all-equal-3Sat to Fold_∞ given by Paterson and Przytycka [17].

The input to the Max-Cut problem is an undirected graph $G = (V, E)$. The goal is to find a *cut*,

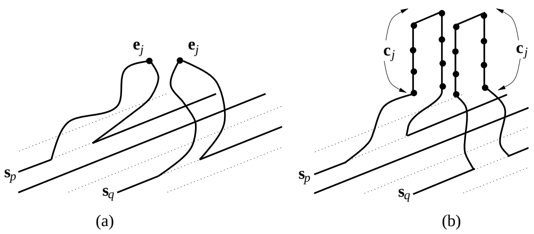

Figure 2: (a) *The two letters* $\mathbf{e}_j$ *can bond iff their strings start at lattice points of opposite parity.* (b) *Two codewords corresponding to an edge bond along the* z *direction.*

i.e., a subset $C \subset V$ of the vertices, such that the number of edges that connect vertices in C with vertices in $V - C$ (the *size* of the cut) is maximized. It follows from the fact that MAX-CUT is **MAX SNP**-complete that there exist constants $0 < \beta < \alpha < 1$, such that it is **NP**-hard to distinguish graphs with m edges that have a cut of size at least αm, from those that have cuts of size at most βm [3]. The best known lower bound for the "gap" $\alpha - \beta$ is $1/22$, as is implicit in [19]. We fix such a pair α, β for the rest of Section 2.

Suppose we are given as input to MAX-CUT a graph $G = (V, E)$ with n vertices, $v_1, \ldots, v_n$, and m edges, $e_1, \ldots, e_m$. Our reduction constructs a set of n strings $S_G = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ over the alphabet $\mathcal{N}$, the natural numbers. The string $\mathbf{s}_i$, corresponding to the vertex v_i, is the concatenation of m blocks $\mathbf{s}_{i1}, \ldots, \mathbf{s}_{im}$, corresponding to the m edges of G. The blocks $\mathbf{s}_{ij}$ are defined as follows:

$$\mathbf{s}_{ij} = \begin{cases} *^n \mathbf{e}_j *^{n+1} & \text{if } v_i \in e_j, \\ ** & \text{otherwise.} \end{cases}$$

Here each $\mathbf{e}_j$ is a distinct letter from the alphabet, and '$*$' is assumed to be a special (neutral) symbol that does not interact with any other symbol, including itself. If such a symbol is not part of the model, we can simply replace each '$*$' with a distinct letter from the (unbounded) alphabet. Note that each of the blocks $\mathbf{s}_{ij}$ is of *even* length. This ensures that the two copies of symbol $\mathbf{e}_j$ can bond iff their strings start at lattice points of *opposite* parity. By associating the two sides of a cut in G with the two possible parities of the starting points of the strings in an embedding of S_G (refer to Figure 2(a); the details can be found in the full paper [13]), it is easy to see that:

Lemma 2.1 *The graph* G *has a cut of size at least* k *iff there is an embedding of* S_G *with a score of* k.

As an immediate consequence we get:

Theorem 2.2 (i) FOLD$_\infty$ *is* **NP**-*hard*;
(ii) MAX-FOLD$_\infty$ *is* **MAX SNP**-*hard.*

2.2 From the infinite to the finite

The reduction given in the previous section suffers from a serious drawback common to all previous hardness results in HP-like models: it requires an *unbounded* alphabet. Our goal here is to obtain a reduction for a fixed, finite alphabet. A natural approach to this is to try to replace each letter in the above reduction by a *codeword* over a fixed finite alphabet. In order that the codewords emulate alphabet symbols, we require that unequal codewords should bond only very weakly with one another, no matter how hard they try by turning and twisting around each other. Thus, the codewords of such a code, which we refer to as a *spatial code*, must satisfy conditions that are much more stringent than those demanded of codewords in classical error-correcting codes. We believe that this concept may be of independent interest. Before formalizing it, we need the following important definition.

Definition 2.3 (Intended and unintended bonds) *Let* $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ *be a multiset of strings over the alphabet* $\{1, 2, \ldots, A\}$. *A bond in an embedding of* S, *formed by the adjacency of the* i_1*-th letter in* $\mathbf{s}_{j_1}$ *and the* i_2*-th letter in* $\mathbf{s}_{j_2}$ *is said to be* intended *if* $i_1 = i_2$ *and* $\mathbf{s}_{j_1} = \mathbf{s}_{j_2}$, *and* unintended *otherwise. The* intended (unintended) *score of the embedding is the number of intended (unintended) bonds formed in it.*

Note in particular that, if all the strings in the set S are distinct, then *all* bonds in an embedding of S are considered to be unintended. We are now ready for the definition of spatial codes.

Definition 2.4 (Spatial Codes) *A set* C *of* m *strings of length* ℓ *over the alphabet* $\{1, 2, \ldots, A\}$ *is said to be a* (c, f)-spatial code *if the unintended score of any embedding of the multiset consisting of* c *copies of each codeword of* C *in* $\mathcal{Z}^3$ *is at most* $f \cdot 3cm\ell$.

Note that if we take c copies of each of m strings of length ℓ, we can always achieve a certain intended score in an obvious way by aligning the copies of the same string next to each other; e.g., in the case $c = 2$ this score would be $m\ell$. But if the strings constitute a (c, f)-spatial code, then no matter how we embed the strings we can only achieve a relatively small unintended score (even if we are willing to sacrifice some of the intended bonds): namely at most a fraction f of the total bonding capacity of m strings of length ℓ, which is clearly bounded above by $3cm\ell$. (The total number of letters in these words is $cm\ell$, and in $\mathcal{Z}^3$ each letter can bond with at most six other letters. This should be divided by two as each potential bond is counted twice.)

To simplify the exposition we assume for the moment that the finite alphabet contains a neutral symbol '$*$', and denote the corresponding problems by FOLD^*_A and MAX-FOLD^*_A. We now claim the following:

Theorem 2.5 *If a $(2, f)$-spatial code consisting of m codewords over the alphabet $\{1, 2, \ldots, A\}$, with $f < (\alpha - \beta)/6$, exists for each m and can be constructed in polynomial time (in m), then:*
(i) FOLD^*_A *is* **NP**-*hard*;
(ii) MAX-FOLD^*_A *is* **MAX SNP**-*hard.*

The proof of Theorem 2.5 is obtained by replacing each active symbol $\mathbf{e}_j$ in the reduction of Section 2.1 by a codeword $\mathbf{c}_j$ from a $(2, f)$-spatial code. A bond between two $\mathbf{e}_j$ symbols can easily be simulated by lining up the corresponding copies of the codewords as shown in Figure 2(b), so the codewords behave essentially like the original symbols $\mathbf{e}_j$. The only problem is the existence of *unintended* bonds between different codewords. The definition of spatial codes ensures that this contribution to the score is small, and hence cannot compensate for the gap $\alpha - \beta$.

We do not know of any deterministic polynomial time algorithm for constructing the spatial codes required in Theorem 2.5. Indeed, this seems like a challenging problem in its own right. However, we are able to give a very simple randomized algorithm for their construction. As Theorem 2.6 of the next section shows, we can simply choose a random set of sufficiently long words over a sufficiently large, but finite, alphabet. Together with Theorem 2.5 above, this immediately proves the **NP**-hardness and **MAX SNP**-hardness of FOLD^*_A and MAX-FOLD^*_A respectively, via *randomized* reductions. This almost completes the proof of Theorem 1.1. The only remaining issue is the elimination of the neutral symbol '$*$'. (Note that this is not trivial as in the case of an unbounded alphabet.) This can be done by replacing each occurrence of '$*$' with a random symbol from the finite alphabet. Since the arguments are very similar to those used in the construction of spatial codes, we omit them here.

2.3 Randomized construction of spatial codes

In this section we demonstrate that almost any set of strings over a sufficiently large finite alphabet forms a good spatial code. More specifically, we show:

Theorem 2.6 *Let C be a set of m random strings of length ℓ over the alphabet $\{1, 2, \ldots, A\}$. Then, for every fixed $c \geq 1$ and $f > 0$, there exist constants $\alpha_1 > 0$ and $\alpha_2 > 0$ determined by c and f such that if $\ell \geq \ln m$,*

$$\Pr[\, C \text{ is a } (c, f)\text{-spatial code}\,] \;>\; 1 - \mathrm{e}^{-(\alpha_1 \ln A - \alpha_2) m\ell} \,.$$

In particular, for every fixed $c \geq 1$ and $f > 0$, if A is large enough and $\ell \geq \ln m$, then most sets of m strings of length ℓ over the alphabet $\{1, 2, \ldots, A\}$ are (c, f)-spatial codes.

Proof: Let C be a set of m random strings of length ℓ over the alphabet $\{1, 2, \ldots, A\}$. Let C^c be the multiset composed of c copies of each codeword. We have to show that, with very high probability, the unintended score of any embedding of C^c is at most $f \cdot 3cm\ell$.

We can consider each letter in each codeword of C to be a random variable, independently and uniformly distributed over the set $\{1, 2, \ldots, A\}$. For an embedding $\mathcal{E}$ of the multiset C^c in $\mathcal{Z}^3$, we let $X_{\mathcal{E}}$ denote the random variable giving the unintended score of this embedding.

Consider an embedding $\mathcal{E}$ of the multiset C^c in $\mathcal{Z}^3$. The *adjacency graph* of this embedding is the graph whose vertex set is the set of $cm\ell$ symbol positions in the strings in C^c. Two such positions are connected by an edge in the graph iff they are adjacent in the embedding but not in the strings. For the purposes of analysing the score in any embedding, it is clearly enough to examine the corresponding adjacency graph, rather than the embedding itself: two embeddings with the same adjacency graphs have both the same intended score and the same unintended score, and hence the same score.

With this observation, the theorem follows almost immediately from the following two lemmas, whose proofs are left to the full paper [13].

Lemma 2.7 *Let $\mathcal{E}$ be an embedding of the multiset C^c in $\mathcal{Z}^3$. Then, for some $\alpha_1, \alpha_3 > 0$ that depend only on c and f, we have*

$$\Pr[X_{\mathcal{E}} \geq f \cdot 3cm\ell] \quad \leq \quad \mathrm{e}^{-(\alpha_1 \ln A - \alpha_3) m\ell} \,.$$

Lemma 2.7 is basically a Chernoff-like large deviation bound: note that $E[X_{\mathcal{E}}]$ is at most $(3cm\ell)/A$. The difficulty in the proof comes from the fact that the potential bonds in $\mathcal{E}$ are not independent.

Lemma 2.8 *If $\ell \geq \ln m$, then the number of different adjacency graphs of all embeddings of the strings in C^c in $\mathcal{Z}^3$ is at most $\mathrm{e}^{\alpha_4 m\ell}$, for some $\alpha_4 > 0$ that depends only on c.*

To complete the proof of Theorem 2.6, we note that the probability that C is *not* a (c, f)-spatial code is bounded by the number of possible adjacency graphs times the probability that the spatial code condition is violated for a particular adjacency graph. Thus combining the bounds from Lemmas 2.7 and 2.8 gives

$$\Pr[\, C \text{ is not a } (c, f)\text{-spatial code}\,] \;\leq\; \mathrm{e}^{\alpha_4 m\ell} \cdot \mathrm{e}^{-(\alpha_1 \ln A - \alpha_3) m\ell} \;=\; \mathrm{e}^{-(\alpha_1 \ln A - \alpha_2) m\ell} \,,$$

for $\ell \geq \ln m$, where $\alpha_2 = \alpha_3 + \alpha_4$. This completes the proof of Theorem 2.6. ■

Remark: To get an idea of how large Theorem 2.6 requires A to be for the existence of a (c, f)-spatial code, we can explicitly calculate the constants posited by Lemmas 2.7 and 2.8, and plug in the best known value of $1/22$ for the gap $\alpha - \beta$, implicit in [19]. Doing this gives us a lower bound on A of something like 10^{300}, an extremely large (but constant!) value. We could improve this bound very substantially with a more careful analysis, but this would not be sufficient to reduce it to realistic biological proportions (of, say, 20, which is the number of different amino acid types). ■

3 Folding a single string

In this section, we extend our techniques to show that the problem $\text{F}\small{\text{OLD}}_A$ of the previous section remains NP-hard under randomized reductions even for a *single* string (rather than a *set* of strings), for a suitably large finite alphabet size A, as claimed in Theorem 1.2. This single-string version of the problem, which we refer to as 1-FOLD, is the one most commonly studied in computational biology (see, e.g., [18]).

3.1 Generalizing the technique

We begin by outlining our overall strategy, which the reader should recognize as a generalization of the approach of Section 2. Our strategy proceeds as follows:

(1) We start with an approximation-preserving reduction from a **MAX SNP**-hard problem to MAX-1-FOLD$^*_\infty$, the single string folding problem over an unbounded alphabet with a neutral symbol. We assume that in this reduction each active symbol appears no more than c times, for some constant c. The reduction implies that there are constants $0 \leq \rho < \sigma \leq 1$ such that it is **NP**-hard to distinguish instances of MAX-1-FOLD$^*_\infty$ that have a score of at least σM from those that have a score of at most ρM, where M is the total number of active symbols in the instance of MAX-1-FOLD$^*_\infty$. (In the case of MAX-FOLD$^*_\infty$, in Section 2.1 we gave such a reduction from MAX-CUT with $c = 2$ and $\rho = \beta/2$, $\sigma = \alpha/2$, where α, β are the gap factors for MAX-CUT.)

(2) Next we show that there are constants $0 \leq \rho' < \sigma' \leq 1$ such that it is **NP**-hard to distinguish instances of MAX-1-FOLD$^*_\infty$ that have a score of at least $\sigma' L$ from those that have a score of at most $\rho' L$, where L is now the *total* length of the instance, counting both active and neutral symbols. The rationale for this is the following. In our randomized construction of Section 2.3 we traded off the number of embeddings (or adjacency graphs) against the probability that the unintended score of some embedding overwhelms the gap (see Lemmas 2.8 and 2.7). Since the number of embeddings here is exponential in L, we need the large deviation probability for the unintended score to be exponentially small in L. For this, we require that the gap be a constant fraction of L. (We shall see how this works in more detail under point 3 below.)

To get such a gap, we *replicate* the active parts of the strings produced by the reduction above. We assume that the active parts of these strings are organized in short (constant-length) contiguous *chunks* which correspond to *gadgets* used in the reduction. A string $\mathbf{s}$ produced by the above reduction is of the form

$$\mathbf{c}_1 \sim \mathbf{c}_2 \sim \mathbf{c}_3 \sim \cdots \sim \mathbf{c}_k \quad ,$$

where $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ are the active chunks and '$\sim$' represents a long padding string. We transform such a string into a string $\mathbf{s}'$ of the form

$$\mathbf{c}_1^1 - \mathbf{c}_1^2 - \cdots - \mathbf{c}_1^\ell \sim \mathbf{c}_2^1 - \mathbf{c}_2^2 - \cdots - \mathbf{c}_2^\ell \sim \cdots \sim \mathbf{c}_k^1 - \mathbf{c}_k^2 - \cdots - \mathbf{c}_k^\ell$$

where $\mathbf{c}_i^1, \ldots, \mathbf{c}_i^\ell$ are replicas of $\mathbf{c}_i$, and '$-$' represents a short (constant-length) padding string. If Σ is the set of active symbols used in the original chunks $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$, we choose ℓ disjoint copies $\Sigma_1, \ldots, \Sigma_\ell$ of Σ. For every $1 \leq j \leq \ell$, the chunks $\mathbf{c}_1^j, \mathbf{c}_2^j, \ldots, \mathbf{c}_k^j$ are obtained from the original chunks $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ by replacing each active symbol of Σ by its equivalent in Σ_j. (In the case of MAX-FOLD$^*_A$, the chunks were simply individual symbols $\mathbf{e}_j$, and the replication involved replacing each one by a contiguous codeword of length ℓ, with padding strings of zero length inbetween.)

As a result of this ℓ-fold replication, the optimal score of the string is multiplied by ℓ. (This property is *not* guaranteed to hold in general, as the short padding between copies of the same chunk may not be sufficient to allow the bonds in all ℓ copies to be formed, and at the same time, an individual copy may be able to score more; but it will hold in our case.) In the process of this replication, the lengths of the long padding strings, represented by '$\sim$' above, are *not* increased. Therefore, by making ℓ sufficiently large (taking ℓ to be the length of the original string is more than enough), we ensure that at least a constant fraction of the symbols in the resulting string are active, and that the optimal score is at least a constant fraction of the total length. This gives us a gap whose size is a constant fraction of the total length, as desired.

(3) Finally, we show that MAX-1-FOLD remains **MAX SNP**-hard when the unbounded alphabet is replaced by a sufficiently large finite alphabet. In other words, we show that MAX-1-FOLD$^*_A$ is **MAX SNP**-hard, under randomized reductions, for some finite alphabet size A. A small additional step shows that the same holds for MAX-1-FOLD$_A$, where no neutral symbols are available (and A is now a little larger).

Let $\Sigma' = \bigcup_{j=1}^{\ell} \Sigma_j$ be the set of all active symbols in the string $\mathbf{s}'$ obtained as above. We construct a string $\mathbf{s}''$ by replacing each symbol of Σ' by a *random* symbol from the finite alphabet $\{1, 2, \ldots, A\}$, where A is a sufficiently large constant. Clearly, the optimal score of $\mathbf{s}''$ is at least as large as the optimal score of $\mathbf{s}'$. Thus, if the optimal score of $\mathbf{s}'$ is at least $\sigma' L$, where L is the length of $\mathbf{s}'$, then the score of $\mathbf{s}''$ is also at least $\sigma' L$. We show, on the other hand, that if the optimal score of $\mathbf{s}'$ is at most $\rho' L$, then with very high probability the optimal score of $\mathbf{s}''$ is less than $\sigma' L$.

To show that the optimal score of $\mathbf{s}''$ is not much larger than the optimal score of $\mathbf{s}'$, we use an analysis similar to the one carried out in Section 2.3. More specifically, let $X_{\mathcal{E}}$ be the *unintended* score of the embedding $\mathcal{E}$ of $\mathbf{s}''$ in $\mathcal{Z}^3$. (A bond in $\mathbf{s}''$ is now said to be unintended if it involves two positions whose symbols in $\mathbf{s}'$ are unequal.) Recall that each symbol of Σ' appears in $\mathbf{s}''$ only a constant number of times. For every $f > 0$, there exist some constants $\alpha_1, \alpha_3 > 0$ such that

$$\Pr[X_{\mathcal{E}} \geq f \cdot 3L] \quad \leq \quad \mathrm{e}^{-(\alpha_1 \ln A - \alpha_3)L} .$$

This follows exactly as in the proof of Lemma 2.7, which relies only on the fact that the degree of the collapsed adjacency graph is bounded by a constant. As the number of different embeddings of $\mathbf{s}''$ is at most 5^L, we get, as in the proof of Theorem 2.6, that

$$\Pr[X_{\mathcal{E}} < f \cdot 3L \quad \text{for all } \mathcal{E}] \quad \geq \quad 1 - \mathrm{e}^{-(\alpha_1 \ln A - \alpha_2)L}$$

for some $\alpha_1, \alpha_2 > 0$. This probability approaches 1 if we take A to be a sufficiently large constant. By choosing f to be less than $(\sigma' - \rho')/3$, we can make the unintended score of every embedding negligible compared to the gap $(\sigma' - \rho')L$, with high probability. This completes the randomized reduction, and shows that MAX-1-FOLD$_A^*$ is **MAX SNP**-hard. We can dispense with the neutral symbol '$*$' exactly as in Section 2.

Remark: It is possible to broaden the definition of spatial codes given in Section 2 to fit the generalized scenario described above, thereby again isolating the role of randomness in the reduction. However, as this broader definition does not appear to be as natural as the original one, we will not present it here. ■

Having outlined our strategy, we fill in the details in the following two subsections. In Section 3.2 we give an approximation-preserving reduction to MAX-1-FOLD$_\infty^*$. Once this is done, the rest of the above procedure will go through more or less automatically; we provide the details in Section 3.3.

3.2 An approximation-preserving reduction over an unbounded alphabet

We start from MAX-3SAT(b), a version of MAX-3SAT in which each variable occurs a fixed constant number b times in the whole formula. In [2], it is shown that this problem is **MAX SNP**-hard for $b = 5$. It is in fact not hard to show, by essentially the same argument, that MAX-3SAT(b) is **MAX SNP**-hard even for $b = 3$.

Theorem 3.1 MAX-3SAT(3) *is* **MAX SNP**-*hard.*

We now present an approximation-preserving reduction from MAX-3SAT(3) to MAX-1-FOLD$_\infty^*$. Given a MAX-3SAT(3) instance ϕ with m clauses $C_1, C_2, \ldots, C_m$ over the variables $x_1, \ldots, x_n$, we construct a string $\mathbf{s}_\phi$ over the unbounded alphabet $\mathcal{Z}$ (with a neutral symbol '$*$'). The string $\mathbf{s}_\phi$ consists of n *rod-flap combinations* of constant size (see Figure 3(a)&(b)), one for each variable, and m *ligands* corresponding to the m clauses, also of constant size (see Figure 3(d)). The rod-flap combinations are connected in sequence with constant length padding (strings of '$*$'s) inbetween, and the m ligands are attached to the resulting string in sequence with $\Theta(m)$ padding for each ligand. The symbols appearing in different ligands and in different rod-flap combinations are all distinct, except for *clause symbols* $\mathbf{c}_j$, one for each clause C_j, which occur both in the corresponding ligand and in the (at most three) rods corresponding to the variables in that clause. Thus, apart from bonds internal to the ligands and to the rod-flap combinations, the only other bonds that can be formed are between related ligands and rods.

The clause symbols in a rod are placed on one of two opposite edges of the rod depending on whether the variable occurs positively or negatively in the clause. The rod-flap combination is so designed that, in its optimal embedding, the flap completely covers one of these two edges of the rod and leaves the other exposed (see Figure 3(c)). The choice of which edge to expose corresponds to making a truth assignment to the associated variable: in the optimal embedding of the string, the clause symbols along the exposed edge of each rod are available for bonding with the corresponding ligands (thus "satisfying" the clause). The $\Theta(m)$ padding with which the ligands are connected to the rest of the string is sufficient to allow them to reach any rod for bonding. We ensure that all the rods have the same parity, which is opposite to that of the ligands. This prevents the same clause symbol in different rods from bonding while permitting bonds between rods and

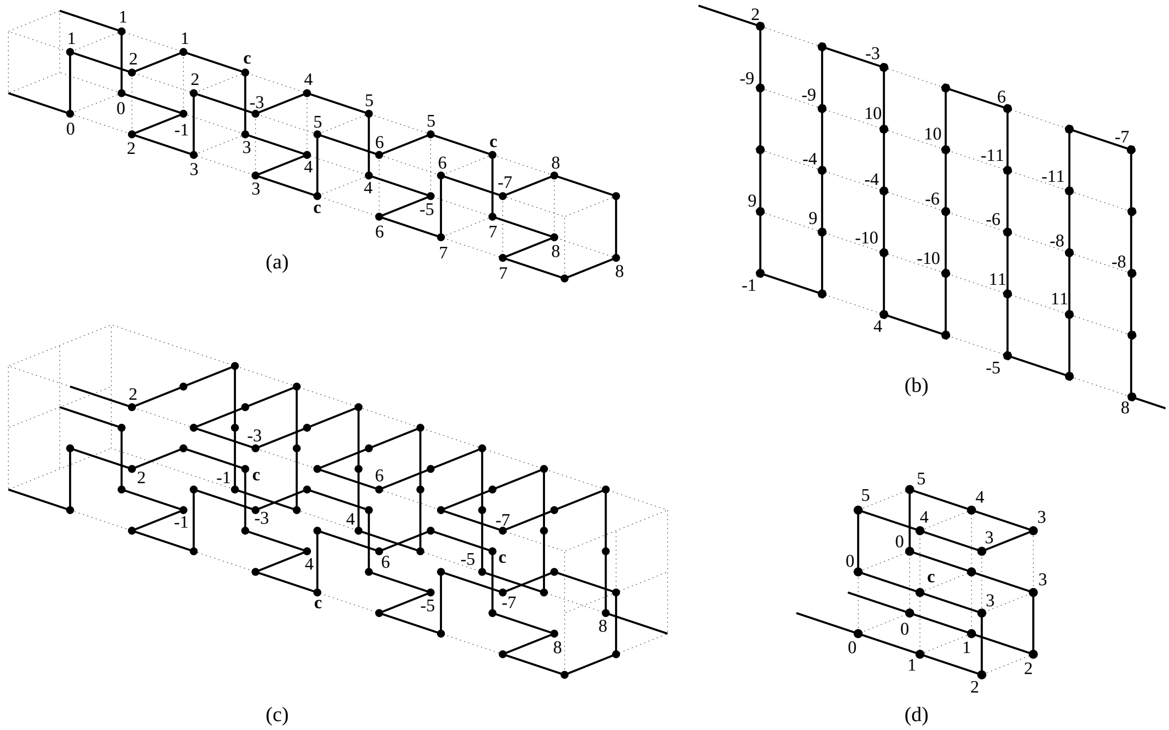

Figure 3: (a) *The symbol pattern in a rod, shown in one of its two optimal embeddings.* (b) *The flap corresponding to the rod.* (c) *One of the two optimal embeddings of the rod-flap combination: only symbols involved in bonds between the rod and the flap are shown.* (d) *The unique optimal embedding of a ligand.*

ligands. Furthermore, the construction of the gadgets ensures that there is no profit in ligands bonding with more than one rod, or in rod-flap pairs deviating from their intended embedding. Figure 4(a) sketches an optimal embedding of the entire string.

This construction ensures that the score of the optimal embedding of the string precisely reflects the maximum number of clauses simultaneously satisfiable in the MAX-3SAT(3) instance, leading to an approximation-preserving reduction:

Proposition 3.2 *Let ϕ be a* MAX-3SAT(3) *instance with m clauses and n variables, and let $\mathbf{s}_\phi$ be the string constructed as described above. If the maximum number of simultaneously satisfiable clauses in ϕ is k, then the optimal score of $\mathbf{s}_\phi$ is $34n + 11m + k$.*

A detailed proof of this proposition is given in the full paper [13].

Since in any instance of MAX-3SAT(3) we have $n \leq 3m$, Proposition 3.2 tells us that the mapping $\phi \to \mathbf{s}_\phi$ is an approximation-preserving reduction from MAX-3SAT(3) to MAX-1-FOLD$^*_\infty$. We therefore get:

Corollary 3.3 MAX-1-FOLD$_\infty$ *is* **MAX SNP**-*hard.*

Remark: This reduction owes its origins to an earlier reduction of Paterson and Przytycka [17, 18], from 3SAT to 1-FOLD$^*_\infty$. Our main innovation here is to make the reduction approximation-perserving, the essential ingredient being the constant-size rod-flap combinations which replace the long "teeth" of Paterson and Przytycka. We note in passing that, in addition to the "helices" (used in the rods) and the "ligands", variants of which were already present in the earlier reduction, our reduction includes a third biological motif, namely the "sheets" used in the flaps. It is unclear whether the presence of these motifs has any biological significance. ■

This concludes the first (and major) part of our strategy as outlined at the beginning of the section. It remains only to explain how to carry out the replication process, which we now do.

3.3 Reduction to a finite alphabet

Recall from point 2 of our general strategy that we need to replicate active portions of the string so that the optimal score becomes a constant fraction of the string length. Note that in $\mathbf{s}_\phi$ the active parts (the

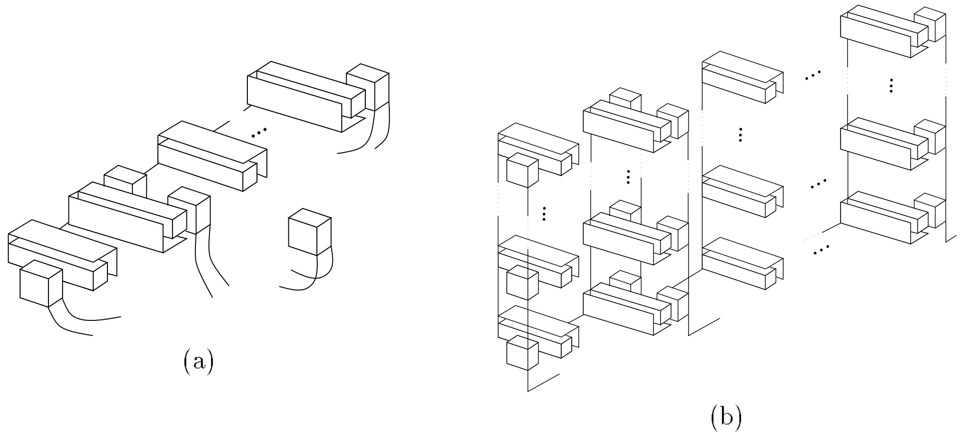

Figure 4: (a) *An optimal embedding of the string* $\mathbf{s}_\phi$. (b) *An optimal embedding of the replicated string* $\mathbf{s}_\phi^\ell$.

ligands, rods and flaps) have net length $\Theta(m)$, while the padding has total length $\Theta(m^2)$, so we will need to replicate each chunk $\ell = \Omega(m)$ times. The "chunks" that we replicate are the gadgets, namely the ligands and the rod-flap combinations. We take ℓ copies of each of these, over disjoint sets of symbols. The ℓ copies of a given rod-flap combination are connected together in sequence, separated by constant length padding strings so that all copies have enough room to fold as intended. Similarly, the ℓ copies of a given ligand are glued together with constant length padding strings so that, if the first copy bonds with the first copy of some rod, then every copy of the ligand in the sequence can bond with the corresponding copy of the rod. The padding between successive sets of rods remains of constant length, and that between successive sets of ligands remains of length $\Theta(m)$. This allows the sets of ligands complete freedom in bonding with sets of rods. A sketch of this ℓ-fold replication of the string $\mathbf{s}_\phi$ (which we denote by $\mathbf{s}_\phi^\ell$) is shown in Figure 4(b), in an optimal embedding. Now Proposition 3.2 yields the following property of $\mathbf{s}_\phi^\ell$:

Corollary 3.4 *Let* ϕ *be a* MAX-3SAT(3) *instance with* m *clauses and* n *variables, and let* $\mathbf{s}_\phi^\ell$ *be the string described above. If the maximum number of simultaneously satisfiable clauses in* ϕ *is* k*, then the optimal score of* $\mathbf{s}_\phi$ *is* $(34n + 11m + k)\ell$.

Now we are essentially done. Since the length of the string $\mathbf{s}_\phi^\ell$ is $\Theta(m\ell + m^2)$, by taking $\ell \geq m$ we can make the optimal score as large as a constant factor times the string length, as required in point 2 of our strategy. As explained in point 3, if we replace the symbols of $\mathbf{s}_\phi^\ell$ with random letters from $\{1, 2, , \ldots, A\}$ for a sufficiently large constant A, we get a (randomized) reduction to MAX-1-FOLD$_A^*$. Finally, we may remove the neutral symbol '$*$' exactly as in the multiple string case, although at the cost of increasing the value of A. We have therefore proved the **MAX SNP**-hardness of MAX-1-FOLD$_A$, as claimed in Theorem 1.2.

4 Extensions and further work

We have presented the first **NP**-hardness, and also **MAX SNP**-hardness results for string folding in an HP-like model with a finite alphabet. This model has various obvious limitations that compromise its biological plausibility. In this final section, we discuss these and comment on the potential for overcoming them.

As we have already observed, our technique is robust with respect to many details of the model, and can in principle be used to convert any well-behaved reduction to string folding in an HP-like model over an unbounded alphabet to a reduction to string folding in the same model over a fixed finite alphabet. The primary criterion for well-behavedness here is that the original reduction be approximation-preserving. We believe that this is not a severe restriction, and that most existing reductions over unbounded alphabets can be modified so as to satisfy it. Obvious candidates include string folding in lattices other than $\mathcal{Z}^3$, such as the 2-dimensional lattice $\mathcal{Z}^2$ or non-bipartite lattices like the triangular or tetrahedral lattice. Reductions over an unbounded alphabet already exist for a wide variety of lattices (see, e.g., [11]); we believe that our techniques can be applied to make the alphabet finite in these cases also.

Our model assumes that the only contributions to the energy arise from adjacencies between identical amino acid monomers. Although this property is often assumed in theoretical models, it is clearly unrealistic; one would want to allow a more general matrix of interactions between different types. We have not investigated in detail how robust our technique is with respect to changes in the interactions. However, we believe that it should still be applicable if the interactions are suitably regular.

An obvious drawback of our technique is that, while the alphabet size required for hardness is finite, it is extremely large. We have made no attempt here to minimize it, and some fine tuning of our arguments would reduce the alphabet size dramatically; however, it appears that a conceptual advance would be required to bring it down to a size of biological proportions (such as 20, the number of different amino acids, or two, the alphabet size of the HP model). Although a proof of **NP**-hardness for the HP model in $\mathcal{Z}^2$ does now exist using a different method [4], the question of hardness of approximation for the string folding problem over a reasonably small alphabet remains open.

Finally, we mention two intriguing open questions related to the concept of a *spatial code*, defined in Section 2.2. Firstly, do there exist spatial codes over alphabets that are substantially smaller than those that we get using our current techniques? Secondly, are there any explicit constructions, i.e., efficient deterministic algorithms for constructing spatial codes? We believe that these questions are interesting in their own right. In addition, positive answers to these questions would presumably yield approximation hardness results for the string folding problem over smaller alphabets.

Acknowledgements

We would like to thank Ken Dill and Sorin Istrail for helpful introductions to protein folding, and Mike Paterson for interesting discussions on snails.

References

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

[2] S. Arora and C. Lund. Hardness of approximations. In *Approximation algorithms for NP-hard problems*, D.S. Hochbaum ed., PWS Publishing Company, Boston, 1996.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and hardness of approximation problems. *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, 1992, pp. 14–23.

[4] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni and M. Yannakakis. On the Complexity of Protein Folding. Manuscript, October 1997. Submitted to *RECOMB 98*.

[5] K.A. Dill. Dominant forces in protein folding. *Biochemistry* **29** (1990), pp. 7133–7155.

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

[7] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM* **43** (1996), pp. 268–292.

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

[9] W.E. Hart and S. Istrail. Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal. *J. Comp. Biol.* **3** (1996), pp. 53–96.

[10] W.E. Hart and S. Istrail. Invariant patterns in crystal lattices: implications for protein folding algorithms. In *Combinatorial Pattern Matching* 1996, Springer Lecture Notes in Computer Science, pp. 288–303.

[11] W.E. Hart and S. Istrail. Robust proofs of NP-hardness for protein folding: general lattices and energy potentials. *J. Comp. Biol.* **4** (1997), pp. 1–20.

[12] W.E. Hart and S. Istrail. Lattice and off-lattice side chain models of protein folding: linear time structure prediction better than 86% of optimal. *J. Comp. Biol.* **4** (1997), pp. 241–259.

[13] A. Nayak, A. Sinclair and U. Zwick. Spatial codes and the hardness of string folding problems. Full version, submitted to *J. Comp. Biol.*, July 1997.

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

[15] J.T. Ngo, J. Marks and M. Karplus. Computational complexity, protein structure prediction, and the Levinthal paradox. In *The protein folding problem and tertiary structure prediction*, K.M. Merz and S.M. Le Grand eds., Birkhäuser, Boston, 1994.

[16] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Computer and System Sciences* **43** (1991), pp. 425–440.

[17] M. Paterson and T. Przytycka. On the complexity of string folding. Research Report CS-RR-286, University of Warwick, 1995.

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

[19] L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming (extended abstract). In *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science*, 1996, pp. 617–626.

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