Spatial Codes and the Hardness of String Folding Problems *

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

June 1997
Revised: August 1998

Abstract

We present a general technique for proving **NP**-hardness (under randomized polynomial time reductions) of 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 these problems (again under randomized polynomial time reductions). This means that obtaining even an *approximate* solution to the protein folding problem, to within some fixed constant factor, is **NP**-hard. Our technique involves replacing the symbols of an unbounded alphabet by *codewords* over a fixed alphabet, and 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.

*A preliminary version of this paper appeared in the Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, January 1998 [15].

†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.

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 [6, 7]. 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-neighbour 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 somewhat more general in that it allows a larger set of monomer types (i.e., a larger alphabet), with only nearest-neighbour contacts between equal types contributing to the score. This is precisely the model considered by Paterson and Przytycka [20, 21].

Protein folding is notoriously hard in any reasonable model, and it is natural to seek evidence for this using the tools of computational complexity. Specifically, one would like to show that the problem is **NP**-*hard* and hence, under standard assumptions, computationally intractable. Ngo, Marks and Karplus [17] present a review of complexity results on the problem, and argue also that **NP**-hardness results can be useful in exposing sources of difficulty for algorithm designers (see also [9, 12, 13, 16, 24]). A number of **NP**-hardness results are known for HP-like models (see, e.g., [20, 21, 12]) 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

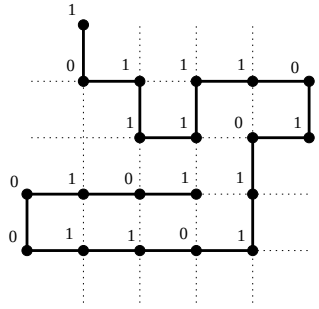


Figure 1: *An embedding of a* 0-1 *string in* $\mathcal{Z}^2$.

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., [21, 12].

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. We actually show a much stronger result: namely, that the string folding problem is **MAX SNP**-*hard* and hence, under the same standard assumptions, cannot even be solved approximately to within some constant factor in reasonable time.[1]

Independently of this work, Crescenzi *et al.* [5] and Berger and Leighton [4] proved the **NP**-hardness of string folding for the HP model in $\mathcal{Z}^2$ and $\mathcal{Z}^3$ respectively. While these results are stronger than ours in that their alphabet size is the smallest possible, they apparently do 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, these authors construct intricate gadgets tailored to the HP model in the 2-d and the 3-d rectangular lattices.

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 [20], in which an unbounded alphabet is used. We reduce from the **MAX SNP**-hard problem Max-Cut [19], 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.* [8] 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(3) [18], 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 [10, 13] 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 score achievable by *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 concept 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

[1] For background on the complexity theoretic concepts used in this paper, see Section 1.3.

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 behaviour is that the reduction be *gap-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 with a gap-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 [21], which is not gap-preserving, so as to make it more robust. 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-neighbour contacts between equal letters. For the purposes of studying 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 adjacent to each other 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* $\mathbf{NP} = \mathbf{ZPP}$;

(ii) *for some constant $\gamma < 1$, if there exists a polynomial time algorithm that approximates* MAX-FOLD$_A$ *within a factor of γ, then* $\mathbf{NP} = \mathbf{ZPP}$.

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 gap-preserving reduction from the MAX-CUT problem, which is known to be **MAX SNP**-hard [19]. We then extend the technique employed in the proof to get our next (stronger) result, namely, the hardness of approximating the single-string version 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**-hard problem MAX-3SAT(3), a version of MAX-3SAT in which each variable appears in at most three clauses [18].

1.3 Complexity theory background and outline of proof techniques

For the benefit of those readers who are not familiar with the area, we present here a short explanation of several basic concepts from complexity theory used in the paper. We do not attempt to do anything like full justice to these ideas; for a more detailed exposition of these topics, see, e.g., [18, 2]. Readers who are already familiar with ideas such as **MAX SNP**-hardness and randomized reductions may safely skip to Section 1.4.

Following widely accepted conventional wisdom, we assert that a computational problem is *intractable*, or does not permit *efficient* solution, if there is no polynomial time algorithm for solving it. **P** denotes the class of decision problems (i.e., problems requiring only a 'yes' or 'no' answer) that have a polynomial time algorithm. A host of problems from a wide variety of areas have defied decades of attempts at efficient solution, and are believed not to be in **P**. In fact, even efficient *randomized* algorithms, which are allowed to make random choices and are required to give the correct answer only with high probability, are not known for these problems. Most of these problems belong to the abstract class **NP**. In order to give evidence for the intractability of a problem, it has thus become standard to show that the problem is in some sense "at least as hard as" any problem in **NP**. The theory of **NP**-hardness formalizes this idea.

A decision problem Π is said to be **NP**-*hard* if every problem Π' in the class **NP** can be efficiently reduced to Π (i.e., if there is a polynomial time algorithm—a *reduction*—that maps each input I' for Π' into an input I for Π such that I is a 'yes' instance of Π if and only if I' is a 'yes' instance of Π'). Of course, to show **NP**-hardness, it is sufficient to exhibit such a reduction for just one problem Π' that is already known to be **NP**-hard. A proof of **NP**-hardness constitutes compelling evidence that a problem is intractable, since the existence of a polynomial time algorithm for the problem would imply that every problem in **NP** can also be solved efficiently (i.e., that **NP** = **P**).

The notion of **NP**-hardness applies equally to optimization problems: if Π is an **NP**-hard optimization problem, then any problem in **NP** can be reduced in polynomial time to Π, such that if an instance I' of the **NP**-problem is mapped to the instance I of Π, then the yes/no answer to I' can be deduced efficiently from the optimum value $\mathrm{opt}(I)$ of I. For optimization problems, however, we may be interested in the complexity of obtaining only an *approximate* solution, within some specified factor of the optimum. An appropriate notion of intractability, **MAX SNP**-*hardness*, was introduced by Papadimitriou and Yannakakis [19] for the study of the complexity of approximation. We explain this concept below.

Let Π be a maximization problem, and let $\gamma \leq 1$. A γ-*approximation algorithm* for Π is an algorithm which, given an instance I of Π, produces a solution of I whose value is at least γ times the value $\mathrm{opt}(I)$ of the optimal solution of I. (There is an entirely analogous notion of approximation for minimization problems.) **MAX SNP** is an abstract class of optimization problems that includes many natural problems which are not only **NP**-hard to solve exactly, but also do not have efficient γ-approximation algorithms for some $\gamma < 1$ (unless **NP** = **P**). The existence of such problems in **MAX SNP** is a consequence of recent breakthroughs in the hardness of approximation, pioneered by Feige *et al.* [8] and Arora *et al.* [3]. We define the term

'**MAX SNP**-hard' recursively as follows. A problem Π is said to be **MAX SNP**-*hard* if there is an efficient *gap-preserving* reduction (as defined below) to Π from one such hard-to-approximate **MAX SNP**-problem, or from another problem already known to be **MAX SNP**-hard.[2] In analogous fashion to **NP**-hardness, the notion of **MAX SNP**-hardness captures the intractability of obtaining *approximate* solutions within a constant factor.

It follows from the work of Arora *et al.* [3] that if Π is **MAX SNP**-hard, then there exist constants $0 \leq \beta < \alpha \leq 1$ and an efficiently computable function f such that (unless $\mathbf{NP} = \mathbf{P}$) there is no polynomial time algorithm that answers 'yes' on inputs I of Π with $\mathrm{opt}(I) \geq \alpha f(I)$ and 'no' on inputs I with $\mathrm{opt}(I) \leq \beta f(I)$. (The algorithm is not required to give a meaningful answer on other instances.) This is often expressed by saying that it is **NP**-hard to distinguish instances with $\mathrm{opt}(I) \geq \alpha f(I)$ from those with $\mathrm{opt}(I) \leq \beta f(I)$, and we often refer to $\alpha - \beta$ as a 'gap' in the approximability of Π. This stronger result clearly implies that approximating Π to within a factor greater than β/α is intractable. A canonical example is the problem Max-Cut, which asks for a partition of the vertices of a graph into two parts so that the number of crossing edges between the two parts is maximized. This problem is **MAX SNP**-hard with $f(G)$ being the number of edges in the input graph G, and the gap $\alpha - \beta = 1/22$. As is shown in [23], given a graph with m edges, it is **NP**-hard to decide whether the maximum cut has at least $17m/22$ edges or at most $16m/22$ edges, and hence also to approximate the size of the largest cut to within a factor greater than $16/17$.

We now define gap-preserving reductions precisely. Let Π and Π' be two maximization problems, and let f' be a function on instances of Π'. Furthermore, let α' and β' be constants such that it is **NP**-hard to distinguish instances of Π' with $\mathrm{opt}(I') \geq \alpha' f'(I')$ from those with $\mathrm{opt}(I') \leq \beta' f'(I')$. A *gap-preserving reduction* from Π' to Π is an algorithm that maps each input I' for Π' to an input I for Π and satisfies the following special property. There are constants $0 \leq \beta < \alpha$ and an efficiently computable function f, such that

1. if $\mathrm{opt}(I') \geq \alpha' f'(I')$ then $\mathrm{opt}(I) \geq \alpha f(I)$;
2. if $\mathrm{opt}(I') \leq \beta' f'(I')$ then $\mathrm{opt}(I) \leq \beta f(I)$.

(Note that the behaviour of the reduction on other instances of Π' can be arbitrary.) Thus the reduction "translates" the gap $\alpha' - \beta'$ for Π' into a gap $\alpha - \beta$ for Π.

To prove that a maximization problem Π is **MAX SNP**-hard, it suffices to give an efficient gap-preserving reduction from some known **MAX SNP**-hard problem Π' to Π. This is essentially what we do in the present paper. In order to show that Max-Fold$_A$, the problem of folding a set of strings over a finite alphabet of size A, and Max-1-Fold$_A$, the problem of folding a single string over a finite alphabet of size A, are **MAX SNP**-hard, we present gap-preserving reductions from known **MAX SNP**-hard problems to these problems. Let us briefly sketch how this goes for Max-Fold$_A$, using the **MAX SNP**-hard problem Max-Cut mentioned above. First, it is a simple matter to give a gap-preserving reduction from Max-Cut to Max-Fold$_\infty$, the problem of folding a set of strings over an *unbounded* alphabet; indeed, as we shall see, the score of an optimal embedding in this reduction will be exactly equal to the size of a maximum cut, so the constants α, β are exactly the same as α', β'. To obtain a finite alphabet, we simply replace each symbol of the unbounded alphabet with a *string* of symbols (or *codeword*) over the finite alphabet. Suppose the codewords have length ℓ (which is not constant but depends on the input). Then, assuming the codewords behave exactly like the symbols they replace, we would simply inflate the score of each embedding by a factor of ℓ. Thus we again have a gap-preserving reduction with the same gap $\alpha - \beta$.

[2] This definition of **MAX SNP**-hardness, which is adapted from [2], differs from that in [19, 18], but suffices for the purposes of showing hardness of approximation.

Not surprisingly life is not quite that simple, because the codewords cannot be expected to behave exactly like single symbols. Specifically, two unequal codewords will in general have some symbols in common, so, unlike the symbols they replace, they may form bonds that contribute to the score. However we might hope that, with a judicious choice of codewords, it is possible to limit such 'unintended' bonds to a small constant fraction of the overall score. If this fraction is small enough, then we can still get a gap-preserving reduction with a slightly smaller gap $\alpha - \beta$. (The constant α will be unchanged, and β will be a little larger than β' due to the effect of unintended bonds.) Note that what we require of the codewords is that unequal codewords bond only very weakly with one another, no matter how hard they try by turning and twisting around each other in space: this is a much stronger condition than that demanded of classical error-correcting codes. We call a set of words with this property a *spatial code*. (The precise definition of spatial codes, given in Section 2.2, is in fact slightly more complicated as it requires certain global properties of *all* the codewords, and not just local properties of any pair.)

There is one further twist. Unfortunately we are not able to explicitly construct spatial codes in polynomial time, so we cannot implement the gap-preserving reduction sketched above. However, we *are* able to efficiently construct *random* sets of codewords which form a spatial code *with high probability*. (The phrase 'with high probability' here should be taken to mean 'with probability tending to 1 as the size of the problem input tends to infinity.') This means that the algorithm we use in our **MAX SNP**-hardness reduction is in fact a randomized algorithm, which works only with high probability. More precisely, condition 1 above still holds with certainty, but condition 2 now holds only with high probability. To reflect this fact, we say that MAX-FOLD$_A$ is **MAX SNP**-hard *under randomized reductions*.

What consequences do randomized reductions have for our **MAX SNP**-hardness results? If MAX-FOLD$_A$ has an efficient γ-approximation algorithm for a certain constant γ close enough to 1, then we cannot quite conclude (as in the case of non-randomized reductions) that **NP** = **P**. However, we can conclude that every problem in **NP** has a polynomial time *randomized* algorithm. This algorithm will give the correct answer with certainty on all 'yes' instances of the problem, and the correct answer with high probability on all 'no' instances. The technical jargon for this conclusion is that **NP** $\subseteq$ **co-RP**. (**co-RP** is the class of decision problems that have a polynomial time randomized algorithm of the above form.) Using simple complexity theoretic arguments, we can further conclude that every problem in **NP** has a randomized algorithm that solves it *without any error* in *expected polynomial time* (or, in other words, **NP** = **ZPP**), which is widely regarded as almost as unlikely as the conclusion **NP** = **P**, for essentially the same reasons. This is precisely what we claimed about MAX-FOLD$_A$ in Theorem 1.1.

To show that MAX-1-FOLD$_A$ is **MAX SNP**-hard under randomized polynomial time reductions (Theorem 1.2) we take a similar route, starting this time with a gap-preserving reduction from MAX-3SAT(3) to MAX-1-FOLD$_\infty$. We then convert it into a randomized gap-preserving reduction from MAX-3SAT(3) to MAX-1-FOLD$_A$, for a finite alphabet size A. The conversion this time uses a slightly more general notion of spatial codes.

1.4 Organization of the paper

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 gap-preserving reduction from MAX-CUT to MAX-FOLD$_\infty$, 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 gap-preserving reduction from MAX-CUT to MAX-FOLD$_A$. For clarity of exposition, our reductions will employ a neutral symbol; we show how we can eliminate the occurrences of this symbol to arrive at our first result, Theorem 1.1 above, in Section 2.4. 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.

2 Folding a set of strings

2.1 A simple reduction

In this section we describe a very simple gap-preserving reduction from MAX-CUT to the string folding problem using an *unbounded* alphabet. We call this (optimization) version of the string folding problem MAX-FOLD$_\infty$, and the decision version FOLD$_\infty$. The reduction is similar to a reduction from NOT-ALL-EQUAL-3SAT to FOLD$_\infty$ given by Paterson and Przytycka [20]. (Note: this reduction does *not* appear in the journal version of their paper [21].)

The input to the MAX-CUT problem is an undirected graph $G = (V, E)$. The goal is to find a *cut*, 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**-hard that there exist constants $0 < \beta < \alpha < 1$, such that the problem (α, β)-CUT of distinguishing graphs with m edges that have a cut of size at least αm from those that have cuts of size at most βm, is **NP**-hard [3]. The best known lower bound for the 'gap' $\alpha - \beta$ is $1/22$, as is implicit in [23]. 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:

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

In this definition, each $\mathbf{e}_j$ is a distinct letter from the alphabet, and '$*$' is assumed to be a special (neutral) symbol that does not contribute to the score. If such a symbol is not assumed to be 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. We now claim:

Lemma 2.1 *The graph G has a cut of size at least k if and only if there is an embedding of S_G in $\mathcal{Z}^3$ with a score of k.*

Proof: Suppose that there is an embedding of S_G in $\mathcal{Z}^3$ with a score of k. The lattice $\mathcal{Z}^3$ is bipartite and its points can be classified as being either *even* or *odd*. We now construct a cut $(C; V - C)$ as follows. Put the vertex v_i in C if the string $\mathbf{s}_i$ starts at an even lattice point, and in $V - C$ otherwise. The only

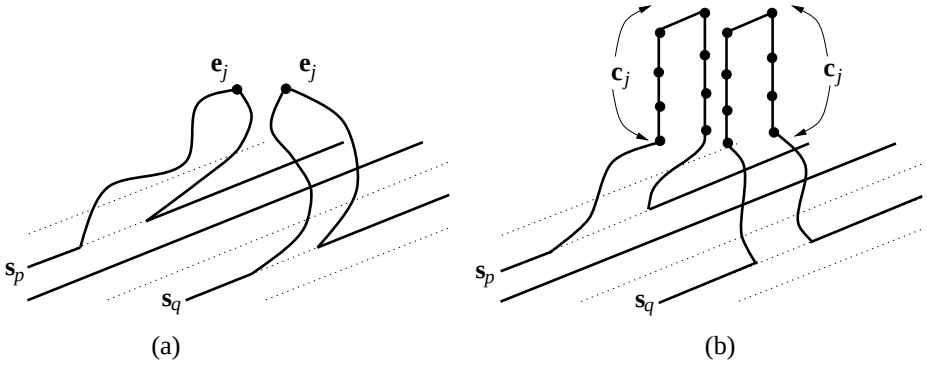


Figure 2: *An illustration of the reduction from* MAX-CUT *to string folding. The wavy lines stand for padding consisting of '$*$'s.* (a) *The two letters* $\mathbf{e}_j$ *can bond iff their strings start at lattice points of opposite parity.* (b) *Two codewords corresponding to an edge bond along the* z *direction.*

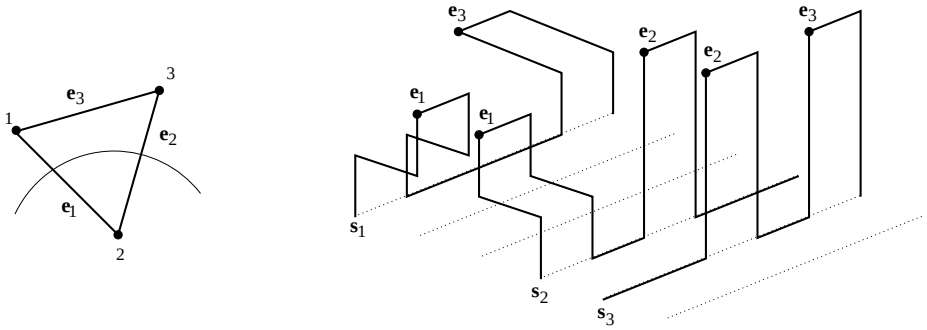


Figure 3: *A graph on three vertices, and an optimal embedding of the corresponding set of strings obtained by applying the reduction described in Section 2.1. The embedding corresponds to the optimal cut* $(\{1,3\};\{2\})$. *Note that there is no way to make the symbols* $\mathbf{e}_3$ *bond without altering the parity of the strings.*

symbols in S_G that can bond are the symbols $\mathbf{e}_j$ that correspond to the edges. Each such symbol appears exactly twice and can therefore contribute at most one bond. Let $e_j = (v_p, v_q)$. If the two copies of $\mathbf{e}_j$ bond, then these two symbols are necessarily on points of the lattice with different parities. It is easy to check that this implies that the starting points of the strings $\mathbf{s}_p$ and $\mathbf{s}_q$ have different parities. Thus, the edge e_j belongs to the cut. The size of the cut is therefore at least k.

Suppose now that the graph G has a cut $(C; V - C)$ of size at least k. It is easy to place the strings $\mathbf{s}_1, \ldots, \mathbf{s}_n$ so that the two copies of $\mathbf{e}_j$ can score if and only if the edge e_j is in the cut. One way of doing this is the following. The strings $\mathbf{s}_1, \ldots, \mathbf{s}_n$ are mostly made to lie in the x-y plane. The string $\mathbf{s}_i$ starts at point $(2i, 0, 0)$, if $v_i \in C$, and at $(2i + 1, 0, 0)$, otherwise. The string $\mathbf{s}_i$ is generally arranged to be parallel to the y axis—all the blocks $\mathbf{s}_{ij}$, for which $v_i \notin e_j$, are placed at points whose y coordinates are $2(j-1)$ and $2j-1$. If the edge $e_j = (v_p, v_q)$ belongs to the cut, then the two copies of the symbol $\mathbf{e}_j$ are placed on lattice points with opposite parities. The padding with '$*$'s makes it easy for the strings $\mathbf{s}_p$ and $\mathbf{s}_q$ to arrange a meeting between these two copies and then return to the line that they started in (see Figure 2(a)). The score of such an embedding is clearly equal to the cut size. ■

Figure 3 shows the instance of MAX-FOLD$_\infty$ obtained when the above reduction is applied to a small

graph.

An immediate consequence of Lemma 2.1 is that the reduction from MAX-CUT to MAX-FOLD$_\infty$ described above is gap-preserving: graphs with cuts larger than αm are mapped to sets of strings which are guaranteed to have score at least αm, and conversely, if every cut in a graph has size at most βm the optimum score of the corresponding set of strings is also bounded by βm. We have thus proved that:

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

(Part (i) follows from the same reduction by setting the threshold score $s = \alpha m$ for the FOLD$_\infty$ instance, and from the fact that the problem (α, β)-CUT is **NP**-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 is to try to replace each letter in the above reduction by a word from an error-correcting code over a fixed finite alphabet. In order that the codewords emulate alphabet symbols, we require that unequal codewords 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 definition.

Definition 2.3 (Intended and unintended score) *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 score *of the embedding is the number of intended bonds formed, and the* unintended score *is the number of unintended bonds formed.*

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 $\{1, \ldots, A\}$ 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 (of the form $g_c m\ell$, where g_c is a constant depending on c) in an obvious way by aligning the copies of the same string next to each other; e.g., for $c = 1, 2, 3, 4$ we have $g_c = 0, 1, 2, 4$ respectively. But if the strings constitute a (c, f)-spatial code, then no matter how we embed the cm strings we can only achieve a relatively small additional (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 '$*$'. We denote the corresponding string folding problems by FOLD$^*_A$ and MAX-FOLD$^*_A$. We show later, in Section 2.4, how to eliminate the use of the neutral symbol; note that this is not a trivial matter as in the case of an unbounded alphabet.

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.*

Note that we do not place any explicit constraint on the length of the codewords in the supposed $(2, f)$-spatial code. However, since they must be constructible in polynomial time, their length must obviously be polynomially bounded in m; and conversely their length cannot be too small as a function of m if they are to satisfy the requirements of a spatial code.

Proof: To prove part (ii) of the theorem, we define a gap-preserving reduction from MAX-CUT to MAX-FOLD$^*_A$. The same reduction yields part (i) when applied to (α, β)-CUT (with the threshold score set to $\alpha m\ell$).

Given a graph $G = (V, E)$ with m edges, we construct a $(2, f)$-spatial code, $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_m\}$, with $f < (\alpha - \beta)/6$. We then follow the reduction described in the previous section, replacing each symbol $\mathbf{e}_j$ by the codeword $\mathbf{c}_j$. (Because of parity considerations, we actually have to assume that the $\mathbf{c}_j$'s have odd length; if this is not the case, we can simply pad each $\mathbf{c}_j$ with a single '$*$' symbol.) Let $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ be the multiset of strings produced. The threshold score attached to this set is $\alpha m\ell$, where ℓ is the length of the codewords.

Suppose first that the input graph $G = (V, E)$ has a cut of size at least αm. We show that the set of strings produced has an embedding with a score of at least $\alpha m\ell$. This is easy: we can obtain such a score by playing according to the rules laid out in the proof of Lemma 2.1. For every edge e_j in the cut, we can make the two copies of the codeword $\mathbf{c}_j$ meet and score ℓ bonds (refer to Figure 2(b)). As there are at least αm edges in the cut, we get a total score of at least $\alpha m\ell$.

For the other implication in the reduction, suppose now that all the cuts of the input graph $G = (V, E)$ are of size at most βm. We have to show that the score of *any* embedding of the strings $\mathbf{s}_1, \ldots, \mathbf{s}_m$ is less than $\kappa m\ell$, for some constant $\kappa < \alpha$. Here we have to rely on the properties of the spatial code.

It is not difficult to see that the *intended* score of any embedding of the strings $\mathbf{s}_1, \ldots, \mathbf{s}_m$ is at most $\beta m\ell$. This follows from the fact that the only intended bonds possible are bonds between corresponding symbols of different copies of the same codeword. The claim then follows from the arguments given in the proof of Lemma 2.1.

We now bound the *unintended* score of any embedding of $\mathbf{s}_1, \ldots, \mathbf{s}_m$. All the *active*, i.e., non-neutral, symbols in this set of strings are contained in the codewords. There are exactly two copies of each codeword. By the definition of spatial codes, the unintended score of any embedding of $\mathbf{s}_1, \ldots, \mathbf{s}_m$ is at most $f \cdot 3cm\ell$. As $c = 2$, the unintended score of each embedding is at most $6fm\ell$.

The total score of each embedding, which is the sum of the intended and unintended scores, is therefore at most $\beta m\ell + 6fm\ell$. Since $f < (\alpha - \beta)/6$ by assumption, the constant $\kappa = \beta + 6f$ we get is less than α, showing that the reduction is indeed gap-preserving. Thus MAX-FOLD$^*_A$ is **MAX SNP**-hard, and approximating it within a factor of $\gamma > (\beta + 6f)/\alpha$ is **NP**-hard. ■

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. We are, however, able to show their existence and 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 very nearly completes the proof of Theorem 1.1: the only remaining detail is to eliminate the use of the neutral symbol '$*$', which we do in Section 2.4.

2.3 Randomized construction of spatial codes

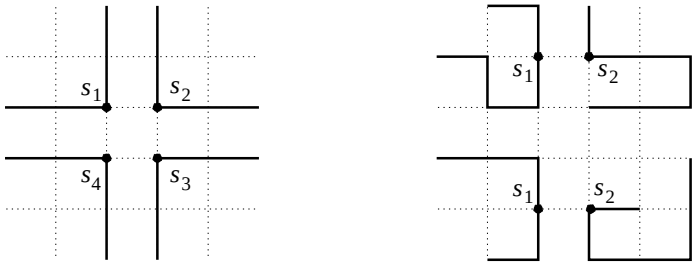
The probabilistic method is a widely used technique for proving the existence of combinatorial structures. In fact, it was by this method that Shannon first showed the existence of good error-correcting codes [22]. In this section, we employ this technique to demonstrate that almost any set of strings over a sufficiently large finite alphabet forms a good spatial code.

Before presenting the formal result, we first give a high level overview of the proof. Consider a set C consisting of m strings, each of length ℓ, which are chosen by assigning to each symbol a letter chosen independently and uniformly at random from the alphabet $\{1, \ldots, A\}$. For this to be a good (c, f)-spatial code, we require that when c copies of each string in C are embedded in $\mathcal{Z}^3$, the maximum unintended score achieved (let us denote it by X) is at most $f \cdot 3cm\ell$ (cf. Definitions 2.3 and 2.4). Since the set of strings is chosen randomly, the maximum unintended score X is a random variable. We claim that $X \leq f \cdot 3cm\ell$ with high probability, and that most sets C therefore form good spatial codes.

To see why this is the case, consider any fixed embedding $\mathcal{E}$ of the strings, and the unintended score $X_\mathcal{E}$ of that embedding. Recall that two symbols at adjacent sites in the embedding contribute to the unintended score only if they belong to different strings or if they belong to different positions in the same string. Any two such symbols score when they are both assigned the same letters from the alphabet $\{1, \ldots, A\}$. This occurs with probability $1/A$ since the letters are chosen independently. Moreover, the number of favourable pairs of symbols in any embedding is at most $3cm\ell$, the maximum number of adjacencies possible (cf. the discussion following Definition 2.4). Thus, each $X_\mathcal{E}$ is a sum of at most $3cm\ell$ Bernoulli trials with bias $1/A$. By linearity of expectation, the *expected* unintended score $E[X_\mathcal{E}]$ is at most $3cm\ell/A$. Now, the maximum unintended score X is the maximum of $X_\mathcal{E}$, taken over all embeddings $\mathcal{E}$ of the strings. If each $X_\mathcal{E}$ is well concentrated around its mean, which is bounded above by $3cm\ell/A$, we can expect X not to deviate far from $3cm\ell/A$. In fact, by the union bound we have

$$\Pr X \geq f \cdot 3cm\ell \;\leq\; (\#\mathcal{E}) \cdot \max_{\mathcal{E}} \Pr X_\mathcal{E} \geq f \cdot 3cm\ell,$$

where $\#\mathcal{E}$ denotes the number of different embeddings of the strings. It is not difficult to see that the number of different embeddings is at most exponential in $cm\ell$: each of the c copies of the m strings can be embedded in at most 6^l different ways in $\mathcal{Z}^3$, if its starting point is fixed, and all possible bond combinations can be realized within a small (polynomial sized) region, so we need only consider a small



(a) (b)

Figure 4: *Dependencies between unintended bonds arise in two ways.* (a) *Cycles: if bonds are formed because the pairs of symbols* $(s_1, s_2), (s_2, s_3)$ *and* (s_3, s_4) *are equal, then necessarily a further bond is formed by* (s_4, s_1). (b) *Repetition: due to the presence of multiple copies of the strings, all adjacencies between different occurrences of the symbols* s_1 *and* s_2 *either score or do not score simultaneously.*

number of starting points. So we get

$$\Pr X \geq f \cdot 3cm\ell \;\leq\; \exp(O(cml)) \cdot \max_{\mathcal{E}} \Pr X_{\mathcal{E}} \geq f \cdot 3cm\ell \,. \qquad (*)$$

To get a meaningful bound from the above, we thus require an exponential (i.e., Chernoff-type) bound on the tail probability of $X_{\mathcal{E}}$.

Recall that for any embedding $\mathcal{E}$, the random variable $X_{\mathcal{E}}$ is a sum of at most $3cm\ell$ Bernoulli trials with bias $1/A$. These events are, however, not all independent, and so we cannot directly invoke the Chernoff bound—consider the two situations depicted in Figure 4. Not surprisingly, however, the dependencies between the events are of a very special nature, and so $X_{\mathcal{E}}$ can be decomposed into a constant number of sums of mutually independent random variables, to each of which we can apply the Chernoff bound. This enables us to get a Chernoff-like bound of the following form for $X_{\mathcal{E}}$:

$$\Pr X_{\mathcal{E}} \geq f \cdot 3cm\ell \;\leq\; \exp(-\Omega(\ln A)m\ell).$$

Substituting for this in $(*)$, we get

$$\Pr X \geq f \cdot 3cm\ell \;\leq\; \exp(O(cml)) \cdot \exp(-\Omega(\ln A)m\ell).$$

Thus, if the alphabet size A is large enough, $X < f \cdot 3cm\ell$ with high probability.

We summarize this in Theorem 2.6, and in its proof we formalize the informal argument presented above.

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" in C. 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 graph 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.

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 2.8 *If $\ell \geq \ln m$, then the number of different adjacency graphs of embeddings of the multiset 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. The above two lemmas provide bounds on these two quantities. Combining them gives

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

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

Proof of Lemma 2.7: Consider an embedding $\mathcal{E}$ of C^c in $\mathcal{Z}^3$. Let G be the adjacency graph of the embedding. The number of possible unintended bonds in the embedding $\mathcal{E}$ is clearly at most the number of edges in G, which is at most $3cm\ell$. Consider a possible unintended bond. The two symbols that may form this unintended bond are assigned letters independently from the alphabet; otherwise the bond is intended. Therefore, the probability that each possible unintended bond materializes is exactly $1/A$. The expected number of unintended bonds is, therefore, at most $3cm\ell/A$. If all possible unintended bonds were independent, then we could easily finish the proof using the Chernoff bound. However, some of these bonds are dependent. Two factors (also illustrated in Figure 4) lead to these dependencies. The first is that each codeword is duplicated c times. The second is that the lattice $\mathcal{Z}^3$ contains cycles, so that the possible unintended bonds would be dependent even if the codewords were not duplicated.

We can, however, sidestep the dependency problem in the following way. Let G' be the *collapsed* adjacency graph of C^c. This graph is formed from the adjacency graph G by identifying the c vertices that correspond to the same symbol in the c copies of a codeword. The collapsed adjacency graph is in fact a multigraph, since we do not remove parallel edges that may be formed by this identification process (though we do

ignore the self-loops; these correspond to intended bonds). As each position in a codeword can bond with at most six other positions (actually at most four, unless it is a first or last symbol of a codeword, in which case the number is five), the degree of G' is at most $d = 6c$.

As the degree of G' is at most d, its *arboricity* is at most d. This means that the edges of G' can be decomposed into at most d forests. An easy way to see this is the following. Choose any spanning forest of G' and remove its edges from the graph. This reduces the degree of each non-isolated vertex by at least one. Repeat the process until no edges are left in the graph. Now let $F_1, F_2, \ldots, F_d$ be the forests obtained in this way. (The last few F_i's may be empty.) The following claim is now easily verified.

Claim 2.9 *The set of possible unintended bonds that correspond to the edges of a forest F_i in the collapsed adjacency graph G' are mutually independent.*

Thus, the total number of unintended bonds is the sum of d binomially distributed random variables. We can therefore finish the argument using the following variant of the Chernoff bound.

Lemma 2.10 *Let $X = \sum_{i=1}^{k} X_i$, where each X_i is a binomially distributed random variable. (The random variables X_i are* not *necessarily independent.) Let $\mu = E[X]$. Then, for every $B > 0$ we have*

$$\Pr[X \geq B] \;\leq\; k \cdot \mathrm{e}^{\left(1-\ln \frac{B}{k\mu}\right)\frac{B}{k}} \;.$$

Proof: If $X \geq B$, then there exists an i, such that $X_i \geq B/k$. So,

$$\Pr[X \geq B] \;\leq\; k \cdot \max_i \Pr[X_i \geq \tfrac{B}{k}] \;. \qquad (**)$$

The conventional Chernoff bound easily implies that if Y is a binomially distributed random variable with mean $E[Y]$ then

$$\Pr[Y \geq B] \;\leq\; \mathrm{e}^{(1-\ln \frac{B}{E[Y]})B} \;.$$

Thus, if $\mu_i = E[X_i] \leq \mu$, then

$$\Pr[X_i \geq \tfrac{B}{k}] \;\leq\; \mathrm{e}^{(1-\ln \frac{B}{k\mu_i})\frac{B}{k}} \;\leq\; \mathrm{e}^{(1-\ln \frac{B}{k\mu})\frac{B}{k}} \;.$$

The bound of the lemma is obtained by combining this bound with $(**)$ above. ∎

We apply Lemma 2.10 to the family of random variables $\{X_i\}$, where X_i is the unintended score contributed by the edges in the forest F_i. We can thus take $k = d$, $B = 3cfm\ell$ and $\mu \leq \frac{3cm\ell}{A}$. With these values, Lemma 2.10 gives

$$\begin{aligned} \Pr[\, X_{\mathcal{E}} \geq f \cdot 3cm\ell \,] \;&\leq\; d \cdot \mathrm{e}^{\left(1-\ln \frac{fA}{d}\right)\cdot \frac{3cfm\ell}{d}} \\ &\leq\; \mathrm{e}^{-(\alpha_1 \ln A - \alpha_3)m\ell} \;, \end{aligned}$$

for suitable choices of $\alpha_1 > 0$ and $\alpha_3 > 0$. This completes the proof of Lemma 2.7. ∎

Proof of Lemma 2.8: It is not difficult to see that any possible adjacency graph can be realized when the cm strings of C^c are placed in a $cm\ell \times cm\ell \times cm\ell$ box in $\mathcal{Z}^3$. (Note that $cm\ell$ is the total length of all the strings.) For each string, we therefore have only $(cm\ell)^3$ different starting points. Ignoring the effect of the other strings, each string can then be embedded in at most $6^{\ell-1}$ ways (in fact, at most $6 \cdot 5^{\ell-2}$ ways,

or $O(\gamma^{\ell-1})$ ways, where $\gamma \simeq 4.68$ is the connective constant of self-avoiding walks in $\mathcal{Z}^3$ [14]). The total number of adjacency graphs is therefore at most $((cm\ell)^3 6^\ell)^{cm}$. Since we assume that $\ell \geq \ln m$, this is less than $e^{\alpha_4 m\ell}$, for some large enough α_4 that depends only on c. This completes the proof of the lemma. ■

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 [23]. Doing this gives us a lower bound on A of something like 10^{300}, an extremely large (though 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). ■

Our randomized construction of spatial codes does not give us an unconditional version of Theorem 2.5, but it does yield the slightly weaker result that FOLD$^*_A$ is **NP**-hard under *randomized* polynomial time reductions, and similarly for the **MAX SNP**-hardness of MAX-FOLD$^*_A$. As pointed out before, these results follow essentially from the reduction described in the proof of Theorem 2.5, except that we use a *randomly chosen* set of words to replace the letters $\mathbf{e}_j$. The rest follows immediately from Theorem 2.6. Note that the randomization introduces a one-sided error in the reduction. If the FOLD$^*_A$ instance is obtained from a positive instance of (α, β)-CUT, it achieves the threshold of $\alpha m\ell$ with certainty. If, on the other hand, it is obtained from a negative (α, β)-CUT instance, it scores at most $(\beta + 6f)m\ell$ with high probability, but not with certainty. This implies that if FOLD$^*_A$ can be solved in polynomial time, then **NP** $\subseteq$ **co-RP**. As mentioned in Section 1.3, this further implies that **NP** = **ZPP**.

This leaves us but a small technical step away from the proof of our first main result, Theorem 1.1: that of eliminating the '$*$'s in the reduction to MAX-FOLD$^*_A$. We deal with this in the next short subsection.

2.4 Eliminating the neutral symbols

Our strategy for eliminating the neutral symbols from the reduction is to replace them with pieces of string over the alphabet $\{1, \ldots, A\}$ that bond very weakly with each other and with the rest of the substrings occurring in the MAX-FOLD$^*_A$ instance. In view of Theorem 2.6, it would be natural to assign random symbols from the alphabet to each occurrence of a '$*$', and to expect that this weak bonding property is achieved. This is indeed the case, provided the neutral symbols account for at most a constant fraction of the total length of the strings.

The total length of the strings obtained in the reduction from a graph with n vertices and m edges is $(2n - 1 + \ell) \cdot 2m + 2 \cdot mn$, of which $2m\ell$ symbols are active and $(2n - 1) \cdot 2m + 2mn$ symbols are neutral. If neutral symbols are allowed, then as we have seen it is sufficient to take $\ell \geq \ln m$ (as required for the existence of a good spatial code in Theorem 2.6). However, with this value of ℓ only a negligible fraction of the symbols in the strings are active. In order to make a constant fraction of the symbols active, we now choose the larger value $\ell = n$ (we assume here, for simplicity, that n is odd). With this choice, the total length of the strings becomes at most $8mn$ while the number of active symbols becomes $2mn$, so at most 75% of the symbols are neutral. Note also that the bonding capacity of the strings is at most $8mn \cdot 3 = 24mn = 24m\ell$.

We can now prove variants of Lemmas 2.7 and 2.8 that state that, with high probability, the number of unintended bonds in any fixed embedding of the resulting multiset of strings is a small fraction of the total bonding capacity of the strings, and that the number of different adjacency graphs of the strings is still relatively small.

Very briefly, the proofs go as follows. By arguing as in the proof of Lemma 2.7, we can show that the probability of the unintended score in any fixed embedding exceeding a fraction f of the total bonding capacity $24ml$ is at most $e^{-(\alpha_5 \ln A - \alpha_6)m\ell}$, for some positive constants α_5 and α_6. If we follow the line of reasoning in the proof of Lemma 2.8, we can show that the number of adjacency graphs for this new set of strings is $\leq (8m\ell)^{3n}6^{8m\ell}$, which is at most $e^{\alpha_7 m\ell}$ for α_7 a large enough constant. Combining these two bounds as before we conclude that, for a sufficiently large value of A, the maximum unintended score of the set of strings is at most $f \cdot 24m\ell$ with high probability. Theorem 1.1 now follows immediately by choosing $f < (\alpha - \beta)/24$ and setting $\gamma = (\beta + 24f)/\alpha$.

3 Folding a single string

In this section, we extend our techniques to show that the problem FOLD$_A$ of the previous section remains NP-hard under randomized polynomial time 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., [21]).

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. Indeed, this material is motivated not only by the inherent importance of single string folding, but also by our wish to illustrate the generality of our techniques. We shall comment more on this at the end of this subsection.

Our strategy proceeds as follows:

1. We start with a gap-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 gap-preserving nature of the reduction implies the following: there are constants $0 \leq \rho < \sigma$ 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 $0 \leq \rho' < \sigma'$ 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 the 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 clearly exponential in the string length 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 times 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

$$\mathbf{c}_1^1 - \mathbf{c}_1^2 - \cdots - \mathbf{c}_1^\ell \sim \mathbf{c}_2^1 - \mathbf{c}_2^2 - \cdots - \mathbf{c}_2^\ell \sim \mathbf{c}_3^1 - \mathbf{c}_3^2 - \cdots - \mathbf{c}_3^\ell \sim \cdots \sim \mathbf{c}_k^1 - \mathbf{c}_k^2 - \cdots - \mathbf{c}_k^\ell \quad ,$$

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 times the total length. This gives us a gap whose size is a constant times 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 the constant 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* letter 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 optimal 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 constants $\alpha_1, \alpha_2 > 0$. This probability approaches 1 if we take A to be a sufficiently large constant. By choosing the constant f to be less than $(\sigma' - \rho')/3$, we can make the unintended score of every embedding small 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 2.4.

Remark: As mentioned earlier, the approach used here to deal with single string folding is a generalization of that used in the previous section to deal with multiple string folding. The new complication arises in the replication process (point 2 above). In the case of multiple string folding, the simplicity of the original gap-preserving reduction allowed us to formulate the replication process in terms of the natural concept of spatial codes. This concept allowed us to isolate the role of randomness in the reduction. It is possible to broaden the definition of spatial codes 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. Our main task will be that in point 1 above, namely establishing a gap-preserving reduction for the problem MAX-1-FOLD$^*_\infty$ over an unbounded alphabet. Existing reductions for single string folding problems do not have the gap-preserving property, and a simple construction such as the one we used for multiple strings in Section 2.1 is apparently not sufficient. Therefore, we will have to do a little work here, which we describe in Section 3.2. 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.

Before proceeding with this, however, we take the opportunity to emphasize the generality of the above approach. Essentially, it provides a way of translating hardness results in an unbounded alphabet model to results in the same model with a fixed finite alphabet. This appears to be a useful technique because it is much simpler to "program" a reduction over an infinite alphabet, where certain gadgets can be assumed not to interact because they have no symbols in common. Almost all that is required for our technique to work is that the reduction to the unbounded alphabet model be gap-preserving, and consist of gadgets that allow for replication. Most significantly, the technique is independent of the details of the model. Thus for example it is robust not only with respect to the number of strings (as we have seen), but also with respect to variations in the lattice and (to some extent) the scoring function.

3.2 A gap-preserving reduction over an unbounded alphabet

MAX-3SAT is the problem of computing the maximum number of clauses that can be satisfied simultaneously, given a *3CNF* Boolean formula. A *3CNF* formula consists of a conjunction of clauses which are disjunctions of at most three variables or their negations. We start from MAX-3SAT(3), a version of MAX-3SAT in which each variable occurs at most three times in the whole formula. This problem is known to be **MAX SNP**-hard (see, for instance, Theorem 13.10 of [18]). It then follows from the work of Arora *et al.* [3] that MAX-3SAT(3) is **NP**-hard to approximate within some constant factor $\gamma < 1$.

We now present a gap-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 $\mathcal{N}$ (with a neutral symbol '$*$'). The string $\mathbf{s}_\phi$ consists of a *rod-flap combination* (see Figures 6 and 7) of constant size for each variable, and a *ligand*, also of constant size, for each clause (see Figure 5). 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, with the exception of *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 that occur 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 8). 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 ligands. Furthermore, the construction of the gadgets ensures that there is no profit in the ligands bonding with more than one rod, or in the rod-flap pairs deviating from their intended embedding. This means 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 a gap-preserving reduction. Figure 9 sketches an optimal embedding of the entire string.

Remark: This reduction owes its origins to an earlier reduction of Paterson and Przytycka [20, 21], from 3SAT to 1-FOLD$_\infty$. Our main innovation here is to make the reduction gap-preserving, 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. ■

We now proceed to give the details of the construction described informally above.

The ligand

Each ligand consists of a string segment of length 18, with the symbols occurring in the pattern shown in Figure 5. Ligands corresponding to different clauses are built out of disjoint sets of symbols. The symbol '**c**' indicates the position where the associated clause symbol $\mathbf{c}_j$ should occur. The ligand is designed so that, in order to achieve the maximum possible score, it has to fold in a unique fashion (up to reflection and rotation). Moreover, this fold is highly robust in the sense that at least *two* bonds are broken in any non-optimal embedding.

Proposition 3.1 *A ligand string has a unique optimal embedding with a score of 11. Moreover, any non-optimal embedding of the ligand has a score of at most 9.*

We defer the proof of this fact to the Appendix.

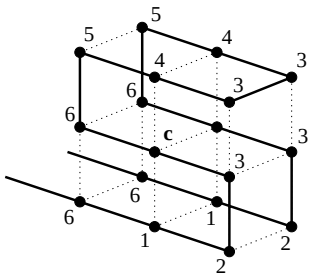


Figure 5: *A clause ligand, shown in its unique optimal embedding.*

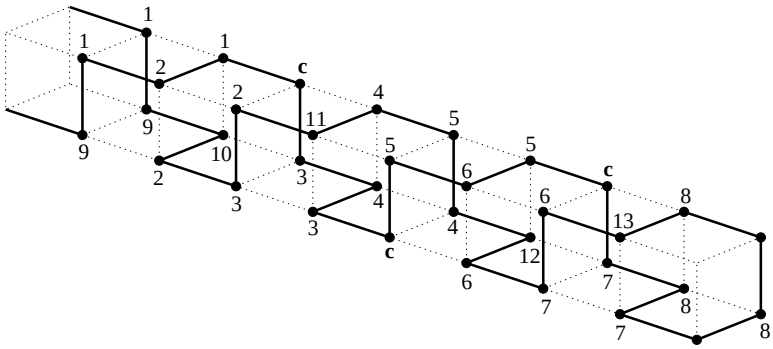


Figure 6: *The symbol pattern in a rod. The rod is shown in one of its two optimal embeddings.*

Note that the optimal embedding of the ligand leaves only one vacant site adjacent to the clause symbol. Thus the above robustness property ensures that, if the ligand bonds with more than one rod, then it loses at least two bonds in the process. But since the ligand can bond with at most three rods (namely, those corresponding to the variables occurring in its clause), this ensures that any advantage gained from its bonding with more than one rod is canceled by the bonds broken within the ligand. Thus we may assume, w.l.o.g., that each ligand bonds with at most one rod, and that the maximum score achievable by all bonds involving the ligand is 12.

The rod-flap combination

The string $\mathbf{s}_\phi$ has a rod-flap combination for each variable x_i. A rod gadget consists of a string with the symbol pattern shown in Figure 6. This is a slight modification of the helical "tooth" construction of Paterson and Przytycka [20, 21]. The '**c**'s indicate the positions of the three symbols corresponding to the clauses in which the variable x_i occurs. Note that we can assume, w.l.o.g., that no variable occurs only positively or only negatively in the formula, so that there is always one occurrence of one type (positive or negative) and two of the other. The clause symbol corresponding to the lone occurrence is placed *between* the other two on the string.

A flap gadget has the symbol pattern shown in Figure 7. The symbols along the top and bottom lines in this figure bond with the symbols on the corresponding rod. The remaining symbols cause the flap to fold like a "sheet" (in the language of protein motifs).

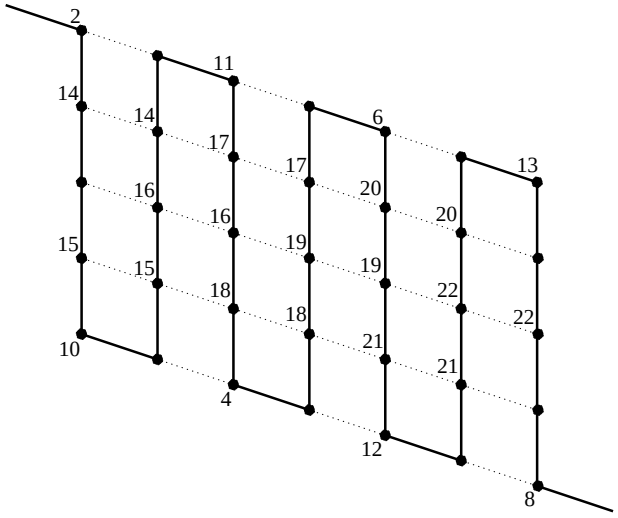


Figure 7: *The flap corresponding to the rod of Figure 6.*

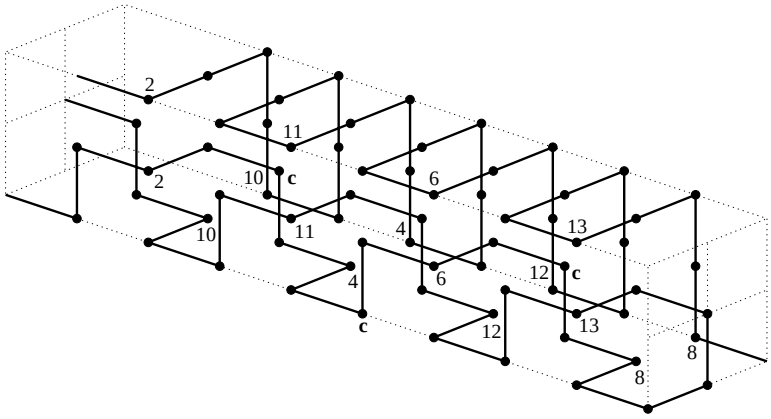


Figure 8: *One of the two possible optimal embeddings of a rod-flap combination. Only symbols involved in bonds between the rod and the flap are shown. The '***c***'s indicate the positions of clause symbols.*

The combination of a rod and its associated flap has only two distinct optimal embeddings (up to rotation and reflection), one of which is shown in Figure 8. In the other optimal embedding, the flap is wrapped around the other two faces of the rod. In the embedding shown, the edge with two clause symbols (corresponding to two negative or two positive occurrences of the variable) is covered and the opposite edge is exposed. This situation is reversed in the other optimal embedding.

Proposition 3.2 *Each rod-flap combination has exactly two optimal embeddings, with a score of* 34*.*

Proposition 3.2, whose proof is again deferred to the Appendix, actually implies that in any optimal embedding of the entire string $\mathbf{s}_\phi$, we may assume w.l.o.g. that every rod-flap combination is itself optimally embedded, and hence corresponds to a valid assignment of a truth value to the corresponding variable. In

particular, it is not advantageous for the combination to "cheat" by exposing clause symbols on both of the two opposite edges of the rod.

To see this, note first that by the discussion following Proposition 3.1 we may partition the ligands into classes according to the (unique) rods they bond with. Call the string portions corresponding to a rod-flap combination and its bonding ligands a "unit." There can be no bonds between different units. In particular, rods carrying the same clause symbol do not bond because of parity considerations mentioned above. Since the ligands are equipped with ample padding, we can rearrange the embedding without changing its score so that different units occupy disjoint regions in $\mathcal{Z}^3$.

Now focus on a single unit, and suppose that its rod-flap combination is *not* optimally embedded, i.e., does not correspond to a valid truth value assignment to its variable. We claim that we can rearrange this embedding to be optimal without decreasing the score. Suppose the number of clause symbols in this rod that bond with ligands is k (so $0 \leq k \leq 3$). Now a moment's thought should convince the reader that at least one of the two optimal embeddings of the rod-flap combination has the property that it leaves exposed at least $\lceil k/2 \rceil$ of these clause symbols. So if we rearrange the combination according to this embedding, the number of bonds between clause symbols that we lose is at most $k - \lceil k/2 \rceil = \lfloor k/2 \rfloor \leq 1$. On the other hand, since the embedding of the combination is now optimal, by Proposition 3.2 we gain at least one bond. Hence we may assume that each rod-flap combination is itself optimally embedded in every optimal embedding of $\mathbf{s}_\phi$.

Putting the gadgets together

It remains only to specify how the above gadgets are assembled together into a single string. The string $\mathbf{s}_\phi$ is constructed by concatenating n rod-flap combinations (one for each variable in ϕ) and m ligands (one for each clause in ϕ) built using disjoint sets of symbols (save for the clause symbols, which occur as described above), with enough padding (strings consisting of the neutral symbol '$*$') inbetween. The padding must be sufficient for each rod-flap combination and each ligand to bond optimally, and for the ligands to bond with any of the rods that contain the same clause symbol as they do. Thus each rod is connected to its associated flap with constant-length padding, and the n rod-flap combinations are connected in sequence with constant-length padding between each adjacent pair. Finally, the m ligands are attached to the resulting string in sequence, with $\Theta(m)$-length padding for each ligand. This is sufficient to allow each ligand to reach any rod with which it wishes to bond. (Recall that n and m are within a constant factor of one another.) We also 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 ligands. Figure 9 contains a sketch of an optimal embedding of the entire string.

In light of the discussions following Propositions 3.1 and 3.2, we have proved:

Proposition 3.3 *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 $34n + 11m + k$.*

Since in any instance of Max-3Sat(3) we have $n \leq 3m$, Proposition 3.3 tells us that the mapping $\phi \to \mathbf{s}_\phi$ is a gap-preserving reduction from Max-3Sat(3) to Max-1-Fold$^*_\infty$. As before, each '$*$' can be replaced by a distinct unused symbol. We may therefore conclude that:

Corollary 3.4 Max-1-Fold$_\infty$ *is* **MAX SNP**-*hard.*

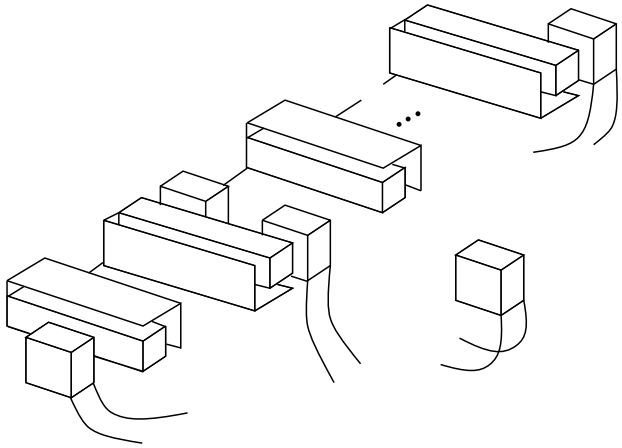


Figure 9: *An optimal embedding of the entire string* $\mathbf{s}_\phi$.

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 times the string length. Note that in $\mathbf{s}_\phi$ the active parts (the 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 is sufficient to allow 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 10, in an optimal embedding.

Now Proposition 3.3 yields the following property of $\mathbf{s}_\phi^\ell$:

Corollary 3.5 *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. Note that the length of the string $\mathbf{s}_\phi^\ell$ is $\Theta(m\ell + m^2)$. Hence, 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, this ensures that if we replace the symbols of $\mathbf{s}_\phi^\ell$ with random letters from the alphabet $\{1, 2, , \ldots, A\}$ for a sufficiently large constant A, we get a (randomized) gap-preserving reduction to Max-1-Fold$^*_A$. Finally, we may remove the neutral symbol '$*$' exactly as in Section 2.4,

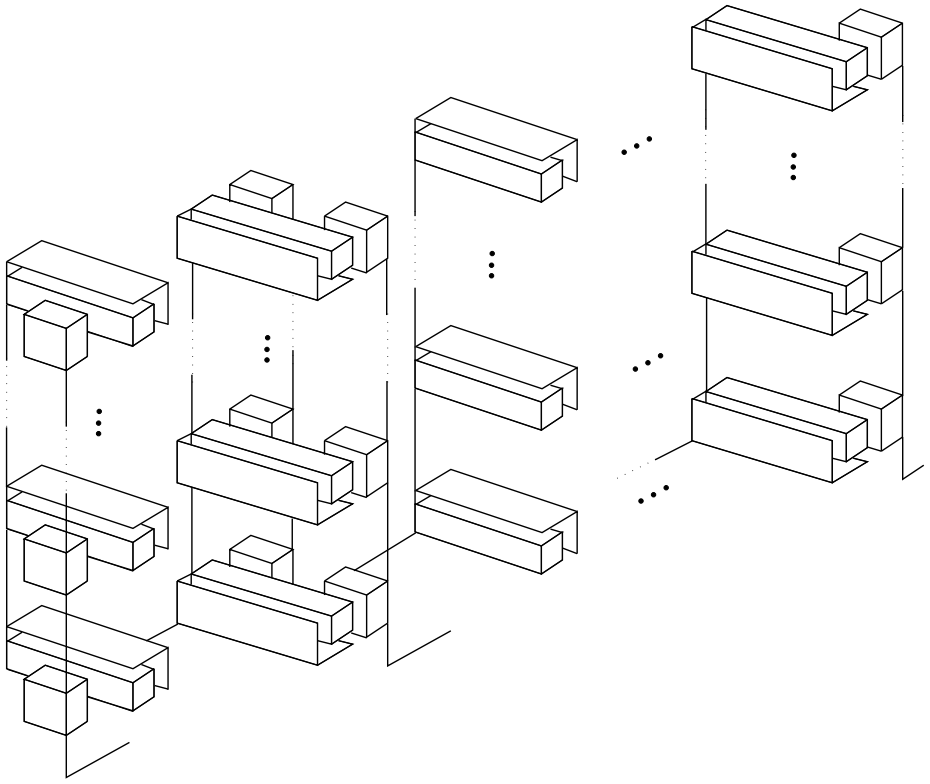


Figure 10: *An optimal embedding of the replicated string* $\mathbf{s}_{\phi}^{\ell}$.

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 Concluding remarks

We have presented **NP**-hardness and approximation 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 gap-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 with an unbounded alphabet already exist for a wide variety of lattices (see, e.g., [12]); 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 required to be extremely large (see the remark at the end of section 2.3). We have made no attempt here to minimize it. Some tuning of our arguments would, with some effort, dramatically reduce the size; 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 true HP model). Although proofs of **NP**-hardness for the HP-model in $\mathcal{Z}^2$and in $\mathcal{Z}^3$ do now exist [5, 4], the question of hardness of approximation for the string folding problem over a reasonably small alphabet remains open.

Finally, we mention two intriguing 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 stronger 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, Mike Paterson for sending us copies of [20] and [21] and for interesting discussions on snails, and Neal Madras for input on self-avoiding walks.

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. *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, 1997, pp. 390–399.

[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, pp. 399–446.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM* **45**(3) (1998), pp. 501–555.

[4] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology* **5** (1998), pp. 27–40.

[5] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni and M. Yannakakis. On the Complexity of Protein Folding. *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, 1998, pp. 597–603. Full version to appear in *Journal of Computational Biology*.

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

[7] 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.

[8] 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.

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

[10] W.E. Hart and S. Istrail. Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal. *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, 1995, pp. 157–168.

[11] 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.

[12] W.E. Hart and S. Istrail. Robust proofs of NP-hardness for protein folding: general lattices and energy potentials. *Journal of Computational Biology* **4** (1997), pp. 1–22.

[13] 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. *Proceedings of the 1st Annual International Conference on Computational Molecular Biology*, ACM Press, 1997, pp. 137–146.

[14] N. Madras and G. Slade. *The self-avoiding walk*. Birkhäuser, Boston, 1993.

[15] A. Nayak, A. Sinclair and U. Zwick. Spatial codes and the hardness of string folding problems. *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998, pp. 639–648.

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

[17] 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.

[18] C.H. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.

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

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

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

[22] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal* **27** (1948), pp. 379–423, 623–656.

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

[24] 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.

Appendix: Proofs of robustness of the gadgets in section 3.2

Proof of Proposition 3.1: Note first that, in the embedding shown in Figure 5, a bond is formed between every possible pair of equal symbols. This immediately implies that the embedding is optimal (and its score is plainly 11). To show uniqueness, we introduce the concept of a "bond graph," following [21]. The *bond graph* of an embedding of a string **s** is the graph whose vertices are the *occurrences* of the symbols of **s** and which has an edge between two symbol occurrences s_i, s_j iff either (i) s_i, s_j are adjacent in **s** (i.e., $|i-j|=1$), or (ii) there is a bond between s_i and s_j in the embedding. The bond graph of the embedding of Figure 5 is shown in Figure 11. Since any optimal embedding achieves all these bonds, it has the *same* bond graph. Therefore, we need only show that the graph of Figure 11 has (up to rotation and reflection) a unique embedding in $\mathcal{Z}^3$.

To see this, note that the bond graph consists of a sequence of eight "squares" (cycles of length four, such as $ABba$), arranged end-to-end in a closed cycle. Clearly each of these squares is necessarily embedded as a unit square in $\mathcal{Z}^3$. Suppose we fix the embedding of the square $ABba$. Then the square $BCcb$ is necessarily embedded adjacent to it, "hinged" along the common edge Bb. The same holds for each successive square proceeding around the cycle. Thus we see that the eight squares form a cylinder of unit height in $\mathcal{Z}^3$, whose ends are planar closed curves of length eight. Finally, note that the bond graph contains four additional edges, which form a chord of length two across each of these curves (between points A-E and points a-e respectively). This additional constraint implies that the planar curve ends are in fact 2×2 squares with corners at B, D, F, H and b, d, f, h. Thus the optimal embedding is unique up to rotation and reflection.

The proof of robustness of the embedding proceeds along the same lines. One simply considers the bond graph of Figure 11 with each one of the eleven bonds omitted in turn. (Note that we do not consider omitting the edges corresponding to adjacencies in the string.) By symmetry there are only four distinct cases. In each case, it is easy to check by an argument similar to that above that the only possible embedding of the bond graph corresponds to the one in Figure 5. Hence at least two bonds are broken in any non-optimal embedding. ■

Proof of Proposition 3.2: We make essential use of the bond graph, as in the previous proof. First we consider the rod (see Figure 6). Note that the embedding shown in Figure 6 has a score of 17, and since all possible bonds are formed, this is optimal. The bond graph of this embedding is shown in Figure 12(a); note that any other optimal embedding has the same bond graph. The proof that this bond graph has an essentially unique embedding in $\mathcal{Z}^3$ is, in effect, the same as Paterson and Przytycka's [20, 21] proof of uniqueness (up to rotation and reflection) of the optimal embedding of their "tooth" construction, and is presented here for completeness.

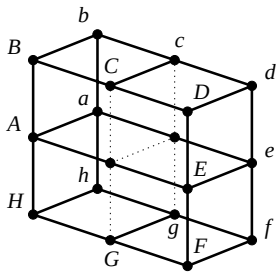


Figure 11: *The bond graph of an optimal ligand embedding.*

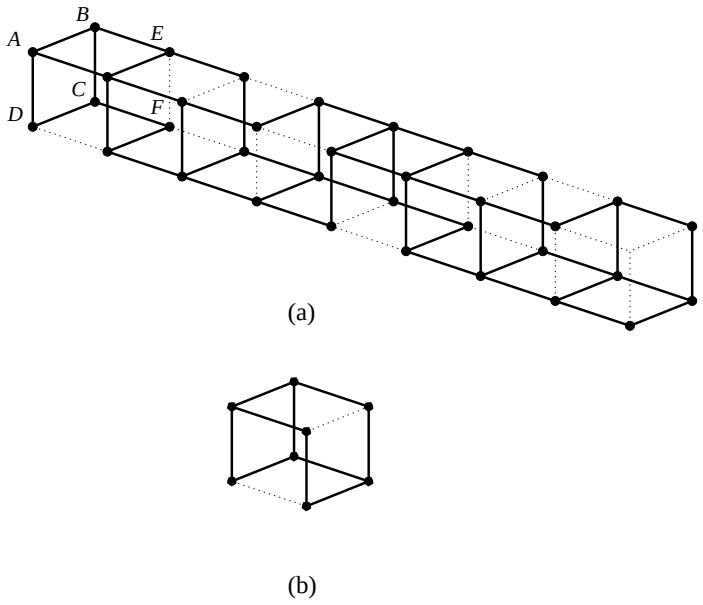


Figure 12: (a) *The bond graph of an optimal rod embedding.* (b) *The bond graph of a "unit cube."*

Note that the bond graph can be viewed as a sequence of "unit cubes" (with some edges omitted). We claim first that each of these portions is necessarily embedded as a unit cube in $\mathcal{Z}^3$. To see this, observe that the bond graph of each cube (except the last one) consists of two squares hinged at a common edge, plus a disjoint path of length three connecting opposite corners of these squares (see Figure 12(b)). The last cube has a disjoint path of length *two* connecting the two squares. That this is indeed the bond graph is immediate for the leftmost cube. For the rest of the cubes, one of the squares is enforced by the embedding of the previous cube.

It is easy to see that either of the two unit cube bond graphs described above has a unique embedding in $\mathcal{Z}^3$ (up to rotation and reflection), namely as a unit cube. Now consider the bond graph of the entire rod (Figure 12(a)). Note that the leftmost face $ABCD$ is necessarily embedded as a square. Once this is fixed, the second square $BEFC$ has the freedom to bend in two ways about the edge BC. The embedding of $BEFC$ determines which of the two possible reflections of the cube is obtained. Note that the rest of the cubes do not enjoy this freedom. We can proceed in this way along the whole rod, one cube at a time. The embedding we obtain is unique up to the choice of embedding of the initial cube. This freedom means that the rod may be embedded either as an anti-clockwise helix, as in Figure 6, or as a similar clockwise helix. For our purposes, these two embeddings are equivalent because the sequences of symbols on pairs of opposite edges of the rod are the same in both cases.

We now turn our attention to the flap (see Figure 7). The embedding of the rod-flap combination shown in Figure 8 has a score of 34 (17 of which are bonds within the rod), and again all possible bonds are formed. Hence any optimal embedding has the same bond graph, shown schematically in Figure 13. In this figure we have drawn the rod as a solid block, since we know from the previous argument that it is necessarily embedded in this way. We wish to argue that there are exactly two embeddings of the flap portion of the bond graph, given the embedding of the rod. The second embedding differs from that of Figure 8 only in that the flap wraps around the rod in the opposite direction.

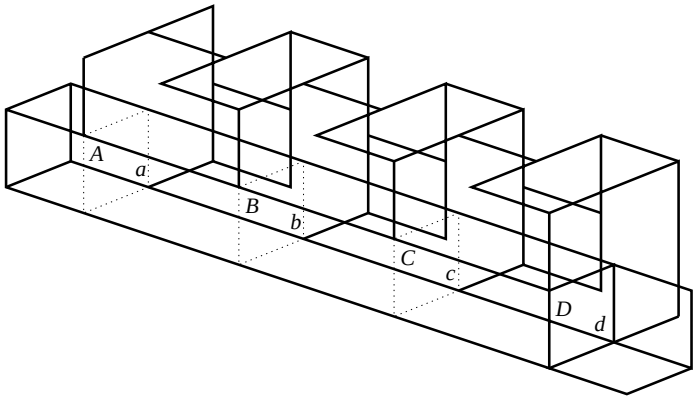


Figure 13: *The bond graph of an optimal rod-flap embedding. The bond graph of the rod itself is shown only schematically.*

To see this, note first that the pair of points (A, a) on opposite edges of the rod are joined by a path of length six in the bond graph. But once the rod is fixed, there are exactly two paths of length six between these two points: namely the one shown in the embedding of Figure 8 and one in which the path wraps around the rod in the opposite direction. Similarly, there are exactly two paths of length six connecting each of the pairs (B, b), (C, c) and (D, d). Now it is easy to see that in fact only two combinations of these four paths are possible: namely, the combinations in which all paths wrap around the rod in the same direction. This follows from the fact that certain intermediate points along the paths are constrained to lie at small distances from one another. Now that these four paths are fixed, it is easy to see that the embedding of the remainder of the flap is forced. ■