PAPER

Bicolored Path Embedding Problems Inspired by Protein Folding Models*

Tianfeng FENG[†a)], ***Nonmember***, **Ryuhei UEHARA**[†b)], ***and*** **Giovanni VIGLIETTA**[†c)], ***Members***

SUMMARY In this paper, we introduce a path embedding problem inspired by the well-known hydrophobic-polar (HP) model of protein folding. A graph is said *bicolored* if each vertex is assigned a label in the set {red, blue}. For a given bicolored path P and a given bicolored graph G, our problem asks whether we can embed P into G in such a way as to match the colors of the vertices. In our model, G represents a protein's "blueprint," and P is an amino acid sequence that has to be folded to form (part of) G. We first show that the bicolored path embedding problem is NP-complete even if G is a rectangular grid (a typical scenario in protein folding models) and P and G have the same number of vertices. By contrast, we prove that the problem becomes tractable if the height of the rectangular grid G is constant, even if the length of P is independent of G. Our proof is constructive: we give a polynomial-time algorithm that computes an embedding (or reports that no embedding exists), which implies that the problem is in XP when parameterized according to the height of G. Additionally, we show that the problem of embedding P into a rectangular grid G in such a way as to maximize the number of *red-red contacts* is NP-hard. (This problem is directly inspired by the HP model of protein folding; it was previously known to be NP-hard if G is not given, and P can be embedded in any way on a grid.) Finally, we show that, given a bicolored graph G, the problem of constructing a path P that embeds in G maximizing red-red contacts is Poly-APX-hard.

key words: *bicolored grid graph, embedding problem, HP (hydrophobic-polar) model, Hamiltonian path problem, protein folding problem*

1. Introduction

1.1 Background and Previous Work

The protein folding problem asks how a protein's amino acid sequence dictates its three-dimensional atomic structure. This problem has wide applications and a long history dating back to the 1960s [8]. From the viewpoint of theoretical computer science, there is ongoing research aiming at revealing insights into reality by working on simplified abstract models.

One of the most popular such models is the *hydrophobic-polar* (HP) model [5]–[7], [10], [13], [16], [19], [22]. A protein in the HP model is represented as an abstract open chain, where each link has unit length and each joint is marked either H (hydrophobic, i.e., non-polar) or P (hydrophilic, i.e., polar). A protein is usually envisioned as a path embedded in a grid within the 2D or 3D lattice, where each joint in the chain maps to a point on the lattice, and each link maps to a single edge. The HP model of energy specifies that a chain desires to maximize the number of *H-H contacts*, which are pairs of H nodes that are adjacent on the lattice but not adjacent along the chain. The *optimal folding problem* in the HP model asks to find an embedding of a sequence of Hs and Ps on the 2D square lattice that maximizes the number of H-H contacts.

Previous results on the HP model mostly concern the 2D square lattice, where commonly used techniques can be criticized for relying on the properties of parity in the lattice. For example, [15], [20], [23] provide several approximation algorithms, all of which bound the maximum number of H-H contacts in terms of the number of odd-parity H nodes and the number of even-parity H nodes in the chain. This is because two H nodes can be embedded in adjacent nodes on the square lattice only if their distance along the chain is odd, i.e., if they have opposite parity. Such observations make sense in the discrete setting, but have no obvious meaning in the real protein folding problem that the theory aims to model. Thus, parity-related arguments should not be taken as the *only* reason as to why an amino acid chain can or cannot fold in a certain way.

Solving the optimal folding problem in a square lattice is shown to be NP-hard in [4]. Although the proof does not suffer from the aforementioned parity-related issues, it constructs a hard-to-fold chain whose properties heavily rely on Hadamard codes and Hamming distances. Again, appealing to these inherently discrete concepts is a departure from the continuous nature of real protein folding.

A final point of criticism to the traditional HP model is that the number of H-H contacts is not the only possible measure that may be used to capture the intricate physical and chemical laws that describe how a real protein folds.

In mathematics and theoretical computer science, there is a vast literature on embeddings of paths and graphs. The general problem of embedding an unlabeled path into a given graph is well-known to be NP-hard [4]. However, the graph embedding problem is efficiently solvable in some special cases. Notably, there is a linear-time algorithm for embedding graphs of constant size into planar graphs [11]. Unfortunately, these results say nothing about *labeled* graphs, which are the focus of the research on protein folding.

Restricting our attention to labeled graphs, there are

Manuscript received October 1, 2021.
Manuscript revised October 25, 2021.
Manuscript publicized December 7, 2021.

[†]The authors are with School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Nomi-shi, 923–1292 Japan.

[*]A preliminary version of this paper appeared in EuroCG 2021.

a) E-mail: ftflluy@jaist.ac.jp (Corresponding author)
b) E-mail: uehara@jaist.ac.jp
c) E-mail: johnny@jaist.ac.jp

DOI: 10.1587/transinf.2021EDP7206

substantially fewer works. To the best of our knowledge, the literature in this field is limited to exponential-time algorithms for the general embedding problem, e.g., [3], [18].

1.2 Our Approach and Motivation

Our critique of the standard HP model has inspired us to formulate the *bicolored path embedding problem*, which, apart from being an interesting graph embedding problem in its own right, may be seen as a new variant of the protein folding problem within the HP model.

In our model, we combine the basic ideas of protein folding with the complementary problem of *protein design*, where the goal is to synthesize a protein of a given shape (and function) from an amino acid sequence. Thus, we provide the "blueprint" of the folded shape of a protein, in the form of an input (grid) graph G with colors assigned to its vertices, and we ask if a given colored path P can be (injectively) embedded in G in such a way that vertex colors match. In other terms, we are effectively asking whether a given amino acid sequence can fold into (part of) a protein with prescribed structure. Since the HP model has nodes of only two types, we assume both G and P to be *bicolored*, say, with colors "red" and "blue." †

The significance of our model is that it more accurately captures some of the crucial and practical problems of protein folding. These problems do not only concern the way a given amino acid chain folds spontaneously, but also involve the design of proteins with desired attributes and shapes. Additionally, in our analysis we strive to avoid any argument that seems too closely related to the arbitrary choices we made when designing our model (for example, none of our proofs relies on the fact that a grid is 2-vertex-colorable, unlike some previous works [15], [20], [23]).

The underlying idea of our research is that, in biological processes, nature "solves" some computational problems related to protein folding in a seemingly efficient way. In order to explain these phenomena, the approach of theoretical computer science is to formulate abstract models of proteins and amino acid chains and study the computational complexity of protein folding problems under these models.

Most of our results (with one notable exception) indicate that several protein folding problems are computationally intractable in our model. In general, knowing that a problem is computationally hard (e.g., NP-hard) should discourage us from seeking an efficient solution. However, in the context of protein folding, a proof of NP-hardness can also provide insights on the deeper reasons why nature works in a certain way. For example, the fact that protein folding is computationally hard in a given model might be evidence that the model is incorrect and should be modified. Nonetheless, it could also mean that the model is accurate, but the instances of the problem that have been proved to be hard never occur in practice. In this case, a hardness result sheds some light on which patterns and configurations are naturally avoided in biological systems, and why.

†With regard to bicolored and monochromatic graphs, we do *not* adhere to established terminology from classical graph coloring theory; for the purposes of this paper, a coloring of a graph is simply a labeling of its vertices, with no extra constraints. In particular, adjacent vertices may have the same color.

1.3 Paper Organization and Results

The paper is organized as follows. In Sect. 2, we give a formal definition of the bicolored path embedding problem and we introduce some preliminary results, observing that the problem is NP-complete in several restricted cases.

In Sect. 3, we consider the case where G is a rectangular grid, which is the standard assumption in the HP model. We prove that the bicolored path embedding problem is NP-complete even if G and P have the same order (i.e., number of vertices). Next, we contrast this hardness result with a polynomial-time algorithm for the case where G is a grid of fixed height; thus, our embedding problem, parameterized according to the height of G, is in XP.

In Sect. 4, we show that maximizing red-red contacts in the bicolored path embedding problem (defined in the same way as H-H contacts in the HP model) is also NP-hard, even if G is a rectangular grid. We remark that, in previous work, it has been established that the problem of maximizing H-H contacts is NP-hard when G is not given, and P can be embedded in any way on a grid [4]. We also prove a complementary result: the problem of constructing a path P that embeds in a given bicolored graph G maximizing red-red contacts is Poly-APX-hard. In particular, it has no polynomial-time approximation algorithm with a sub-polynomial approximation ratio, unless P = NP.

Finally, in Sect. 5 we conclude the paper with some directions for future research.

A preliminary version of this paper appeared in the 37th European Workshop on Computational Geometry (EuroCG 2021) [12]. This extended version contains missing proofs, a revised and improved Sect. 3.2, and the new Sect. 4.2.

2. Definitions and Preliminaries

In this paper, a graph is said "bicolored" if each of its vertices is assigned one of two possible colors, e.g., red or blue:

Definition 1. *A* bicolored graph *is a labeled undirected graph* $G = (V, E, \omega)$*, where* $\omega\colon V \to \{red, blue\}$.

If the image of ω is {blue}, then G is *monochromatic*. A *bicolored path* is a bicolored graph with the topology of a path, i.e., such that $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{\{v_i, v_{i+1}\} \mid 1 \le i < n\}$.

If $P = (V, E, \omega)$ is a bicolored path with $V = \{v_1, v_2, \ldots, v_n\}$ and $G = (V', E', \omega')$ is a bicolored graph, we say that a function $f\colon V \to V'$ is an *embedding* of P into G if:

- f is injective, i.e., $i \neq j \implies f(v_i) \neq f(v_j)$.
- f maps edges of P to edges of G, i.e., $\{f(v_i), f(v_{i+1})\} \in$

E' for all $1 \le i < n$.
- f respects colors, i.e., $\omega(v) = \omega'(f(v))$ for all $v \in V$.

Definition 2. *The* bicolored path embedding problem *asks whether a given bicolored path P has an embedding into a given bicolored graph G.*

In the context of the bicolored path embedding problem, the graph G is called the *blueprint*.

Our first observation is that the NP-complete *Hamiltonian path* problem (i.e., given a graph, decide if there is a walk that visits every vertex exactly once [14]) is a special case of our bicolored path embedding problem. Namely, if both P and G are monochromatic and have the same order, then an embedding of P into G is precisely a Hamiltonian path in G.

It follows that the problem of finding a bijective embedding of a monochromatic path P is NP-complete. Furthermore, it remains NP-complete for all classes of blueprints G where the Hamiltonian path problem is NP-complete. These include *split graphs*, which are graphs whose vertices can be partitioned into a clique and an independent set. Finding a Hamiltonian path in a split graph is NP-complete even if the clique and the independent set have the same order, as shown in [21].

In particular, such graphs are *dense*, i.e., they have a number of edges that is quadratic in the number of vertices. The fact that our path embedding problem is NP-complete even for dense blueprints is somewhat surprising: intuitively, a blueprint G with many edges should allow greater leeway in the construction of an embedding of P. As it turns out, a greater amount of freedom does not necessarily translate into our ability to easily find embeddings.

A second class of interest is that of *grid graphs*, sometimes also called *lattice graphs*:

Definition 3. *A* grid graph *is an induced subgraph of a regular tiling of the plane.*

Note that there are only three possible regular tilings of the plane: the square lattice, the triangular lattice, and the hexagonal lattice.

Grid graphs are the typical setting of the standard HP model. It is known that the Hamiltonian path problem is NP-complete even if G is a grid graph in the square lattice, in the triangular lattice, or in the hexagonal lattice [1]. Thus, so is our monochromatic path embedding problem.

Furthermore, the *bicolored* path embedding problem is NP-complete when G is a *solid* grid graph. Formally, let us define $R(a, b)$ as the grid graph in the square lattice whose vertex set is $\{(i, j) \mid 1 \le i \le a \text{ and } 1 \le j \le b\}$.

Definition 4. *A* rectangular *grid graph is any subgraph of the square lattice that is isomorphic to $R(a, b)$ for some integers a and b.*

Rectangular grid graph are also called *solid* grid graphs in the square lattice.

Given any grid graph G' on n vertices in the square lattice, we can find the smallest integers a and b such that the graph $G = R(a, b)$ contains an induced subgraph G'' isomorphic to G' (intuitively, G is the "bounding rectangle" of G''). We color in blue all vertices of G'', and we color in red all other vertices of G. This operation is called "completing" G' to a solid grid graph G. Now, we can embed a path P of n blue vertices into G if and only if G' has a Hamiltonian path: this proves that the problem is NP-complete.

We can easily generalize the previous observation to solid graphs in the triangular or hexagonal lattice, which are defined similarly, as equilateral triangles and equilateral hexagons, respectively. This NP-completeness result will be strengthened in the next section, where we consider *bijective* embeddings.

3. Embeddings in Rectangular Grids

In this section we focus on *rectangular* blueprints, i.e., blueprints that are rectangular grid graphs, as defined in the previous section. As already mentioned, this is the typical setting of the traditional HP model of protein folding.

3.1 Bijective Embeddings

Let us first consider the case where the bicolored blueprint G is a "precise" description of a protein, i.e., it has to be matched exactly by the amino acid sequence represented by the bicolored path P. In other words, G and P have the same number of vertices, and the embedding should therefore be bijective.

As observed at the end of Sect. 2, the non-bijective bicolored embedding problem in a rectangular grid is NP-complete. On the other hand, the bijective embedding problem in a rectangular grid is polynomial-time solvable if P and G are monochromatic: indeed, this is equivalent to the Hamiltonian path problem in a rectangular grid, which is solved in [17].

In the following theorem, we will close the gap between the two aforementioned results: We will show that the *bijective bicolored* path embedding problem is NP-complete.

Theorem 1. *The bicolored path embedding problem is NP-complete even if the blueprint G is a rectangular grid with the same number of vertices as the path P.*

Proof. We will give an NP-hardness reduction from the Hamiltonian path problem on grid graphs in the square lattice, which is NP-complete [17].

(1) Construction of the reduction

We start from a rectangular grid $R(m', n')$ with an induced subgraph G' (which is an instance of the Hamiltonian path problem), and we construct the blueprint G by "expanding" each vertex v of $R(m', n')$ into a $(k + 2) \times (k + 2)$ block B_v (where k is a large-enough even constant, defined later). If v is not a vertex of G', then all vertices of B_v are blue; if v is a vertex of G', then B_v is illustrated in Fig. 1 (right): its four central vertices are red, and all other vertices are blue. The

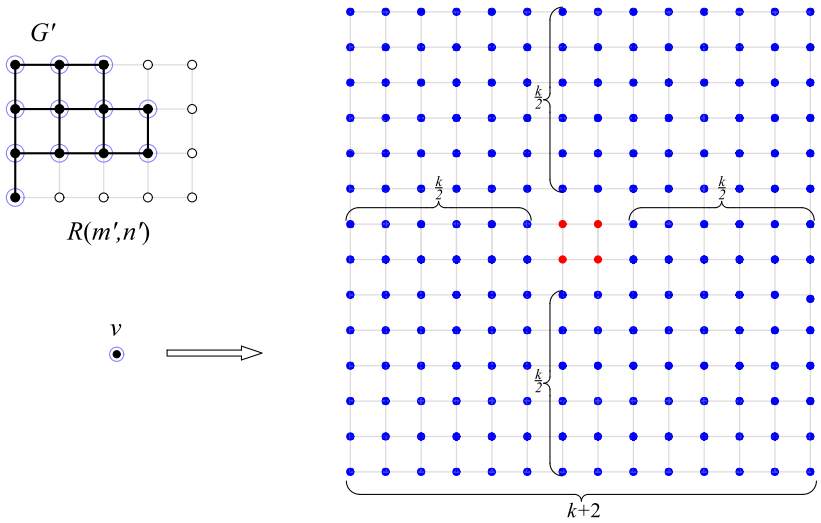

Fig. 1 Example of a grid graph G' and the transformation of a vertex v of G' into the $(k+2)\times(k+2)$ block B_v (in this example, $m'=5$, $n'=4$, and $k=12$)

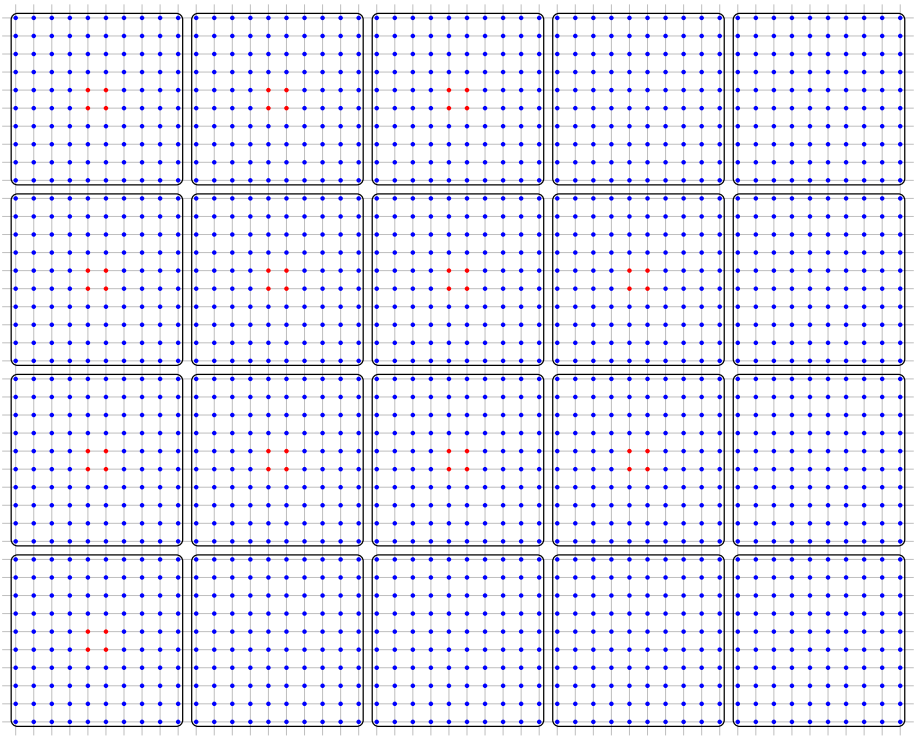

Fig. 2 Complete construction of G from the graph G' of Fig. 1 (now with $k=8$): each of the circled blocks represents a vertex in the original graph G'

order of G is therefore $(k+2)\cdot m' \times (k+2)\cdot n'$; an example of the full construction is shown in Fig. 2.

The path P is constructed as follows. Let P' be a path consisting of 4 red vertices followed by $2k$ blue vertices. P is made up of n consecutive copies of P', where n is the order of G', followed by a trail of blue vertices such that the total length of P matches the order of G. Namely, the final trail of P consists of $(k+2)^2\cdot m'n' - (2k+4)n$ blue vertices. Refer to Fig. 3 for an example.

(2) Embedding the first part of the path

In order to embed P into G, we have to start from a set of four red vertices in some block B_v, and then move to another set of four red vertices in some other block B_w. Since we must traverse exactly $2k$ blue vertices between these two red sets, this is possible only if v and w are adjacent in G' (note that a "diagonal" move would take $2k+1$ steps on blue

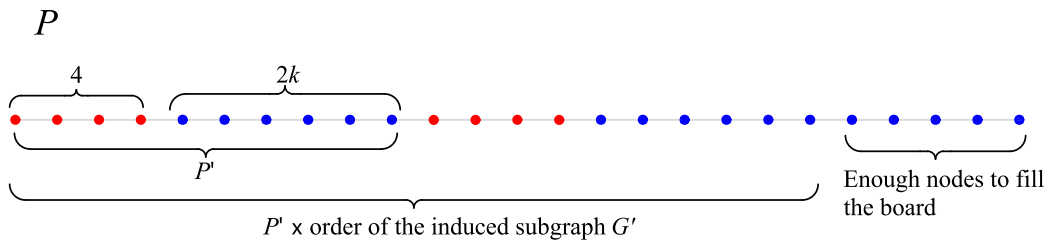

Fig. 3 Construction of the path P (in this example, $k = 3$ and G' has order 2)

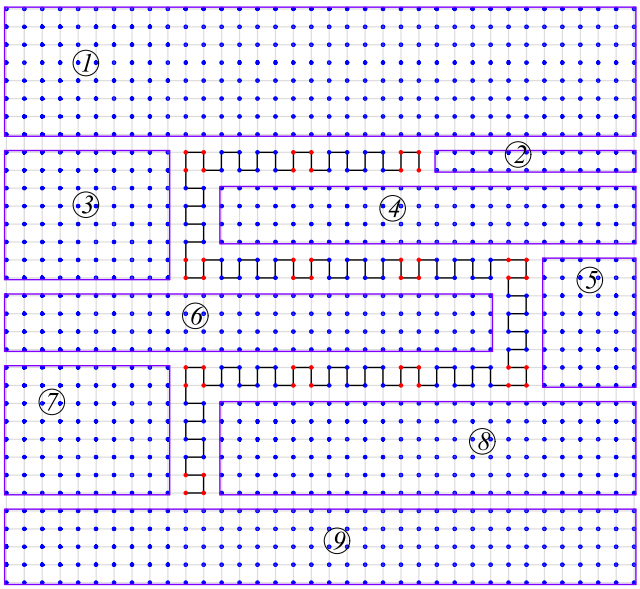

Fig. 4 Example of a partition into rectangles of the region not covered by the zig-zagging copies of P'

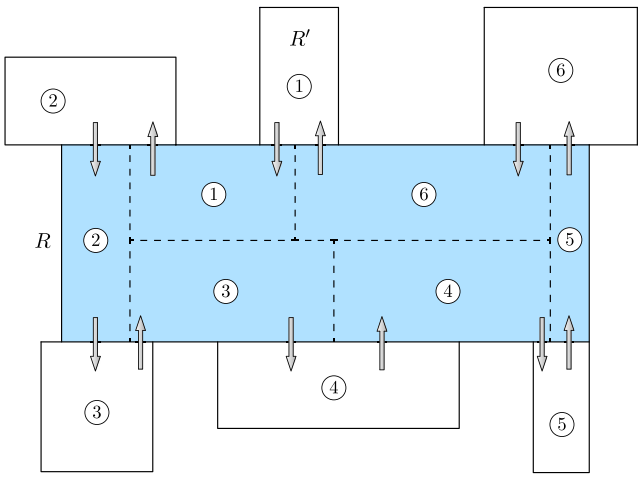

Fig. 5 The shaded rectangle R is subdivided into six tiles, one for each neighboring rectangle. The numbers and arrows show the order in which tiles and neighboring rectangles are covered. The path comes from the parent rectangle R', covers a tile, and moves to the next unvisited rectangle, etc. When R is fully covered, the path returns to the parent R'.

vertices). Thus, embedding P into G is impossible if G' is not Hamiltonian.

Assume now that G' is Hamiltonian. We can embed all copies of P' into G by "mimicking" a Hamiltonian path in G' and moving from one set of red vertices to the next by covering the $2 \times k$ rectangle between them in a zig-zag fashion. Eventually, the region of G covered by all the copies of P' looks like a winding "tube" of width 2, as sketched in Fig. 4.

(3) Embedding the trailing blue vertices

Now we have to cover the remaining part of G with the trailing sequence of blue vertices of P. Observe that this part of G is connected, because the copies of P' were embedded according to a Hamiltonian path, which has no cycles, and therefore did not disconnect G.

In order to cover this last region, we partition it into maximal "horizontal rectangles," i.e., in such a way that no two rectangles touch each other along vertical edges. Figure 4 shows an example of the partition. Then we do a depth-first traversal of these rectangles. When we visit a new rectangle R (perhaps coming from its parent rectangle R'), we cover R as exemplified in Fig. 5: we further divide it into smaller rectangular "tiles," one for each unvisited neighboring rectangle. After completely covering a tile, we continue the depth-first traversal by visiting its adjacent rectangle R'' in the partition. When we backtrack from R'' and get back to R, we move to the next tile of R, and so on.

Note that the last tile we visit is again adjacent to R', and thus we are able to backtrack once all tiles of R have been completely covered (if R is the root of the spanning tree, we just terminate). It is straightforward to prove by induction that this embedding algorithm completely covers all rectangles in the partition.

(4) Detailed construction of tiles

We still need to prove that it is possible to construct the tiles in such a way that each of them can be covered completely before moving on to the next rectangle. We will use a result from [17], where the grid graphs containing a Hamiltonian path with assigned endpoints have been characterized. The characterization includes some special cases of small order, but since our k is a large constant, we can ignore them.

What we can gather from [17] is that, if the order (i.e., the number of vertices) of a tile is even and one of its sides is longer than four vertices, then there is a Hamiltonian path

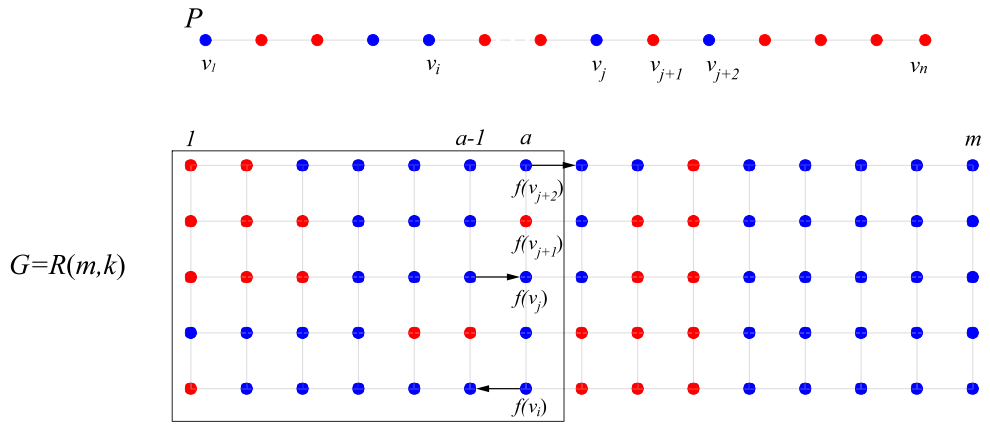

Fig. 6 Example of a sub-problem of the dynamic-programming algorithm for rectangular blueprints. The sub-problem specifies which vertices of the path P should be embedded in the ath column of G, and where: this is indicated by the function f. Each arrow represents a *direction bit*: the vertex v_{i+1} should be mapped to the left of $f(v_i)$, etc. The value of the direction bit at $f(v_{j+1})$ is irrelevant, because both v_j and v_{j+2} are mapped to the ath column. The sub-problem asks if there exists a partial embedding of P in the sub-grid going from the first column to the ath column that matches vertex colors and satisfies the constraints imposed by f and the direction bits.

in the tile with any assigned endpoints having odd distance along the grid. Because k is a large even constant, we can indeed subdivide each rectangle in the appropriate number of tiles, arranged as exemplified in Fig. 5, each of which has even order and at least one side longer than four vertices (choosing $k = 32$ abundantly suffices).

When covering a tile, we want to start on an edge and either end on the same edge or on the opposite edge. For example, referring to Fig. 5, when covering tile 1 we want to enter and exit from the same edge, but when covering tile 2 we want to enter from an edge and exit from the opposite one. So, it is sufficient to choose any pair of starting and ending vertices (along the appropriate edges) having odd distance, and the characterization in [17] guarantees that there is a Hamiltonian path in the tile having these starting and ending vertices. It follows that we can embed P into G. □

3.2 Fixed-Height Rectangular Blueprints

We can contrast our previous hardness results with an embedding algorithm that runs in polynomial time, provided that the blueprint G is a rectangular grid of fixed height k.

Lemma 1. *For all integers $a > 2$ and $b > 0$,*

$$\min\{(a+1)^b, (b+1)^a\} < e \cdot a^b.$$

Proof. It is well known from elementary calculus that, if $k > 0$, the function $f_k(x) = (1 + k/x)^x$ is monotonically increasing for $x > 0$, and its limit as x approaches $+\infty$ is e^k. Hence, by rearranging terms, we have, for every $x, k > 0$,

$$(x+k)^x < e^k x^x. \tag{1}$$

If $a \geq b$, we plug $k = 1$ and $x = a$ in Eq. (1), obtaining

$$(a+1)^b = ((a+1)^a)^{\frac{b}{a}} < (e \cdot a^a)^{\frac{b}{a}} = e^{\frac{b}{a}} \cdot a^b \leq e \cdot a^b.$$

If $a < b$, by a similar reasoning, we have $(b+1)^a < e \cdot b^a$. To conclude, it is now sufficient to prove that $b^a \leq a^b$. This is done by plugging $k = b - a$ and $x = a$ in Eq. (1):

$$b^a = (a + (b-a))^a < e^{b-a} \cdot a^a < a^{b-a} \cdot a^a = a^b,$$

recalling that, by assumption, $e < 3 \leq a$. □

Theorem 2. *Given a bicolored rectangular grid G of order $m \times k$ and a bicolored path P of order n, the embedding problem for G and P can be solved in $O(2^k n^{2k} m)$ time.*

Proof. Let G be a bicolored $m \times k$ grid, and let P be a bicolored path of n vertices. If $n \leq 2$, the problem can be trivially solved in $O(km)$ time by searching G for one or two adjacent vertices with colors matching P. Hence, let us assume that $n > 2$.

(1) Sub-problem specification

Our approach is based on dynamic programming, where a sub-problem consists in embedding part of P (not necessarily all of P) into a sub-grid of G going from the first column to the ath column, with $1 \leq a \leq m$. A sub-problem's specification also contains a description of the intersection between a hypothetical embedding of P and the ath column of G: for each vertex w in the ath column, the sub-problem specifies which vertex v_i of P is mapped to w (if any), as well as an extra bit of information: the *direction bit*. This bit encodes whether the left or right neighbor of v_i along P should be mapped to the left neighbor of w (if such information is incompatible with the rest of the specification, the direction bit is ignored). Figure 6 sketches a sub-problem with a function f specifying which vertices of P are mapped into the ath column.

Thus, the total number of sub-problems is $2^k c_{n,k} m$,

where the factor m represents the possible choices of a, and $c_{n,k}$ is the number of ways a subset of vertices of P can be injectively mapped into a column of G. Informally, we can say that $c_{n,k}$ is the number of ways two paths of length n and k (representing P and a column of G, respectively) can "intersect."

(2) Solving sub-problems

The output to a sub-problem is "Yes" if an embedding of part of P satisfying the given constraints exists, "No" if it does not exist, and "N/A" if the sub-problem specifies no intersection on the ath column, and it is not possible to embed P entirely to the left of the ath column (this implies that P must be embedded entirely to the right of the ath column, but we are still unable to determine if this is possible).

Solving a sub-problem S for column a amounts to finding a sub-problem S' for column $a-1$ with a "Yes" answer such that the specifications of S and S' are compatible. In other words, the mappings described by S and S' on columns a and $a-1$ should (i) match the colors in G and P, and (ii) match with each other. For example, assume that the sub-problem S is the one illustrated in Fig. 6, where the direction bit at $f(v_i)$ indicates that the vertex v_{i+1} of P should be mapped to the left neighbor w' of $w = f(v_i)$ (where w is in column a). Then, the sub-problem S' should agree with this specification: namely, its function f' should indicate that $f'(v_{i+1}) = w'$ (which is in column $a-1$).

(3) Full algorithm

Summarizing, the algorithm for solving a sub-problem S for column a is as follows:

- If $a = 1$, then:
  - If S specifies that the embedding of P does not intersect column 1, then return "N/A."
  - Else, if S specifies that the embedding of P intersects column 1 in a way that (i) matches vertex colors, (ii) whenever it maps two consecutive vertices v_i and v_{i+1} of P to column 1, it maps them to adjacent vertices, and (iii) the direction bits of S specify that the embedding of P continues to the right (whenever this makes sense), then return "Yes."
  - Else, return "No."
- If $a > 1$, and S specifies that the embedding of P does not intersect column a, then:
  - If there is a compatible sub-problem S' for column $a-1$ with answer "Yes," then return "Yes" (by "compatible" we mean that the direction bits of S' imply that no vertex of P mapped to column $a-1$ should have a neighbor mapped to column a).
  - Else, return "N/A."
- If $a > 1$, and S specifies that the embedding of P has some intersections with column a, then:
  - If S specifies that the embedding of P intersects column a in a way that (i) matches vertex colors, (ii) whenever it maps two consecutive vertices v_i and v_{i+1} of P to column a, it maps them to adjacent vertices, (iii) there is a sub-problem S' for column $a-1$ with answer "Yes" or "N/A" that is compatible with S, and (iv) if $a = n$, the direction bits of S specify that the embedding of P continues to the left (whenever this makes sense), then return "Yes."
  - Else, return "No."

(4) Optimizations and remarks

As an optimization, we do not have to look up *all* sub-problems S' for column $a-1$, but only the ones whose direction bits are compatible with S. In other words, we only need to choose which vertices of P are mapped to the column $a-1$ and where, and the correct direction bits can be inferred. Hence, in order to solve S, it is sufficient to look up at most $c_{n,k}$ sub-problems.

Also, for each sub-problem S', the compatibility test between S' and S can be done in constant amortized time. Indeed, the sub-problems S' are enumerated by locally changing the function that maps points on the column $a-1$ to vertices of P; as each of these local changes takes place, the corresponding compatibility check is performed in constant time. Hence, S can be solved in $O(c_{n,k})$ time.

As there are $2^k c_{n,k} m$ sub-problems in total, it takes $O(2^k c_{n,k}^2 m)$ time to solve all of them. In the end, the algorithm returns "Yes" if there is a sub-problem for $a = n$ with a "Yes" answer; it returns "No" otherwise.

(5) Correctness and running time

The correctness of this algorithm can be proved straightforwardly by induction. Note that the distinction between "N/A" and "Yes" implies that, if the final answer is "Yes," then at least some vertices of P have indeed been embedded somewhere in G. If this is the case, then the compatibility tests between columns guarantee that all of P has been correctly embedded.

In order to show that our algorithm has the desired running time, it remains to prove that $c_{n,k} = O(n^k)$. Recall that $c_{n,k}$ is the number of ways P can intersect a column of G. We can give two upper bounds on this number. Each of the k vertices in a column of G may intersect one of the n vertices of P or none of them. This yields at most $(n+1)^k$ different configurations in total, and thus $c_{n,k} \leq (n+1)^k$. Note that this is insufficient to conclude that $c_{n,k} = O(n^k)$, because k is not a constant. Let us give a second upper bound: each of the n vertices of P may be mapped either to one of the k vertices in the given column of G or to a different column. This yields $c_{n,k} \leq (k+1)^n$. Now, Lemma 1, with $a = n$ and $b = k$, gives

$$c_{n,k} \leq \min\{(n+1)^k, (k+1)^n\} < e \cdot n^k = O(n^k),$$

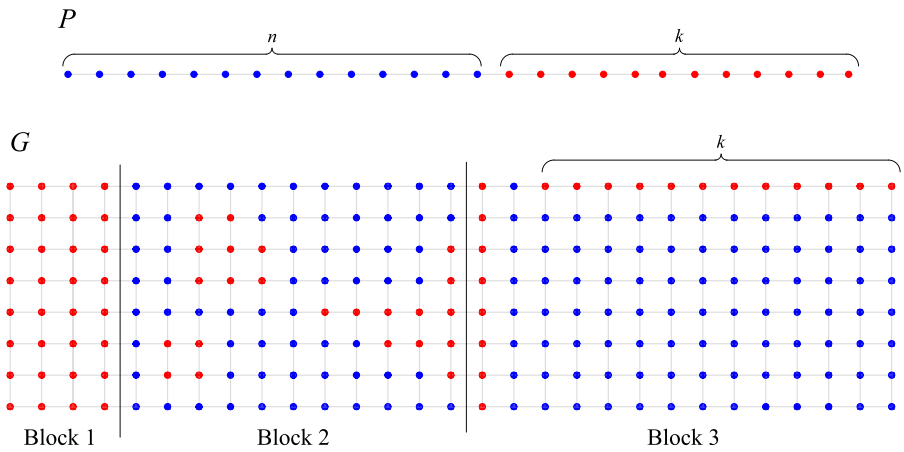

Fig. 7 Sketch of the NP-hardness reduction for the problem of maximizing red-red contacts (the value of n should match the number of blue vertices in Block 2)

as required (recall that we were assuming that $n > 2$). □

We immediately have the following:

Corollary 1. *The bicolored path embedding problem where the blueprint G is a rectangular grid, parameterized according to the height of G, is in XP.*

Proof. According to Theorem 2, if G has order $m \times k$ and P has order n, there is an algorithm that solves the embedding problem for G and P in $O(2^k n^{2k} m)$ time. Now, if k is a constant, the running time of the algorithm is $O(n^{2k} m)$, hence polynomial. □

4. Maximizing Red-Red Contacts

Finally, let us turn to the problem of maximizing red-red contacts in the context of the bicolored path embedding problem. Recall that, according to the HP model of energy, an amino acid chain tends to fold in a way that maximizes the number of H nodes that are close together in the folded state, even if they are not adjacent along the chain. In other words, when G and P are given, we seek an embedding of P into G that covers a large number of adjacent red vertices of G without traversing the edges that connect them with each other.

Definition 5. *A* red-red contact *in an embedding of P into G is a pair of adjacent red vertices u, v in G such that the embedding of P covers both u and v, but does* not *contain the edge* $\{u, v\}$.

4.1 Solid Grid Graphs

The problem of maximizing red-red contacts in the bicolored path embedding problem is also NP-hard, even when restricted to instances where the path P is guaranteed to be embeddable into G, and even when G is a solid grid graph (in a square, triangular, or hexagonal lattice).

Theorem 3. *Given a bicolored solid grid graph G and a bicolored path P that can be embedded in G, it is NP-hard to find an embedding of P in G that maximizes red-red contacts.*

Proof. We will describe a reduction from the Hamiltonian path problem in the case of a square lattice. A similar construction can be used for triangular and hexagonal lattices, as well.

Given a connected input grid graph G' on n vertices, we construct a rectangular grid graph G by juxtaposing three *blocks*, each of which is in turn a rectangular grid graph. Figure 7 shows a sketch of the whole construction.

Block 2 of G is constructed by completing G' to a rectangular grid graph, as we did at the end of Sect. 2. That is, Block 2 of G is the smallest rectangular graph $R(a, b)$ containing (an isomorphic copy of) G' as an induced subgraph; we color in blue the n vertices of this subgraph, and we color in red the $r = ab - n$ remaining vertices in $R(a, b)$.

Next we define $k = r + b + 1$, and we construct Block 1 as a grid graph isomorphic to $R(\lceil k/b \rceil, b)$ whose vertices are all red. Note that Block 1 has at least k vertices.

Block 3 of G is isomorphic to $R(\max\{k, n\} + 2, b)$, and is colored as shown in Fig. 7. Namely, the column adjacent to Block 2, i.e., the leftmost column, is all red; each of the k rightmost columns has the topmost vertex in red and all other vertices in blue; all other columns are entirely blue.

Finally, the path P consists of n blue vertices followed by k red vertices.

Without loss of generality, we may assume that, if a Hamiltonian path exists in G', one of its endpoints s must be on the perimeter of the bounding rectangle of G'. (This is a well-known fact in Hamiltonicity theory; see for example [17].) When constructing G, we can embed G' into Block 2 in such a way that s is in the column adjacent to Block 1. Thus, if G' has indeed a Hamiltonian path, we can embed the blue part of P in Block 2 and the red part of P in Block 1, which produces a large number of red-red contacts.

On the other hand, if G' does not have a Hamiltonian

path, we can only embed P in Block 3, which yields no red-red contacts. This is because embedding some red vertices of P in Block 1 would force us to embed all the blue vertices in Block 2, which is impossible because G' is not Hamiltonian. Also, if we embedded some red vertices in the leftmost column of Block 3, we would have to embed all of them in this column or in Block 2 (because G' is connected, and thus there is no path of red vertices connecting Block 1 with Block 3). However, there are only $r + b$ red vertices in this region, and therefore we cannot fit all the $k = r + b + 1$ red vertices of P. The only possibility is to embed these k red vertices in the topmost row of Block 3, which always yields a feasible embedding, as Block 3 has a large-enough blue region to fit the n blue vertices of P, as well.

In conclusion, P can always be embedded in G; however, finding an embedding that produces any red-red contacts at all is NP-hard. □

The above reduction implies that the maximum number of red-red contacts is not only NP-hard to compute exactly, but also to *approximate*. In the next section, we will make this observation more precise with a strong inapproximability result.

4.2 General Graphs

For a bicolored path P and a bicolored graph G, let $m_{P,G}$ be the maximum number of red-red contacts across all embeddings of P into G. In the previous section we showed that computing $m_{P,G}$ is NP-hard, even assuming that the solution space is non-empty. Now, in the spirit of protein synthesis, we formulate the following problem:

Definition 6. *Given a bicolored graph G, the* bicolored synthesis problem *asks for the bicolored path P of the same order as G that maximizes $m_{P,G}$.*

Translated into the language of protein folding, we are given the "form" of a protein (i.e., a bicolored graph G), and we ask for the amino acid chain that is most likely to fold into a protein of that particular form.

We will show that this problem is Poly-APX-hard, i.e., the optimum is NP-hard to approximate within a subpolynomial ratio.

Theorem 4. *The bicolored synthesis problem is Poly-APX-hard.*

Proof. We will give an approximation-preserving reduction from the *Independent Set* problem, which is Poly-APX-complete [2]. For the reduction, we borrow the *edge gadget* from [14], which is illustrated in Fig. 8, top. The top six vertices (next to the letter u) constitute the "top half" of the gadget; the other six are the "bottom half". If this gadget is part of a larger graph, there are only three ways a Hamiltonian path can traverse it, as shown in the figure.

Now, given a connected graph $G' = (V', E')$, where $V' = \{v_1, \ldots, v_n\}$, we will construct a bicolored graph G that

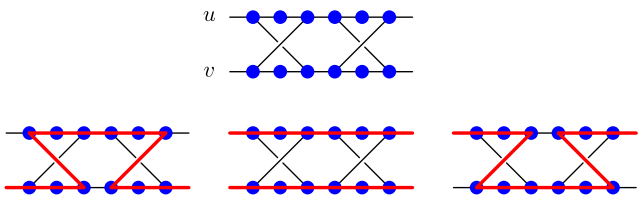

Fig. 8 Illustration of the *edge gadget* used in Theorem 4 (see [14]). The bottom part of the figure shows the three possible ways a Hamiltonian path can traverse this gadget.

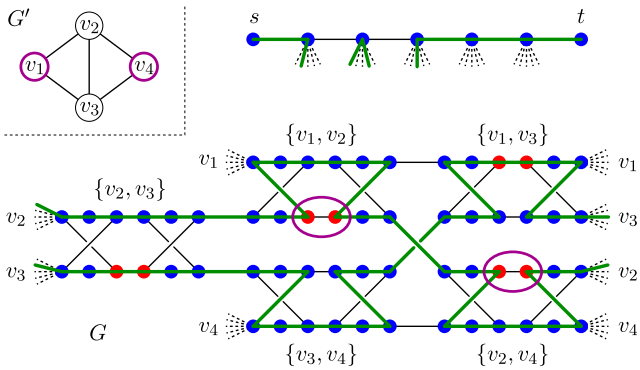

Fig. 9 Example of the approximation-preserving reduction used in Theorem 4. As the dashed edges suggest, each vertex in G labeled $v_1, \ldots, v_4$ is adjacent to all vertices in the top *selector gadget*, except s and t. The vertices of G' circled in purple constitute an independent set that corresponds to a path embedded in G (drawn in green) with as many red-red contacts.

implements our approximation-preserving reduction as follows (an example is shown in Fig. 9). First we construct an edge gadget for each edge in E'. Then we connect edge gadgets together in such a way that, for each vertex $v_i \in V'$, there is a path, called "strand," that traverses (either the top or the bottom half of) each of the edge gadgets corresponding to edges incident to v_i in G'. For example, in Fig. 9, the three gadgets labeled $\{v_2, v_3\}$, $\{v_1, v_2\}$, and $\{v_2, v_4\}$ are connected together in sequence, forming a strand whose endpoints are labeled v_2 in the figure. This represents the fact that the vertex $v_2 \in V'$ is adjacent to v_1, v_3, and v_4 in G'. It follows that each edge gadget is shared by precisely two strands.

Next, we construct a *selector gadget*, shown at the top of Fig. 9, which simply consists of a path of $n+3$ vertices, the endpoints of which are called s and t. We connect each vertex of the selector gadget, except s and t, to both endpoints of all the previously constructed strands; these connections are represented by dashed edges in the figure.

Finally, we color all vertices of G blue except for $2n$ of them, as follows. For each vertex $v_i \in V'$, we choose one adjacent vertex v_j, we identify the edge gadget corresponding to $\{v_i, v_j\}$, and we color in red the two central vertices in the gadget that belong to the strand of v_j, as shown in Fig. 9.

This completes the construction. We will now prove that G' has an independent set $S \subseteq V'$ of k vertices if and only if there is a bicolored path P of the same order as G that can be bijectively embedded in G forming exactly

k red-red contacts. This will imply that our reduction is approximation-preserving. In the example in Fig. 9, the independent set is $S = \{v_1, v_4\}$.

Note that the existence of an embedded path P that forms k red-red contacts is equivalent to the existence of a Hamiltonian path in G that avoids exactly k edges whose endpoints are both red. Now, any Hamiltonian path in G must have endpoints in s and t, and use the selector gadget to access some of the strands. Once a strand has been chosen, it must be followed until the end; after that, the path goes to the next vertex of the selector gadget, and then into another strand. Along the strand of vertex v_i, each encountered edge gadget $\{v_i, v_j\}$ has to be covered in one of two possible ways, depending on whether the strand of v_j will be traversed or not. If the strand of v_j is not going to be traversed, the path must cover all vertices in the edge gadget $\{v_i, v_j\}$; for an example, see the edge gadget $\{v_1, v_2\}$ in the figure, where the strand of v_2 is traversed and the strand of v_1 is not. In this case, if the edge gadget contains two red vertices corresponding to v_j, these vertices will form a red-red contact.

Thus, if a Hamiltonian path forms two red-red contacts corresponding to vertices v_a and v_b, it means that it does not traverse the strands of v_a and v_b. Hence v_a and v_b are not adjacent in G', otherwise there would be an edge gadget $\{v_a, v_b\}$ in G that is not covered by the path. So, the set of red-red contacts determined by a Hamiltonian path corresponds to an independent set of G'. Conversely, if S is an independent set, traversing the strands of the vertices in $V' \setminus S$ yields a Hamiltonian path with $|S|$ red-red contacts. □

5. Conclusions

We proposed the bicolored path embedding problem, inspired by protein folding in the HP model. We showed that the problem is NP-hard in several settings, and polynomial-time solvable if the blueprint is a grid graph of fixed height.

Our hardness results indicate that certain general problems related to protein folding are probably intractable by computers. Interpreting some of these results in practical terms is ultimately a philosophical matter: since nature does indeed seem to solve NP-hard protein folding problems in an efficient way, this may indicate that our ways of modeling computation (e.g., Turing machines) do not correctly reflect the way nature works. It may also indicate that discrete models of protein folding such as the HP model do not faithfully capture the essence of real protein folding, or that our NP-hardness reductions produce instances of amino acid chains and proteins that are unlikely to be found in real biological systems. To address the latter issue, we suggest studying the same problems under the *smoothed analysis* paradigm: adding small amounts of random noise to an amino acid chain may be sufficient to eliminate the special patterns that cause protein folding to be computationally intractable.

Another way of interpreting our hardness results is as an "upper bound" on what can be done efficiently by computers. For example, Theorem 4 states that guessing an amino acid chain that is likely to fold into a protein of a given arbitrary shape is a hopelessly hard problem. This knowledge should discourage us from attempting to find efficient algorithms for the general problem, and direct us toward special cases or relaxations of the problem. For instance, it would be interesting to know if Theorem 4 remains true when G is a grid graph: we leave this as an open question.

Another open question is whether the running time of the dynamic-programming algorithm in Sect. 3.2 can be improved. Also, it would be interesting to find other natural classes of blueprints for which the bicolored path embedding problem is polynomial-time solvable.

Acknowledgments

The authors are grateful to the anonymous reviewers for greatly improving the quality of this paper with helpful comments and suggestions. This work is partially supported by JSPS KAKENHI Grant Numbers 17H06287 and 18H04091.

References

[1] E.M. Arkin, S.P. Fekete, K. Islam, H. Meijer, J.S.B. Mitchell, Y. Núñez-Rodríguez, V. Polishchuk, D. Rappaport, and H. Xiao, "Not being (super) thin or solid is hard: A study of grid Hamiltonicity," Computational Geometry, vol.42, no.6–7, pp.582–605, Aug. 2009.

[2] C. Bazgan, B. Escoffier, and V.T. Paschos, "Completeness in standard and differential approximation classes: Poly-(D)APX- and (D)PTAS-completeness," Theoretical Computer Science, vol.339, no.2-3, pp.272–292, June 2005.

[3] P.-A. Champin and C. Solnon, "Measuring the similarity of labeled graphs," Proc. the 5th Int. Conf. Case-Based Reasoning Research and Development (ICCBR'03), pp.80–95, 2003.

[4] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis, "On the complexity of protein folding," J. Computational Biology, vol.5, no.3, pp.423–465, 1998.

[5] K.A. Dill, "Theory for the folding and stability of globular proteins," Biochemistry, vol.24, no.6, pp.1501–1509, March 1985.

[6] K.A. Dill, S.B. Ozkan, M.S. Shell, and T.R. Weikl, "The protein folding problem," Annual Review of Biophysics, vol.37, pp.289–316, 2008.

[7] K.A. Dill, S.B. Ozkan, T.R. Weikl, J.D. Chodera, and V.A. Voelz, "The protein folding problem: when will it be solved?" Current Opinion in Structural Biology, vol.17, no.3, pp.342–346, June 2007.

[8] K.A. Dill and J.L. MacCallum, "The protein-folding problem, 50 years on," Science, vol.338, no.6110, pp.1042–1046, Nov. 2012.

[9] E.D. Demaine and J. O'Rourke, Geometric folding algorithms: Linkages, origami, polyhedra, Cambridge University Press, July 2007.

[10] Y. Duan and P.A. Kollman, "Computational protein folding: From lattice to all-atom," IBM Systems Journal, vol.40, no.2, pp.297–309, 2001.

[11] D. Eppstein, "Subgraph Isomorphism in Planar Graphs and Related Problems," J. Graph Algorithms and Applications, vol.3, no.3, pp.1–27, 1999.

[12] T. Feng, R. Uehara, and G. Viglietta, "Bicolored path embedding problems in protein folding models," Proc. 37th European Workshop on Computational Geometry (EuroCG'21), pp.24:1–24:9, 2021.

[13] K.F. Lau and K.A. Dill, "A lattice statistical mechanics model of the conformational and sequence spaces of proteins," Macromolecules,

vol.22, no.10, pp.3986–3997, Oct. 1989.
[14] M.R. Garey and D.S. Johnson, Computers and intractability, Freeman, San Francisco, 1979.
[15] W.E. Hart and S.C. Istrail, "Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal," J. Computational Biology, vol.3, no.1, pp.53–96, April 1996.
[16] W.E. Hart and A. Newman, "The computational complexity of protein structure prediction in simple lattice models," in Handbook of Computational Molecular Biology, CRC Press, 1–24, 2001.
[17] A. Itai, C.H. Papadimitriou, and J.L. Szwarcfiter, "Hamilton paths in grid graphs," SIAM J. Computing, vol.11, no.4, pp.676–686, 1982.
[18] S.-M. Hsieh, C.-C. Hsu, and L.-F. Hsu, "Efficient method to perform isomorphism testing of labeled graphs," Proc. the 6th Int. Conf. Computational Science and Its Applications (ICCSA'06), pp.422–431, 2006.
[19] T. Jiang, Q. Cui, G. Shi, and S. Ma, "Protein folding simulations of the hydrophobic- hydrophilic model by combining tabu search with genetic algorithms," J. Chemical Physics, vol.119, no.8, pp.4592–4596, 2003.
[20] G. Mauri, G. Pavesi, and A. Piccolboni, "Approximation algorithms for protein folding prediction," Proc. the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99), pp.945–946, Jan. 1999.
[21] H. Müller, "Hamiltonian circuits in chordal bipartite graphs," Discrete Mathematics, vol.156, no.1-3, pp.291–298, Sept. 1996.
[22] A. Neumaier, "Molecular modeling of proteins and mathematical prediction of protein structure," SIAM Review, vol.39, no.3, pp.407–460, 1997.
[23] A. Newman, "A new algorithm for protein folding in the HP model," Proc. the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02), pp.876–884, Jan. 2002.

Tianfeng Feng is a Ph.D. student at the School of Information Science, Japan Advanced Institute of Science and Technology (JAIST). She is a member of Uehara Laboratory.

Ryuhei Uehara is a professor at the School of Information Science, JAIST. He received his B.E., M.E., and Ph.D. degrees from the University of Electro-Communications, Japan, in 1989, 1991, and 1998, respectively. He was a researcher in CANON Inc. during 1991–1993. In 1993, he joined Tokyo Woman's Christian University as an assistant professor. He was a lecturer during 1998–2001, and an associate professor during 2001–2004 at Komazawa University. He moved to JAIST in 2004. His research interests include computational complexity, algorithms and data structures, and graph algorithms. He is especially engrossed in computational origami, games and puzzles from the viewpoints of theoretical computer science. He is a member of IPSJ and IEICE. He is the chair of the Japan Chapter of EATCS.

Giovanni Viglietta received his Ph.D. in Computer Science from the University of Pisa in 2013. He is now Assistant Professor at the Japan Advanced Institute of Science and Technology (JAIST).