Backtracking, formalization, and non-monotonicity in protein folding models [*]

Bjørn Kjos-Hanssen

February 1, 2024

Abstract

We show that the optimal energy in the HP protein folding model is not monotone under concatenation in the following lattices: 2D and 3D rectangular, triangular, and hexagonal. The computation uses recursive backtracking, a technique that we formally verify in the proof assistant Lean.

Furthermore, we demonstrate how to approximate optimal energy by restricting the set of allowed directions.

Contents

1 Introduction **2**

2 Nonmonotonicity **2**

2.1 2D rectangular lattice 3

2.2 3D rectangular lattice 3

2.3 The triangular lattice (hexagonal king's graph) 4

2.4 The hexagonal (brick wall) lattice 5

2.5 Four-dimensional grids and beyond 6

3 Triangular lattice approximation by restricted move sets **7**

4 Further results **9**

5 Formal verification **10**

[*]This work was partially supported by grants from the Simons Foundation (#704836 to Bjørn Kjos-Hanssen) and Decision Research Corporation (University of Hawai'i Foundation Account #129-4770-4). Undergraduates Sophia Kop, Tung Nguyen and Chien Hua Shu contributed to the Python coding on this project.

1 Introduction

The hydrophobic-polar (HP) protein folding model was proposed by Ken Dill in 1985 and uses four moves (left, right, up, down) in its 2D incarnation. We show that the three-move (left, right, down) restriction of the hydrophobic-polar protein folding model has a polynomial-time solvable optimization problem. The approach of viewing protein folding in terms of "moves" is analogous to the famous computer "snake game" and there exist several implementations [7].

The basic setup is that a protein is modeled as a sequence from the alphabet $\{H, P\}$. When two occurrences of H are next to each other in the lattice but not in the sequence, a point is achieved. An optimal folding is one that achieves the maximum number of points. In this article we identify $H = 0$ and $P = 1$. Fraenkel [4, 5] proved some results towards hydrophobic-polar protein folding being NP-hard. A definitive result was obtained in [3]. We show that the hydrophobic-polar protein folding model is non-monotone under concatenation in 2D, 3D, and hexagonal grids.

We give an algorithm (Theorem 8) that solves the three-move optimization problem in five minutes on realistic-length polypeptides of length 500. For example, we can use it on benchmark sequences as in Lesh, Mitzenmacher, and Whitesides (2003) [11]: S64, S85, S100a, S100b.

We show that a triangular lattice analogue of the three-move model gives an approximation algorithm for the triangular lattice optimization problem (Theorem 7). Such algorithms were known from work of Agarwala et al. [1], but our algorithm is "organic" in that it comes directly from a restricted-move version of the graph. That is, one can ask for approximation algorithms that only use restricted resources, thus showing that the restricted resource problem approximates the full problem. When A and B are maximization problems we can hope to approximate B by showing that A is efficiently optimizable and approximates B to a constant factor. In general, in mathematics we have object A of interest and seek to show that it is approximately equal to an object B that is more readily computed. It is an extra bonus if B is itself already of independent interest.

The idea of restricting the set of moves the way we do is natural from the point of view a folding as a map $f : [0, n-1]_{\mathbb{N}} \to G$ between metric spaces, where the metric on $\mathbb{N}$ is the usual one, and the metric on a graph G is the graph metric. A folding is then an injective morphism of metric spaces, i.e., it satisfies $d(f(x), f(y)) \le d(x, y)$. If G is a sublattice of a coordinatized graph such as $\mathbb{Z}^2$ we naturally consider directions for $f(i+1)$ given $f(i)$. For example, for $\mathbb{Z}^2$, $f(i+1) - f(i) \in \{(0, \pm 1), (\pm 1, 0)\}$. To simplify the setup it is then natural to restrict the set of moves to, for example, $\{(0, 1), (\pm 1, 0)\}$.

2 Nonmonotonicity

Definition 1. *Let $P_{2D}(x), P_{3D}(x), P_{\text{tri}}(x), P_{\text{hex}}(x)$ denote the maximum number of points achievable for a word x in the 2D rectangular, 3D rectangular,* *triangular, and hexagonal lattices, respectively.*

Stecher conjectured that the HP folding problem is "weakly monotone", that is, if we add a prefix or suffix to a given sequence, our point count should never decrease.

Conjecture 2 ([13])**.** *Let P be one of the function in Definition 1 and let x, y be binary words. Then $P(x) \leq P(xy)$ and $P(y) \leq P(xy)$.*

Theorem 3. *The value function is non-monotone in the following HP models, with an x violating monotonicity specified in each case.*

1. *2D rectangular lattice (Section 2.1): if $x = (01)^3 00$ or $(01)^3 10$ then $P_{2D}(x) = 3 > 2 = P_{2D}(x1)$.*
2. *3D rectangular lattice (Section 2.2): if $x = (01)^5 10$ or $(01)^5 00$ then $P_{3D}(x) = 5 > 4 = P_{3D}(x1)$.*
3. *Triangular lattice (Section 2.3): if $x = (01)^5 00$ then $P_{\text{tri}}(x) = 11 > 10 = P_{\text{tri}}(x1)$.*
4. *Hexagonal lattice (Section 2.4): if $x = 010^3 10110$ then $P_{\text{hex}}(x) = 2 > 1 = P_{\text{hex}}(x1)$.*

The calculations were done using recursive backtracking [12].

2.1 2D rectangular lattice

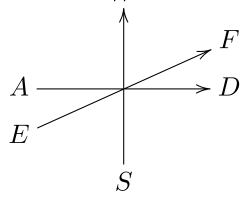
Let $P(w)$ be the number of points we can achieve for w. Then $P(1w) \leq P(w)$, and $P(1w) < P(w)$ can occur. We have $P(01010100) = 3 > 2 = P(010101001)$ using the folding in (1).

$$\begin{tikzcd}
 & 0 \arrow[r] & 1 \arrow[d] \\
0 \arrow[r] & 0 & 0 \arrow[d] \\
1 \arrow[u] & 0 \arrow[l] & 1 \arrow[l]
\end{tikzcd} \tag{1}$$

Our computations show that 01010100 and 01010110 are the only examples that refute monotonicity at length 8.

2.2 3D rectangular lattice

For the 3D rectangular lattice, the six permissible moves corresponding to rays from the origin in the standard basis for $\mathbb{R}^3$ may be labeled as shown in (2), inspired by QWERTY keyboard layout:



(2)

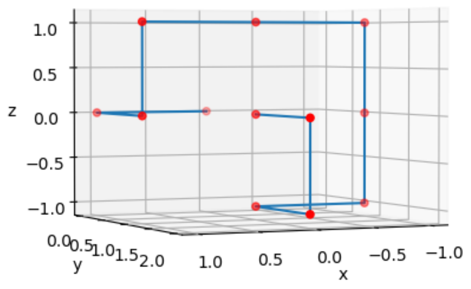


Figure 1: A witnessing folding for the inequality $P_{3D}((01)^5 10) \geq 5$.

Let x be the word $(01)^5 10$ and $y = x1$. Then we find $P(x) = 5$ and $P(y) = 4$. There are a total of five non-isomorphic sequences of moves for $P(x) = 5$: DWWAFSAEEDF, DWFAWEASEDF, DWFAWEESAFD, DWFAAEWDESF, and DWFAAEEDWFS. The latter sequence of moves is shown in Figure 1.

We have $P(x1) \leq P(x)$ for all x, since any folding of $x1$ can become a folding of x by removing the last 1, without any loss of points.

A remaining question is whether $P(x) \leq P(x0)$ for all x. The same obstacle seems to exist as shows that for some x, $P(x) > P(x1)$, but on the other hand, that extra 0 could be useful in other ways. We find that the same counterexample works: $(01)^5 100$ also has only $P = 4$. (Also, for 2D the same example as before works, $P((01)^3 10) = 3$ and $P((01)^3 100) = 2$.)

Thus, $P(x)$ and $P(x0)$ are actually incomparable in general. Note also that we have $P(x1) \leq P(x0)$ for all x. The situation is illustrated in (3).

$$\begin{tikzcd}
x \mapsto P(x) \arrow[rr, bend right=10, "\not\leq"'] & & x \mapsto P(x0) \arrow[ll, bend right=10, "\not\geq"'] \\
& x \mapsto P(x1) \arrow[ul, ">"] \arrow[ur, "<"'] &
\end{tikzcd} \tag{3}$$

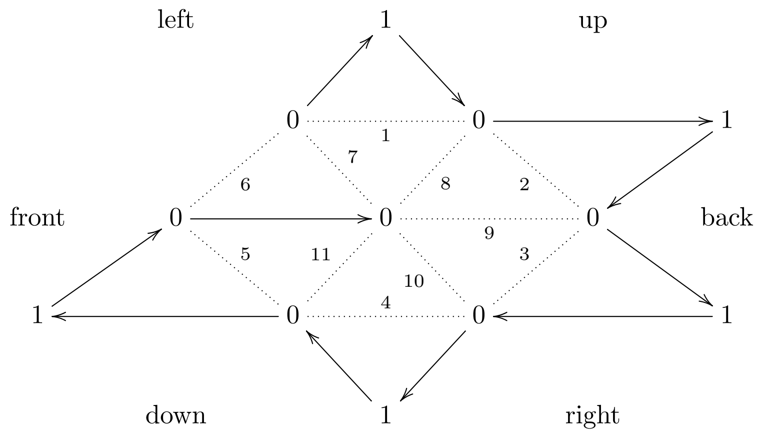
2.3 The triangular lattice (hexagonal king's graph)

The triangular lattice is also the hexagonal king's graph, i.e., the graph defined by moves of a king on a hexagonal chess board. A triangular lattice is formed by tiling the plane with equilateral triangles. The triangular lattice graph is then the graph whose vertices are the vertices of these triangles, and whose edges are the edges of these triangles. It can be realized as the set of points (4).

$$\{(n, m\sqrt{3}) \mid m, n \in \mathbb{Z}\} \cup \left\{ \left(n + \frac{1}{2}, m\sqrt{3} + \frac{\sqrt{3}}{2} \right) \middle| m, n \in \mathbb{Z} \right\}. \tag{4}$$

We refute monotonicity for triangular lattices. There is no example of x, y with $P(x1y) < P(xy)$, in other words, the 1 can always be "pushed out of the way", when $|xy| \leq 7$. However, $P_{\text{tri}}((01)^5 00) = 11$ (eleven), as witnessed by the folding in (5), and $P_{\text{tri}}((01)^5 001) = 10$ (ten), refuting monotonicity for triangular lattices. Ten points are achieved by the fold DWWZWZWZZDDX, among others.

The same word $(01)^5 00$ versus $(01)^5 001$ serves as a counterexample for both the hex and 3D models. What's more, the same witness can be used in both cases if we view a 3D coordinate system from a certain angle (5).

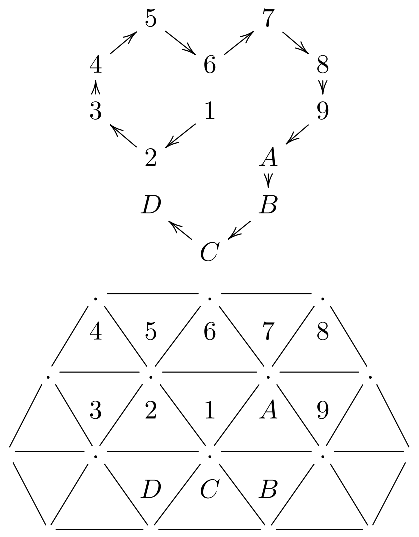


(5)

2.4 The hexagonal (brick wall) lattice

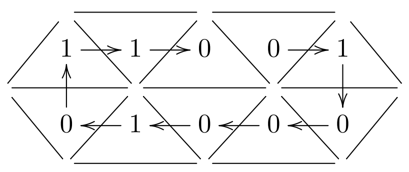
The hexagonal lattice may also suggestively be called the brick wall lattice; it also has a dual version composed of triangles. These three perspectives are illustrated in a demonstration of the fact that $P_{\text{hex}}(0^{13}) \geq 3$ in (6).

$$\begin{tikzcd}
4 \arrow[r] \arrow[d] & 5 \arrow[r] & 6 \arrow[r] \arrow[d, dash, dotted] & 7 \arrow[r] & 8 \arrow[d] \\
3 & 2 \arrow[l] \arrow[d, dash, dotted] & 1 \arrow[l] \arrow[r, dash, dotted] & A \arrow[d] & 9 \arrow[l] \\
 & D & C \arrow[l] & B \arrow[l] &
\end{tikzcd}$$



(6)

We obtain two points for $x = 010^3 10110$, see (7). Two points are not possible for $x1$, or even for $x0$.



(7)

$$\begin{tikzcd}
1 \arrow[r] & 1 \arrow[r] & 0 \arrow[r, dash, dotted] \arrow[d, dash, dotted] & 0 \arrow[r] & 1 \arrow[d] \\
0 \arrow[u] & 1 \arrow[l] & 0 \arrow[l] & 0 \arrow[l] & 0 \arrow[l]
\end{tikzcd}$$

In fact, the same example shows that for $x = 0111010100$, we have $P_{\text{hex}}(x) = 2$ and $P_{\text{hex}}(x000) = 1$. We expect a bound on k for which there exists an x with $P_{\text{hex}}(x0^k) < P_{\text{hex}}(x)$, though.

2.5 Four-dimensional grids and beyond

We are not able to refute monotonicity for 4D rectangular grids, although this is surely simply a matter of computational power. On the other hand, for infinite-dimensional rectangular grids, monotonicity does hold, since we can always pass to a higher dimension to fit the next bit.

The power of 4D does not become evident until length 13:

Theorem 4. *We have $P_{4D}(0^{13}) > P_{3D}(0^{13})$. For each word x of length 12 or less, $P_{4D}(x) = P_{3D}(x)$.*

Proof. In (8), the first 8 zeros are fold three-dimensionally. In order to utilize the last zero efficiently and obtain maximum points, it is necessary to use coordinates of the form $xy11$.

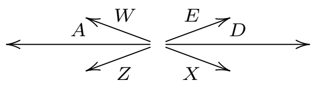
$$\begin{tikzcd}
 & 0100 \arrow[dl] & & 1100 \arrow[ll] \\
0110 \arrow[rr] & & 1110 \arrow[dd] & \\
 & 0000 \arrow[rr] & & 1000 \arrow[uu] \\
0010 \arrow[ddd, squiggly, bend right] & & 1010 \arrow[ll] & \\
0111 \arrow[rr] & & 1111 \arrow[dd] & \\
 & & & 1001 \\
0011 \arrow[uu] & & 1011 \arrow[ur] &
\end{tikzcd} \tag{8}$$

□

3 Triangular lattice approximation by restricted move sets

Definition 5. *A problem is in APX if there is $c > 1$ and an optimization algorithm that on an instance i yields a number $\mathrm{ALG}(i) \le \mathrm{OPT}(i)$ with $\mathrm{OPT}(i) \le c\,\mathrm{ALG}(i)$, where $\mathrm{OPT}(i)$ is the optimal solution. Following Arora and Barak ([2, Definition 11.1], although their definition is specialized to the SAT problem) we say that for $\rho \le 1$, a ρ-approximation algorithm is one that finds a solution which scores at least $\rho\,\mathrm{OPT}(i)$ points.*

The standard move set for the triangular "hexagonal king's graph" lattice is W, E, D, X, Z, A. Motivated by QWERTY keyboard layout, these directions can be written as in (9).



(9)

We demonstrate an approximation algorithm utilizing four of these, excluding two that are neighbors. For definiteness, let us exclude W and E.

Theorem 6. *For any word x with $x(i) = x(j) = 0$, $i + 1 < j$, there is a folding of x in the triangular lattice which achieves a point for (i, j).*

Proof. If $i - j = 2k + 2$ is even, we use the sequence of moves $E^{k+1}XZ^k$. If $i - j = 2k + 3$ is odd, we use the sequence of moves $E^{k+1}DZ^{k+1}$. □

Jiang and Zhu (2005) [6] give a 1/6-approximation algorithm for the hexagonal (brick wall) lattice. Agarwala et al. in 1997 [1] gave approximation algorithms for the triangular lattice. Here we give another approximation algorithm, of a special and simple kind using restricted move sets.

Theorem 7. *The triangular lattice protein folding optimization problem is in APX and has a* $\frac{1}{25}$*-approximation algorithm which is realized by folding with a restricted move set* $\{A, Z, D, X\}$.

Proof. Let w be a binary word, let $Z(w)$ be the number of zeros of w, and let $P(w)$ be the number of points achievable in the triangular lattice. Clearly $P(w) \le 5Z(w)$ since as we go through w bit by bit, each bit adds at most 5 points to the total: each vertex has 6 neighbors, and one of them was used by the preceding bit in w.

On the other hand, we have

$$\lfloor (Z(w) - 1)/2 \rfloor \le P(w)$$

(the minus 1 since the first 0 does not immediately contribute a point) using the strategy of Theorem 6: if w is $0^{n_1}1^{m_1}0^{n_2}1^{m_2}\dots$ we use D $n_1/2$ times followed by X or Z depending on parity, followed by A $n_1/2$ times. We use alternatingly the left and the right side of the 0.

The only time we do not get a point out of a 0 is when it is immediately followed by another 0. That is, 0s following 0s are treated as 1s in the algorithm layout and written as $\underline{0}$.

Thus, no zeros are underlined, except those that immediately follow a non-underlined zero in the sequence. For the word $1^6 0 1^2 0^2 1^2 0^3 1^6 00$ we thus rewrite it as $1^6 0 1^2 0\underline{0} 1^2 0\underline{0}0 1^6 0\underline{0}$ and fold so that the non-underlined zeros form a diagonal line (10).

$$\begin{tikzcd}
1 \arrow[rr] & & 1 \arrow[rr] & & 1 \arrow[rr] & & 1 \arrow[rr] & & 1 \arrow[rr] & & \textcircled{0} \arrow[rr] & & 1 \arrow[dl] \\
& & & & & 1 \arrow[dr] & & \underline{0} \arrow[ll] & & \textcircled{0} \arrow[ll] & & 1 \arrow[ll] & \\
& & & & & & 1 \arrow[rr] & & \textcircled{0} \arrow[dr] & & & & \\
& 1 \arrow[dl] & & 1 \arrow[ll] & & 1 \arrow[ll] & & \textcircled{0} \arrow[ll] & & \underline{0} \arrow[ll] & & & \\
1 \arrow[rr] & & 1 \arrow[rr] & & 1 \arrow[rr] & & \textcircled{0} \arrow[rr] & & \underline{0} & & & &
\end{tikzcd} \tag{10}$$

Our approximation algorithm is thus:

1. If

   (a) (i) w has at most one zero, or

   (b) (ii) has exactly two zeros, and they are right next to each other in w,

   output any folding of w (say, laying w out from left to right).

2. If w has exactly two zeros, and they are not right next to each other in w, output a folding which achieves 1 point as in Theorem 6.

3. If w has three or more zeros, output a folding that achieves $\lfloor (Z(w)-1)/2 \rfloor$ points.

Since $Z(w)/5 \leq \lfloor (Z(w)-1)/2 \rfloor$ and $P(w) \leq 5Z(w)$, this is a $\frac{1}{25}$-approximation algorithm.

$\square$

Theorem 8. *2D rectangular HP protein folding with movements restricted to $\{A, S, D\}$ has an efficiently solvable optimization problem.*

Proof. We use dynamic programming (memoization): we only need to consider the lengths `ult` and `pen` of the current horizontal line and the one above it. See [8] for an implementation. $\square$

4 Further results

Optimal Lipschitz constants. For the 2D rectangular lattice we have $P(x0) \leq P(x) + 3$. This is optimal, as $x = 0101010$ gives $P(x0) = 3$ and $P(x) = 0$. The moves made are DSSAAWD.

For the triangular lattice we have $P(x0) \leq P(x) + 2$. This is optimal as $x = 000001111$ gives $P(x0) = 2$, $P(x) = 0$. The moves made are DDSASAASD.

For the hexagonal and 3D rectangular lattices we have $P(x0) \leq P(x) + 5$.

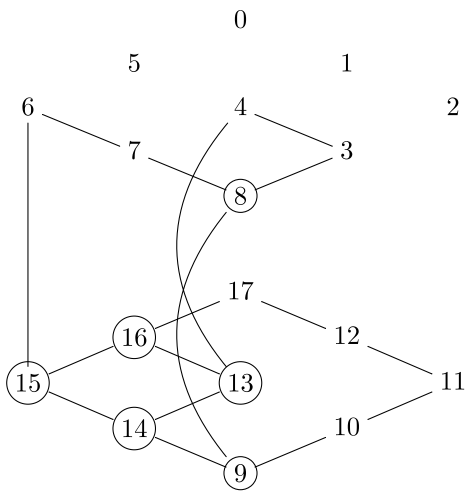
Question 9. *Is there an x with $P(x0) = P(x) + 5$ for $P \in \{P_{\text{hex}}, P_{3D}\}$?*

Lipschitz constants on the negative side. In the triangular lattice we find an example with $P(x) = P(x0) = 3$ but $P(x1) = 1$: $x = 0000011011100$ with $|x| = 13$. The moves made are DSAAAASASDDSD.

Question 10. *Is $P_{\text{tri}}(x) - P_{\text{tri}}(x1)$ unbounded as x varies?*

Non-planarity and outerplanarity. Knottedness is of interest for protein folding. Here we consider the related concept of planarity. All folds in the hexagonal, triangular, and 2D rectangular lattices are embedded in the plane and hence planar. Not surprisingly, this does not persist when we move to the 3D rectangular lattice. Recall that by Kuratowski's theorem, a graph is planar if and only if it does not have a minor isomorphic to $K_{3,3}$ or K_5. Here, $K_{m,n}$ is the complete bipartite graph on m and n vertices, and K_n is the complete graph on n vertices.

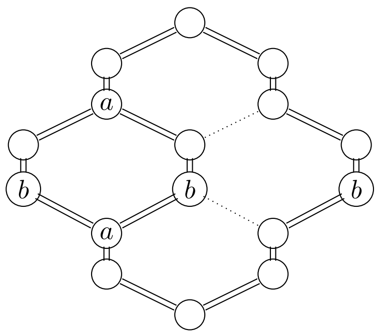
The word 0^{18} achieves 16 points using the fold indicated by the numbers $0, 1, \ldots, 17$ below. The graph whose edges are $(i, i+1)$ and all contacts is non-planar. It has a copy of $K_{3,3}$ as indicated, with vertices $8, 9, 13, 14, 15, 16$ where each of $8, 14, 16$ is connected to each of $9, 13, 15$.



(11)

Outerplanar graphs may be characterized by the two forbidden minors K_4 and $K_{2,3}$. The bottom layer of (11) shows that protein folding in the 2D rectangular lattice is not outerplanar. The vertices $\{9, 13, 14, 15, 16\}$ form a $K_{2,3}$ since all elements of $\{9, 13, 15\}$ are connected to both elements of $\{14, 16\}$.

The hexagonal lattice is also not outerplanar, as it has the $K_{2,3}$ minor shown in (12).



(12)

5 Formal verification

We verify some of our results in the proof assistant Lean 4 ([9]). The algorithmic technique of backtracking is formalized as follows. Since the number of words having a property `P` with a given suffix `w` is most easily evaluated when `w` is of full length, we need the datatype `Gap`.

```
def Gap (b L k : ℕ) : Type := Vector (Fin b) (L - k)

def Gap_cons {b n L:ℕ} (a:Fin b) (w : Gap b L.succ n.succ)
                    (h: ¬ n ≥ L.succ) : Gap b L.succ n
  := ⟨a :: w.1, by {rw [List.length_cons];simp;exact (Nat.succ_sub
    (Nat.not_lt.mp h)).symm}⟩
```

```

def Gap_nil {k b L : ℕ} (h : k ≥ L.succ) : Gap b L.succ k
  := ⟨List.nil, by {rw [Nat.sub_eq_zero_of_le h];rfl}⟩
```

Mathematically, it would perhaps be even more natural to consider the number of words satisfying `P` with a given *prefix*, but the constructor of `List` in Lean adds symbols on the left, which makes using suffixes slightly more convenient.

Next `num_by_backtracking` calculates the number of words (of `List` type) that satisfy the conjunction of the predicates `P` and `Q`, where the monotonicity of `P` is used to shortcut futile searches.

```
def num_by_backtracking {k b L:ℕ}
  (P: List (Fin b) → Prop) [DecidablePred P]
  (Q: List (Fin b) → Prop) [DecidablePred Q]
  (w : Gap b L.succ k) : ℕ :=
by {
  induction k
  exact ((ite (P w.1 ∧ Q w.1) 1 0))   /- Base case -/
  exact  /- Inductive case -/
    (ite (P w.1)) (dite (n ≥ L.succ)
      (fun h ↦                              n_ih (Gap_nil        h) )
      (fun h ↦ List.sum (List.ofFn (fun a ↦ (n_ih (Gap_cons a w h)))))
    ) 0
}
```

The function `num_by_backtracking` is recognized as decidable by Lean, so that we can use it for computations. To use the results in theorems we want to specify exactly what the function does, and under what conditions. For instance, if `P` is not monotone then `num_by_backtracking` is useless. Thus, we use the following type.

```
structure MonoPred (b:ℕ) where
  P : List (Fin b) → Prop
  preserved_under_suffixes (u v : List (Fin b)): u <:+ v → P v → P u
  Q (l: List (Fin b)) : Prop := True
```

Note that `Q` is given a default value which can be overridden as desired.

In `backtracking_verification` we characterize what `num_by_backtracking` does when `P` is monotone. The current proof is several hundred lines of code. The parameter `b` is the branching rate of a tree.

```
theorem backtracking_verification {k b L:ℕ}
  (bound : k ≤ L.succ) (M:MonoPred b)
  [DecidablePred M.P] [DecidablePred M.Q]
  (w : Vector (Fin b) (L.succ-k)):
  Fintype.card {
    v : Vector (Fin b) L.succ // (M.P v.1 ∧ M.Q v.1) ∧ w.1 <:+ v.1
  }  = num_by_backtracking M.P M.Q w
```

Using a variation on `num_by_backtracking` in [10] that returns the set of witnesses rather than their number, we confirm formally our refutation of Stecher's conjecture from Section 2.1.

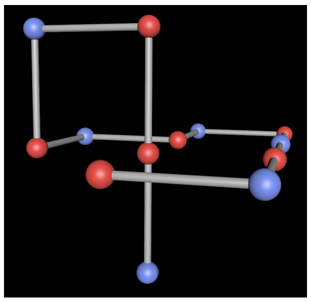


Figure 2: A slipknotted optimal witness for the length-13 word 1001010101010 using the move sequence FFAEWDWDSSSA. This image can be reconstructed using the URL `https://math.hawaii.edu/~bjoern/?sheets=9&xs=29&ys=9&string=1001010101010&page=labbyfold&moves=ffaewdwdsssa`.

Future work. For $(01)^5001$ in the 3D rectangular we obtain a "slipknotted" witness for 4 points in Figure 2. It will be interesting to see the extent of knotted behavior in these models.

References

[1] Richa Agarwala, Serafim Batzoglou, Vlado Dančík, Scott E. Decatur, Martin Farach, Sridhar Hannenhalli, S. Muthukrishnan, and Steven Skiena. Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model. In *Proceedings of the First Annual International Conference on Computational Molecular Biology*, RECOMB '97, page 1–2, New York, NY, USA, 1997. Association for Computing Machinery.

[2] S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2006.

[3] Pierluigi Crescenzi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the complexity of protein folding. *Journal of Computational Biology : a journal of computational molecular cell biology*, 5:423–65, 02 1998.

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

[5] Aviezri S. Fraenkel. Protein folding, spin glass and computational complexity. In *DNA Based Computers*, 1997.

[6] Minghui Jiang and Binhai Zhu. Protein folding on the hexagonal lattice in the Hp model. *J. Bioinform. Comput. Biol.*, 3(1):19–34, 2005.

[7] Bjørn Kjos-Hanssen. HP protein folding games. `https://math.hawaii.edu/~bjoern/`. Accessed: 2023-11-03.

[8] Bjørn Kjos-Hanssen. Python code for "Nonmonotonicity of the value function in HP protein folding models". `https://github.com/bjoernkjoshanssen/protein`. Accessed: 2023-11-03.

[9] Bjørn Kjos-Hanssen. Formalized backtracking. `https://github.com/bjoernkjoshanssen/bay/BacktrackingVerification.lean`.

[10] Bjørn Kjos-Hanssen. Stecher's conjecture. `https://github.com/bjoernkjoshanssen/bay/StecherConjecture.lean`.

[11] N. Lesh, M. Mitzenmacher, and S. Whitesides. A complete and effective move set for simplified protein folding. In *7th Annual International Conference on Research in Computational Molecular Biology (RECOMB) 2003*, pages 188–195. ACM Press, 2003.

[12] Chien Hua Shu. ProteinFolding. `https://github.com/shu4dev/ProteinFolding`. Accessed: 2023-11-03.

[13] Jack Stecher. personal communication.