Bounding A Protein's Free Energy In Lattice Models Via Linear Programming

Robert Carr [*] William E. Hart [*] Alantha Newman [†]

April 14, 2004

Abstract

The established HP lattice 2D and 3D models have been useful abstractions in understanding protein structure. In these models, a protein folds to maximize H-H contacts (minimize free energy). We examine and compare integer programming models for the 2D lattice, whose linear relaxations provide non-trivial upper bounds on the maximum number of contacts. These bounds can be used in a branch-and-bound approach to solve the problem optimally, and are of independent interest as well. In particular, we seek to beat the simple combinatorial bound that arises from the lattice being bipartite.

[*]Discrete Algorithms and Math Department, Sandia National Laboratories, Albuquerque, NM. email: `rdcarr,wehart@sandia.gov`.

[†]Laboratory for Computer Science, MIT, Cambridge, MA. email: `alantha@theory.lcs.mit.edu`.

1 Introduction

We discuss discrete optimization approaches to the problem of protein folding in the Hydrophobic-Hydrophilic (HP) model. The widely-studied HP model was introduced by Ken Dill [5, 6]. This model abstracts the dominant force in protein folding: the hydrophobic interaction. The hydrophobicity of an amino acid measures its affinity for water, and the hydrophobic amino acid residues of a protein form a tightly clustered core. In the HP model, each amino acid residue is classified as an H (hydrophobic) or a P (hydrophilic). The model further simplifies the problem by restricting the feasible foldings to the 2D or 3D square lattice. An optimal conformation for a string of amino acid residues in this model is one that maximizes the number of H-H contacts, i.e. pairs of H's that occupy adjacent lattice points but are not adjacent on the string.

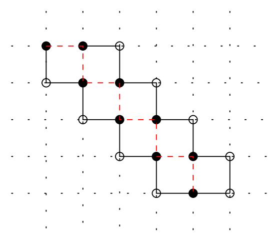


Figure 1: An optimal folding for the string 101010101001010101. 0's and 1's are denoted by unfilled and filled dots, respectively. Contacts are denoted by the dashed lines.

The problem of protein folding in the HP model is combinatorially equivalent to folding a string of 0's and 1's on the square lattice to form a self-avoiding walk that maximizes the number of pairs of adjacent 1's, i.e. with H=1 and P=0. For example, suppose we have the string 101010101001010101. An optimal folding for this string on the 2D square lattice is shown in Figure 1. This folding has eight *contacts* or pairs of 1's that are placed on adjacent lattice points but are not neighbors in the string.

Finding an optimal folding is NP-hard for both the 2D and 3D square lattices [4, 3]. Computing an exact upper bound on the optimal number of contacts is therefore probably very difficult to do efficiently, so we would like find the best *efficiently computable* upper bound. The better the upper bound–the closer the upper bound is to the maximum number of contacts–the more information we have about the actual optimal value. Despite the fact that the best-known efficiently computable upper bound is quite straightforward and was introduced at least a decade ago, it has yet to be improved upon.

To explain this upper bound, we introduce some notation that we will use throughout the paper. Let $S = s_1 s_2 \ldots s_n$ be a binary string in $\{0, 1\}^n$. We refer to each 1 in an odd position on the string as an *odd-1* and each 1 in an even position on the string as an *even-1*. Let $\mathcal{O}[S]$ represent the number of odd-1's in S and let $\mathcal{E}[S]$ represent the number of even-1's in S. Since both the 2D and 3D square lattices are bipartite graphs, each odd-1 can have contacts with only even-1's. Similarly, each even-1 can have contacts with only odd-1's. Thus, a simple upper bound for the 2D problem is:

$$2 \cdot \min\{\mathcal{E}[S], \mathcal{O}[S]\} + 2. \tag{1}$$

For the 3D problem, an analogous argument leads to an upper bound of $4 \cdot \min\{E[S], \mathcal{O}[S]\} + 2$.

These simple upper bounds can be used to obtain algorithms with approximation guarantees of 1/4 [8] and 1/3 [10] for the 2D problem and 3/8 [8] for the 3D problem. The approximation guarantee for each of these algorithms is based on the observation that it yields a folding with at least 1/4, 1/3, and 3/8, respectively, as many contacts as permitted by the upper bound. Our goal is to improve upon this upper bound, which could potentially lead to improved approximation algorithms and more efficient exact algorithms (using for instance a branch and bound technique).

Overview In this paper, we investigate various integer programs that model the 2D problem and examine the upper bounds given by their respective linear programming relaxations. Our formulations are similar to those studied previously in [7], which appears to contain the only other description of this problem as an integer program. In Section 2, we give some background on linear programming and its applications to finding upper bounds for hard combinatorial maximization problems. In Section 2.1 and Section 2.2, we define some notation and explain how we we use variables in our integer programs. In Section 2.3, we give a basic integer program for our problem and explain why there is a one-to-one correspondence between integer solutions and actual foldings of strings on the 2D square lattice. In Section 2.4, we introduce a slightly modified approach to our formulations that uses aggregate variables. In Section 2.5, we show how to obtain a linear programming relaxation from the IP in Section 2.3 and discuss ways to measure the quality of the upper bound provided by a linear programming relaxation. We show that the upper bound given by this relaxation is actually quite poor and needs to be strengthened. In Section 3.1, we present improved linear models and prove that they are at least as good as the best-known upper bound. In Section 3.2, we give an example of a string for which the linear programming relaxation is off by a factor of 2 and in Section 4 we make some suggestions for how to further strengthen the relaxation. Finally, in Section 5, we give some experimental results which show that the upper bound provided by the linear programming relaxation can beat the simple upper bound.

2 Integer Program Formulations

Linear Programming is often used to find upper bounds on the cost of optimal solutions to hard combinatorial (maximization) problems. The method is to (i) find an integer program that describes the problem, (ii) relax the integrality constraints to obtain a linear programming relaxation for the problem, (iii) solve the linear program to compute a bound on an optimal integral solution. Although solving an integer program can be computationally difficult, we can solve linear programs efficiently. Thus, this is one approach to finding efficiently computable upper bounds. We consider a simple integer program for our problem. First, we introduce some necessary notation.

2.1 Notation

Let I be the set of indices in a given binary string S of length n, i.e. $I = \{1, \dots n\}$. We break down I as follows:

$\mathcal{E}$ is the set of indices of elements in even positions.
$\mathcal{O}$ is the set of indices of elements in odd positions.

We break down $\mathcal{E}$ and $\mathcal{O}$ further as follows:

$H_{\mathcal{O}}$ is the set of indices of odd-1's in S,
$H_{\mathcal{E}}$ is the set of indices of even-1's in S,
$P_{\mathcal{O}}$ is the set of indices of odd-0's in S,
$P_{\mathcal{E}}$ is the set of indices of even-0's in S.

Thus, $H_{\mathcal{E}} \cup P_{\mathcal{E}} = \mathcal{E}$,$H_{\mathcal{O}} \cup P_{\mathcal{O}} = \mathcal{O}$ and $\mathcal{E} \cup \mathcal{O} = I$.

Let V represent the set of feasible vertices in the lattice, i.e. a vertex occurs at each intersection of a horizontal and vertical line in the lattice. We assume that one of the points (e.g. the odd point closest to the middle) on the string is assigned to a particular lattice point, which defines the feasible region of vertices in the lattice. In other words, once this middle element is fixed, there are only a finite number of lattice points to which we can assign the other elements of the string. We classify the points in V as follows:

$V_{\mathcal{E}}$ is the set of even lattice points in V,
$V_{\mathcal{O}}$ is the set of odd lattice points in V.

Let $\delta(v)$ denote the set of feasible vertices adjacent to v, which, in 2D, consists of at most four lattice points. The set of feasible edges in the lattice is denoted by E, which is the set of (v, w) such that $v \in V_{\mathcal{O}}$ and $w \in V_{\mathcal{E}}, w \in \delta(v)$.

2.2 Variables for IP and LP Formulations

Now we define and explain the function of the variables that we use in our various integer programs. By convention, we always use i and v to refer to indices for odd elements on the string and odd lattice points, respectively. Similarly, we always use j and w to refer to indices for even elements on the string and even lattice points, respectively.

The variable $h_{(iv)(jw)}$ indicates whether or not there is a contact between hydrophobic elements i and j on edge (v, w). For example, if there is a contact between i and j across edge (v, w) then $h_{(iv)(jw)}$ is 1, and if there is no contact between i and j on edge (v, w), then $h_{(iv)(jw)}$ is 0.

The variable $h_{(v,w)}$ represents the total number of contacts formed across edge (v, w). In an integer solution, if there is a contact between $i \in H_{\mathcal{O}}$ and $j \in H_{\mathcal{E}}$ on edge (v, w), such that $i \neq j \pm 1$, then the value of $h_{(vw)}$ is 1. If there are no contacts across edge (v, w), then the value of $h_{(vw)}$ is 0. Note that there is a relationship hold between the variables $h_{(iv)(jw)}$ and $h_{(vw)}$:

$$h_{(v,w)} = \sum_{i \in H_{\mathcal{O}}} \sum_{j \in H_{\mathcal{E}}, j \neq i \pm 1} h_{(iv)(jw)}. \tag{2}$$

The variable x_{iv} indicates whether or not the element i is placed on vertex point v. In an integer solution, x_{iv} is set to 1 if element i is placed on lattice point v and 0 otherwise. Since the square lattice is bipartite, without loss of generality, we can assume that odd elements are placed only on odd lattice points and even elements are placed only on even lattice points. Thus, we distinguish between these two cases and create variables x_{iv} for the odd case and x_{jw} for the even case. Note that any string folding corresponds to a 0-1 assignment of the variables $\{x_{iv}, x_{jw}\}$. However, note that not every 0-1 assignment to the variables $\{x_{iv}, x_{jw}\}$ corresponds to a folding, which is why we need to further constrain these variables.

2.3 A Simple Integer Program

The integer program IP$_1$ is a simple formulation for the 2D folding problem. Lemma 1 states that there is a one-to-one correspondence between foldings and integer solutions to the following integer program.

IP$_1$:

$$\max \sum_{(v,w) \in E} \sum_{i \in H_{\mathcal{O}}} \sum_{j \in H_{\mathcal{E}}, j \neq i \pm 1} h_{(iv)(jw)}$$

$$\begin{array}{rcll}
\textit{subject to}: \quad \displaystyle\sum_{v \in V} x_{iv} & = & 1, \quad \forall i \in I & (3) \\
\displaystyle\sum_{i \in I} x_{iv} & \leq & 1, \quad \forall v \in V & (4) \\
\displaystyle\sum_{w \in \delta(v)} x_{i+1,w} & \geq & x_{iv}, \quad \forall i \in I \setminus \{n\}, v \in V & (5) \\
\displaystyle\sum_{j \in H_{\mathcal{E}}, j \neq i \pm 1} h_{(iv)(jw)} & \leq & x_{iv}, \quad \forall i \in H_{\mathcal{O}}, (v,w) \in E & (6) \\
\displaystyle\sum_{i \in H_{\mathcal{O}}, i \neq j \pm 1} h_{(iv)(jw)} & \leq & x_{jw}, \quad \forall j \in H_{\mathcal{E}}, (v,w) \in E & (7) \\
h_{(iv)(jw)},\ x_{iv},\ x_{jw} & \in & \{0,1\}, \quad \forall i \in H_{\mathcal{O}}, j \in H_{\mathcal{E}}, (v,w) \in E. & (8)
\end{array}$$

Lemma 1 *There is a one-to-one correspondence between foldings and integer solutions for IP$_1$.*

Proof: First, we show that every folding corresponds to an integer solution. In a valid folding, each element is placed on a unique lattice point and every lattice point has at most one element, so constraints (3) and (4) are satisfied. Consecutive elements on the string are placed on adjacent lattice points, so constraint (5) is satisfied.

Second, we show that an integer solution corresponds to a valid folding. For each element i, there is exactly one v such that $x_{iv} = 1$ (constraint (3)). Moreover, each lattice point v contains at most one element (constraint (4)). Constraint (5) guarantees that each consecutive element on the string is placed on an adjacent lattice point to its neighbor on the string. Thus, we have a valid folding. □

Constraints (6) and (7) require that there must be an even-1 and odd-1 on adjacent lattice points if there is a contact on that edge. Note that since only the $\{x_{iv}, x_{jw}\}$ variables are needed to describe a valid folding, these last two constraints are only used to limit the number of contacts given in the objective function. Constraint (8) enforces the integrality of all the variables. It is possible that we only need to force the x variables to be integer and this will automatically enforce the h variables to be integer.

2.4 Aggregate Constraints

We can obtain another integer program and its corresponding linear programming relaxation by replacing constraints (6) and (7) in IP_1 and LP_1 with the aggregate constraints (9) and (10).

$$\sum_{i \in H_{\mathcal{O}}} \sum_{j \in H_{\mathcal{E}}, j \neq i \pm 1} h_{(iv)(jw)} \leq \sum_{i \in H_{\mathcal{O}}} x_{iv} \quad \forall (v,w) \in E \tag{9}$$

$$\sum_{j \in H_{\mathcal{E}}} \sum_{i \in H_{\mathcal{O}}, i \neq j \pm 1} h_{(iv)(jw)} \leq \sum_{j \in H_{\mathcal{E}}} x_{jw} \quad \forall (v,w) \in E \tag{10}$$

We can use the variables $h_{(vw)}$ to simplify these constraints. Thus, if we use constraints (9) and (10) to replace constraints (6) and (7), then LP_1 would contain fewer h variables. Recall the definition of $h_{(vw)}$ from Equation (2).

2.5 A Linear Programming Relaxation

We obtain a linear programming relaxation by relaxing constraint (8) in IP_1 to the following:

$$0 \leq x_{iv}, x_{jw} \leq 1. \tag{11}$$

A linear programming formulation provides an upper bound on a maximum integral solution and can be solved much more efficiently than an integer program. One way to measure the quality of an integer program for a maximization problem is to determine the upper bound guaranteed by its linear relaxation. In general, the tighter (better) the bound provided by the linear relaxation, the higher the quality of the integer programming formulation.

There are other ways to formulate the problem as an integer program. For example, in IP_1, we could replace constraint (5) with constraint (12), which is shown below.

$$\sum_{w \in \delta(v)} x_{i-1,w} \geq x_{iv} \quad \forall i \in I \setminus \{1\}, v \in V. \tag{12}$$

This would also result in an integer programming formulation. Alternatively, we can include both constraints (12) and (5). Including both these constraints leads to a tighter linear program than including only one of these constraints. This is stated in Lemma 2, which is proved in the Appendix. We add constraint (12) to IP_1 and refer to its corresponding linear programming relaxation as LP_1.

Lemma 2 *Including both constraints* (5) *and constraint* (12) *results in a tighter linear program (i.e. can provide a better upper bound) than including only one constraint.*

Unfortunately, the relaxation discussed so far may not provide fractional solutions that are very close to integral solutions. As noted in Section 1, the upper bound on the number of contacts in a string S is $2 \cdot \min\{\mathcal{O}[S], \mathcal{E}[S]\} + 2$. These relaxations can yield a fractional answer that is twice as large as this upper bound. In this section, we show that the *integrality gap* of IP_1 is poor. The integrality gap of a relaxation is the worst case ratio of the fractional and integral optimal solution values over all possible non-negative cost functions. Lemma 3 is proved in the Appendix.

Lemma 3 *The objective value of LP_1 is each at least* $4 \cdot \min(\mathcal{O}[S], \mathcal{E}[S])(1 - \frac{1}{\sqrt{n}})$ *for any string S of length n.*

Thus, the integrality gap for both formulations is arbitrarily close to 4 since there are strings for which the optimal folding achieves only $(1 + o(1)) \min\{\mathcal{O}[S], \mathcal{E}[S]\}$ contacts [10].

3 Improved Linear Models

In order to obtain a linear programming relaxation that provides a tighter upper bound, we can add more constraints to strengthen our linear program. First, we want to find an example of when the current linear program provides a poor bound and then we can try to find additional constraints that addresses this weakness.

3.1 Additional Constraints

Figure 2 depicts a situation in which adding new constraints may help. In Figure 2, the variables $x_{iv}, x_{j+1,v}, x_{i+1,w}$ and x_{jw} each have value $\frac{1}{2}$. If $i, j+1 \in H_{\mathcal{O}}$ and $j, i+1 \in H_{\mathcal{E}}$, then $h_{(iv)(jw)}$ and $h_{(j+1,v)(i+1,w)}$ can each be assigned a value as high as $\frac{1}{2}$.

However, in an integral solution, if element i were placed on lattice point v and element $i+1$ were placed on lattice point w, then the edge (v, w) could not be used for any contacts since it is occupied by the actual string. Even in a fractional solution, the value of the contacts that occur across edge (v, w) should not be 1, since at least a fraction of the string is occupying the edge.

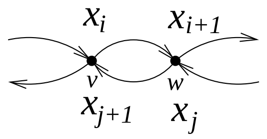
In order to make the optimal LP value closer to the optimal integer value of a folding, we add constraints that we refer to as *backbone* constraints. We use the following variables: the variable $E_{(iv)(i+1,w)}$ set to 1 means that element i is on lattice position v and element $i+1$ is on lattice element w. Since these variables are only for *consecutive* elements on the string, we can abbreviate them as follows:

$$E^{+}_{ivw} = E_{(iv)(i+1,w)}, \qquad\qquad E^{-}_{ivw} = E_{(iv)(i-1,w)}.$$

Then we can add the six sets of valid inequalities (13) and (14) to obtain a new linear program, which we refer to as LP_2.

LP$_2$:

$$
\begin{array}{rcll}
\max \displaystyle\sum_{(v,w)\in E} h_{(vw)} & & & \\
\textit{subject to}: \quad \displaystyle\sum_{v\in V_{\mathcal{O}}} x_{iv} & = & 1, \quad \forall i \in H_{\mathcal{O}} & \\
\displaystyle\sum_{v\in V_{\mathcal{E}}} x_{jw} & = & 1, \quad \forall j \in H_{\mathcal{E}} & \\
\displaystyle\sum_{i\in H_{\mathcal{O}}} x_{iv} & \leq & 1, \quad \forall v \in V_{\mathcal{O}} & \\
\displaystyle\sum_{j\in H_{\mathcal{E}}} x_{jw} & \leq & 1, \quad \forall w \in V_{\mathcal{E}} & \\
\displaystyle\sum_{w\in\delta(v)} E^{-}_{ivw} & = & x_{iv}, \quad \forall i \in H_{\mathcal{O}}, v \in V_{\mathcal{O}} & (13) \\
\displaystyle\sum_{w\in\delta(v)} E^{+}_{ivw} & = & x_{iv} \quad \forall i \in H_{\mathcal{O}}, v \in V_{\mathcal{O}} & \\
\displaystyle\sum_{v\in\delta(w)} E^{-}_{j+1,vw} & = & x_{jw}, \quad \forall j \in H_{\mathcal{E}}, w \in V_{\mathcal{E}} & \\
\displaystyle\sum_{v\in\delta(w)} E^{+}_{j-1,vw} & = & x_{jw}, \quad \forall j \in H_{\mathcal{E}}, w \in V_{\mathcal{E}} & \\
\displaystyle\sum_{i\in H_{\mathcal{O}}} E^{-}_{ivw} + \sum_{i\in H_{\mathcal{O}}} E^{+}_{ivw} + h_{(v,w)} & \leq & \displaystyle\sum_{i\in H_{\mathcal{O}}} x_{iv}, \quad \forall v \in V_{\mathcal{O}} & (14) \\
\displaystyle\sum_{j\in H_{\mathcal{E}}} E^{-}_{j+1,vw} + \sum_{j\in H_{\mathcal{E}}} E^{+}_{j-1,vw} + h_{(v,w)} & \leq & \displaystyle\sum_{j\in H_{\mathcal{E}}} x_{jw}, \quad \forall v \in V_{\mathcal{E}} & \\
0 & \leq & E^{\pm}_{ivw}, x_{iv}, x_{jw}, h_{(vw)} \ \leq 1, \quad \forall i \in H_{\mathcal{O}}, j \in H_{\mathcal{E}}, (v,w) \in E. &
\end{array}
$$



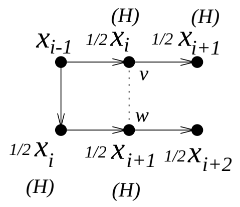


Figure 2: The figure on the left shows an example in which *backbone* constraints can be added to the linear programming formulation to give a better bound on an optimal folding. In this example, both element i and elements i and $j+1$ are each partially assigned to lattice point v, i.e. $x_{iv} = x_{j+1,v} = \frac{1}{2}$, and elements $i+1$ and j are each partially assigned to lattice point w, i.e. $x_{jw} = x_{i+1,w} = \frac{1}{2}$.

Figure 3: The figure on the right depicts a situation in which the variable $h_{(v,w)}$ can have value at least $\frac{1}{2}$ in LP$_2$. In LP$_3$, the contribution of edge (v,w) would be 0 since the variable $h_{(iv)(i+1,w)}$ is not defined, i.e. it is implicitly 0.

Note the connectivity constraints, (12) and (5), are missing from LP$_2$. As stated in Lemma 4, which is proved in the Appendix, these constraints are implied by constraints (13) and (14).

Lemma 4 *Backbone constraints imply the connectivity constraints, i.e. constraints* (13) *imply constraints* (12) *and* (5).

We can now show that LP$_2$ provides an upper bound that is at least as good as the upper bound (1) explained in Section 1. Lemma 5 is proved in the Appendix.

Lemma 5 *The optimal solution for LP$_2$ is at most* $2 \cdot \min\{\mathcal{O}[S], \mathcal{E}[S]\} + 2$.

LP$_2$ does *not* always give a solution whose objective value is at least $2 \cdot \min\{\mathcal{O}[S], \mathcal{E}[S]\} + 2$. It may give a solution whose objective value is strictly better. For example, if we consider the string of 20 consecutive 1's, from the simple combinatorial upper bound, we know an optimal folding can have no more than 21 contacts. However, an optimal folding can actually have no more than 14.5 contacts according to our AMPL implementation of LP$_2$. (See Section 5 for An alternate formulation for the linear program above would entail using the four index h variables $h_{(iv)(jw)}$ instead of the two index h variables $h_{(vw)}$.

$$\begin{aligned} E^-_{ivw} + E^+_{ivw} + \sum_{j \in H_\mathcal{E}} h_{(iv)(jw)} &\leq x_{iv} \quad \forall i \in H_\mathcal{O}, (v,w) \in E, \\ E^-_{j+1,vw} + E^+_{j-1,vw} + \sum_{i \in H_\mathcal{O}} h_{(iv)(jw)} &\leq x_{jw} \quad \forall j \in H_\mathcal{E}, (v,w) \in E. \end{aligned} \tag{15}$$

We can substitute constraints (15) for constraints (14). We refer to the resulting integer and linear program as IP$_3$ and LP$_3$, respectively.

Lemma 6 *Suppose the string S contains no consecutive 1's. Then the upper bound provided by LP$_3$ is no tighter than the upper bound provided by LP$_2$, i.e. substituting constraints* (15) *for constraints* (14) *does not lead to a tighter relaxation.*

If the string S contains consecutive 1's, then the proof of Lemma 6 does not go through. Furthermore, we can construct an example in which LP$_2$ and LP$_3$ have different objective values. Figure 3 gives an example in which LP$_2$ has a higher objective function than that of LP$_3$.
The only difference between LP$_2$ and LP$_3$ is that LP$_3$ does not allow "contacts" between adjacent elements on the string. Let $f(S)$ represent the number of pairs of consecutive 1's in S, e.g. the string $S = 01110$ has two pairs of consecutive 1's, so $f(S) = 2$. Then the relationship between the values of LP$_2$ and LP$_3$ for a string S is stated in Lemma 7, which is proved in the Appendix.

Lemma 7 *LP$_2$ - $f(S) \leq$ LP$_3$ $\leq$ LP$_2$ $\leq 2 \cdot \min\{\mathcal{O}[S], \mathcal{E}[S]\} + 2$.*

3.2 Integrality Gaps

We can show that the integrality gap for LP$_2$ and LP$_3$ is $2 - \epsilon$ for any $\epsilon > 0$. We use the string $S = \{0\}^q\{01\}^k\{0\}^{2q}\{1000\}^k\{0\}q$. We let k denote a positive integer and $q = 4k^2$. In [10], it is shown that no folding of S has more than $(1+o(1))\mathcal{O}[S]$ contacts. However, we can easily construct a fractional solution for LP$_2$ for which the objective function is $2\mathcal{O}[S]$.

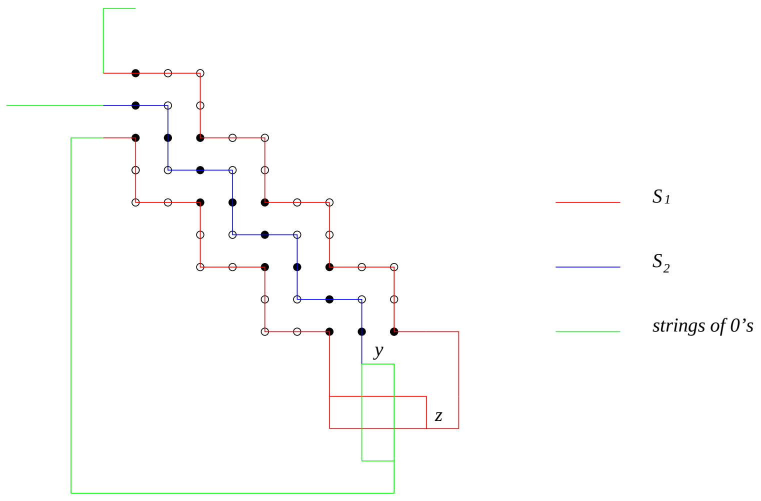


Figure 4: Let $S_1 = \{01\}^k$ and let $S_2 = \{0001\}^k$. The string splits in half at points y and z, which allows the string to cross itself, something not allowed in an integral solution.

4 Six Index Constraints

Another idea for strengthening the linear program is to add *six-index* constraints. One reason to use such constraints is that they would invalidate the solution given in Figure 4, thus strengthening the linear program. Suppose we let the variable $h_{(iv)(jw)(ku)}$ be a 1 if there is a contact between i and j on edge (v, w) and between j and k on edge (w, u). Then we can have the following constraint for collinear v, w, u. Recall that n denotes the length of the input string.

$$h_{(iv)(jw)(ku)} = 0 \quad \forall i, j, k : (\frac{|i-k|}{4})^2 \min\{n-j, j-1\}.$$

The idea behind this constraint is as follows: Suppose i and k are distance d apart on the string and both form a contact with j. Suppose i, j, k are placed on lattice points v, w, u, respectively, where v, w, u are collinear. Since the string cannot cross itself, the distance from j to the last point on the string n (or the nearest endpoint) must be less than distance $d^2/4$, since this is the maximum number of lattice points that can be in the region bounded by the substring connecting i to k.

We cannot simply add this constraint to LP_2 or LP$_3$ because we are not optimizing over the six-index h-variables or *double constraints*. However, this constraint might still be used to obtain information about the optimal folding of a string, because if a string has more than $\mathcal{O}[S]$ contacts, then it must have double contacts. We define a *double contact* as two contacts that are adjacent to each other, i.e. contacts formed on edges (v, w) and (w, u) where v, w, u are either collinear or form a right angle. In other words, if a folding has more than $\mathcal{O}[S]$ contacts, then some 1's must have more than 1 contact. Thus, we could add the following constraints to LP$_2$ for all adjacent v, w, u:

$$h_{(iv)(jw)(ku)} \leq h_{(iv)(jw)},$$
$$h_{(iv)(jw)(ku)} \leq h_{(jw)(ku)}.$$

And we could replace the objective function with the following:

$$\max \sum_{i,k \in H_{\mathcal{O}}} \sum_{j \in H_E} \sum_{adjacent\ v,w,u} h_{(iv)(jw)(ku)} + \sum_{i,k \in H_{\mathcal{E}}} \sum_{j \in H_O} \sum_{adjacent\ v,w,u} h_{(iv)(jw)(ku)}.$$

If the solution for LP$_2$ or LP$_2$ with this objective functions were 0, then we would know that an optimal folding contains only $\max\{\mathcal{O}[S], \mathcal{E}[S]\}$ contacts.

5 Experimental Results

In this section, we present experimental results for LP$_2$. We ran LP$_2$ on some of the benchmarks for the problem in the 2D HP model. These were taken from: www.cs.sandia.gov/tech_reports/compbio/tortilla-hp-benchmarks.html. We ran LP$_2$ on the following strings:

```
1.  hphpphhphpphphhpphph
2.  hhpphpphpphpphpphpphpphh
3.  pphpphhpppphhpppphhpppphh
4.  ppphhpphhppppphhhhhhhpphhpppphhpphpp
5.  pphpphhpphhppppphhhhhhhhhhpppppphhpphhpphpphhhhh
6.  hhhpphphphpphphphpph
```

String	length	upper bound	LP$_3$	Opt
1	20	11	10.67529996	9
2	24	11	11	9
3	25	8	8	8
4	36	16	14.89908257	14
5	48	25	24.88770748	22
6	20	11	10.76264643	10

6 Discussion

The challenge that we introduce here is to compute better upper bounds for the 2D folding problem using linear programming or otherwise. Our integer and linear programming models provide a promising direction for solving the 2D folding problem to optimality using branch-and-bound. However, because of the large size of the linear program (i.e. number of variables), we likely need tighter linear programming bounds to make these techniques practical. One way to address these scalability issues would be to use branch-and-bound on some special subset of the variables. For example, we have empirically observed that if we use branch-and-bound to enforce the integrality constraints on the odd variables, i.e. the $\{x_{iv}\}$ variables, then even variables, i.e. the $\{x_{jw}\}$ variables, are also integral in the resulting solution. Thus, we conjecture that if we have an optimal solution for LP$_2$ such that all the $\{x_{iv}\}$ variables are integral, then we can use this solution (e.g. round this solution) to obtain a fully integral solution with the same objective value. If this conjecture is true, we can restrict branching to the set of odd variables, thus cutting down the time necessary to compute an exact solution.

Another possible application of our integer and linear programming formulations is to find actual foldings that are better than those obtained in approximation algorithms but perhaps not provably optimal. Backofen has used exact methods from constraint logic programming to obtain compact conformations, i.e. solutions, for these folding problems [2]. If we can further constrain our integer programs to the solution space of compact foldings, then we may be able to reduce the time needed to find a solution.

7 Acknowledgments

This work was performed in part at Sandia National Laboratories. Sandia is a multipurpose laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000. This work was partially funded by the US Department of Energy's *Genomes to Life* program (www.doegenomestolife.org), under project "Carbon Sequestration in Synechococcus Sp.: From Molecular Machines to Hierarchical Modeling," (www.genomes-to-life.org).

We would like to acknowledge Harvey Greenberg, Cindy Phillips, Sorin Istrail, Jonathan Eckstein and Naomi Cameron for their helpful discussions on IP formulations for this problem.

References

[1] R. Agarwala, S. Batzoglou, V. Dancik, S. 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", *Journal of Computational Biology* (1997) Vol. 4(2):275-296.

[2] Rolf Backofen, "Optimization Techniques for the Protein Structure Prediction Problem", *Ph.D. Thesis, Ludwig-Maximilians-Universität München* (2000).

[3] Bonnie Berger and Tom Leighton, "Protein Folding in the Hydrophobic-Hydrophilic (HP) Model is NP-Complete", *Proceedings of the 2nd Conference on Computational Molecular Biology (RECOMB '98).*

[4] P. Crescenzi, D. Goldman, C. Papadimitiou, A. Piccolboni, and M. Yannakakis, "On the Complexity of Protein Folding", *Proceedings of the 2nd Conference on Computational Molecular Biology (RECOMB '98).*

[5] K. A. Dill, "Theory for the Folding and Stability of Globular Proteins", *Biochemistry* (1985) Vol. 24:1501.

[6] K. A. Dill, "Dominant Forces in Protein Folding, *Biochemistry* (1990) Vol. 29:7133-7155.

[7] H. J. Greenberg, W. E. Hart, and G. Lancia, "Opportunities for Combinatorial Optimization in Computational Biology", *INFORMS Journal of Computing, To appear.*

[8] William Hart and Sorin Istrail, "Fast Protein Folding in the Hydrophobic-Hydrophilic Model within Three-Eighths of Optimal", *Journal of Computational Biology* Vol. 3, No. 1, 1996:53-96.

[9] Giancarlo Mauri, Antonio Piccolboni, and Giulio Pavesi, "Approximation Algorithms for Protein Folding Prediction", *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA '99).*

[10] Alantha Newman, "A New Algorithm for Protein Folding in the HP Model", *Proceedings of SODA, 2002*, 876-884.

8 Appendix

Proof of Lemma 2: We show that constraint (12) does not imply constraint (5) or vice-versa. To do this we give a feasible LP solution for a string of length 9 such that constraint (5) is obeyed but constraint (12) is violated.

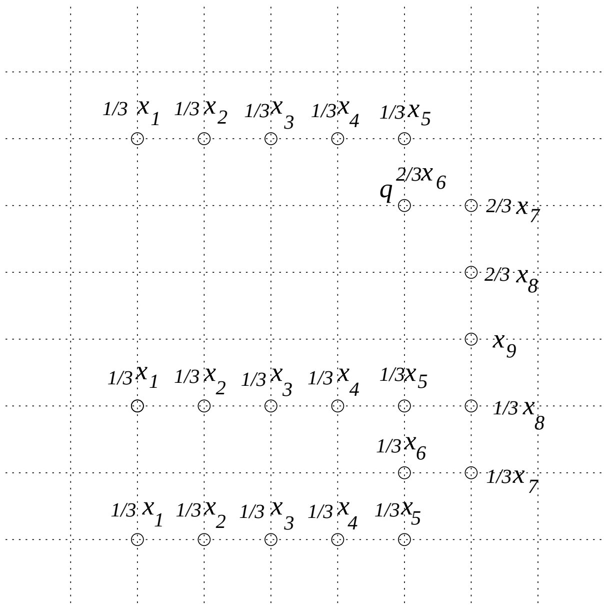


Figure 5: Constraint (12) is violated for $i = 6,\ v = q$. However, note that no other constraints (e.g. constraint (5)) are violated.

Such a feasible solution is shown in Figure 5. The values shown in Figure 5 are the fractions of each x_i that are placed at the labeled lattice points, i.e. the x_{iv} values. Let $i = 6,\ v = q$. Note that constraint (12) is violated for x_{6q} since $x_{6q} = 2/3$ and $\sum_{w \in \delta(q)} x_{5w} = 1/3$. Note that constraint (5) is not violated for any of the x_{iv} variables. We can repeat this argument for the string labeled in the reverse order and we would obtain an example in which constraint (12) is not violated but constraint (5) is violated. Thus neither constraint is implied by the other. □

Proof of Lemma 3: To show this, we give a solution for LP_1 that is valid for any string S and that has an objective value of $4 \cdot \min\{\mathcal{O}[S], \mathcal{E}[S]\}$. We let n represent the number of elements in S, i.e. the length of S. Without loss of generality, assume $\mathcal{O}[S] \leq \mathcal{E}[S]$ and let m be the number of lattice points, i.e. $|V_{\mathcal{O}}| = |V_{\mathcal{E}}| = \frac{m}{2}$. We also assume $n \leq m$, i.e. the string can actually be folded onto the lattice. We let $x_{iv} = \frac{2}{m}$ for all $i \in H_{\mathcal{O}}, v \in V_{\mathcal{O}}$ and $x_{jw} = \frac{2}{m}$ for all $j \in H_{\mathcal{E}}, w \in V_{\mathcal{E}}$. Then we let $h_{(iv)(jw)} = \frac{2}{(\mathcal{E}[S])m}$ for all $i \in H_{\mathcal{O}}, j \in H_{\mathcal{E}}, v \in V_{\mathcal{O}}, w \in V_{\mathcal{E}}$.

Note that constraint (3) is satisfied since for each i, there are $\frac{m}{2}$ possible $v \in V$ with the same parity. Constraint (4) will be satisfied because we have:

$$\sum_{i \in I} x_{iv} = \sum_{i \in H_{\mathcal{O}}} x_{iv} \leq \frac{m}{2} \cdot \frac{2}{m} \leq 1.$$

Constraints (12) and (5) will be satisfied as long as each lattice point v has at least one neighbor. Constraint (6) is satisfied since for $i \in H_{\mathcal{O}}$, we have $\frac{2}{(\mathcal{E}[S])m} \cdot \mathcal{O}[S] \leq \frac{2}{m}$ and for even i, we have $\frac{2}{(\mathcal{E}[S])m} \cdot \mathcal{E}[S] = \frac{2}{m}$. The number of $h_{(iv)(jw)}$ variables is $\mathcal{E}[S] \cdot \mathcal{O}[S] \cdot 4(\frac{m}{2})$; there are $\mathcal{E}[S] \cdot \mathcal{O}[S]$ pairs of 1's such that odd-1's are paired with even-1's. There are $\frac{m-\sqrt{m}}{2}$ odd lattice points each with 4 neighbors, i.e. the m lattice points form a convex region, so each odd lattice point, except those on the border, serves as an endpoint for 4 edges, so we have a total of $4(\frac{m-\sqrt{m}}{2})$ edges. Thus, the objective value is:

$$\max \sum_{i \in H_{\mathcal{O}}} \sum_{j \in H_{\mathcal{E}}} \sum_{v \in V_{\mathcal{O}}} \sum_{w \in \delta(v)} h_{(iv)(jw)} \leq (\mathcal{O}[S] - 2) \cdot \mathcal{E}[S] \cdot \frac{m - \sqrt{m}}{2} \cdot 4 \cdot \frac{2}{\mathcal{E}[S]m} = (4\mathcal{O}[S] - 8)(1 - \frac{1}{\sqrt{m}}).$$

Since $m \geq n$, this implies the lemma. So the value of the objective function is arbitrarily close to $4 \cdot \min\{\mathcal{O}[S], \mathcal{E}[S]\}$ for sufficiently large n, i.e. sufficiently long enough strings. □

Proof of Lemma 4: From the backbone constraints, we have:

$$x_{iv} = \sum_{w \in \delta(v)} E^{-}_{ivw}.$$

For each variable $x_{i-1,w}$, we also have:

$$x_{i-1,w} = \sum_{u \in \delta(w)} E^{+}_{i-1,wu}.$$

This last constraint implies that $x_{i-1,w} \geq E^{+}_{i-1,wv}$, since $v \in \delta(w)$. Note that $E^{+}_{i-1,wv} = E^{-}_{ivw}$. For each of terms in the first constraint in this proof, we can obtain the inequality $x_{i-1,w} \geq E^{-}_{ivw}$. Thus, we have the desired inequality:

$$x_{iv} \leq \sum_{w \in \delta(v)} x_{i-1,w}.$$

We can repeat this argument to derive constraint (5). $\square$

Proof of Lemma 5: The optimal solution for the linear program is $\sum_{(v,w)\in E} h_{(vw)}$. Without loss of generality, we assume $\mathcal{O}[S] \leq \mathcal{E}[S]$. Recall that constraint (14) is in the linear program. We rewrite this constraint as follows:

$$h_{(vw)} \leq \sum_{i \in H_{\mathcal{O}}} x_{iv} - \sum_{i \in H_{\mathcal{O}}} E^{-}_{ivw} - \sum_{i \in H_{\mathcal{O}}} E^{+}_{ivw}.$$

Summing over all the edges, we have:

$$\sum_{(v,w)\in E} h_{(vw)} \leq \sum_{(v,w)\in E} \sum_{i \in H_{\mathcal{O}}} x_{iv} - \sum_{(v,w)\in E} \sum_{i \in H_{\mathcal{O}}} E^{-}_{ivw} - \sum_{(v,w)\in E} \sum_{i \in H_{\mathcal{O}}} E^{+}_{ivw}.$$

The first sum is upper bounded by $4\mathcal{O}[S]$. To show this, first we note that:

$$\sum_{v \in V_{\mathcal{O}}} x_{iv} = 1.$$

If we sum over all edges, as opposed to all odd vertices, note that each odd vertex $v \in V_{\mathcal{O}}$ is an endpoint in at most 4 edges. Thus, we have:

$$\sum_{(v,w)\in E} x_{iv} = \sum_{v \in V_{\mathcal{O}}} \sum_{w \in \delta(v)} x_{iv} = \sum_{w \in \delta(v)} \sum_{v \in V_{\mathcal{O}}} x_{iv} = \sum_{w \in \delta(v)} 1 \leq 4,$$

$$\sum_{(v,w)\in E} \sum_{i \in H_{\mathcal{O}}} x_{iv} = \sum_{i \in H_{\mathcal{O}}} \sum_{(v,w)\in E} x_{iv} \leq \sum_{i \in H_{\mathcal{O}}} 4 = 4\mathcal{O}[S].$$

Now we analyze the following sum:

$$\sum_{(v,w)\in E} \sum_{i \in H_{\mathcal{O}}, i \neq 1} E^{-}_{ivw} = \sum_{i \in H_{\mathcal{O}}, i \neq 1} \sum_{(v,w)\in E} E^{-}_{ivw}.$$

Each variable E^{-}_{ivw} is associated with a unique odd vertex, i.e. the odd vertex v. We have the following constraints for each odd vertex:

$$\sum_{w \in \delta(v)} E^{-}_{ivw} = x_{iv} \quad \forall i \in H_{\mathcal{O}}, v \in V_{\mathcal{O}}.$$

Thus, we can rewrite the sum as follows:

$$\sum_{i\in H_{\mathcal{O}},i\neq 1}\sum_{(v,w)\in E} E^-_{ivw} = \sum_{i\in H_{\mathcal{O}},i\neq 1}\sum_{v\in V_{\mathcal{O}}}\sum_{w\in\delta(v)} E^-_{ivw} = \sum_{i\in H_{\mathcal{O}},i\neq 1}\sum_{v\in V_{\mathcal{O}}} x_{iv} = \sum_{i\in H_{\mathcal{O}},i\neq 1} 1 = \mathcal{O}[S]-1.$$

Note that:

$$\sum_{(v,w)\in E} E^-_{ivw} = \sum_{(v,w)\in E} E^+_{ivw}.$$

Thus,

$$\sum_{i\in H_{\mathcal{O}},i\neq 1}\sum_{(v,w)\in E} E^-_{ivw} = \sum_{i\in H_{\mathcal{O}},i\neq n}\sum_{(v,w)\in E} E^+_{ivw} = \mathcal{O}[S]-1.$$

Therefore, we have:

$$\sum_{(v,w)\in E} h_{(vw)} \leq 4\mathcal{O}[S] - (\mathcal{O}[S]-1) - (\mathcal{O}[S]-1) \leq 2\mathcal{O}[S]+2.$$

So the maximum value of the objective function is $2\cdot\min\{\mathcal{O}[S],\mathcal{E}[S]\}+2$. □

Proof of Lemma 6: We assume that there are no consecutive 1's in the string S. We show that given a set of $\{h_{(vw)}\}$ that satisfy all the constraints in LP$_2$ for the string S, we can find a set $h_{(iv)(jw)}$ such that $\{h_{(iv)(jw)}\}$ satisfy all the constraints in LP$_3$ for the string S. We define $h_{(vw)}$ as in Equation (2). From any solution for LP$_3$, we can obtain a solution for LP$_2$ with the same objective value. Conversely, we need to show that for any solution $\{h_{(vw)}\}$ for LP$_2$, we can find a solution $\{h_{(iv)(jw)}\}$ for LP$_3$ with the same objective value.

We use the variables f_{iv} and f_{jw}, which we define below:

$$\begin{aligned} f_{ivw} &= x_{iv} - E^-_{ivw} - E^+_{ivw}, \\ f_{jvw} &= x_{jw} - E^-_{j+1,vw} - E^+_{j-1,vw}. \end{aligned}$$

If we sum the f_{ivw} variables over all $i\in H_{\mathcal{O}}$ and the f_{jvw} variables over all $j\in H_{\mathcal{E}}$, then we have:

$$\begin{aligned} \sum_{i\in H_{\mathcal{O}}} f_{ivw} &= \sum_{i\in H_{\mathcal{O}}} (x_{iv} - E^-_{ivw} - E^+_{ivw}), \\ \sum_{j\in H_{\mathcal{E}}} f_{jvw} &= \sum_{j\in H_{\mathcal{E}}} (x_{jw} - E^-_{j+1,vw} - E^+_{j-1,vw}). \end{aligned}$$

Consider the following table for an arbitrary edge $(v,w)\in E$. Assume there are k i's in $H_{\mathcal{O}}$ labeled $i_1\ldots i_k$ and assume there are m j's in $H_{\mathcal{E}}$ labeled $j_1\ldots j_m$.

$$\begin{array}{lllllllll}
 & i: & i_1 & i_2 & i_3 & \ldots & i_k & & \\
j: & & & & & & & & \\
j_1 & & h_{(i_1v)(j_1w)} & h_{(i_2v)(j_1w)} & h_{(i_3v)(j_1w)} & \ldots & h_{(i_kv)(j_1w)} & \leq & f_{j_1vw} \\
j_2 & & h_{(i_1v)(j_2w)} & h_{(i_2v)(j_2w)} & h_{(i_3v)(j_2w)} & \ldots & h_{(i_kv)(j_2w)} & \leq & f_{j_2vw} \\
j_3 & & h_{(i_1v)(j_3w)} & h_{(i_2v)(j_3w)} & h_{(i_3v)(j_3w)} & \ldots & h_{(i_kv)(j_3w)} & \leq & f_{j_3vw} \\
. & & . & . & . & & . & & . \\
. & & . & . & . & & . & & . \\
. & & . & . & . & & . & & . \\
j_m & & h_{(i_1v)(j_mw)} & h_{(i_2v)(j_mw)} & h_{(i_3v)(j_mw)} & \ldots & h_{(i_kv)(j_mw)} & \leq & f_{j_mvw} \\
 & & \leq & \leq & \leq & \ldots & \leq & & \\
 & & f_{i_1vw} & f_{i_2vw} & f_{i_3vw} & \ldots & f_{i_kvw} & &
\end{array}$$

Note that if there are no consecutive 1's in S, then there will be no $h_{(iv)(jw)}$ variables in the above table in which $j = i + 1$ or $j = i - 1$. If there were such variables, then they must be assigned 0. But since there are no consecutive 1's in S, all the $h_{(iv)(jw)}$ variables in the table can be non-zero.

Without loss of generality, for some i, j, v, w, assume $\sum_{i \in H_{\mathcal{O}}} f_{ivw} \leq \sum_{j \in H_{\mathcal{E}}} f_{jvw}$. We want to distribute the value $h_{(v,w)}$ among the $h_{(iv)(jw)}$ variables. We can set the variable $h_{(i_1v)(j_1w)}$ to $\min\{f_{i_1vw}, f_{j_1vw}\}$. Then we can set the variable $h_{(i_1v)(j_2w)}$ to be as large as possible so that $h_{(i_1v)(j_1w)} + h_{(i_1v)(j_2w)} \leq f_{i_1v}$, etc.

We assign values to the $h_{(iv)(jw)}$ variables in the first column so that the sum of the variables in the first column is equal to f_{1vw}. We can do this by setting $h_{(i_1v)(j_2w)}$ to be as large as possible such that it is at most f_{2vw} and at most f_{1vw}. Then we set $h_{(i_1v)(j_3w)}$ to be as large as possible so that the sum of the three variables is no more than f_{i_1vw} and $h_{(i_1v)(j_3w)}$ is no greater than $f_{(j_3vw)}$. We repeat this for the rest of the $h_{(i_1v)(jw)}$ variables for $j \in H_{\mathcal{E}}$. When we are done, we have the following:

$$\sum_{j \in H_{\mathcal{E}}} h_{(i_1v)(jw)} = f_{i_1vw}.$$

Then we repeat for f_{i_2vw}, etc. Recall that the sum of the f_{ivw}'s is no more than the sum of the f_{jvw}'s. Thus, we can always find an assignment for the $h_{(iv)(jw)}$'s such that none of the constraints are violated. If for some f_{ivw}, we could not find a set of $h_{(iv)(jw)}$ variables to assign the value (because doing so would violate constraint (14)) then we would have a contradiction, since this would mean that the sum of the f_{jvw}'s is less than the sum of the f_{ivw}'s. $\square$

Proof of Lemma 7: The last inequality was proved in Lemma 5. The second inequality follows from the observation that given any solution for LP$_3$, we can obtain a solution for LP$_2$ with the same objective value just by setting the $h_{(vw)}$ variables as in Equation (2). The first inequality follows from Lemma 6 and the slight modification of it's proof in which we allow variables $h_{(iv)(jw)}$ for $j = i \pm 1$. Then we simply let all such variables equal 0, decreasing the objective function by at most $f(S)$. $\square$