Is protein folding problem really a NP-complete one ? First investigations

Jacques M. Bahi[a], Wojciech Bienia[c], Nathalie Côté[b], Christophe Guyeux[a,1,*]

[a]*FEMTO-ST Institute, UMR 6174 CNRS, University of Franche-Comté, Besançon, France*
[b]*Laboratoire de Biologie du Développement, UMR 7622, Université Pierre et Marie Curie, Paris, France*
[c]*G-SCOP Laboratory, ENSIMAG, 46 av. Félix Viallet, F-38031 Grenoble Cedex 1, France*

Abstract

To determine the 3D conformation of proteins is a necessity to understand their functions or interactions with other molecules. It is commonly admitted that, when proteins fold from their primary linear structures to their final 3D conformations, they tend to choose the ones that minimize their free energy. To find the 3D conformation of a protein knowing its amino acid sequence, bioinformaticians use various models of different resolutions and artificial intelligence tools, as the protein folding prediction problem is a NP complete one. More precisely, to determine the backbone structure of the protein using the low resolution models (2D HP square and 3D HP cubic), by finding the conformation that minimize free energy, is intractable exactly [6]. Both the proof of NP-completeness and the 2D prediction consider that acceptable conformations have to satisfy a self-avoiding walk (SAW) requirement, as two different amino acids cannot occupy a same position in the lattice. It is shown in this document that the SAW requirement considered when proving NP-completeness is different from the SAW requirement used in various prediction programs, and that they are different from the real biological requirement. Indeed, the proof of NP completeness and the predictions *in silico* consider conformations that are not possible in practice. Consequences of this fact are investigated in this research

[*]Corresponding author
[1]Authors in alphabetic order

work.
Keywords: Protein folding problem, Self-Avoiding Walk requirement, NP-completeness, Graph theory, Pivot moves

1. Introduction

Proteins are polymers formed by different kinds of amino acids. During or after proteins have been synthesized by ribosomes, they fold to form a specific tridimensional shape. This 3D geometric pattern defines their biological functionality, properties, and so on. For instance, the hemoglobin is able to carry oxygen to the blood stream thanks to its 3D conformation. However, contrary to the mapping from DNA to the amino acids sequence, the complex folding of this sequence is not yet understood. In fact, Anfinsen's "Thermodynamic Hypothesis" claims that the chosen 3D conformation corresponds to the lowest free energy minimum of the considered protein [2]. Efficient constraint programming methods can solve the problem for reasonably sized sequences [14]. But the conformation that minimizes this free energy is most of the time impossible to find in practice, at least for large proteins, due to the number of possible conformations. Indeed the Protein Structure Prediction (PSP) problem is a NP-complete one [11, 6]. This is why conformations of proteins are *predicted*: the 3D structures that minimize the free energy of the protein under consideration are found by using computational intelligence tools like genetic algorithms [16], ant colonies [26], particle swarm [23], memetic algorithms [19], constraint programming [14, 22], or neural networks [15], etc. These computational intelligence tools are coupled with protein energy models (like AMBER, DISCOVER, or ECEPP/3) to find a conformation that approximately minimize the free energy of a given protein. Furthermore, to face the complexity of the PSP problem, authors who try to predict the protein folding process use models of various resolutions. For instance, in coarse grain, single-bead models, an amino acid is considered as a single bead, or point. These low resolution models are often used as the first stage of the 3D structure prediction: the backbone of the 3D conformation is determined. Then, high resolution models come next for further exploration. Such a prediction strategy is commonly used in PSP softwares like ROSETTA [7, 10] or TASSER [28].

In this paper, which is a supplement of [4, 5], we investigate the 2D HP square lattice model. Let us recall that this popular model is used to test methods and as a first 2D HP lattice folding stage in some protein folding prediction algorithms [16, 9, 20, 27, 18]. It focuses only on hydrophobicity by separating the amino acids into two sets: hydrophobic (H) and hydrophilic (or polar P) [13]. These amino acids occupy vertices of a square lattice, and the 2D low resolution conformation of the given protein is thus represented by a self avoiding walk (SAW) on this lattice. Variations of this model are frequently investigated: 2D or 3D lattices, with square, cubic, triangular, or face-centered-cube shapes. However, at each time, a SAW requirement for the targeted conformation is required. The PSP problem takes place in that context: given a sequence of hydrophobic and hydrophilic amino acids, to find the self avoiding walk on the lattice that maximizes the number of hydrophobic neighbors is a NP complete problem [11].

We will show in this document that this SAW requirement can be understood in various different ways, even in the 2D square lattice model. The first understanding of this requirement in the 2D model, called SAW_1 in the remainder of this paper, has been chosen by authors of [11] when they have established the proof of NP-completeness for the PSP problem. It corresponds to the famous "excluded volume" requirement, and it has been already well-studied by the discrete mathematics community (see, for instance, the book of Madras and Slade [21]). It possesses a dynamical formulation we call it SAW_2 in this document. The SAW_3 set is frequently chosen by bioinformaticians when they try to predict the backbone conformation of proteins using a low resolution model. Finally, the last one proposed here is perhaps the most realistic one, even if it still remains far from the biological folding operation. We will demonstrate that these four sets are not equal. In particular, we will establish that SAW_4 is strictly included into SAW_3, which is strictly included into $SAW_1 = SAW_2$.

So the NP-completeness proof has been realized in a strictly larger set than the one used for prediction, which is strictly larger than the set of biologically possible conformations. Concrete examples of 2D conformations that are in a SAW_i without being in another SAW_j will be given, and characterizations of these sets, in terms of graphs, will finally be proposed.

The remainder of this paper is structured as follows. In the next section we recall some notations and terminologies on the 2D HP square lattice model, chosen here to simplify explanations. In Section 3, the dynamical system used to describe the folding process in the 2D model, initially presented in [4, 5], is recalled. In Sect. 4, various ways to understand the so-called self-avoiding walk (SAW) requirement are detailed. Their relations and inclusions are investigated in the next section. Section 6 presents a graph approach to determine the size ratios between the four SAW requirements defined previously, and the consequences of their strict inclusions are discussed. This paper ends by a conclusion section, in which our contribution is summarized and intended future work is presented.

2. Background

In the sequel S^n denotes the n^{th} term of a sequence S and V_i the i^{th} component of a vector V. The k^{th} composition of a single function f is represented by $f^k = f \circ ... \circ f$. The set of congruence classes modulo 4 is denoted as $\mathbb{Z}/4\mathbb{Z}$. Finally, given two integers $a < b$, the following notation is used: $[\![a;b]\!] = \{a, a+1, \ldots, b\}$.

2.1. 2D Hydrophilic-Hydrophobic (HP) Model

In the HP model, hydrophobic interactions are supposed to dominate protein folding. This model was formerly introduced by Dill, who consider in [13] that the protein core freeing up energy is formed by hydrophobic amino acids, whereas hydrophilic amino acids tend to move in the outer surface due to their affinity with the solvent (see Fig. 1).

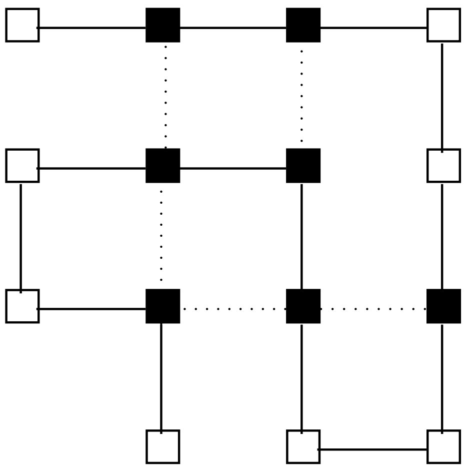


Figure 1: Hydrophilic-hydrophobic model (black squares are hydrophobic residues)

In this model, a protein conformation is a “self-avoiding walk (SAW)”, as the walks studied in [21], on a 2D or 3D lattice such that its energy E, depending on topological neighboring contacts between hydrophobic amino acids that are not contiguous in the primary structure, is minimal. In other words, for an amino-acid sequence P of length N and for the set $\mathcal{C}(P)$ of all SAW conformations of P, the chosen conformation will be $C^* = min\left\{E(C) \mid C \in \mathcal{C}(P)\right\}$ [25]. In that context and for a conformation C, $E(C) = -q$ where q is equal to the number of topological hydrophobic neighbors. For example, $E(c) = -5$ in Fig. 1.

Protein Encoding

Additionally to the direct coordinate presentation in the lattice, at least two other isomorphic encoding strategies for HP models are possible: relative encoding and absolute encoding. In relative encoding [17], the move direction is defined relative to the direction of the previous move (forward, backward, left, or right). Alternatively, in absolute encoding [3], which is the encoding chosen in this paper, the direct coordinate presentation is replaced by letters or numbers representing directions with respect to the lattice structure.

For absolute encoding in the 2D square lattice, the permitted moves are: east $\rightarrow$ (denoted by 0), south $\downarrow$ (1), west $\leftarrow$ (2), and north $\uparrow$ (3). A 2D conformation C of $\mathsf{N}+1$ residues for a protein P is then an element C of $\mathbb{Z}/4\mathbb{Z}^{\mathsf{N}}$, with a first component equal to 0 (east) [17]. For instance, in Fig. 1, the 2D absolute encoding is 00011123322101 (starting from the upper left corner), whereas 001232 corresponds to the following path in the square lattice: (0,0), (1,0), (2,0), (2,-1), (1,-1), (1,0), (0,0). In that situation, at most 4^{N} conformations are possible when considering $\mathsf{N}+1$ residues, even if some of them invalidate the SAW requirement as defined in [21].

3. A Dynamical System for the 2D HP Square Lattice Model

Protein minimum energy structure can be considered statistically or dynamically. In the latter case, one speaks in this article of "protein folding". We recall here how to model the folding process in the 2D model, or pivot moves, as a dynamical system. Readers are referred to [4, 5] for further explanations and to investigate the dynamical behavior of the proteins pivot moves in this 2D model (it is indeed proven to be chaotic, as defined by Devaney [12]).

3.1. Initial Premises

Let us start with preliminaries introducing some concepts that will be useful in our approach.

The primary structure of a given protein P with $\mathsf{N}+1$ residues is coded by $00\ldots0$ (N times) in absolute encoding. Its final 2D conformation has an absolute encoding equal to $0C_1^*\ldots C_{\mathsf{N}-1}^*$, where $\forall i, C_i^* \in \mathbb{Z}/4\mathbb{Z}$, is such that $E(C^*) = min\left\{E(C)/C \in \mathcal{C}(P)\right\}$. This final conformation depends on the repartition of hydrophilic and hydrophobic amino acids in the initial sequence.

Moreover, we suppose that, if the residue number $n+1$ is at the east of the residue number n in absolute encoding ($\rightarrow$) and if a fold (pivot move) occurs after n, then the east move can only by changed into north ($\uparrow$) or south ($\downarrow$). That means, in our simplistic model, only rotations or pivot moves of $+\frac{\pi}{2}$ or $-\frac{\pi}{2}$ are possible.

Consequently, for a given residue that has to be updated, only one of the two possibilities below can appear for its absolute encoding during a pivot move:

- $0 \longmapsto 1$ (that is, east becomes north), $1 \longmapsto 2, 2 \longmapsto 3$, or $3 \longmapsto 0$ for a pivot move in the clockwise direction, or
- $1 \longmapsto 0, 2 \longmapsto 1, 3 \longmapsto 2$, or $0 \longmapsto 3$ for an anticlockwise.

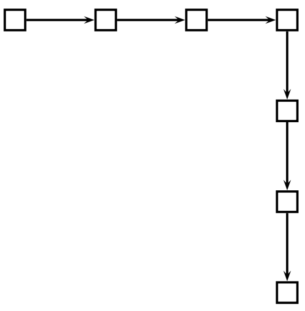
This fact leads to the following definition:

Definition 1 The *clockwise fold function* is the function $f : \mathbb{Z}/4\mathbb{Z} \longrightarrow \mathbb{Z}/4\mathbb{Z}$ defined by $f(x) = x + 1 \pmod 4$.

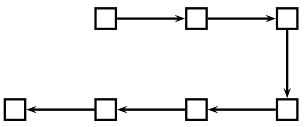
Obviously the anticlockwise fold function is $f^{-1}(x) = x - 1 \pmod 4$.

Thus at the n^{th} folding time, a residue k is chosen and its absolute move is changed by using either f or f^{-1}. As a consequence, *all of the absolute moves must be updated from the coordinate k until the last one N by using the same folding function.*

Example 1 If the current conformation is $C = 000111$, i.e.,



and if the third residue is chosen to fold (pivot move) by a rotation of $-\frac{\pi}{2}$ (mapping f), the new conformation will be $(C_1, C_2, f(C_3), f(C_4), f(C_5), f(C_6))$, which is $(0, 0, 1, 2, 2, 2)$. That is,



These considerations lead to the formalization described thereafter.

3.2. Formalization and Notations

Let $\mathsf{N} + 1$ be a fixed number of amino acids, where $\mathsf{N} \in \mathbb{N}^* = \{1, 2, 3, \ldots\}$. We define

$$\check{\mathcal{X}} = \mathbb{Z}/4\mathbb{Z}^{\mathsf{N}} \times [\![-\mathsf{N}; \mathsf{N}]\!]^{\mathbb{N}}$$

as the phase space of all possible folding processes. An element $X = (C, F)$ of this dynamical folding space is constituted by:

- A conformation of the $\mathsf{N}+1$ residues in absolute encoding: $C = (C_1, \ldots, C_{\mathsf{N}}) \in \mathbb{Z}/4\mathbb{Z}^{\mathsf{N}}$. Note that we do not require self-avoiding walks here.
- A sequence $F \in [\![-\mathsf{N}; \mathsf{N}]\!]^{\mathbb{N}}$ of future pivot moves such that, when $F_i \in [\![-\mathsf{N}; \mathsf{N}]\!]$ is k, it means that it occurs:
  - a pivot move after the $k-$th residue by a rotation of $-\frac{\pi}{2}$ (mapping f) at the $i-$th step, if $k = F_i > 0$,
  - no fold at time i if $k = 0$,
  - a pivot move after the $|k|-$th residue by a rotation of $\frac{\pi}{2}$ (*i.e.*, f^{-1}) at the $i-$th time, if $k < 0$.

On this phase space, the protein folding dynamic in the 2D model can be formalized as follows.

Denote by i the map that transforms a folding sequence in its first term (*i.e.*, in the first folding operation):

$$\begin{array}{rccc} i: & [\![-\mathsf{N}; \mathsf{N}]\!]^{\mathbb{N}} & \longrightarrow & [\![-\mathsf{N}; \mathsf{N}]\!] \\ & F & \longmapsto & F^0, \end{array}$$

by σ the shift function over $[\![-\mathsf{N}; \mathsf{N}]\!]^{\mathbb{N}}$, that is to say,

$$\begin{array}{rccc} \sigma: & [\![-\mathsf{N}; \mathsf{N}]\!]^{\mathbb{N}} & \longrightarrow & [\![-\mathsf{N}; \mathsf{N}]\!]^{\mathbb{N}} \\ & \left(F^k\right)_{k\in\mathbb{N}} & \longmapsto & \left(F^{k+1}\right)_{k\in\mathbb{N}}, \end{array}$$

and by *sign* the function:

$$sign(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{else.} \end{cases}$$

Remark that the shift function removes the first folding operation (a pivot move) from the folding sequence F once it has been achieved.

Consider now the map $G : \check{\mathcal{X}} \rightarrow \check{\mathcal{X}}$ defined by:

$$G\left((C,F)\right) = \left(f_{i(F)}(C), \sigma(F)\right),$$

where $\forall k \in [\![-\mathsf{N};\mathsf{N}]\!]$, $f_k : \mathbb{Z}/4\mathbb{Z}^{\mathsf{N}} \rightarrow \mathbb{Z}/4\mathbb{Z}^{\mathsf{N}}$ is defined by: $f_k(C_1,\ldots,C_{\mathsf{N}}) = (C_1,\ldots,C_{|k|-1}, f^{sign(k)}(C_{|k|}),\ldots,f^{sign(k)}(C_{\mathsf{N}}))$. Thus the folding process of a protein P in the 2D HP square lattice model, with initial conformation equal to $(0,0,\ldots,0)$ in absolute encoding and a folding sequence equal to $(F^i)_{i\in\mathbb{N}}$, is defined by the following dynamical system over $\check{\mathcal{X}}$:

$$\begin{cases} X^0 = ((0,0,\ldots,0),F) \\ X^{n+1} = G(X^n), \forall n \in \mathbb{N}. \end{cases}$$

In other words, at each step n, if $X^n = (C,F)$, we take the first folding operation to realize, that is $i(F) = F^0 \in [\![-\mathsf{N};\mathsf{N}]\!]$, we update the current conformation C by rotating all of the residues coming after the $|i(F)|-$th one, which means that we replace the conformation C with $f_{i(F)}(C)$. Lastly, we remove this rotation (the first term F^0) from the folding sequence F: F becomes $\sigma(F)$.

Example 2 Let us reconsider Example 1. The unique iteration of this folding process transforms a point of $\check{X}$ having the form $\left((0,0,0,1,1,1);(3,F^1,F^2,\ldots)\right)$ in $G\left((0,0,0,1,1,1),(+3,F^1,F^2,\ldots)\right)$, which is equal to $\left((0,0,1,2,2,2),(F^1,F^2,\ldots)\right)$.

Remark 1 Such a formalization allows the study of proteins that never stop to fold, for instance due to never-ending interactions with the environment.

Remark 2 A protein P that has finished to fold, if such a protein exists, has the form $(C,(0,0,0,\ldots))$, where C is the final 2D structure of P. In this case, we can assimilate a folding sequence that is convergent to 0, *i.e.*, of the form $(F^0,\ldots,F^n,0\ldots)$, with the finite sequence $(F^0,\ldots,F^n)$.

We will now introduce the SAW requirement in our formulation of the folding process in the 2D model.

4. The SAW Requirement

4.1. The paths without crossing

Let $\mathcal{P}$ denotes the 2D plane,

$$\begin{array}{cccc} p: & \mathbb{Z}/4\mathbb{Z}^{\mathsf{N}} & \rightarrow & \mathcal{P}^{\mathsf{N}+1} \\ & (C_1,\ldots,C_{\mathsf{N}}) & \mapsto & (X_0,\ldots,X_{\mathsf{N}}) \end{array}$$

where $X_0 = (0,0)$, and

$$X_{i+1} = \begin{cases} X_i + (1,0) & \text{if } c_i = 0, \\ X_i + (0,-1) & \text{if } c_i = 1, \\ X_i + (-1,0) & \text{if } c_i = 2, \\ X_i + (0,1) & \text{if } c_i = 3. \end{cases}$$

The map p transforms an absolute encoding in its 2D representation. For instance, $p((0,0,0,1,1,1))$ is ((0,0);(1,0);(2,0);(3,0);(3,-1);(3,-2);(3,-3)), that is, the first figure of Example 1.

Now, for each $(P_0, \ldots, P_\mathsf{N})$ of $\mathcal{P}^{\mathsf{N}+1}$, we denote by

$$\textit{support}((P_0, \ldots, P_\mathsf{N}))$$

the set (without repetition): $\{P_0, \ldots, P_\mathsf{N}\}$. For instance,

$$\textit{support}\,((0,0);(0,1);(0,0);(0,1)) = \{(0,0);(0,1)\}\,.$$

Then,

Definition 2 A conformation $(C_1, \ldots, C_\mathsf{N}) \in \mathbb{Z}/4\mathbb{Z}^\mathsf{N}$ is a *path without crossing* iff the cardinality of $\textit{support}(p((C_1, \ldots, C_\mathsf{N})))$ is $\mathsf{N}+1$.

This path without crossing is sometimes referred as "excluded volume" requirement in the literature. It only means that no vertex can be occupied by more than one protein monomer. We can finally remark that Definition 2 concerns only one conformation, and not a *sequence* of conformations that occurs in a folding process.

4.2. Defining the SAW Requirements in the 2D model

The next stage in the formalization of the protein folding process in the 2D model as a dynamical system is to take into account the self-avoiding walk (SAW) requirement, by restricting the set $\mathbb{Z}/4\mathbb{Z}^\mathsf{N}$ of all possible conformations to one of its subset. That is, to define precisely the set $\mathcal{C}(P)$ of acceptable conformations of a protein P having $\mathsf{N}+1$ residues. This stage needs a clear definition of the SAW requirement. However, as stated above, Definition 2 only focus on a given conformation, but not on a complete folding process. In our opinion, this requirement applied to the whole folding process can be understood at least in four ways.

In the first and least restrictive approach, we call it "SAW_1", we only require that the studied conformation satisfies the Definition 2.

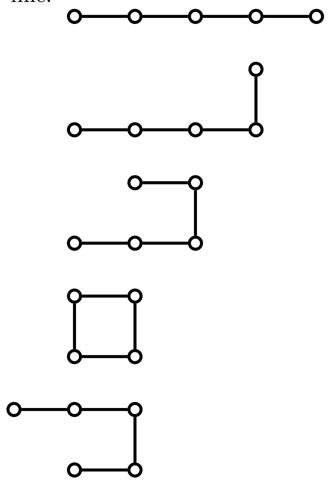
Definition 3 (SAW_1) A conformation c of $\mathbb{Z}/4\mathbb{Z}^{\mathsf{N}}$ satisfies the first self-avoiding walk requirement ($c \in SAW_1(\mathsf{N})$) if this conformation is a path without crossing.

It is not regarded whether this conformation is the result of a folding process that has started from $(0, 0, \ldots, 0)$. Such a SAW requirement has been chosen by authors of [11] when they have proven the NP-completeness of the PSP problem. It is usually the SAW requirement of biomathematicians, corresponding to the self-avoiding walks studied in the book of Madras and Slade [21]. It is easy to convince ourselves that conformations of SAW_1 are the conformations that can be obtained by any chain growth algorithm, like in [8].

As stated before, protein minimum energy structure can be considered statically or dynamically. In the latter case, we speak here of "protein folding", since this concerns the dynamic process of folding. When folding on a lattice model, there is an underlying algorithm, such as Monte Carlo or genetic algorithm, and an allowed move set. In the following, for the sake of simplicity, only pivot moves are investigated, but the corner and crankshaft moves should be further investigated [24].

Basically, in the protein folding literature, there are methods that require the "excluded volume" condition during the dynamic folding procedure, and those that do not require this condition. This is why the second proposed approach called SAW_2 requires that, starting from the initial condition $(0, 0, \ldots, 0)$, we obtain by a succession of pivot moves a final conformation being a path without crossing. In other words, we want that the final tree corresponding to the true 2D conformation has 2 vertices with 1 edge and $\mathsf{N} - 2$ vertices with 2 edges. For instance, the folding process of Figure 2 is acceptable in SAW_2, even if it presents a cross in an intermediate conformation. Such an approach corresponds

Figure 2: Folding process acceptable in SAW_2 but not in SAW_3. The folding sequence (-4,-3,-2,+4), having 3 anticlockwise and 1 clockwise pivot moves, is applied here on the conformation 0000 represented as the upper line.



to programs that start from the initial conformation $(0, 0, \ldots, 0)$, fold it several times according to their embedding functions, and then obtain a final conformation on which the SAW property is checked: only the last conformation has to satisfy the Definition 2. More precisely,

Definition 4 ($\boldsymbol{SAW_2}$) A conformation c of $\mathbb{Z}/4\mathbb{Z}^{\mathsf{N}}$ satisfies the second self-avoiding walk requirement SAW_2 if $c \in SAW_1(\mathsf{N})$ and a finite sequence $(F^1, F^2, \ldots, F^n)$ of $[\![-\mathsf{N}, \mathsf{N}]\!]$ can be found such that

$$(c, (0, 0, \ldots)) = G^n \left((0, 0, \ldots, 0), \left(F^1, F^2, \ldots, F^n, 0, \ldots\right)\right).$$

$SAW_2(\mathsf{N})$ will denote the set of all conformations satisfying this requirement.

In the next approach, namely the SAW_3 requirement, it is demanded that each intermediate conformation, between the initial one and the returned (final) one, satisfies the Definition 2. It restricts the set of all conformations $\mathbb{Z}/4\mathbb{Z}^{\mathsf{N}}$, for a given N, to the subset $\mathfrak{C}_{\mathsf{N}}$ of conformations $(C_1, \ldots, C_{\mathsf{N}})$ such that $\exists n \in \mathbb{N}^*$, $\exists k_1, \ldots, k_n \in [\![-\mathsf{N}; \mathsf{N}]\!]$,

$$(C_1, \ldots, C_{\mathsf{N}}) = G^n \left((0, 0, \ldots, 0); (k_1, \ldots, k_n)\right)$$

and $\forall i \leqslant n$, the conformation $G^i \left((0, \ldots, 0); (k_1, \ldots, k_n)\right)$ is a path without crossing. Let us define it,

Definition 5 ($\boldsymbol{SAW_3}$) A conformation c of $\mathbb{Z}/4\mathbb{Z}^{\mathsf{N}}$ satisfies the third self-avoiding walk requirement if $c \in SAW_1(\mathsf{N})$ and a finite sequence $(F^1, F^2, \ldots, F^n)$ of $[\![-\mathsf{N}, \mathsf{N}]\!]$ can be found such that:

- $\forall k \in [\![1, n]\!]$, the conformation c_k of $G^k\left((0,0,\ldots,0), \left(F^1, F^2, \ldots, F^n, 0, \ldots\right)\right)$ is in $SAW_1(\mathsf{N})$, that is, it is a path without crossing.
- $(c, (0,0,\ldots)) = G^n\left((0,0,\ldots,0), \left(F^1, F^2, \ldots, F^n, 0, \ldots\right)\right)$.

$SAW_3(\mathsf{N})$ will denote the set of all conformations satisfying this requirement.

The "SAW requirement" in the bioinformatics literature refers either to the SAW_2 or to the SAW_3 folding process requirement [9, 16, 18]. For instance in [20], random sequences of $[\![0, 3]\!]$ are picked and the excluded volume requirement (as recalled previously, no vertex can be occupied by more than one protein monomer) is then checked, meaning that this research work takes place into SAW_2. Contrarily, in [27], the Monte Carlo search for folding simulations algorithm repeats the step: "from conformation S_i with energy E_i make a pivot move to get S_j with E_j" until S_j is valid, so Unger and Moult are in SAW_3. Algorithms that refine progressively their solutions (following a genetic algorithm or a swarm particle approach for instance) are often of this kind. In these SAW_3 related approaches, the acceptable conformations are obtained starting from the initial conformation $(0, 0, \ldots, 0)$ and are such that all the intermediate conformations satisfy the Definition 2.

Finally, the SAW_4 approach is a SAW_3 requirement in which there is no intersection of vertex or edge during the transformation of one conformation to another. For instance, the transformation of Figure 3 is authorized in the SAW_3 approach but refused in the SAW_4 one: during the rotation around the residue having a cross, the structure after this residue will intersect the remainder of the "protein". In this last approach it is impossible, for a protein folding from one plane conformation to another plane one, to use the whole space to achieve this folding.

This last requirement is the closest approach of a true natural protein folding. It is related to researches that consider more complicated moves than the simple pivot move [24].

Figure 3: Folding process acceptable in SAW_3 but not in SAW_4. It is in SAW_3 as 33330011111033333322221111111100333 (the right panel) is 3333001111103333332222111111$f^{-1}(1)f^{-1}(1)f^{-1}(0)f^{-1}(0)f^{-1}(0)f^{-1}(0)$, which corresponds to a clockwise pivot move of residue number 28 in SAW_3. Figure 4 explains why this folding process is not acceptable in SAW_4.

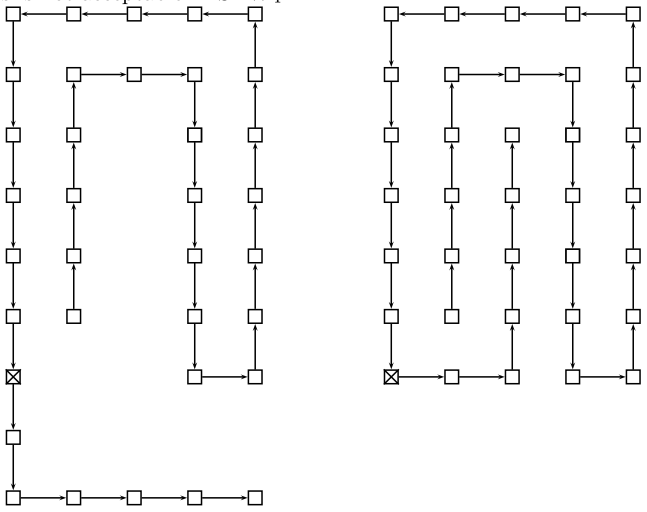
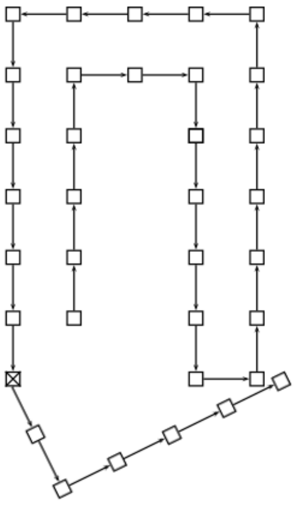


Figure 4: It is impossible to make the rotation around the crossed square, in such a way that the tail does not intersect the head structure during the rotation, so the folding process of Fig. 3 is not in SAW_4.



5. Relations between the SAW requirements

For $i \in \{1,2,3,4\}$, the set $\bigcup_{n \in \mathbb{N}^*} SAW_i(n)$ will be simply written SAW_i. The following inclusions hold obviously:

$$SAW_4 \subseteq SAW_3 \subseteq SAW_2 \subseteq SAW_1$$

due to the definitions of the SAW requirements presented in the previous section. Additionally, Figure 3 shows that $SAW_4 \neq SAW_3$, thus we have,

Proposition 1 $SAW_4 \subsetneq SAW_3 \subseteq SAW_2 \subseteq SAW_1$.

Let us investigate more precisely the links between SAW_1, SAW_2, and SAW_3.

5.1. SAW_1 is SAW_2

Let us now prove that,

Proposition 2 $\forall n \in \mathbb{N}, SAW_1(n) = SAW_2(n)$.

PROOF We need to prove that $SAW_1(n) \subset SAW_2(n)$, *i.e.*, that any conformation of $SAW_1(n)$ can be obtained from $(0,0,..,0)$ by operating a sequence of (anti)clockwise pivot moves.

Obviously, to start from the conformation $(0,0,..,0)$ is equivalent than to start with the conformation $(c,c,...,c)$, where $c \in \{0,1,2,3\}$. Thus the initial configuration is characterized by the absence of a change in the values (the initial sequence is a constant one).

We will now prove the result by a mathematical induction on the number k of changes in the sequence.

- The base case is obvious, as the 4 conformations with no change are in $SAW_1(n) \cap SAW_2(n)$.
- Let us suppose that the statement holds for some $k \geqslant 1$. Let $c = (c_1, c_2, ..., c_n)$ be a conformation having exactly k changes, that is, the cardinality of the set $D(c) = \left\{i \in [\![1, n-1]\!] / c_{i+1} \neq c_i\right\}$ is k. Let us denote by $p(c)$ the first change in this sequence: $p(c) = min\{D(c)\}$. We can apply the folding operation that suppress the difference between $c_{p(c)}$ and $c_{p(c)+1}$. For instance, if $c_{p(c)+1} = c_{p(c)} - 1$ $(mod\ 4)$, then a clockwise pivot move on position $c_{p(c)+1}$ will remove this difference. So the conformation $c' = \left(c_1, c_2, \ldots, c_{p(c)}, f\left(c_{p(c)+1}\right), \ldots, f(c_n)\right)$ has $k-1$ changes. By the induction hypothesis, c' can be obtained from $(j, j, j, \ldots, j)$, where $j \in \{0,1,2,3\}$ by a succession of clockwise and anticlockwise pivot move. We can conclude that it is the case for c too.

Indeed the notion of “pivot moves” is well-known in the literature on protein folding. It was already supposed that pivot moves provide an ergodic move set, meaning that by a sequence of pivot moves one can transform any conformation into any other conformation, when only requiring that the ending conformation satisfies the excluded volume requirement. The contribution of this section is simply a rigorous proof of such an assumption.

5.2. SAW_2 is not SAW_3

To determine whether SAW_2 is equal to SAW_3, we have firstly followed a computational approach, by using the Python language. A first function (the function *conformations* of Listing 1 in the appendix) has been written to return the list of all possible conformations, even if they are not paths without crossing. In other words, this function produces all sequences of compass directions of length n (thus $conformations(n) = \mathbb{Z}/4\mathbb{Z}^n$). Then a generator *saw1_conformations(n)* has been constructed, making it possible to obtain all the SAW_1 conformations (see Listing 2). It is based on the fact that such a conformation of length n must have a support of cardinality equal to n.

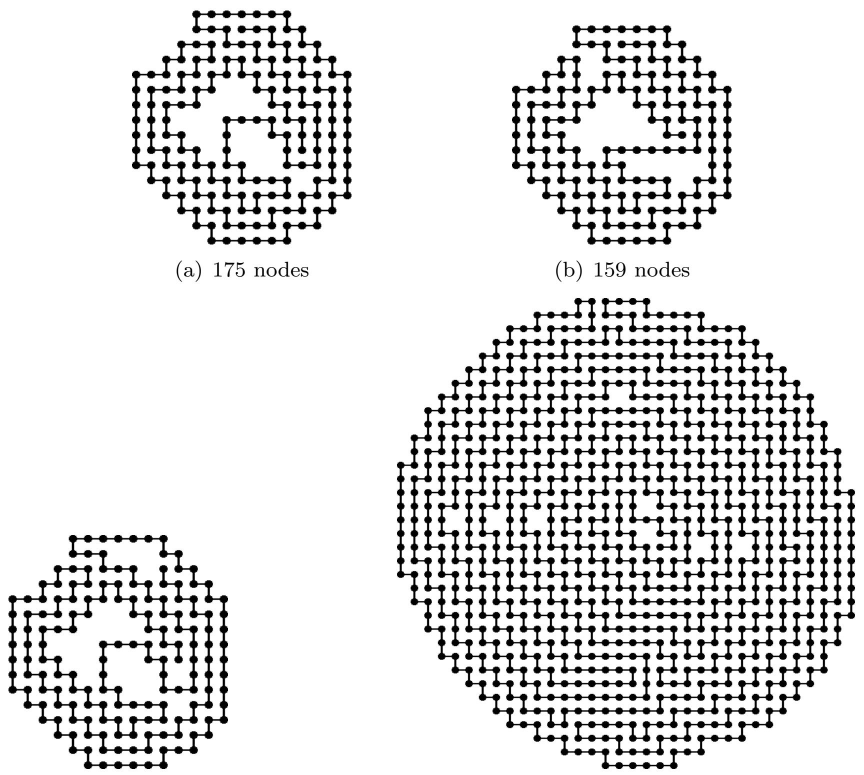
Finally, a program (Algorithm 3) has been written to check experimentally whether an element of $SAW_1 = SAW_2$ is in $SAW3$. This is a systematic approach: for each residue of the candidate conformation, we try to make a clockwise pivot move and an anticlockwise one. If the obtained conformation is a path without crossing then the candidate is rejected. On the contrary, if it is never possible to unfold the protein, whatever the considered residue, then the candidate is in SAW_2 without being in SAW_3.

Figure 5 gives four examples of conformations that are in SAW_2 without being in SAW_3 (the unique ones authors have found via the programs given in the appendix). These counterexamples prove that,

Proposition 3 $\exists n \in \mathbb{N}^*, SAW_2(n) \neq SAW_3(n)$.

5.2.1. Consequences of the strict inclusion

Proposition 1 can be rewritten as follows,



(a) 175 nodes

(b) 159 nodes

(c) 169 nodes

(d) 914 nodes

Figure 5: Examples of conformations in SAW_2 without being in SAW_3

Proposition 4 $SAW_4 \subsetneq SAW_3 \subsetneq SAW_2 = SAW_1$.

As stated previously, the NP-completeness holds for SAW_1. However SAW_1 is a strictly larger set than SAW_3. SAW_3 is a set frequently used for protein structure prediction. As SAW_3 is strictly smaller than SAW_1, it is not sure that the considered problem still remains a NP complete one. Incidentally, it is not clear that only prediction is possible. Indeed, proteins have “only” tens to thousands amino acids. If SAW_3 is very small compared to SAW_1, then perhaps exact methods as SAT solvers can be more widely considered ?

Moreover, SAW_3 is strictly larger than SAW_4, which is a 2D model slightly closer than true real protein folding. This strict inclusion reinforces the fact that the NP-completeness statement must be regarded another time, to determine if this prediction problem is indeed a NP-complete one or not. Furthermore, prediction tools could reduce the set of all possibilities by taking place into SAW_4 instead of SAW_3, thus improving the confidence put in the returned conformations.

All of these questionings are strongly linked to the size ratio between each SAW_i: the probability the NP-completeness proof remains valid in SAW_3 or SAW_4 decreases when these ratios increase. This is why we will investigate more deeply, in the next section, the relation between SAW_2 and SAW_3

6. A Graph Approach of the SAW_i Ratios Problem

Let us denote by $\mathfrak{G}_0(n)$ the directed graph having 4^n vertices, such that:

- these vertices are elements of $[\![0,3]\!]^n$,
- there is a directed edge from the vertex $(e_1,\ldots,e_n)$ to the vertex $(f_1,\ldots,f_n)$ if and only if $\exists k \in [\![1,n]\!]$ and $\exists i \in \{-1,1\}$ such that $(f_1,\ldots,f_n)$ is equal to:
  - either $(e_1,\ldots,e_k,e_k+1 \pmod 4,\ldots,e_n+1 \pmod 4)$
  - or $(e_1,\ldots,e_k,e_k-1 \pmod 4,\ldots,e_n-1 \pmod 4)$.

Obviously, in $\mathfrak{G}_0(n)$, if there is a directed edge from the vertex i to the vertex j, then there is another edge from j to i too. Such a graph is depicted in Fig. 6, in which some edges are dotted to represent the fact that this graph is as a torus: we can go from the vertex 22 to the vertex 33 for instance. The rule of construction of this graph is detailed in Figure 7.

Let us now define another digraph as follows. $\mathfrak{G}(n)$ is the subgraph of $\mathfrak{G}_0(n)$ obtained by removing the vertices that do not correspond to a "path without crossing" according to Madras and Slade [21]. In other words, we remove from $\mathfrak{G}_0(n)$ vertices that do not satisfy the $SAW_1(n)$ requirement. For instance, the digraph $\mathfrak{G}(2)$ associated to $\mathfrak{G}_0(2)$ is depicted in Fig. 9, whereas Figure 10 contains both $\mathfrak{G}(3)$ and the removed vertices. Its construction rules are recalled in Fig. 8.

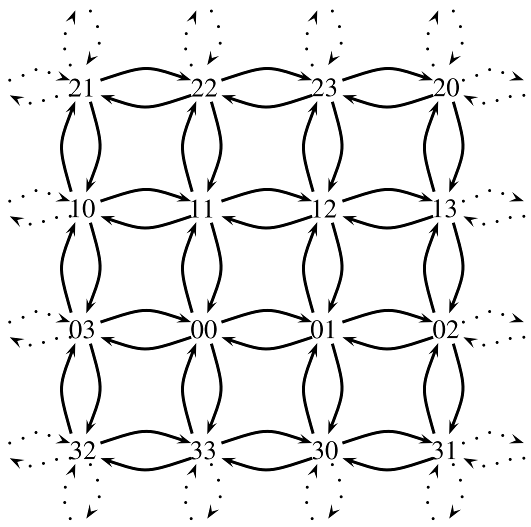


Figure 6: The digraph $\mathfrak{G}_0(2)$

$$\begin{tikzcd}
 & ((i+1) \bmod 4, (j+1) \bmod 4) & \\
(i,(j-1) \bmod 4) & (i,j) \arrow[l] \arrow[r] \arrow[u] \arrow[d] & (i,(j+1) \bmod 4) \\
 & ((i-1) \bmod 4, (j-1) \bmod 4) &
\end{tikzcd}$$

Figure 7: Rules of $\mathfrak{G}_0(2)$

$$\begin{tikzcd}
 & ((i+1) \bmod 4, (j+1) \bmod 4, (k+1) \bmod 4) & \\
 & & (i, (j+1) \bmod 4, (k+1) \bmod 4) \\
(i,j,(k-1) \bmod 4) & (i,j,k) \arrow[l] \arrow[r] \arrow[uu] \arrow[dd] \arrow[ur] \arrow[dl] & (i,j,(k+1) \bmod 4) \\
(i, (j-1) \bmod 4, (k-1) \bmod 4) & & \\
 & ((i-1) \bmod 4, (j-1) \bmod 4, (k-1) \bmod 4) &
\end{tikzcd}$$

Figure 8: Rules of $\mathfrak{G}_0(3)$

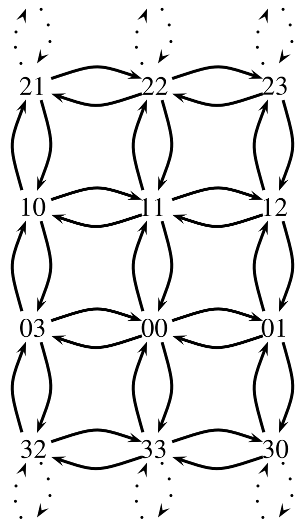


Figure 9: The digraph $\mathfrak{G}(2)$

The links between $\mathfrak{G}(n)$ and the SAW requirements can be summarized as follows:

- The vertices of the graph $\mathfrak{G}_0(n)$ represent all the possible walks of length n in the 2D square lattice.
- The vertices that are preserved in $\mathfrak{G}(n)$ are the conformations of $SAW_1(n) = SAW_2(n)$.
- Two adjacent vertices i and j in $\mathfrak{G}(n)$ are such that it is possible to change the conformation i in j in only one pivot move.
- Finally, a conformation of $SAW_3(n)$ is a vertex of $\mathfrak{G}(n)$ that is reachable from the vertex $000\ldots0$ by following a path in $\mathfrak{G}(n)$.

For instance, the conformation $(2, 2, 3)$ is in $SAW_3(3)$ because we can find a walk from 000 to 223 in $\mathfrak{G}(n)$. The following result is obvious,

Theorem 1 *$SAW_3(n)$ corresponds to the connected component of $000\ldots0$ in $\mathfrak{G}(n)$, whereas $SAW_2(n)$ is the set of vertices of $\mathfrak{G}(n)$. Thus we have:*

$$SAW_2(n) = SAW_3(n) \Longleftrightarrow \mathfrak{G}(n) \textit{ is (strongly) connected.}$$

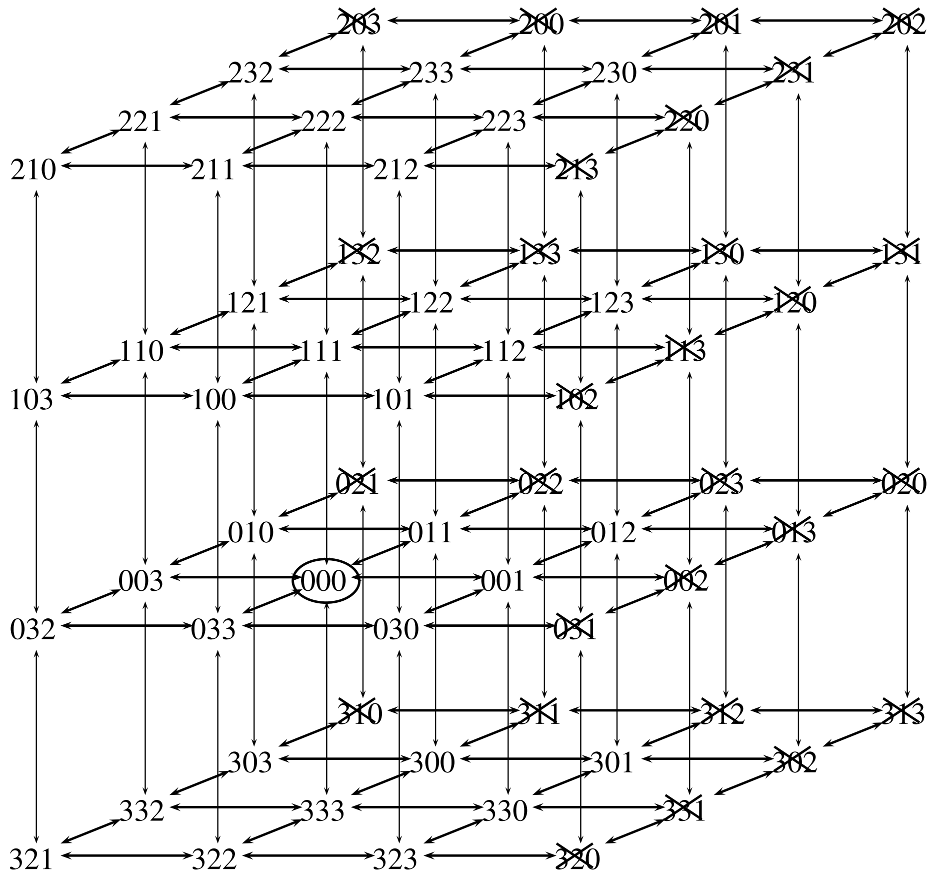


Figure 10: The digraph $\mathfrak{G}(3)$

The previous section shows that the connected component of $000\dots0$ in $\mathfrak{G}(158)$, $\mathfrak{G}(168)$, $\mathfrak{G}(175)$, and $\mathfrak{G}(914)$ are not equal to $\mathfrak{G}(158)$, $\mathfrak{G}(168)$, $\mathfrak{G}(175)$, and $\mathfrak{G}(914)$ respectively. In other words, these graphs are not connected ones.

Indeed, being able to make one pivot move in a given conformation of size n is equivalent to make a move from one edge to another adjacent one in the graph $\mathfrak{G}(n)$. The set of all conformations that are attainable from a given conformation c by a succession of folding processes are thus exactly the connected component of c. This is why the elements of SAW_3 are exactly the connected component of the origin $000\dots00$.

Furthermore, the program described in Section 5.2 is only able to find connected components reduced to one vertex. Obviously, it should be possible to find larger connected components that have not the origin in their set of connected vertices. These vertices are the conformations that are in $SAW_2 \setminus SAW_3$. In other words, if $\dot{c}$ is the connected component of c,

$$SAW_2(n) \setminus SAW_3(n) = \{c \in \mathfrak{G}(n) \text{ s.t. } 000\ldots00 \notin \dot{c}\} \tag{1}$$

Such components are composed by conformations that can be folded several times, but that are not able to be transformed into the line $0000\ldots00$. These programs presented previously are thus only able to determine conformations in the set

$$\{c \in \mathfrak{G}(n) \text{ s.t. } 000\ldots00 \notin \dot{c} \text{ and cardinality of } \dot{c} \text{ is } 1\} \tag{2}$$

which is certainly strictly included into $SAW_2(n) \setminus SAW_3(n)$. The authors' intention is to improve these programs in a future work, in order to determine if the connected component of a given vertex contains the origin or not. The problem making it difficult to obtain such components is the construction of $\mathfrak{G}(n)$. Until now, we:

- list the 4^n possible walks;
- define nodes of the graph from this list, by testing if the walk is a path without crossing;
- for each node of the graph, we obtain the list of its $2 \times n$ possible neighbors;
- an edge between the considered vertex and one of its possible neighbors is added if and only if this neighbor is a path without crossing.

Then we compare the size of the connected component of the origin to the number of vertices into the graph (this latter is indeed the number of n-step self-avoiding walks on square lattice as defined in Madras and Slane, that corresponds to the Sloane's A001411 highly non-trival sequence; it is known that there are α^n self avoiding walks, with upper and lower bounds on the value α).

n	Size of the connected comp. of $00\dots0$	Nodes in $\mathfrak{S}(n)$	Nodes in $\mathfrak{S}_0(n)$
1	4	4	4
2	12	12	16
3	36	36	64
4	100	100	256
5	284	284	1024
6	780	780	4096
7	2172	2172	16384
8	5916	5916	65536
9	16268	16268	262144
10	44100	44100	10485576
11	120292	120292	4194304

Table 1: Sizes ratio between $SAW_2(n)$ and $SAW_3(n)$ for small n

If the difference is large, then the proof of completeness is irrelevant. Obviously, our computational approach can only provide results for small n corresponding to peptides, not proteins. These results are listed into Table 1 and the ratio is represented in Figure 11.

One can deduce from these results that for small n, there is only one connected component in $\mathfrak{S}(n)$, and thus $SAW_2(n) = SAW_3(n)$ for $n \leqslant 11$. On the contrary, the previous section shows that $SAW_2(n) \neq SAW_3(n)$ for n equal to $158, 169, 175$, and 914. It seems as if a stall appears between $n = 11$ and $n = 158$ making a rupture in the connectivity of $\mathfrak{S}(n)$: too much vertices from $\mathfrak{S}_0(n)$ have been removed to preserve its connectivity when defining $\mathfrak{S}(n)$. As the difference between the sizes of $\mathfrak{S}_0(n)$ and $\mathfrak{S}(n)$ increases more and more, we can reasonably suppose that the remaining nodes are more and more isolated, leading to the appearance of several connected components, and to the reduction of the size of the component of the origin.

7. Conclusion

In this paper, the 2D HP square lattice model used for low resolution prediction has been investigated. We have shown that its SAW requirement can be understood in at least four different ways. Then we have demonstrated that these four sets are not equal. In particular, SAW_4 is strictly included into

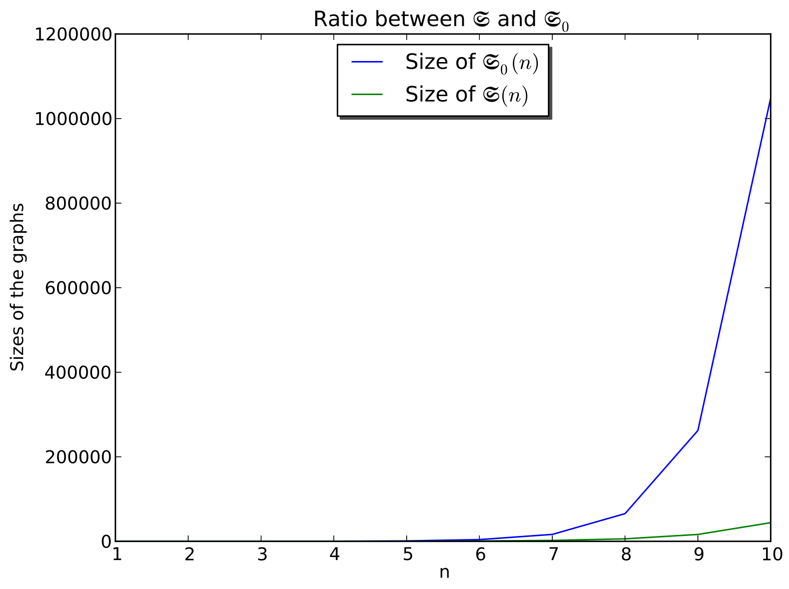


Figure 11: Number of nodes removed in $\mathfrak{G}_0(n)$

SAW_3, which is strictly included into SAW_1. So the NP-completeness proof has been realized in a larger set that the one used for prediction, which is larger than the set of biologically possible conformations. Concrete examples have been given, and characterizations of these sets have finally been proposed.

At this point, we can claim that the NP-completeness of the protein folding prediction problem does not hold due to the fact that it has been established for a set that is not natural in the biological world: it encompasses too much conformations as it takes place into SAW_2. However, this discussion still remains qualitative, and if the size of SAW_3 is very large, then the PSP problem is probably an NP-complete one (even if the proof still remains to do).

We will try to compare in a future work the size of SAW_2, which is the Sloane's A001411 sequence, to the size of the connected component of the origin. The third dimension will be investigated, and mathematical results of the self-avoiding walks belonging into SAW_3 will be regarded. Conclusion of these studies will then be dressed, and solutions to improve the quality of the protein structure prediction will finally be investigated.

[1] *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010*. IEEE, 2010.

[2] Christian B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[3] R. Backofen, S. Will, and P. Clote. Algorithmic approach to quantifying the hydrophobic force contribution in protein folding, 1999.

[4] Jacques Bahi, Nathalie Côté, and Christophe Guyeux. Chaos of protein folding. In *IJCNN 2011, Int. Joint Conf. on Neural Networks*, pages 1948–1954, San Jose, California, United States, July 2011.

[5] Jacques Bahi, Nathalie Côté, Christophe Guyeux, and Michel Salomon. Protein folding in the 2D hydrophobic-hydrophilic (HP) square lattice model is chaotic. *Cognitive Computation*, 4(1):98–114, 2012.

[6] Bonnie Berger and Tom Leighton. Protein folding in the hydrophobic-hydrophilic (hp) is np-complete. In *Proceedings of the second annual international conference on Computational molecular biology*, RECOMB '98, pages 30–39, New York, NY, USA, 1998. ACM.

[7] Richard Bonneau and David Baker. Ab initio protein structure prediction: Progress and prospects. *Annual Review of Biophysics and Biomolecular Structure*, 30(1):173–189, 2001.

[8] Erich Bornberg-Bauer. Chain growth algorithms for hp-type lattice proteins. In *Proceedings of the first annual international conference on Computational molecular biology*, RECOMB '97, pages 47–55, New York, NY, USA, 1997. ACM.

[9] Michael Braxenthaler, R. Ron Unger, Ditza Auerbach, and John Moult. Chaos in protein dynamics. *Proteins-structure Function and Bioinformatics*, 29:417–425, 1997.

[10] Dylan Chivian, David E. Kim, Lars Malmstrm, Jack Schonbrun, Carol A. Rohl, and David Baker. Prediction of casp6 structures using automated robetta protocols. *Proteins*, 61(S7):157–166, 2005.

[11] Pierluigi Crescenzi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the complexity of protein folding (extended abstract). In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 597–603, New York, NY, USA, 1998. ACM.

[12] Robert L. Devaney. *An Introduction to Chaotic Dynamical Systems*. Addison-Wesley, Redwood City, CA, 2nd edition, 1989.

[13] KA Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–9–, March 1985.

[14] I. Dotu, M Cebrián, P. Van Hentenryck, and P.Clote. On lattice protein structure prediction revisited. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6):1620–32, Nov–Dec 2011.

[15] I. Dubchak, I. Muchnik, S. R. Holbrook, and S. H. Kim. Prediction of protein folding class using global description of amino acid sequence. *Proc Natl Acad Sci U S A*, 92(19):8700–8704, Sep 1995.

[16] Trent Higgs, Bela Stantic, Tamjidul Hoque, and Abdul Sattar. Genetic algorithm feature-based resampling for protein structure prediction. In *IEEE Congress on Evolutionary Computation* [1], pages 1–8.

[17] Md. Hoque, Madhu Chetty, and Abdul Sattar. Genetic algorithm in ab initio protein structure prediction using low resolution model: A review. In Amandeep Sidhu and Tharam Dillon, editors, *Biomedical Data and Applications*, volume 224 of *Studies in Computational Intelligence*, pages 317–342. Springer Berlin Heidelberg, 2009.

[18] Dragos Horvath and Camelia Chira. Simplified chain folding models as metaheuristic benchmark for tuning real protein folding algorithms? In *IEEE Congress on Evolutionary Computation* [1], pages 1–8.

[19] Md. Kamrul Islam and Madhu Chetty. Novel memetic algorithm for protein structure prediction. In *Proceedings of the 22nd Australasian Joint Conference on Advances in Artificial Intelligence*, AI '09, pages 412–421, Berlin, Heidelberg, 2009. Springer-Verlag.

[20] Md. Kamrul Islam and Madhu Chetty. Clustered memetic algorithm for protein structure prediction. In *IEEE Congress on Evolutionary Computation* [1], pages 1–8.

[21] Neal Madras and Gordon Slade. *The Self-avoiding walk*. Probability and its applications. Birkhauser, 1993.

[22] M. Mann, S. Will, and R. Backofen. Cpsp-tools–exact and complete algorithms for high-throughput 3d lattice protein studies. *BMC Bioinformatics*, 7:9:230, May 2008.

[23] Luis Germán Pérez-Hernández, Katya Rodríguez-Vázquez, and Ramón Garduño-Juárez. Estimation of 3d protein structure by means of parallel particle swarm optimization. In *IEEE Congress on Evolutionary Computation* [1], pages 1–8.

[24] Andrej S[breve]ali, Eugene Shakhnovich, and Martin Karplus. How does a protein fold? *Nature*, 369(6477):248–251, May 1994.

[25] Alena Shmygelska and Holger Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6(1):30, 2005.

[26] Alena Shmygelska and Holger H Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem, 2005 Feb.

[27] Ron Unger and John Moult. Genetic algorithm for 3d protein folding simulations. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 581–588, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[28] Yang Zhang, Adrian K. Arakaki, and Jeffrey Skolnick. Tasser: An automated method for the prediction of protein tertiary structures in casp6. *Proteins*, 61(S7):91–98, 2005.

Appendix

This appendix contains the Python programs that have helped the authors during their investigations of the respective SAW_i.

7.1. The list of all possible conformations

Python function called *conformations* (Listing 1) produces the list of all possible conformations (satisfying or not the excluded volume requirement) as follows: the conformations of length n are the conformations of size $n-1$ with 0, 1, 2, or 3 added to their tails (recursive call). The return is a list of conformations, that is, a list of integers lists.

Listing 1: Obtaining all the conformations

```
def conformations(n):
    if n==1:
        return [[0]]
    else:
    L = []
    for k in conformations(n-1):
        for i in range(4):
            L.append(k+[i])
    return L
```

7.2. Obtaining the SAW_1 conformations

To obtain the conformations belonging into SAW_1, we first introduce the function *points* which aim is to produce the list of points (two coordinates) of the square lattice that corresponds to a given conformation C. This is simply the function p of Section 4.1.

Function *is_saw1* returns a Boolean: it is true if and only if the conformation C is in SAW_1. To do so, the list of its points in the lattice (its support) is produced, and it is regarded whether this list contains twice a same point (in other words, if the support has the same size than the list of points).

Finally, *saw1_conformations* produces a generator. It returns the next SAW_1 conformation at each call of the *next* method on the generator. To do so, an exhaustive iteration of the list produced by *conformations* is realized, and the *is_saw1* function is applied to each element of this list, to test if this element is in SAW_1.

Listing 2: Finding the SAW_1 elements

```
def points(C):
    L = [(0,0)]
    for c in C:
        P = L[-1]
        if c == 0: L.append((P[0]+1,P[1]))
        elif c == 1: L.append((P[0],P[1]-1))
        elif c == 2: L.append((P[0]-1,P[1]))
        elif c == 3: L.append((P[0],P[1]+1))
     return L


def is_saw1(C):
    L = points(C)
    return len(L) == len(list(set(L)))


def saw1_conformations(n):
    for k in conformations(n):
        if is_saw1(k):
            yield k
```

7.3. Investigating the SAW_3 set

To determine if a conformation in SAW_2 is in SAW_3 too, we try all the possible pivot moves (either in the clockwise direction, or in the anticlockwise). The *fold* function tests, considering a conformation called *protein*, a pivot move on residue number *position* following the given *direction* (+1 or -1, if clockwise or not). Function *is_in_SAW3* applies the *fold* function to each residue of the candidate, and for the two possible directions. The function returns True if and only if no pivot move is possible (the function can return erroneous False responses for conformations that can be unfolded a few, but never until the line 0000...00).

Listing 3: Testing whether a SAW_2 element is in SAW_3

```
def fold(protein, position, direction):
    if position == 0:
        return protein
    new_conformation = []
    for k in range(len(protein)):
        if k<abs(position):
            new_conformation.append(protein[k])
        else:
            new_conformation.append((protein[k]+direction)%4)
    return new_conformation


def is_in_SAW3(candidate):
    for k in range(1,len(candidate)):
        if is_saw1(fold(candidate, k, -1)):
            return True
        elif is_saw1(fold(candidate, k, +1)):
            return True
    return False
```