FITTING PROTEIN CHAINS TO CUBIC LATTICE IS NP-COMPLETE

JÁN MAŇUCH

School of Computing Science,
Simon Fraser University,
Burnaby, BC, V5A 1S6, Canada
E-mail: jmanuch@sfu.ca

DAYA RAM GAUR

Department of Math and Computer Science,
University of Lethbridge,
Lethbridge, AB, T1K 3M4, Canada
E-mail: gaur@cs.uleth.ca

It is known that folding a protein chain into the cubic lattice is an NP-complete problem. We consider a seemingly easier problem, given a 3D fold of a protein chain (coordinates of its C_α atoms), we want to find the closest lattice approximation of this fold. This problem has been studied under names such as "lattice approximation of a protein chain", "the protein chain fitting problem" and "building protein lattice models". We show that this problem is NP-complete for the cubic lattice with side 3.8Å and the coordinate root mean-square deviation.

1. Introduction

A protein is a linear sequence of amino acids which when placed into a solvent forms a 3D structure (fold). One of the main problems in proteomics is to computationally determine the structure of a protein given a sequence of amino acids. This problem appears extremely hard even when very simplified models are considered. For instance in Dill's HP-model[6], it is assumed that the "centers" of amino acids (C_α atoms) of the protein structure occupy vertices of a given lattice. A fold of a protein then can be represented as a path in the lattice. The second simplification, is the energy function of the fold. Instead of considering all forces affecting the folding process, only hydrophobic interactions between amino acids neighboring in the lattice are considered. It was shown in Refs. 1 and 4 that protein folding is NP-complete even in this simplified model.

Even though protein folding in lattice models is NP-complete, it is more computationally feasible than in the general non-lattice models as the lattice significantly limits the degree of freedom. In fact, lattice models are widely used in investigation of folding kinetics and thermodynamics[5,7] and for computer investigation of protein folding[15,20]. However, even if the optimal (native) fold in a certain lattice model is found it could be quite far from the real fold of the protein. Identifying lattice models which have a potential to produce folds close to real 3D structures is an important question in structural proteomics studied

in handful of papers[3,11,10,19,18,16,17,12,14] to cite a few.

To measure the accuracy of representation of lattice models, the following procedure is commonly used: (1) select a test set of proteins with known 3D structure (for instance from PDB[2]); (2) find the closest lattice representation of each protein minimizing the overall distance of the lattice representation to the exact structure, as measured by the coordinate root mean squared deviation (c-RMS) or by the distance rms deviation (d-RMS); (3) calculate average of the c-RMS (d-RMS) values over all proteins in the test set. The crucial part of this procedure is the computation of the closest lattice representation (of the chain) of a given protein structure. This problem is also referred to as "the discretization of a protein backbone", "lattice approximation of 3D structure of a chain molecule", "constructing lattice models of protein chains", "modeling protein structures on a lattice", "discrete state model fitting to X-ray structures" or "fitting of a protein chain to a lattice". In this paper, we call the problem *protein chain lattice fitting problem* (the PCLF problem). Also note that an algorithm for the PCLF problem is an essential part of genetic protein folding algorithm, cf. Ref. 17.

The first algorithm for the PCLF problem proposed in Ref. 3 enumerates all possible conformations and picks the best one. Dynamic programming based algorithms were presented in several papers, cf. Refs. 19, 18 and 17. A greedy approach keeping about 500 "best" lattice folds was used in Ref. 16 and another greedy approach was used in Ref. 14. A completely different approach using the self-consistent mean field theory was presented in Ref. 12. All these algorithms either exhaustively enumerate all conformations, which can be applied only on very short proteins, or produce approximate solutions. It is questionable how reliable is the comparison of accuracy of various lattices based on an approximate algorithm. A chosen approximation algorithm might have a better approximation ratio for certain types of lattices which would consequently show higher accuracy for those lattice than other ones. Therefore, it would be highly desirable to develop a fast (polynomial) and exact algorithm solving the protein chain fitting problem.

In this paper, we show that the protein chain lattice fitting problem is NP-complete for the cubic lattice with side 3.8Å and c-RMS deviation. Although this result does not immediately imply that the problem is intractable for other lattices as well, it would be very unlikely if it is not the case.

2. Formalization of the problem

Given is a protein as a sequence of 3D points and a regular lattice embedded into space (each lattice vertex is a 3D point). The goal is to map every point of the protein to a lattice point such that consecutive protein points are mapped to lattices points connected by an edge, the mapping is injective and the "distance" between the sequences of protein points and mapped points is minimized. The following properties of proteins whose structure is available in PDB[2] are well known:

- the distances between consecutive points of a protein vary very little from 3.8Å;
- the distances between non-consecutive points of a protein is at least 3.8Å.

We will assume that the lengths of the edges of the lattice are equal to 3.8Å. We can then easily scale the whole setting so that the distances between consecutive protein points and between two neighboring lattice points are one. Hence, we can formalize the protein chain fitting problem as follows:

Protein Chain Lattice Fitting (PCLF) Problem
Instance: Equilateral lattice L with side 1, a sequence of points $p = p_1, \ldots, p_n$ such that

(P1) $d(p_i, p_{i+1}) = 1$, for every $1 \leq i \leq n$, and
(P2) $d(p_i, p_j) \geq 1$ for every $2 \leq i + 1 < j \leq n$,

a distance measure α on the sequence of points, and a number K.
Question: Is there a path $l = l_1, \ldots, l_n$ in L such that $\alpha(p, l) \leq K$?

$\alpha(p, l)$ represents the quality of the lattice representation of a given protein structure. Two most common distance measures used to measure this quality are the *coordinate root mean square deviation* (c-RMS) and the *distance root mean square deviation* (d-RMS), defined as follows

$$\text{c-RMS}(p, l) = \sqrt{\frac{\sum_{i=1}^{n} d^2(p_i, l_i)}{n}}, \quad \text{d-RMS}(p, l) = \sqrt{\frac{\sum_{1 \leq i < j \leq n} [d(p_i, p_j) - d(l_i, l_j)]^2}{n(n-1)/2}}.$$

We show that the PCLF problem is NP-complete for the cubic lattice and the c-RMS measure. We would like to point out that the assumption that $l = l_1, \ldots, l_n$ is a *path* in the lattice is a crucial assumption. In fact it can be proved that if l is a *walk* in the lattice, the problem can be solved in polynomial time[9]. We use a reduction from a special variant of the satisfiability problem shown to be NP-complete in Ref. 13.

Var-linked planar 3-SAT (VLP-3-SAT)
Instance: A formula Φ with a set C of clauses over a set X of variables in a conjunctive normal form such that:

(S1) Every clause contains at most three variables.
(S2) The set X of variables allows a linear ordering, $x_1, \ldots, x_n$ such that the graph $G_\Phi = (C \cup X, \{xc;\ x \in c \in C \text{ or } \neg x \in c \in C\} \cup \{x_i x_{i+1};\ i = 1, \ldots, n\})$ is planar (here $x_{n+1} = x_1$).

Question: Is Φ satisfiable?

Note that for a different variant of planar 3-SAT, in which clauses are linked in a circled chain instead of variables (clause-linked), it was shown in Ref. 8 that it can be assumed that each variable occurs in exactly three clauses, once negated and twice positive. A similar assumption for the var-linked version of the planar 3-SAT problem simplifies our proof. We provide the justification for this assumption in the next paragraph.

Consider an instance of the VLP-3-SAT problem. It is possible to draw G_Φ in a way that all the variables in X lie on one vertical line, and clauses lie either to the left or to the right of this line and connect directly to the variables without crossing the vertical line.

Henceforth, we distinguish between left and right clauses. Now, similar to the construction in Ref. 8, we replace every variable x which does not have exactly 3 occurrences such that at least one of them is positive and one negative, with variables $l_1, \ldots, l_k$ and $r_1, \ldots, r_m$, where x was connected to left clauses $L_1, \ldots, L_k$ and right clauses $R_1, \ldots, R_m$. Obviously, we can assume that $k + m \geq 2$. Figure 1 shows the connections before and after the replacement. Note that we have introduced new clauses $(l_i \vee \neg l_{i+1})$, $(l_k \vee \neg r_m)$, $(r_{i+1} \vee \neg r_i)$ and $(r_1 \vee \neg l_1)$, which guarantees that all new the variables have the same value in any satisfiable assignment. Therefore, the new 3-SAT formula is satisfiable if and only if the original one was, and every variable has at least one positive and at least one negative occurrence. Finally, replacing the variables with negative occurrences by their negations we obtain a var-linked planar 3-SAT formula satisfying the following condition:

(S3) Every variable occurs in exactly three clauses, once negated and twice positive.

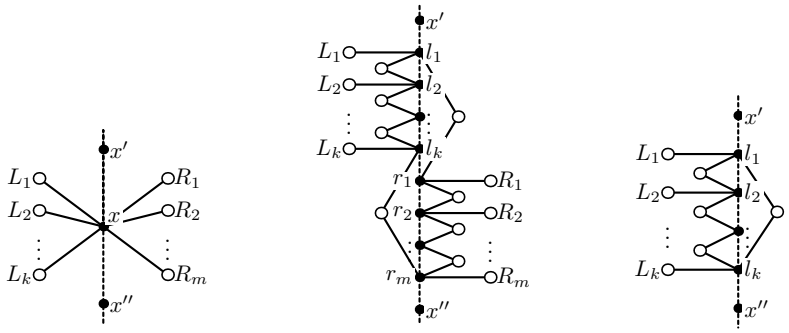


Figure 1. Replacement of variable x with sequence of variables: (a) the original configuration; (b) a new configuration in case $k, m > 0$; (c) a new configuration in case $m = 0$.

3. NP-completeness of the PCLF problem for the cubic lattice and the c-RMS measure

To prove the NP-completeness of the PCLF problem, we consider special instances of the problem. Next we give a simple lower bound on the c-RMS of protein sequence p and its lattice approximation l. For every p_i, let $L(p_i)$ denote the set of lattice vertices which are the closest to p. Let d_i be the distance of p_i from $L(p_i)$, i.e., $d_i = d(p_i, q)$, where $q \in L(p_i)$. Let $D = \sqrt{\frac{\sum_{i=1}^{n} d_i^2}{n}}$. Obviously, for any lattice path l, c-RMS$(p, l) \geq D$. In the special instances we set K to D, i.e., we get an affirmative answer to the instance if and only if there is a path l in the lattice such that $l_i \in L(p_i)$. We show that this happens if and only if the formula Φ for which we built the instance is satisfiable. Since the cubic lattice $L_\square$ is equilateral, we can assume that the vertices of $L_\square$ are all integral points, i.e., $V(L_\square) = \{[x, y, z];\ x, y, z \in \mathbb{Z}\}$ and $E(L_\square) = \{([x, y, z], [x', y', z']);\ |x - x'| + |y -$

$y'|+|z-z'|=1\}$. A point p for which $L(p)$ is not a singleton will be called a *flexible* point. Obviously, flexible points will play an important role in the construction. It is sufficient to prove the following lemma.

Lemma 3.1. *It is NP-complete to decide whether for a given sequence of points $p = p_1, \ldots, p_n$ there is a path $l = l_1, \ldots, l_n$ in $L_\square$ such that for every $i = 1, \ldots, n$, $l_i \in L(p_i)$.*

Proof. We establish the NP-completeness by a reduction from the VLP-3-SAT problem. Let Φ be a formula and G_Φ a planar drawing of Φ such that every variable occurs twice positive and once negated (cf. the definition of VLP-3-SAT and the discussion after the definition). We construct p in two phases. In the first phase, we construct several subsequences of p lying either in or close to (within distance 3 from) the plane $z = 0$, each ending in $L_\square$-points. These sub-sequences closely follow the planar drawing G_Φ. In the second phase, these subsequence are connected to one sequence p using only $L_\square$-points not lying in the plane $z = 0$. Obviously, this can be done without any problem and in any solution they have to be mapped to the same points, therefore we omit the explicit description of the second phase.

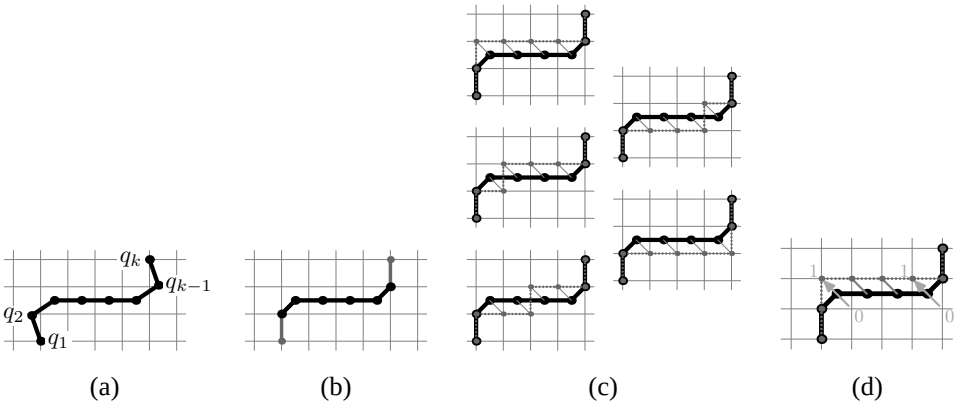


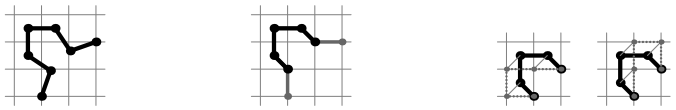
Figure 2. Illustration of a "wire": (a) a real subsequence (a distance between two consecutive points is 1); (b) a schematic drawing of subsequence (the gray edges will be usually omitted); (c) 5 possible lattice approximations of the subsequence (depicted with dotted lines); (d) pulling the wire on the left hand side forces a unique state of the wire.

The first basic building block of the construction is a "*wire*", cf. Figure 2(a). The end points of the wire, q_1 and q_k are lattice points, as required. The middle points $q_3, \ldots, q_{k-2}$ are flexible points, each having four choices for the closest lattice point, since they lie in the centers of square faces of plane $z = 0$, i.e., in the set $L_{1/2} = [1/2, 1/2, 0] + V(L_\square)$. The second and the penultimate points, q_2 and q_{k-1}, lie neither in $L_\square$ nor in $L_{1/2}$, as they connect lattice points to flexible points. We will call such points *connecting points*. Even though they do not lie directly in the lattice, their closest lattice points sets $L(q_2)$ and $L(q_{k-1})$ are singletons, i.e., in every solution, they are uniquely mapped to the

lattice points. Therefore, in our schematic drawing, the end points of the wire are usually omitted and the connecting points are shifted to the lattice points to which they are uniquely mapped, cf. Figure 2(b).

Next we show that the connecting points satisfying conditions (P1) and (P2) do exist. Wlog assume that $q_1 = [1, 0]$ and the leftmost flexible point is $[5/2, 3/2]$ then $q_2 = [3/4 + \sqrt{15}/20, 5/4 - 3\sqrt{15}/20] = [0.9436, 0.669]$. It can be verified that q_2 obeys the distance constraints (P1) and (P2).

Figure 2(c) shows all the possible ways a "wire" can be mapped to the lattice points sequence forming a path in $L_\square$. An important property of a wire is that forcing the first (leftmost) flexible point (by some other gadget of the construction) to position 1, as depicted in Figure 2(d), yields a unique state of the wire, and most importantly, makes the last flexible point to map into position 1 as well (which can affect state of another gadget on this end of wire). One can imagine that the wire is sending a signal from left to the right (or symmetrically, from right to the left). More formally, consider two boolean variables s and t. Let $s = 1$ iff the first flexible point of the wire is in position 1 and let $t = 1$ iff the last flexible point is in position 1. Then in every solution, we have $s \implies t$.



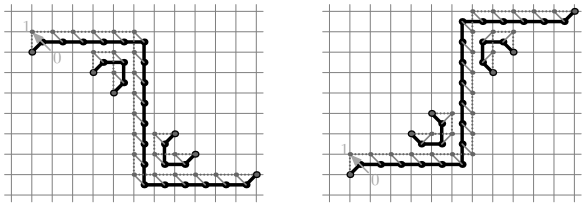
(a) (b) (c)

Figure 3. Illustration of a "flipper": (a) a real subsequence; (b) a schematic drawing (the gray edges will be usually omitted); (c) two possible states of the subsequence.

The second basic building block is a "*flipper*", cf. Figure 3(a). Note that the distance between two connecting points of the subsequence is $\frac{3}{\sqrt{2}} - \frac{\sqrt{6}}{\sqrt{5}} > 1$, i.e., condition (P2) is satisfied. In the schematic drawing we again omit the end points and shift the second and the second last point to positions where they are uniquely mapped, cf. Figure 3(b). Figure 3(c) shows all the possible states for a flipper.

To model the graph G_Φ we replace each vertex (clause) $c \in C$ by a "clause gadget" and every vertex (variable) $x \in X$ with a "variable gadget". We will use two different types of gadgets for clauses depending on the number of literals, and several different types of gadgets for variables depending on occurrences of variables in the formula Φ (positive or negative, in left or right clauses, as well as their relative order). The variable gadgets are placed vertically on top of each other and there are no connections between them as in G_Φ. For each edge between a clause and a variable, we have a wire connecting the corresponding clause and variable gadgets. It is not always possible to draw G_Φ in a way that all clause-variable edges are horizontal. Therefore, we need a wire which bends and correctly sends a signal from one end to the other. Such wires can be constructed using only two bends. Figure 4 shows how using two flippers we can achieve exactly this. Consider Figure 4(a), if the top part of the wire is forced to be in state 1 (pulled to the top) then the

bottom part of is also pulled. If the top part of the wire is in state 0, then the top flipper is in $\sqsupset$ state, this forces the bottom flipper to be in $\sqcup$ state, this forces the bottom part of the wire to be in state 0.



(a) (b)

Figure 4. Illustration of a wire which is shifted from one horizontal line to another: (a) down shift; (b) up shift. The dotted line shows the unique state when the first flexible point is forced to position 1.

Consider a left clause $c \in C$ (for right clauses, we would use a symmetrical design). First, assume that the clause contains two literals. The gadget for such a clause is depicted in Figure 5. The point q, common to the two wires which appears as a lattice point in the schematic drawings (a)–(c) has to be shifted off the lattice (by 1/4 in both coordinates) as shown in Figure 5(d), otherwise the connecting points p and r would be closer to each other than 1 which would violate condition (P2). Enumerating all the possibilities (using a computer program), we found out that there are 42 states for this gadget, 2 states with the upper wire pulled (the blue arrow is in position 1) and the lower wire not pulled (position 0), cf. Figure 5(a) for one of them, 2 states with the reversed situation of the wires, cf. Figure 5(b), and the remaining have both wires are pulled, cf. Figure 5(c) for instance. If s and t are boolean variables representing the position of the last flexible point of the wires, then in any solution, we have $s \vee t$ is satisfiable, and vice versa, for any values of s and t such that $s \vee t$ is satisfied, there is a solution with these values. Consider the case when the gadget in Figure 5 has both the top and the bottom wires in state 0 (not pulled). The state of the bottom wire forces the bottom flipper to be in $\sqcup$ state, and the flipper in the middle to be in $\sqcap$ state. The state of the top wire forces the top flipper to be in $\sqsubset$ state, but point q can be occupied by only one flipper, this leads to a contradiction. Hence, both the wires cannot be in 0 state.

Next, assume that the clause contains three literals. We want to design a gadget with 3 wires coming out of it such that it allows all and only those states in which at least one of the wires is pulled. It seems hard to design such a planar gadget, therefore we use a 3D-version of the flipper depicted in Figure 6(a). Note that we again need two connecting points q_2 and q_3 on left end of the sequence. Points q_2 and q_3 are exactly above each other at distance 1. If q_2 were placed on a lattice vertex, it would be too close to the point q_7 on the other end of the subsequence. Figure 6(b) shows all possible states of the flipper. Observe that in each state, the mapped sequence goes through exactly one of the points p, q, r. Two of them, p and q lie directly in the plane $z = 0$. For point r, we use one vertically placed flipper to transfer information that r is occupied to plane $z = 0$, cf. Figure 7(a–b).

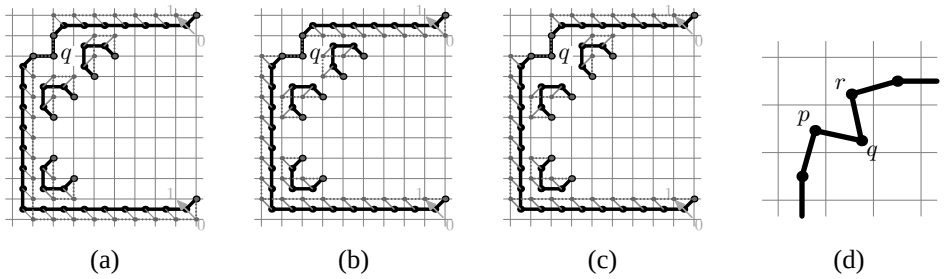


Figure 5. Illustration of a 2-clause gadget: (a) a state when the upper wire is pulled; (b) a state when the lower wire is pulled; (c) a state when both wires are pulled; (d) the real sequence around the point where the two wires meet (upper left corner). The two wires coming out of it (on the right) would end in corresponding variable gadgets (after possible shifts).

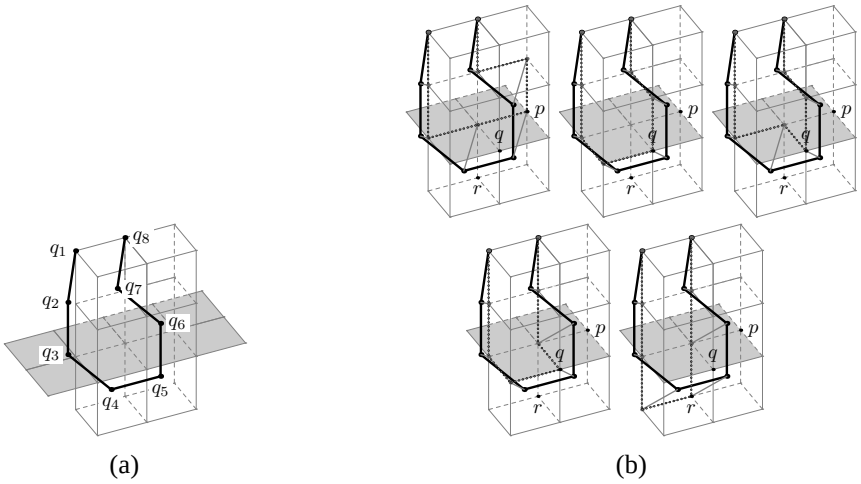


Figure 6. Illustration of a 3D-flipper: (a) a real subsequence; (b) 5 possible states of the 3D-flipper. The grey area indicates the plane $z = 0$ where the ordinary gadgets are placed.

Figure 7(a) shows the unique state of the vertical flipper in the case when the 3D-flipper is in the state using point r. Obviously, in this case points r_1 and r_2, both lying in the plane $z = 0$ are used. In any other state of the 3D-flipper, the vertical flipper can be in either of the two states. Figure 7(b) shows an example, when it is in the state not using points r_1 and r_2. Figure 7(c) shows five possible situations in the plane $z = 0$ caused by the non-planar part of the 3-clause gadget. Observe that for any of the points p, q and r_2, there is a solution such that this point is occupied while the other two are not. Given 3 wires, and using several flippers it is possible to ensure that at least one of the wires is pulled, cf. Figure 7(d). If s_1, s_2, s_3 are boolean variables representing positions of the rightmost flexible points of the three wires, then in any solution, we have $s_1 \vee s_2 \vee s_3$ is satisfiable, and vice versa, for any values of s_1, s_2, s_3 such that $s_1 \vee s_2 \vee s_3$ is satisfied, there is a solution with these values. Consider the case when all the three wires are in state 0, then

the flippers are forced to occupy points p, q, r_2 but at least one of these points is needed by the non-planar part of the 3-clause gadget, a contradiction. All the other states are valid, and again have been checked using a computer program.

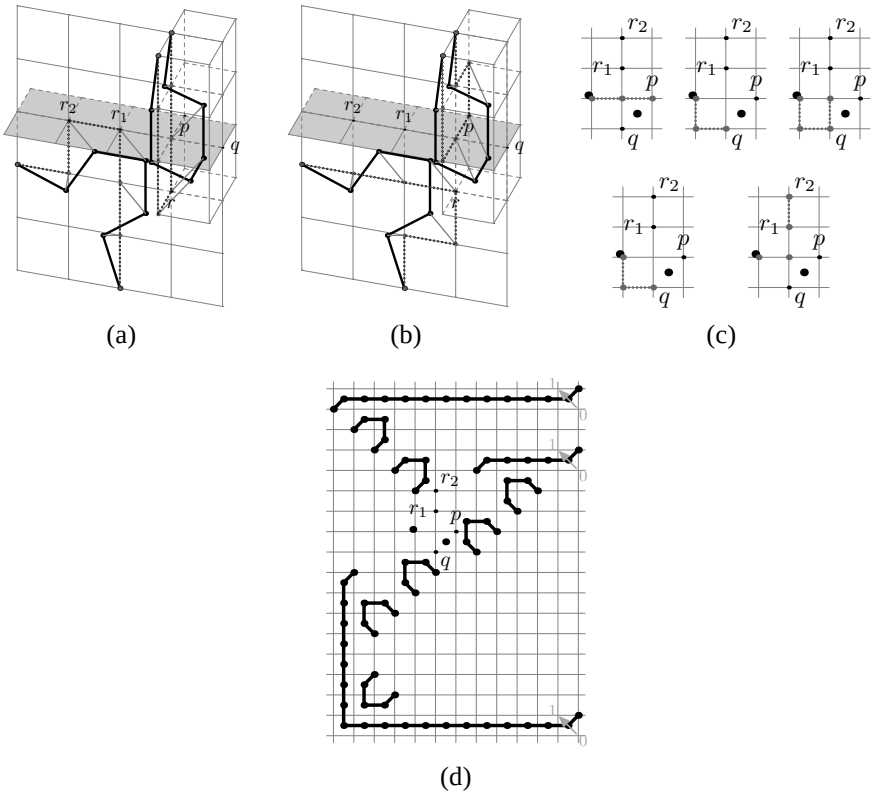


Figure 7. Illustration of a 3-clause gadget: (a–b) the part of the gadget outside of the plane $z = 0$; (c) five possible situations in the plane $z = 0$ caused by the non-planar part of the gadget (two larger black dots mark the places where the subsequence of the 3D-flipper crosses the plane $z = 0$); (d) the planar part of the gadget.

Finally, we construct the variable gadgets. Recall that each $x \in X$ occurs in exactly 3 clauses, twice positive and once negated. Figure 8 shows all the cases (up to symmetries) of how the neighborhood of a variable x occurring in 3 clauses looks like in a planar drawing of G_Φ.

All variable gadgets are variations of the gadget depicted in Figure 9(a). There are 2439 states for this gadget. Let s_1, s_2, s_3, s_4 be boolean variables whose values depend on the positions of flexible points at the end of wires as depicted in the figure. If s_1 is in state 1, the flippers force s_2 into state 0. Symmetrically, if s_4 is in state 1, then the flippers force s_3 to be in state 0. Also note that if s_1 is in state 1 then s_3 is in state 0, and symmetrically if s_2 is in state 1 then s_4 is in state 0. Therefore in any solution for the gadget we have that the formula $(s_1 \vee s_4) \oplus (s_2 \vee s_3)$ is satisfied, and vice versa, for any satisfiable assignment

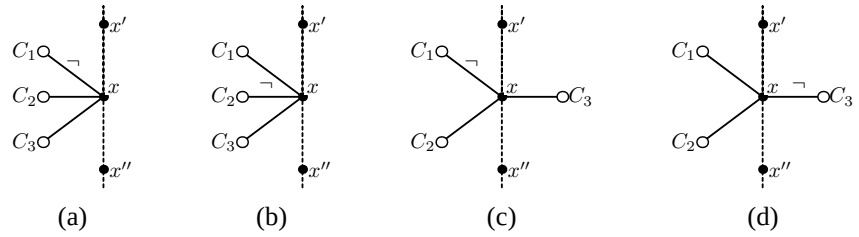


Figure 8. Neighborhood of a variable $x \in X$ in a planar drawing of G_Φ. For negated occurrence we put the symbol ¬ on the edge connecting the clause and x.

of the formula there is a valid state for the gadget. Once again, all the valid states for the variable gadget have been verified using a computer program.

Now, consider the occurrence of some variable x. For every positive occurrence of x in a clause $c \in C$, connect the wire coming out of the gadget of c to the end of wire marked with s_1 or s_4 (depending on whether it is a left or right clause), and for every negative occurrence, to the end marked with s_2 or s_3, cf. Figure 9(b). Obviously, this strategy can be directly applied in the case (c) depicted in Figure 8. Note that one wire coming out of the variable gadget stays unused, and remains unconnected to any clause.

Given a solution to the PCLF problem, we determine value for variable x in a satisfiable assignment as follows: if $s_1 = 1$ or $s_4 = 1$ then $x = 1$; if $s_2 = 1$ or $s_3 = 1$ then $x = 0$; otherwise (all s_1, s_2, s_3, s_4 have 0 values), the value for x can be chosen arbitrarily. Note that this assignment to variables X based on any solution to the PCLF problem, guarantees that every clause is satisfied. Indeed, in every clause gadget atleast one of the wires is in state 1 (pulled) this correponds to a literal in the corresponding clause to be true (by construction).

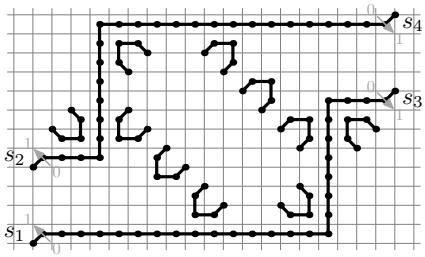
On the other hand, for every satisfiable assignment to variables in X of the formula Φ, set each clause gadget to the state in which they are pulling each wire which corresponds to a literal satisfied by the assignment, and set each variable gadget to the state in which $s_1 = s_4 = 1$ if the corresponding variable x has value 1, or $s_2 = s_3 = 1$ otherwise.

Finally, for the remaining cases of the neighborhood of x, cf. Figure 8(a)(b)(d), we need to bend one wire from the right hand side of the configuration to the left hand side. For the case (a), the complete variable gadget is depicted in Figure 10. For other two cases, the construction is analogous.

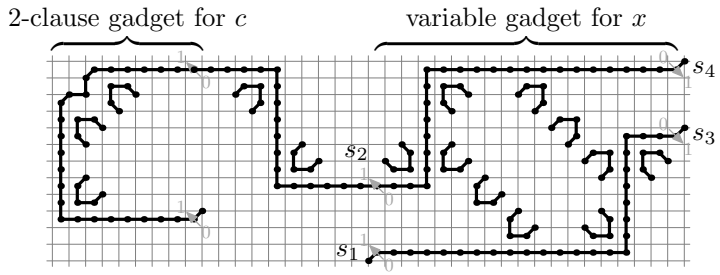
It follows by the construction that the formula Φ is satisfiable if and only if there exists a solution to the constructed PCLF problem. It is also clear that the construction can be done in polynomial time and space. Therefore, the PCLF problem is NP-complete. □

4. Conclusions

We have proved that the protein chain lattice fitting problem is NP-complete for the cubic lattice with side 3.8Å and the coordinate root mean square deviation (c-RMS) as the distance measure. From the theoretical point of view it would be very interesting to further investigate the complexity for the 2D square lattice with side 3.8Å. The proof of NP-

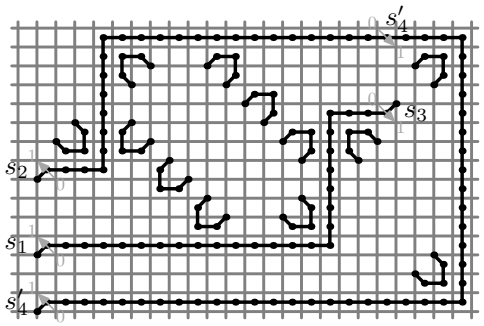


(a)



(b)

Figure 9. (a) A common substructure to all variable gadgets. (b) Example of a connection between a left 2-clause gadget for $c = \neg x \vee y$ and the variable gadget for x.



(a)

Figure 10. A variant of the variable gadget used in the case (a) depicted in Figure 8. Note that if the wire is pulled at position s'_4, it is also pulled at position s_4.

completeness presented in this paper mostly uses one plane of the 3D cubic lattice ($z = 0$) and is based on the planar 3-SAT problem. However, it cannot be applied directly to the square lattice for two reasons: (1) we were unable to design the 3-clause gadget without using the third dimension; (2) connecting the gadgets into one protein string without using the third dimension seems to be a nontrivial task.

From the practical point view, the questions whether our result applies to different types of lattices or cubic lattices with sides different from 3.8Å or d-RMS used as the distance

measure between two 3D structures are more important. It can be shown that a greedy algorithm can perform arbitrary bad for constructed sequences of points (although, the performance on proteins from PDB is better). It would be interesting to study whether the existing DP-based algorithms have bounded performance ratio, or to design a new algorithm with constant performance ratio.

References

1. B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *J. Comp. Biol.*, 5:27–40, 1998.
2. H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The protein data bank. *Nucleic Acids Res*, 28(1):235–242, 2000.
3. D. G. Covell and R. L. Jernigan. Conformations of folded proteins in restricted spaces. *Biochemistry*, 29:3287–3294, 1990.
4. P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. In *Proc. of STOC'98*, pages 597–603, 1998.
5. H. DA and L. M. Exploring conformational space with a simple lattice model for protein structure. *J. Mol. Biol.*, 243(4):668–682, 1994.
6. K. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.
7. K. A. Dill, S. Bromberg, K. Yue, K. M. Fiebig, D. P. Yee, P. D. Thomas, and H. S. Chan. Principles of protein folding: A perspective from simple exact models. *Protein Science*, 4:561–602, 1995.
8. M. Fellows, J. Kratochvíl, M. Middendorf, and F. Pfeiffer. The complexity of induced minors and related problems. *Algorithmica*, 13(3):266–282, 1995.
9. D. Gaur, J. Maňuch, and D. Thomas. Polynomial algorithm for aligning proteins to cubic lattice as a walk (manuscript).
10. A. Godzik, A. Kolinski, and J. Skolnick. Lattice representations of globular proteins: How good are they? *J. Comp. Chem.*, 14:1194–1202, 1993.
11. D. A. Hinds and M. Levitt. A lattice model for protein structure prediction at low resolution. *Proc. Natl. Acad. Sci. USA*, 89:2536–2540, 1992.
12. P. Koehl and M. Delarue. Building protein lattice models using self-consistent mean field theory. *J. Chem. Physics*, 108(22):9540–9549, 1998.
13. D. Lichtenstein. Planar formulae and their uses. *SIAM Journal of Computing*, 11(2):329–343, 1982.
14. C. Mead, J. Maňuch, X. Huang, B. Bhattacharyya, L. Stacho, and A. Gupta. Investigating lattice structure for inverse protein folding (abstract). *FEBS Journal*, 272 (s1):4739_1_380, 2005.
15. K. P and L. M. A brighter future for protein structure prediction. *Nat. Struct. Biol.*, 6(2):108–111, 1999.
16. B. H. Park and M. Levitt. The complexity and accuracy of discrete state models of protein structure. *J. Mol. Biol.*, 249:493–507, 1995.
17. A. A. Rabow and H. A. Sheraga. Improved genetic algorithm for the protein folding problem by use of a Cartesian combination operator. *Protein Science*, 5:1800–1815, 1996.
18. B. A. Reva, D. S. Rykunov, A. J. Olson, and A. V. Finkelstein. Constructing lattice models of protein chains with side groups. *Journal of Comp. Biology*, 2(4):527–535, 1995.
19. D. S. Rykunov, B. A. Reva, and A. V. Finkelstein. Accurate general method for lattice approximation of three-dimensional structure of a chain molecule. *Proteins: Structure, Function and Genetics*, 22:100–109, 1995.
20. Y. Xia, E. Huang, M. Levitt, and R. Samudrala. Ab initio construction of protein tertiary sctructures using a hierarchical approach. *J. Mol. Biol.*, 300:171–185, 2000.