




Evolving cellular automata schemes for protein folding modeling using the Rosetta atomic representation

Daniel Varela^{1,2} · José Santos³ 

Received: 10 June 2021 / Revised: 24 November 2021 / Accepted: 31 December 2021 /
Published online: 16 January 2022
© The Author(s) 2022

Abstract

Protein folding is the dynamic process by which a protein folds into its final native structure. This is different to the traditional problem of the prediction of the final protein structure, since it requires a modeling of how protein components interact over time to obtain the final folded structure. In this study we test whether a model of the folding process can be obtained exclusively through machine learning. To this end, protein folding is considered as an emergent process and the cellular automata tool is used to model the folding process. A neural cellular automaton is defined, using a connectionist model that acts as a cellular automaton through the protein chain to define the dynamic folding. Differential evolution is used to automatically obtain the optimized neural cellular automata that provide protein folding. We tested the methods with the Rosetta coarse-grained atomic model of protein representation, using different proteins to analyze the modeling of folding and the structure refinement that the modeling can provide, showing the potential advantages that such methods offer, but also difficulties that arise.

Keywords Protein folding · Neural cellular automata · Differential evolution

Area Editor: James A. Foster

✉ José Santos
jose.santos@udc.es

Daniel Varela
daniel.varela@biochemistry.lu.se

- ¹ Department of Biochemistry and Structural Biology, University of Lund, Lund, Sweden
- ² Department of Computer Science and Information Technologies, University of A Coruña, A Coruña, Spain
- ³ CITIC (Centre for Information and Communications Technology Research), Department of Computer Science and Information Technologies, University of A Coruña, A Coruña, Spain

1 Introduction

A protein reaches its native and functional structure through the dynamic physical process of protein folding, which is the result of physical and chemical interactions over time between the protein amino acids. However, existing research into the use of computational approaches for protein structure has focused on predicting the final folded structure, this because the final native structure is related to the function of the protein.

Computational Protein Structure Prediction (PSP) is necessary to reduce the increasing “sequence/structure gap” between the number of proteins with a known sequence (order of millions [40]) and the number of proteins whose structure is already resolved (more than 177,000 in 2021 [26]). Protein structures publicly available in the Protein Data Bank (PDB) [26] are the result of laborious, expensive and time-consuming methods such as X-ray crystallography (in which protein samples have to be crystalized) and Nuclear Magnetic Resonance (NMR) (in which protein samples are in solution) [16]. Even with the rapid increase in structures resolved with the latest cryo-electron microscopy technique (protein samples are frozen using liquid nitrogen) [3], to reduce the gap, the computational prediction of the protein structure is required. With computational prediction, there are methods that rely on known resolved proteins, such as prediction based on the homology between the sequences of the target protein and those of proteins with a known structure, and threading methods that seek to fit a target sequence into a library of resolved structures [38]. At the other end of the range of PSP computational approaches, the “ab initio” prediction is the most challenging, since it relies only on the protein’s primary structure (its amino acid sequence). This ab initio prediction relies on the thermodynamic hypothesis that states that the protein structure with the lowest Gibbs free energy corresponds to the native structure, and that the native conformation is determined solely using the information of the amino acid sequence (Anfinsen’s dogma) [1].

Given a protein representation model and an energy model associated with protein conformations, the ab initio approach becomes a means of identifying the conformation that minimizes energy. Hence, there has been intense research on the use of search methods for the ab initio PSP, especially with the use of metaheuristics from the field of natural computation, and in particular evolutionary algorithms, with simplified lattice models of protein representation [19, 31, 39, 42, 46] (mentioning just a few) as well as with off-lattice atomic models [9, 24]. The prediction of the final native structure enables the practical application of drug search (which only requires the final folded structure). However, the objective of the present study is not PSP itself. We also need to know how proteins fold dynamically. From a biological point of view, knowledge of how proteins fold is necessary to understand, for example, protein misfolding that can be involved in serious diseases. On the other hand, we want to know if it is possible, from a machine learning perspective, to obtain a folding model without resorting to a priori decisions of the designer, that is, by learning exclusively from available data, which indeed is the motivation of the present study. Such a folding

model could be used, for instance, for the refinement of protein conformations (e.g., conformations predicted with PSP methods), hence obtaining structures closer to the real native structure.

Levinthal's paradox [18] states that it is not possible for a protein to reach its native structure by means of a random search of the enormous number of possible structures. Nevertheless, proteins can spontaneously fold into their native conformations on short timescales (these generally of the order of milliseconds or seconds). In fact, as Kmiecik et al. [13] claim, the number of possible conformations is drastically reduced due to preferences of secondary structure and other geometric characteristics of the amino acid chains, facilitating relatively fast folding to folded structures.

In protein folding modeling, the traditional approach is the use of molecular dynamics simulations [25], which attempt to capture the main atom interactions in order to define a force field and thus to derive the moves of atoms based on Newton's equations. The problem with these approaches is that those force field potentials are non-exact. As Englander and Mayne [6] note, "The forces that direct protein folding are delicately balanced, inter-locking, and not describable in exact terms". Consequently, errors are progressively propagated in the high-time consuming simulations of very short moves. Furthermore, and as also observed by Kmiecik et al. [13], even using a supercomputer dedicated to simulations of atomic interactions in molecular dynamics, it is only possible to simulate the folding process with short proteins. For example, a typical simulation of a system size of about 10^5 - 10^6 atoms for a simulation of several nanoseconds will likely require 10^6 - 10^7 time steps and will take several days of computing time in a workstation/cluster [25].

Apart from molecular dynamics simulations, there are few studies that have experimented with modeling the temporal folding process, rather than looking directly for the final folded conformation. For example, Krasnogor et al. [14], using a simple lattice scheme of protein representation, explored the use of tools such as Lindenmayer systems and Cellular Automata (CA)¹ to see whether these models could provide the transitions to obtain a native folded structure. However, in this initial study with CA there was no connection between the amino acid nature of the primary structure and the CA rules, and thus their study [14] only focused on testing the possibility of using (evolved) CA rules to provide final folded structures.

Calabretta et al. [2] modeled the temporal process evolving matrices of folding potentials between amino acids, where each element of the matrix determines the force of repulsion or attraction between amino acids at a given distance (100 Å). When the matrix is evolved, the fitness function considers the difference (in the rotation angles in the backbone chain) of the final folded structure relative to the actual native structure. The authors only tested the method using a short fragment (13 amino acids) of the protein *crambin*, which generates an alpha-helix as in the native structure.

Danks et al. [4] presented a Lindenmayer system to provide the folding. The L-system establishes an association between amino acid states and Secondary Structure Elements (SSEs) of the chain. The (stochastic) rules of the L-system alter the

¹ The acronym CA will be used for both Cellular Automata and Cellular Automaton

state of an amino acid depending on its own amino acid type and on neighboring amino acid types (on both sides of the protein chain). Using typical torsion angles of the backbone chain in the seven secondary structure elements considered, it is possible to rebuild the structure of a protein in each application of the L-system rules. Using four proteins, corresponding to each major structural class, the authors showed that a preference for local structure can emerge for some amino acids in the protein chain.

Unlike molecular dynamics simulations, in the present study we will consider how to obtain a model of the folding process automatically using machine learning, a model that can also be easily adapted to different conformational representations of proteins. For example, our previous work on the line of protein folding modeling focused on the use of lattice models of protein representation, such as the HP model [32–34] and the Face-Centered Cubic (FCC) lattice model [43]. In these studies [32–34, 43] the CA models overcome the problem previously discussed in Krasnogor et al.'s study [14], as there is now a connection between the CA scheme and the specific amino acids of the primary structure to which the CA rules are applied. The modeling of folding performed in lattice models can be extended to atomic models, where new possibilities and difficulties appear, which is the main objective addressed in the present paper.

Moreover, protein folding is considered here as an emergent process, the result of the emergent consequence of the interactions of protein components over time. Therefore, the process can be modeled with traditional tools used in Artificial Life to model the emergent property, classical tools such as cellular automata [11]. Consequently, we used the traditional CA scheme to model changes in the dihedral angles of the atomic protein conformation, applying an optimized cellular automaton sequentially and iteratively through several time iterations to decide the changes in the dihedral angles of the amino acid sequence. The coarse-grained protein representation model of the Rosetta system [29, 30] (one of the most successful software packages in PSP) was used in the CA-based folding modeling. However, instead of traditional CA rules that specify the next state of a grid element based on its previous state and the state of the neighboring elements, a simple Artificial Neural Network (ANN) was used to implement the rule set, which is why we refer to these ANNs as “neural cellular automata”. These neural-CA are optimized by an Evolutionary Algorithm (EA) (Differential Evolution - DE [27]) in order to obtain the final folded structure. Preliminary results with the atomic model are given in [41], while the present study describes in more detail the extension of the methods to the atomic model, as well as discussing results with different proteins. Specifically, we will describe, for the first time, the protein structure refinement that optimized ANNs can provide, also considering different options to evolve the neural-CA models. An analysis of the results to provide folding with different starting conformations is also included. All these novel aspects allow us to show the possibilities and problems that appear in the folding modeling when the Rosetta atomic scheme is used.

The remainder of the article is structured as follows. Section 2 details the methods used for the folding modeling, with a brief introduction to the main Rosetta concepts used in its *ab initio* PSP protocol, such as its protein energy and protein

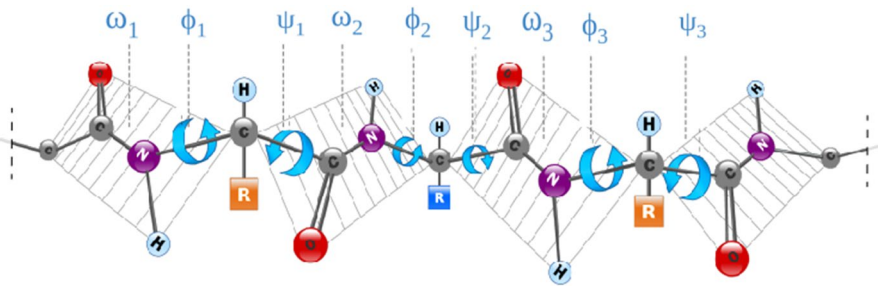


Fig. 1 Rosetta’s low-level coarse-grained protein representation. This only considers the main backbone atoms, while pseudo-atoms represent the lateral residues. The three dihedral angles between the main amino acid atoms, ω (between atoms of the peptide bond), ϕ and ψ , encode each protein conformation

representation models. These Rosetta concepts are used in our folding modeling. Section 2 also details the neural-CA modeling of the folding process, as well as the evolutionary algorithm (DE) used to automatically obtain the models based on neural-CA that provide the folding. Section 3 describes the experiments and results performed with different proteins in this modeling, detailing the main aspects related to energy transitions and conformational refinement in the modeling. Finally, Section 4 provides a discussion of the main conclusions that can be drawn from the experiments and the methods employed.

2 Methods

This section summarizes the methods used to model the folding process. All the software of the defined methods can be downloaded from [37].

2.1 Basic aspects of Rosetta

Two protein representations are used by the Rosetta system: coarse-grained and all-atom. The coarse-grained models group atoms into beads and assign an energy function between the beads in order to reproduce some properties of the protein structure [23]. In Rosetta, its low-resolution coarse-grained representation only considers the main backbone atoms (with their dihedral angles), whereas the side chains are represented by a pseudo-atom located at their center of mass (Fig. 1). Therefore, protein conformations are defined in the space of dihedral angles, with three degrees of freedom (ω , ϕ and ψ) in each amino acid of the protein chain. Rotation *Chi* angles for side chains are also considered in the all-atom representation. Other degrees of freedom are set to fixed, “ideal” values (e.g., all bond lengths and angles are fixed) [13].

For protein structure prediction, the Rosetta ab initio protocol [29, 30], with the low-resolution protein representation, employs a search technique in which a Monte Carlo procedure decides if the dihedral angles of small protein fragments can replace the original ones [12, 29]. A protein fragment is a group of consecutive amino acids of a resolved protein. Those fragments are drawn from

Table 1 Coarse-grained Rosetta's energy terms

env	Residue environment (solvation)
Pair	Residue pair interactions (electrostatics, disulfides)
ss_pair	Strand pairing (hydrogen bonding)
Sheet	Strand arrangement into sheets
hs_pair	Helix-strand packing
rsigma	Strand pairing based on distance/register
rg	Radius of gyration (van der Waals attraction; solvation)
cenpack	Packing density
cbeta	$C\beta$ density (solvation; correction for excluded volume effect introduced by simulation)
vdw	Steric repulsion

experimentally determined structures (a non-redundant set of proteins). The fragments are selected taking into account their sequence similarity with respect to the window of consecutive residues in the target protein into which the fragments will be inserted. This position for fragment insertion into the target is randomly chosen. Rosetta uses fragment regions 9 and 3 residues long. These fragments are extracted for each position on the target, a process that is prior to the *ab initio* run and that generates a library of fragments (particular for each target protein). Typically, the fragment libraries used in Rosetta contain about 200 fragments for each amino acid position of the protein.

The decision as to whether the dihedral angles of a selected fragment replace those of the target protein is based on the Metropolis criterion [20]. This criterion always accepts changes that improve energy (lower values), while occasionally it accepts dihedral angle changes that worsen energy (energy increase), with the probability of accepting the fragment depending on the energy increase relative to the previous state in the target protein. This probability is given by $\exp(\frac{-\Delta E}{kT})$, where k is the Boltzmann constant and T a temperature parameter (the fixed value $T = 2$ is used, but with the possibility of temperature re-heating). This procedure helps the search for protein conformations with the fragment substitution technique to escape local minima.

Regarding the energy model, Rosetta uses knowledge-based and physics energy terms [17]. Knowledge-based potentials imply empirical terms obtained from a statistical analysis of the structures already resolved in PDB [26]. The interesting property is that these knowledge-based terms require less computational time. Physics-based energy terms [10] are based on bond lengths and angles, torsion angles, electrostatic and van der Waals interactions.

In Rosetta, a protein conformation has an associated energy, defined as a linear weighted combination of such energy components that model the molecular forces acting between the amino acid atoms of that conformation. These scoring terms are, in most cases, knowledge-based.

In the Rosetta energy model, steric overlap between the atoms of the backbone and the side-chain is penalized, while van der Waals interactions are modeled with a Lennard-Jones potential [29]. Other Rosetta's energy terms correspond

Table 2 Stages of Rosetta ab initio protocol

Stage 1	Starting from a fully extended conformation, this stage inserts 9-mer fragments until all the backbone dihedral angles are modified at least once and considering a maximum of 2,000 cycles (fragment insertion attempts). During this stage, the energy function (called <i>score0</i>) only takes into account the steric-clash term to avoid overlap between backbone atoms and the centroids of lateral residues.
Stage 2	Stage2 employs 9-mer fragment insertions over 2,000 cycles, but uses a more complex score function, <i>score1</i> , which adds specific pair interactions and hydrophobic burial terms, along with scores of secondary structure.
Stage 3	The third Rosetta stage runs 10 iterations of 2,000 cycles of fragment (9-mer) insertion attempts. Rosetta combines in this stage two score functions, <i>score2</i> and <i>score5</i> . These functions focus on secondary structure terms and compactness. A convergence check determines the structural similarity of the current conformation with respect to a reference one regularly updated. If there is not enough structural variation after 100 fragment insertions are accepted, stage 3 ends.
Stage 4	The final Rosetta stage employs 3-mer fragment insertions over 12,000 cycles, split into 3 iterations of 4,000 cycles for each one. In this stage, <i>score3</i> (which considers all energy components) is used.

to electrostatics effects and solvation, hydrogen bonding, repulsion and scores related to secondary structure (e.g., helix-strand packing and strand pairing).

Table 1 shows a very brief definition of each energy term, while the detailed definition of the energy terms can be found, for example, in [29], and the weight sets for the individual energy terms for the definition of every Rosetta score are detailed in [30]. The Rosetta score function called *score3* is the one that integrates all the energy components. Nevertheless, the weight set is changed according to the stage of the Rosetta ab initio protocol (Table 2).

A problem that must be taken into consideration is that, as is well-known, Rosetta’s knowledge-based energy model is inaccurate since the native conformation is not necessarily located in the minimum of energy. For example, Shmygelska and Levitt [36] show deficiencies in the energy function corresponding to the low-resolution protein representation of Rosetta, since the structures with lower energy do not have to correspond with the most native-like.

In search of protein conformations with minimal energy, the Metropolis Monte Carlo procedure is run many times. For this purpose, the ab initio protocol is divided into four stages (as detailed in Table 2). Rosetta uses the protein coarse-grained representation and its fragment insertion technique (with the Metropolis criterion [20]) throughout these four stages, to generate new structural conformations. Table 2 includes a brief summary of each stage with the most important details.

Moreover, the number of fragment insertions attempts at each stage can be changed with the parameter *increase_cycles*, which multiplies to the default values of insertion cycles in the four stages of the ab initio procedure. Since the Metropolis Monte Carlo process is stochastic, the 4-stage ab initio protocol is run thousands of times. The final conformations (“decoys”) in this ab initio protocol (a clustering process can be applied to decipher the most representative decoy set), can be refined in a second “Ab initio relax” procedure that uses the full Rosetta’s atomic model,

process that performs the reconstruction of the protein side chain with an all-atom energy minimization.

2.2 Neural cellular automata

Since protein folding is considered an emergent process, a cellular automaton scheme is used to model the process. The idea is that the CA scheme provides the conformation changes of the protein over time to obtain a final folded structure corresponding to the native structure. As in classical CA, where the CA rules are applied to define the next state of each element in a grid-like environment and over time iterations, now the CA scheme that provides the folding will be applied over several time iterations to all the elements of the protein chain (dihedral angles of amino acids). The difference is that the CA scheme is now implemented with a feed-forward ANN and, therefore, we define it as a neural-CA model. This ANN must provide changes that correspond to continuous values of the dihedral angles of the protein's coarse-grained representation [30]. The second difference is that the neural cellular automaton model will receive the input information from the conformational energy space, rather than the spatial surroundings of the element (neighbor states) to which the classical CA rule set model is applied. The next subsection explains how this ANN model is obtained by means of an evolutionary algorithm.

The neural-CA process can be summarized as follows: With the coarse-grained representation model, the neural cellular automaton is applied sequentially to the dihedral angles of all the amino acids. However, the neural-CA can only change the angles ϕ and ψ , since the third dihedral angle of the amino acid (ω) remains fixed (180°), because this angle of the peptide bond cannot rotate. The input information for the ANN is obtained by taking into account the consequences of perturbations at the dihedral angles (ϕ or ψ) at which the ANN is applied. The same ANN defines both dihedral angle changes. This process is repeated iteratively from the first dihedral angle of the initial amino acid to the final dihedral angle of the final amino acid; that is, repeating the same sequential process along the protein chain and through different temporal iterations while the protein progressively folds into a final structure.

2.2.1 Artificial neural network inputs and output

The neural cellular automaton scheme is applied sequentially to determine the changes of ϕ and ψ , as illustrated in Fig. 2. The ANN determines the angle changes with information extracted from the energy landscape, which is more natural as proteins navigate through the complex energy landscape to fold to the native structure [45]. Moreover, this is more useful than using spatial information, as the energy landscape encapsulates the details of the spatial protein conformation into a more compact representation. This information on the energy landscape is obtained when different perturbations are considered in the angle at which the ANN is applied. The angle perturbations are defined considering a *MAC* (*Maximum Angle Change*) value. With this partial information of the energy landscape, the ANN output defines

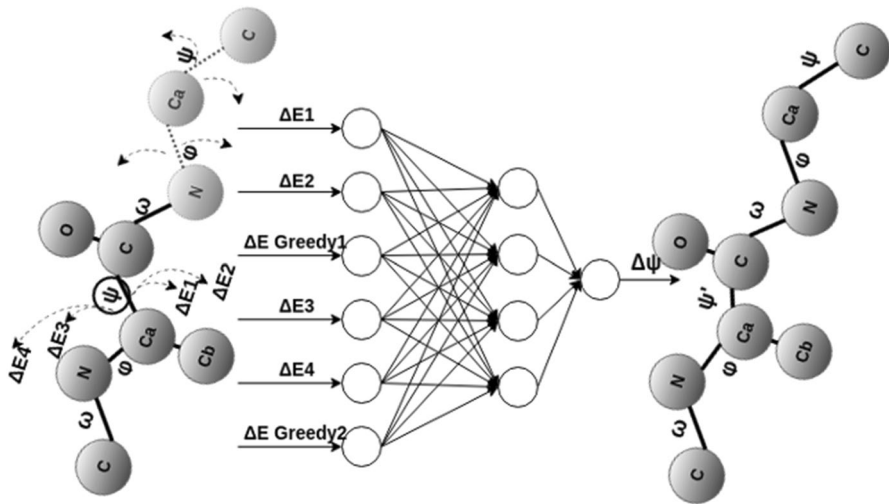


Fig. 2 Neural cellular automaton that provides the iterative folding. The ANN determines the changes of angles ϕ and ψ . The ANN inputs are defined considering 4 perturbations in the dihedral angle to which the neural network model is to be applied, taking into account the difference between the energies of the protein conformations before and after the angular perturbations. Two additional ANN inputs correspond to the difference in conformational energy if greedy changes are applied in the next angles and after the largest angle perturbations

the corresponding dihedral angle change (increase or decrease of its value, also constrained to the range $[-MAC, MAC]$).

The ANN inputs are defined as follows:

1. The neural cellular automaton is applied, in the amino acid i of the protein chain, to one of its dihedral angles (ϕ or ψ) (Fig. 2). In this dihedral angle, four disturbances are considered: MAC , $MAC/2$, $-MAC/2$ and $-MAC$. The conformational energy difference (positive, negative, or zero) is calculated by considering the protein conformations before and after each angle perturbation. These four energy increases ($\Delta E1$, $\Delta E2$, $\Delta E3$ and $\Delta E4$ in Fig. 2) are inputs to the ANN.
2. The partial view of the conformational energy space above can be enhanced with more information on the consequences of an angle change for subsequent changes in the next angles. It must be taken into account that the energy landscape is dynamic through the folding process since, once an angle change is performed, the subsequent energy landscape corresponds to a different protein conformation. With this idea in mind, for the highest perturbations ($-MAC$ and MAC) in the dihedral angle to which the ANN is applied, several “greedy” changes are considered in the next N (ϕ or ψ) dihedral angles. A greedy change here means that those posterior angle changes are chosen between two possibilities, $-MAC/2$ and $MAC/2$, selecting the one that provides the resulting conformation with the lowest energy. These possible values in the angle changes are used, in that these correspond to the average values (positive and negative) that the ANN can provide.

That is, at the current angle to which the neural network is applied, one of the largest perturbations ($-MAC$ and MAC) is considered. After this, N greedy moves are considered in the next angles. The difference in conformational energy is calculated again taking into account the final and initial (without any disturbance) conformations of the protein.

The two additional inputs (ΔE Greedy1 and ΔE Greedy2 in Fig. 2) correspond with these energy differences. In this way, the ANN has more information regarding the energy landscape to determine the most appropriate angle change, since it has a partial view of the “future” consequences of the change in the angle at which the neural network is applied, that is, the ANN has a limited view of what the subsequent energy landscape would look like. When the ANN is applied in the posterior dihedral angles, obviously the ANN can decide a different change to the greedy one, again taking into account the information of the posterior energy landscape in next angles.

Figure 2 illustrates the general process. When the neural cellular automaton is applied to an angle (ψ in the example in Fig. 2), it receives the six inputs with those noted energy differences. The ANN output determines the most appropriate angle change with that partial information extracted from the energy landscape. The hidden and output nodes use the standard sigmoid transfer function, decoding the output value in the range $[-MAC, MAC]$. The number of nodes in the hidden layer is set so that the association between energy increases and appropriate angle changes can be learned while avoiding overfitting.

Note that the update of the angles is therefore sequential. This is so because, with a parallel update, the interpretation of the inputs to the network would no longer be as explained, taking into account the consequences of the perturbations in the angle at which the ANN is applied (which requires the rest of the protein conformation to be intact). This modeling constraint is contrary to the actual parallel folding process, but facilitates the ANN processing with correct input information to decide the appropriate angle changes.

2.3 Evolutionary algorithm (Differential Evolution) and fitness function

The neural-CA model that defines the folding is obtained automatically by an evolutionary algorithm: Differential Evolution [27]. DE is a population-based search method, where the genetic population encodes possible solutions to the problem, in this case ANNs that are employed as cellular automata to provide the folding process of a protein chain. It should be noted that the objective of the this study is not the comparison of different evolutionary algorithms for the application, since our objective focuses on the modeling of the folding process. DE was selected since it is a robust method and with proven advantages over other EAs methods [5, 8] and also yields better results in PSP compared to other EAs [31].

DE maintains a population of vectors that encode solutions to a problem. An alternative candidate is chosen for each solution, and the one with the best fitness is passed on to the next generation. The key aspect of DE is the generation of the

candidate or trial solutions, since these are defined from the difference of two vectors of the population. The DE algorithm is especially suitable for optimization problems in which the solutions are coded with real values (such as the current problem). Algorithm 2.1 pseudo-code specifies the main steps of the standard DE algorithm.

A limited number of parameters is required for the DE implementation. In addition to the population size, two parameters (F and CR) are used to generate the “trial” or “candidate” vectors (y) for each “target” vector x of the population. F is the weight factor (typically in $[0, 2]$), which is used to define the “mutant” or “donor” vector ($x_1 + F(x_2 - x_3)$). This is generated from the difference between two vectors (randomly selected) in the current population, the difference added to the “base vector” x_1 (also randomly selected). The next step involves a crossover operation between the target vector and the mutant vector to define the trial vector. Each component of the trial vector (y) is generated considering a crossover probability (CR) in the “binomial” crossover, taking into account that a least one vector element in y comes from the mutant vector thanks to the index R (Algorithm 2.1).

Algorithm 2.1: DIFFERENTIAL EVOLUTION(*Population*)

```

for each Individual  $\in$  Population
  do { Individual  $\leftarrow$  INITIALIZERANDOMPOSITIONS()
  repeat
    for each Individual  $x \in$  Population
       $\left\{ \begin{array}{l} x_1, x_2, x_3 \leftarrow \text{GETRANDOMINDIVIDUAL}(\textit{Population}) \\ // x_1, x_2, x_3 \text{ must be distinct from each other and } x \\ R \leftarrow \text{GETRANDOM}(1, n) // n \text{ is the dimensionality of the problem} \\ \textbf{for each } i \in 1 : n \\ \textbf{do} \left\{ \begin{array}{l} // \text{Compute individual's potentially new position } y = [y_1, \dots, y_n] \text{ (trial vector)} \\ r_i \leftarrow \text{GETRANDOM}(0, 1) // \text{uniformly in open range } (0, 1) \\ \textbf{if } ((i = R) \parallel (r_i < CR)) // CR - \text{crossover probability} \\ \quad y_i = x_{1_i} + F(x_{2_i} - x_{3_i}) // F - \text{weight factor} \\ \quad \textbf{else } y_i = x_i \end{array} \right. \\ \textbf{if } (f(y) \leq f(x)) \quad x = y // \text{if } y \text{ has better or equal fitness, replace } x \text{ with } y \end{array} \right.$ 
    until TERMINATIONCRITERION()
  return (GETLOWESTFITNESS(Population)) // return the best candidate solution

```

In the selection process, the target vector (x) and the trial vector (y) are compared (in terms of fitness) to select the one that survives, keeping the population size constant in the next generation. In this way, the algorithm has built-in elitism, since the best solution found is improved or maintained over generations.

Different DE schemes have been defined with the combination of mutation variants and crossover operators. Standard schemes include *DE/rand/1/bin* (1 specifies the number of differences used to define the donor vector whereas *bin* specifies the crossover type), which selects the base vector x_1 randomly when defining the mutant vector, and the scheme *DE/best/1/bin*, which chooses the best solution in the current population for the base vector. The key aspect of this algorithm is the adaptive nature of the exploration level used in the search for better solutions, given by the vector differences when generating the mutant vectors. At the beginning of the DE process, the differences tend to be large, progressively narrowing over the following

generations as the population concentrates on the best-found areas of the fitness landscape.

2.3.1 Neural-CA encoding and fitness

DE is used to optimize an ANN (which acts as a cellular automaton) that progressively provides, for the folding process, the dihedral angle changes that it decides on each situation. A simple feed-forward ANN (Fig. 2), with fixed topology, is used to implement the neural cellular automaton. Therefore, the DE population corresponds to possible ANNs, and each population vector encodes an ANN weight set. The ANN weights are encoded in the range $[-1, 1]$ and can be decoded in a different range so that the nodes can be saturated with the received input information, as detailed in the experiments.

Each solution of the population (encoded ANN) is applied sequentially to the dihedral angles of a protein, from the first angle ϕ in the initial amino acid to the angle ψ of the final amino acid (Fig. 2). As discussed above, this procedure is repeated through several steps, defining a “step” as the sequential application of the encoded ANN to all angles ϕ and ψ of the protein. A maximum number of steps is employed and, in addition, a control is considered at the end of each step: if the final folded conformation is worse compared to the final structure from the previous step (in terms of energy), then the iterative process ends, returning the final folded protein structure (and its fitness) from that previous step.

Therefore, with each encoded ANN, the folding process provided by the ANN ends with a final protein structure and the energy (Rosetta *score3* [30]) of this folded conformation defines the fitness of the encoded ANN. This iterative process can start with an initially unfolded chain, in which all angles ϕ and ψ have the same value (175°) (the dihedral angle ω is fixed at (180°), or it can also begin from a partially folded structure (as in the experiments described below).

It should be noted that the energy scores used for the fitness definition in the EA optimization and for the ANN inputs with the calculation of differences in conformational energy (before and after angle perturbations) can be different. Nevertheless, the experiments detailed in the next section use the same score for both. Rosetta *score3* was selected in both cases, as it corresponds to the full energy function of the coarse-grained representation, which includes all the individual energy terms [30].

One aspect to consider is the reason for using an evolutionary algorithm to obtain the optimized ANN, rather than training it with standard ANN training algorithms for feed-forward ANNs (with less computational time required). The reason is that a “training set” is necessary for the ANN’s supervised learning. This means that the appropriate targets (desired outputs, i.e., appropriate angle changes) for many different combinations of possible ANN inputs (energy increases after angle perturbations) must be set beforehand. The great advantage of using an evolutionary algorithm is that the designer only establishes the fitness taking into account the final folding obtained by each encoded ANN, that is, without any other requirement for the intermediate states, which makes the use of simulated evolution much more appropriate.

Finally, it is worth noting the similarity of this fitness assignment (to each encoded neural cellular automaton) with respect to morphological processes usually employed in evolutionary computation and Artificial Life applications, when a compact genotype defines the development of a final complex phenotype, for example a complex morphology in a simulated robot [15]. Here the genotype is the encoded ANN (applied as a classical cellular automaton to the protein chain), whereas the final complex phenotype is the final folded conformation after application of the ANN to the dihedral angles of the whole protein chain and in different temporal steps.

3 Results

3.1 Experimental setup

DE [27] was used for the optimization of a neural cellular automaton when this defines the folding process of a protein. DE population size was set at 60, using standard values for the weight factor ($F = 0.9$) and for the crossover probability ($CR = 0.9$) [27]. Moreover, scheme *DE/rand/1/bin* was employed, since this provides low selective pressure. DE was run over 100 generations to optimize the neural-CA. These parameters were selected experimentally to provide the best results in the proteins considered and without premature convergence.

The ANN solutions of the population are coded with weights at the interval $[-1, 1]$. The final ANN weights are established by multiplying the encoded weight value by a constant ($MAX_VALUE=3$), which allows us to obtain values in the entire range of the standard sigmoid transfer function of the neural network nodes, that is, it allows the nodes to be saturated with the input information received.

Regarding the ANN processing, the angle perturbations were generated with a value of 10° for the parameter *MAC*, experimentally selected to provide smooth angle changes. When calculating the ANN inputs, Rosetta *score3* was the only energy function considered. Moreover, $N = 2$ was used (Sect. 2.2). This means that, for the calculation of the ANN inputs ΔE Greedy1 and ΔE Greedy2 (Fig. 2), greedy changes are considered at the next two dihedral angles ϕ and ψ , as explained in Sect. 2.2, above. Finally, the maximum number of steps was fixed at 20, which means that the ANN is applied to all angles ϕ and ψ (from the beginning to the end of the protein chain) for a maximum of 20 times.

PDB proteins PDB:5WOD (α protein, 2 helices, 38 amino acids), PDB:1E0M (β protein, 3 beta strands, 37 amino acids) and PDB:1D5Q ($\alpha - \beta$ protein, 1 helix, 2 beta strands, 27 amino acids) were used in the experiments. These proteins were resolved with NMR, and therefore have lower resolution with respect to proteins resolved with X-ray crystallography. In the latter case, the deposited structures probably better reflect the final folded structure [16]. That is, we are assuming that the PDB structures of these NMR-resolved proteins correspond to the final folding, although these may correspond to transitional structures. However, these PDB structures are only considered as a reference to calculate the distances of the folded structures (by means of the optimized ANN) with respect to them.

The neural-CA models were evolved, for each protein, considering as fitness (for each of the encoded ANN models) the Rosetta energy (*score3*) of the final conformation after the neural-CA application to one or more initial conformations. If several initial conformations are considered, the ANN is applied independently to those conformations. Therefore, the ANN fitness is the average of the final energies of the final conformations when the ANN is applied to each initial conformation. Three experiments/options were carried out in this regard:

1. The ANN was evolved considering only an initial conformation that is fully unfolded.
2. The ANN was evolved considering three starting conformations that are partially folded. These initial conformations are selected with the application of the Rosetta ab initio protocol to the unfolded conformation three times, saving the structures at the end of Stage 2 (Table 2 in Sect. 2.1).
3. The ANN was evolved considering both the initial unfolded conformation and the three partially folded conformations.

3.2 Energy evolution in the folding process

In the first experiment, the neural cellular automaton is evolved using options *ii* and *iii*. As indicated, in option *ii*, the ANN is evolved considering three partially folded conformations (Rosetta ab initio Stage 2), whereas option *iii* adds the fully unfolded conformation to the set of starting conformations considered to calculate the fitness of each encoded ANN.

After the evolutionary optimization of the neural cellular automaton for each protein,² the dihedral angles (ϕ and ψ) of the amino acids were modified sequentially by the ANN application and beginning with a particular initial structure. Figure 3 includes the evolution of the different individual energy terms used by the Rosetta energy model through the folding process provided by the evolved and optimized ANNs. The subfigures on the left correspond to an ANN optimized with option *iii* and applied to the initial unfolded conformation. The subfigures on the right correspond to an optimized ANN with option *ii*, which is applied to a partially folded initial conformation (result of Rosetta ab initio Stage 2). The values of the energy terms are measured when the ANN has been applied to the two angles ϕ - ψ in each amino acid. The graph also includes, as a reference, the values of the different energy terms of the native structure (values to the right of labels in the subfigures in Fig. 3).

Starting with the unfolded conformation (Fig. 3, left), most of the energy terms tend to advance progressively to lower values, as can clearly be seen by the terms

² Typical computing times are 5.97 hours when the ANN is evolved with option *i*, 22.73 hours when the ANN is evolved with option *ii* and 28.30 hours with option *iii* (protein PDB:1D5Q as target, the code was not parallelized). The experiments were run in the Supercomputing Center of Galicia (www.cesga.es), with Intel Xeon E5-2680 v3 processors at 2.50GHz and 1GB of RAM. Once the ANN is evolved, its application to the protein dihedral angles with a maximum of 20 steps requires an average time of 20 seconds (protein PDB:1D5Q as target).

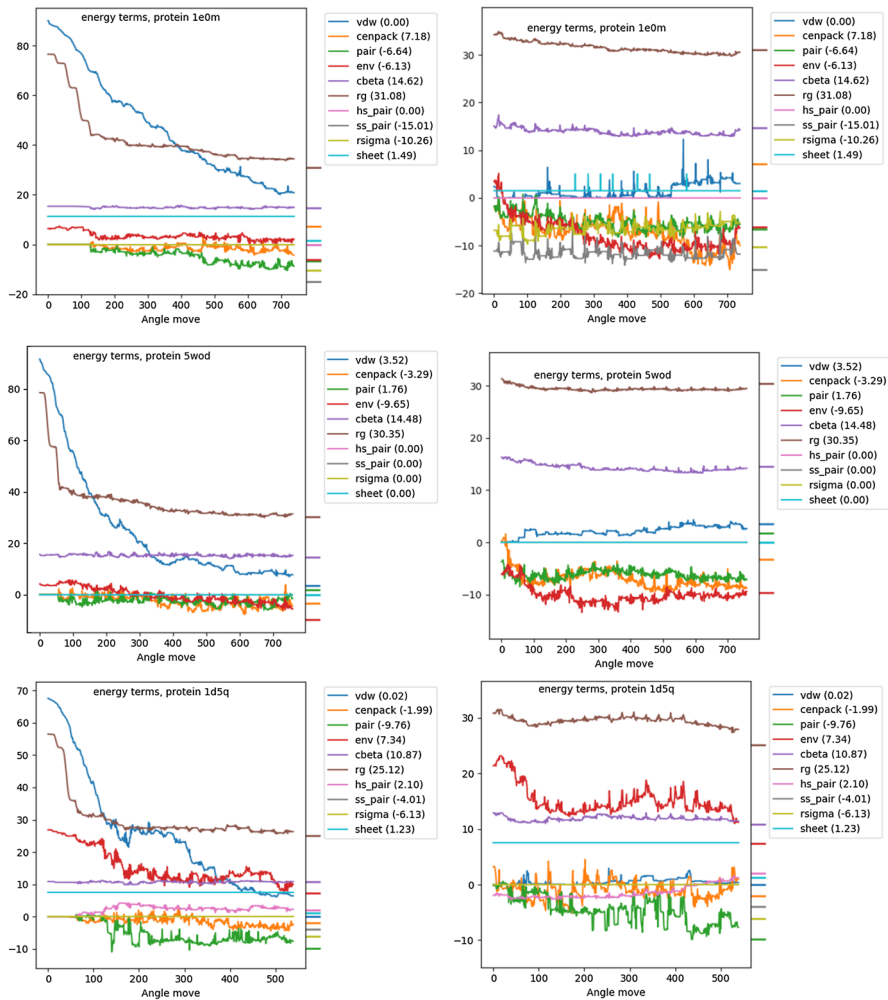


Fig. 3 Progression of individual Rosetta energy terms through the folding process, with proteins PDB:1E0M (upper figures), PDB:5WOD (figures in the middle row) and PDB:1D5Q (bottom figures). An optimized ANN with option *iii* is applied to the initial unfolded conformation (left) and an optimized ANN with option *ii* is applied to a partially folded conformation (right). The *x*-axis corresponds to the ANN application at the angles ϕ and ψ and over the temporal steps. The values to the right of the labels (in parentheses) correspond to the energy term values of the PDB native conformation

wdw (which considers only steric repulsion) and *rg* (which includes van der Waals forces and rewards compact structures), indicating that the protein is folding into a compact structure avoiding collisions. If the starting structure is a partially folded structure (Fig. 3, right), most of the terms show a slower progression with small perturbations, even with slight increases in the overall progression in the term *wdw*.

Regarding the specific terms related to secondary structure, (statistical) energy terms like *sheet* can only have a limited number of discrete values (e.g., in the term

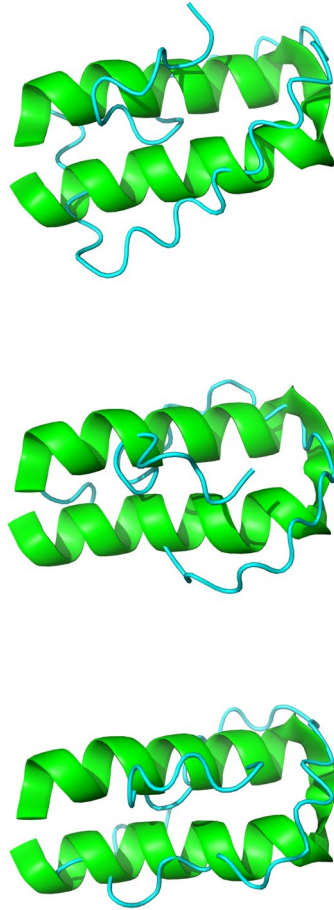


Fig. 4 Final conformations with protein PDB:5WOD after application of the optimized ANN to the unfolded conformation, when the ANN was evolved with option *i* (right), option *ii* (center) and option *iii* (left). The native structure is shown in green (Color figure online)

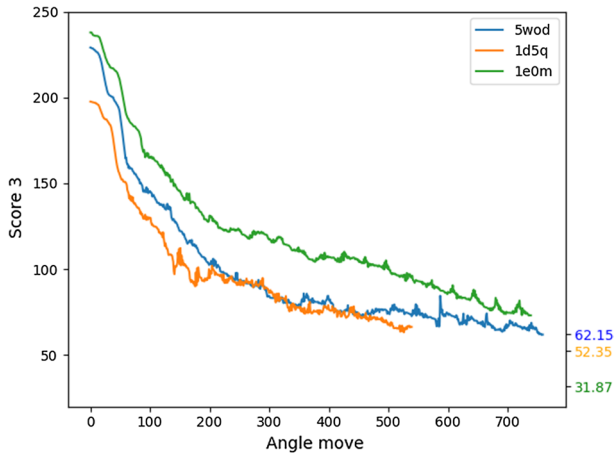


Fig. 5 Progression of the energy (*score3*) through the folding process with proteins PDB:1E0M, PDB:5WOD and PDB:1D5Q (starting with the unfolded conformation) with the best ANN evolved with option *iii*. The *x*-axis corresponds to the sequential changes of the angles ϕ and ψ through the temporal steps. The values on the right correspond with the *score3* values of the PDB native structure

sheet, it depends on the number of protein sheets). It is difficult for these terms to be activated. The reason for this is that the dihedral angles must have very precise values to activate the term (e.g., very close to the angles corresponding to a sheet), which explains the constant value or the sudden appearance of different discrete values. For example, with β protein PDB:1E0M, if the protein starts with the unfolded conformation, the optimized ANN is not able to correctly establish the perfect values to activate the energy terms related to strands: *sheet*, which favors the grouping in sheets of individual beta strands; *rsigma*, which scores the pairs of strands according to the distance between them and the register of the two strands; and *ss_pair* (hydrogen bonding between beta strands) [29]. Therefore, these present a constant value (like zero in the case of *rsigma* and *ss_pair*). However, starting from a partially folded conformation, the terms are activated, since the secondary structures (with the corresponding dihedral angles) are better established. This can be seen with protein PDB:1E0M and option *ii* in the subfigure on the right, where these terms present perturbations during the folding process. With α protein PDB:5WOD, these terms logically have a zero value, while the other energy terms drive the ANN folding process to low-energy conformational areas. Finally, with $\alpha - \beta$ protein PDB:1D5Q, the term *hs_pair* (helix-strand packing) presents oscillations with a final value close to the value of the PDB native structure and in both cases (options *ii* and *iii*).

Figure 4 shows an example of the final folded conformations with protein PDB:5WOD, applying the evolved ANN with the three options to the unfolded conformation, which shows a better adjustment of the final folded structure to the PDB native conformation with options *i* and *iii* (as discussed below) but without setting the angles precise enough to determine the helices.

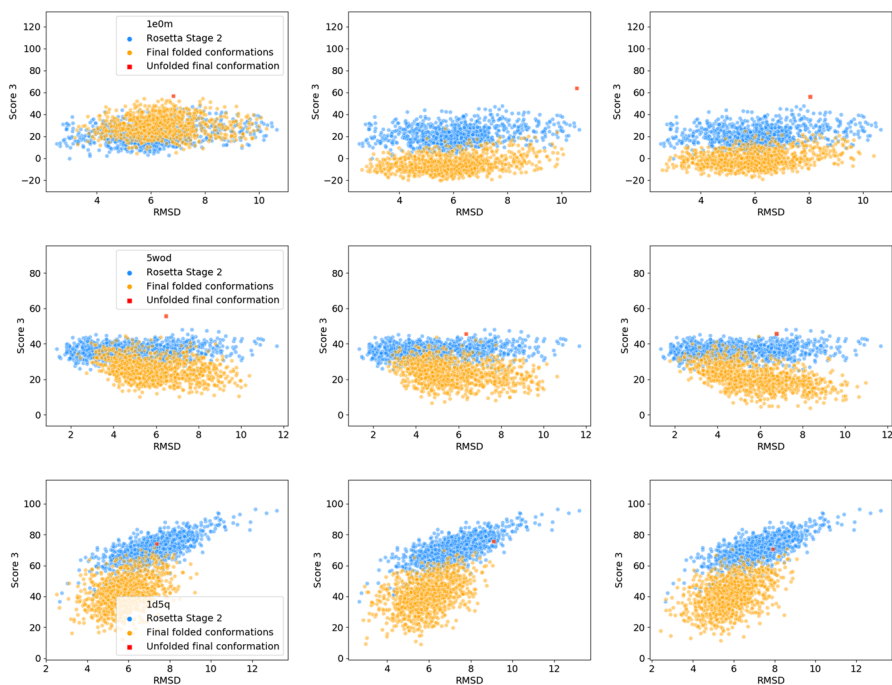


Fig. 6 Energy (*score3*, axis *y*) vs. RMSD (from the native structure, axis *x*, in Å), of the final folded conformations (yellow) after application of the optimized ANN to 1,000 initial conformations with protein PDB:1E0M (upper figures), protein PDB:5WOD (figures in the middle row) and PDB:1D5Q (bottom figures). These starting structures correspond to 1,000 Rosetta solutions at the end of Stage 2 (blue). The point in red corresponds to the final folded conformation when the ANN is applied to the initial unfolded conformation. Left: ANN evolved considering only the initial unfolded conformation (option *i*); Center: ANN evolved considering three initial partially folded conformations (option *ii*); Right: ANN evolved considering the unfolded conformation and three initial partially folded conformations (option *iii*) (Color figure online)

The dihedral angles could be better refined by adding a new term to the *score3* energy, which would measure the correspondence between the predicted SSE (e.g., using the predictor PSIPRED [28]) and the resulting SSE from the current angles of the structure being folded, as was done in [22, 41]. This term would add a reinforcement value to favor solutions that have predicted secondary structures that match the folding structure, that is, to benefit well-formed secondary structures [22]. However, this possibility was not considered initially, as it would require the experimental tuning of the weights associated with the *score3* energy terms and the new reinforcement term of SSE correspondence.

Figure 5 shows the evolution of *score3* in the protein conformations obtained through the folding process, starting with the unfolded conformation, when the best ANN optimized with option *iii* is used. Note that Rosetta energy *score3* was also used as fitness to optimize the corresponding neural-CA for each protein. The *score3* energy is plotted after the sequential application of the ANN to both angles (ϕ and ψ) of the amino acids. The number of angle moves is different in the three

proteins due to their different number of amino acids (and consequently, the number of angles to move in the maximum of 20 steps of the ANN application). Starting with the unfolded conformation, Figure 5 shows how the structure folds progressively towards lower energy regions. Moreover, Figure 5 shows that the evolved ANN does not consider a purely greedy strategy, which would always select angle changes that decrease the *score3* energy. On the contrary, there are fluctuations in energy, with some changes increasing energy that are compensated for in following angle changes in order to progress to areas of low energy. That is, the ANN can provide the progression of the structure, through the funnel-like and rugged protein energy landscape [46], towards the required low-energy areas.

3.3 Folding process starting with different initial conformations

The following experiment checks how the folding process defined by the optimized ANN can also provide a refinement of an initial structure. For this, 1,000 initial partially folded conformations were employed, these provided by running Rosetta *ab initio* protocol 1,000 times and selecting the structures at the end of Stage 2 (standard parameters in the Rosetta protocol [30]).

Figure 6 shows the results of the refinement of these 1,000 initial structures after application of the evolved ANN for each protein. Three cases were considered applying three optimized ANNs, which were evolved with the three options explained above (Sect. 3.1). Figure 6 shows the energy value (*score3*) of each conformation on axis *y*. The distance from each conformation to the native structure is measured by the Root Mean Square Deviation (RMSD) (in Å), which is shown on axis *x*. The RMSD is measured considering the positions of the C-alpha atoms of both structures (candidate conformation and native structure). This standard graph provides the information necessary to assess the distribution of distances (RMSD) of the optimized protein conformations, along with the optimization (in energy terms) obtained in those final solutions.

Rosetta *ab initio* protocol is stochastic, providing the 1,000 partially folded conformations shown (in blue) in Fig. 6. Note the deceptiveness of the Rosetta energy model, in the sense that the best energy conformations do not necessarily have to correspond to those closest to the native one. This problem of deceptiveness is well-known. For example, Shmygelska and Levitt [36] point out some of the deficiencies of the Rosetta energy model, which presents false local minima and general flatness in the energy landscape in the area close to the native states.

When option *i* is used (Fig. 6, left), the application of the evolved ANN provides a general improvement in the energy of the initial conformations in proteins PDB:5WOD and PDB:1D5Q. For protein PDB:1E0M, the ANN even worsens the energy in some initial conformations. It must be taken into account that the ANN was trained/optimized to provide the folding process starting only with the unfolded conformation (option *i*). In this case, the ANN has to learn to apply large angle changes (in the limited and allowed range) in order to fold the initial unfolded conformation, where large structural changes are needed to provide non-zero energy changes (inputs to the ANN), so that the ANN can decide the appropriate angle

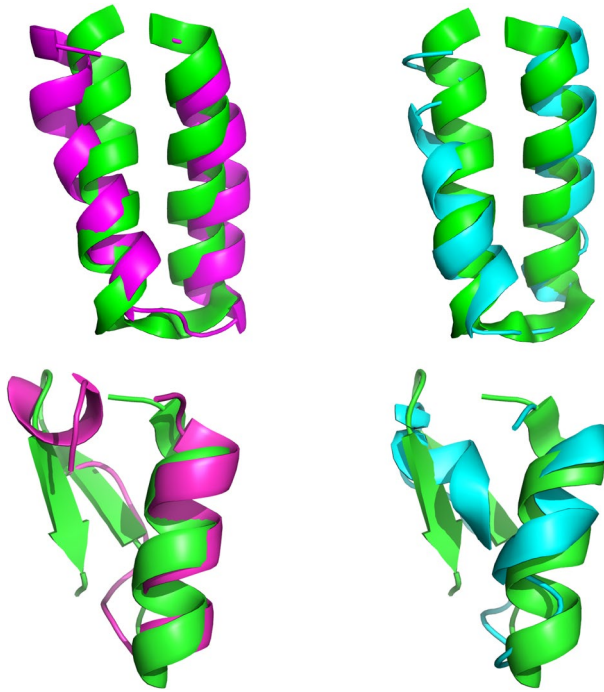


Fig. 7 Snapshots of folded structures with proteins PDB:5WOD (upper figures) and PDB:1D5Q (bottom figures). Left: Initial starting conformation (pink) at the end of Rosetta Stage 2. Right: Final optimized conformation (blue) after the ANN refinement. The native structure is shown in green (Color figure online)

change in each situation. Nevertheless, the ANN was not optimized to provide smoother angle changes for more detailed refinement and, consequently, some partially folded initial conformations could be unrefined.

On the contrary, with option *ii* (Fig. 6, center), the ANN does not need to learn how to fold the protein when it is initially unfolded. Therefore, it can be evolved to provide better refinement of the initial partially folded structures. In this case, there is a general improvement in the energy of the initial conformations and in all proteins. However, note that the refinement of the initial unfolded conformation (red point in subfigures) is worse compared to the previous option in proteins PDB:1E0M and PDB:1D5Q, especially in RMSD terms with protein PDB:1E0M.

Finally, option *iii* (Fig. 6, right) provides a tradeoff between the previous options, since the ANN is evolved to provide the folding process starting with both the unfolded and partially folded conformations. There is, again, a general improvement in the energy of the final solutions and, at the same time, this alternative provides, for the initial unfolded conformation, a final solution close to that obtained with option *i*.

There is one final consideration to highlight. The improvement in energy in the initial conformations has a different behavior in terms of RMSD in the three proteins. With protein PDB:1D5Q, the improvement in energy tends to correspond

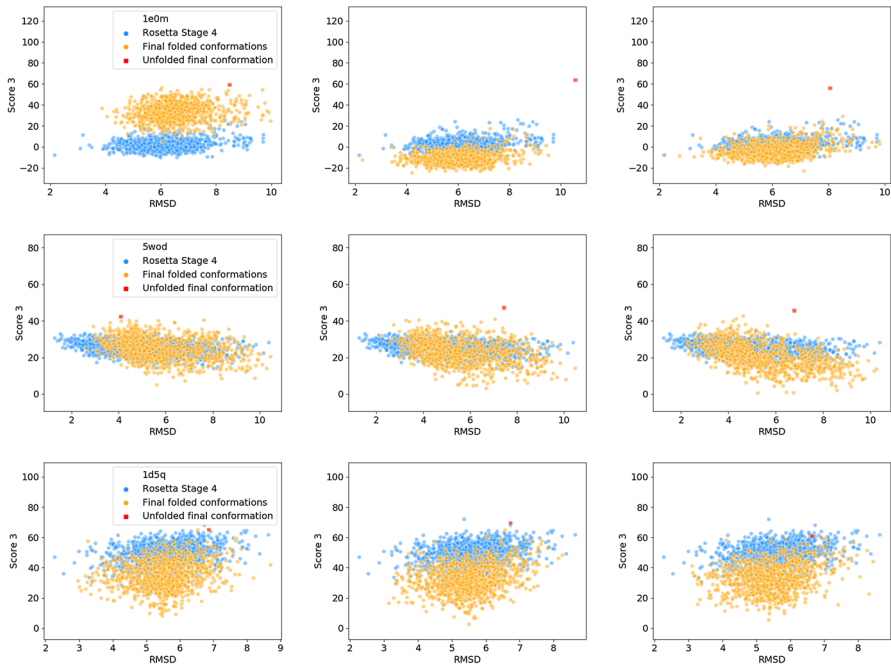


Fig. 8 Energy (*score3*, axis *y*) vs. RMSD (from the native structure, axis *x*, in Å), of the final folded conformations (yellow) after application of the optimized ANN to 1000 initial conformations with protein PDB:1E0M (upper figures), protein PDB:5WOD (figures in the middle row) and PDB:1D5Q (bottom figures). These starting structures correspond to 1000 Rosetta solutions at the end of Stage 4 (blue). The point in red corresponds to the final folded conformation when the ANN is applied to the initial unfolded conformation. Left: ANN evolved considering only the initial unfolded conformation (option *i*); Center: ANN evolved considering three initial partially folded conformations (option *ii_v2*); Right: ANN evolved considering the unfolded conformation and three initial partially folded conformations (option *iii_v2*) (Color figure online)

with solutions closer to the native structure. With protein PDB:1E0M, the energy improvement provided by the modeled folding process does not worsen the RMSD distribution of the final folded structures. However, with protein PDB:5WOD, the RMSD distributions are narrower, and the initial Rosetta conformations closest to the native structure are further away after the ANN application to them. This is a consequence of the energy landscape deceptiveness for this protein, where there is an area corresponding to local minima (around 5–10 Å in RMSD terms) that attracts the changing structures during the folding process.

Figure 7 shows an example of the refinement provided by the folding process with proteins PDB:5WOD and PDB:1D5Q, using the best evolved ANN with option *iii*. Figure 7 includes snapshots of an initial conformation at the end of Rosetta Stage 2 (left part), superimposed on the native structure. The initial structures selected are those that provide the best RMSD value in the final refined structure. With protein PDB:1D5Q, the initial structure has a *score3* value of 65.01 and an RMSD value of 4.77 Å. The optimized result, after the folding provided by the ANN, is shown on the right. This final folded and refined structure has a *score3* value of 17.59 and

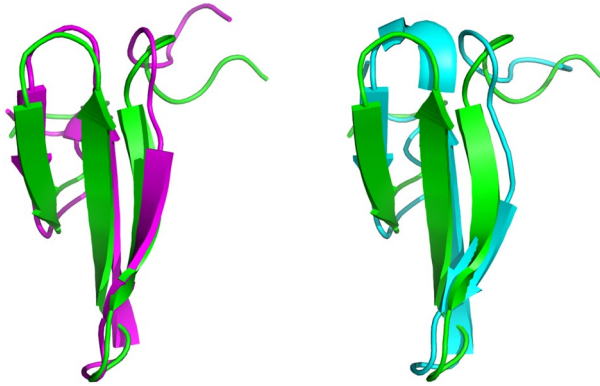


Fig. 9 Snapshots of folded structures with protein PDB:1E0M. Left: Initial starting conformation (pink) at the end of Rosetta Stage 4. Right: Final optimized conformation (blue) after the ANN refinement. The native structure is shown in green (Color figure online)

also a lower RMSD value of 2.44 Å with respect to the initial structure, showing the improvement provided by the ANN process, although without obtaining the correct angles to determine the sheets in both cases. Similarly, with protein PDB:5WOD, its initial structure has a *score3* value of 45.01 and an RMSD value of 3.18 Å. The refined structure has a *score3* value of 30.25 and also a lower RMSD value of 1.77 Å with respect to the initial structure (although, for this protein, the RMSD of many refined conformations is worse with respect to their starting conformations, due to the energy landscape deceptiveness, as discussed above).

Finally, to verify the detailed refinement capability provided by the optimized ANNs, the previous experiment was repeated with the same options, except that the initial partially folded conformations correspond to 1000 folded structures at the end of Rosetta Stage 4 (this final stage with fewer fragment insertion attempts than Rosetta *ab initio*, since parameter *increase_cycles* was set to 0.01 at that stage, Sect. 2.1). Three additional folded structures at the end of Rosetta Stage 4 were used to evolve/optimize the ANN. Thus, we called options *ii* and *iii* as their second version (*ii_v2* and *iii_v2*), to emphasize that these are just a variant as only the 3 partially folded structures (used to evolve the ANN) are exchanged for others that have been more refined with Rosetta. Figure 8 includes the results with the neural-CA optimized with option *i* and these new options (*ii_v2* and *iii_v2*).

First, the 1,000 Rosetta solutions at the end of Stage 4 are logically better optimized in energy terms with respect to the previous case with partially folded structures at the end of Stage 2. This can be clearly seen by comparing the initial Rosetta solutions between Figs. 6 and 8 and in all proteins. Nevertheless, the ANN can refine such conformations again towards areas of lower energy, and pointing out the same considerations as in the previous case: 1) when evolving the ANN with option *i* (only considering the unfolded structure), the folding process cannot appropriately refine the partially folded structures in protein PDB:1E0M; 2) the option that uses as starting conformations, for the optimization of the ANN, partially folded structures and the unfolded one (option *iii_v2*) presents the best tradeoff to provide the folding

and refinement of different starting conformations; 3) finally, the previous comment regarding the deceptiveness of the energy landscape with protein PDB:5WOD can be reiterated here, since the area of best energy drives the folding structure further away from the native structure.

Figure 9 includes a final example of this refinement provided by the ANN process with protein PDB:1E0M. Figure 9 includes snapshots of an initial conformation at the end of Rosetta Stage 4 (left part), superimposed on the native structure. The initial structure has a *score3* value of -7.96 and an RMSD value of 2.16 \AA . The best ANN, trained with option *ii_v2*, was used for the refinement of the initial structure. The final folded and refined structure (right part) has a *score3* value of -12.52 and a nearly equal RMSD value of 2.29 \AA with respect to the initial structure. Similar results are obtained with option *iii_v2*, which shows the improvement provided by the ANN process, although it is somewhat minor with respect to the previous case, since the starting structures are now better refined.

4 Discussion and conclusions

In this study, protein folding was considered as an emergent process, and therefore one that can be modeled with classical tools for the study of emergent behavior, such as cellular automata. Nevertheless, instead of using classical CA implemented with rule sets, feed-forward neural networks were used to implement the CA (neural-CA). Thus, the cellular automaton model incorporates the ANN generalization property, and the ANN can be thought of as a black box that encodes the classical CA rule set. Another difference is that the information used by neural-CA is obtained from the energy landscape, rather than from the spatial neighborhood of a grid site element to which the classical CA rule set is applied. The general process can also be considered similar to the processes of developing complex phenotypes from compact genotypes. This parallelism comes from the fact that the final folded structure can be seen as the final phenotype defined by the ANN genotype that guides the development (folding process) of the final phenotype.

Using the low-level representation model of Rosetta, an evolved neural cellular automaton sequentially modifies two of the amino acid dihedral angles (ϕ and ψ), from the beginning to the end of the protein chain and, as in classical CA, this process is iterated over time. Validation is performed by testing whether the proposed methods can provide the final conformations starting with the unfolded (or partially folded) conformation, since there is no experimental information (resolved conformations) corresponding to an ordered sequence of temporal states of a protein. The experiments reported with evolved neural-CA for short proteins showed that optimized neural cellular automata can model the protein folding process using only information from the energy landscape, notwithstanding the defects of the Rosetta energy function with its low-resolution representation. Therefore, contrary to the molecular dynamics alternative [7], which uses a priori modeling of the physico-chemical interactions of the protein elements, machine learning methods have made it possible to automatically obtain a model for the folding process, this from the

available data of resolved proteins. Consequently, our aim was not to compete with PSP methods, but rather to test machine learning and how it might provide an automatic model of the folding process.

Moreover, the modeling of the folding process with evolved neural-CA can serve to refine or improve an initial structural model, as was done with the initial partially folded structures here. This aspect of refinement is considered a new development and constitutes progress since CASP11 (Critical Assessment of Structure Prediction) in terms of the state of the art in structure modeling [21]. When the ANN is optimized to provide the folding of the initial unfolded conformation and also partially folded structures, the results show the appropriate tradeoff to obtain a compact low-energy fold for that fully unfolded conformation, as well as a refinement of other partially folded structures. This refinement can be especially important for unresolved proteins, where the CA models can provide refined structures that minimize conformational energy (e.g., as mentioned in the Introduction, from initial conformations predicted with PSP methods). In this use of neural-CA for refinement, it is worth noting the difference with respect to PSP protocols based on protein fragments (such as Rosetta *ab initio*), since information from resolved proteins (or in the form of small fragments) is not used. Because only the primary sequence information and the energy landscape are considered, the folding model follows a pure *ab initio* procedure.

In addition, the results show that, if the energy landscape is deceptive, the refinement (in energy terms) that the optimized ANN is able to provide might also, at the same time, move the initial structure away from the native conformation (in RMSD terms). When the aim is only the prediction of the final folded structure (PSP), the strategy to tackle the deceptiveness of the energy model is to try to provide conformational models (decoys) that minimize energy but which, at the same time, present a diversified structural distribution. The use of niching methods in evolutionary computation is a straightforward means of addressing the problem [44], since these niching methods enforce the distribution of the population (in this case of PSP, possible protein conformations) in different areas or niches of the energy landscape corresponding to different energy minima (and possibly to structural variants close to the native structure). However, this is not the objective in the present study, since the ANN only uses information from the (imperfect) Rosetta energy landscape to fold the structures towards low-energy areas. The ANN process is deterministic, but if some stochasticity were to be considered, the ANN process could be considered as another *ab initio* PSP strategy, with the possibility of forcing such diversified structural variants in the resulting protein decoys.

Some potential extensions of the current study might be undertaken as future work in this modeling with the Rosetta model. First, it should be noted that the energy changes that are used as inputs to the neural-CA take into account the effect of the angle perturbations in the complete chain (Fig. 2). This implies that it is difficult that an evolved neural cellular automaton determines an angle change that decreases energy in the local neighborhood if the effect on the whole chain is negative (increase in energy), although it would be possible that following changes might repair the problems (e.g., clashes) that cause the increase in energy in areas some

distance away from the angle where the ANN was applied. This limitation arises from the constraints of Rosetta with its energy model, since it is not possible to calculate energy increases (after a perturbation) in the local environment of an amino acid (the energy score can only be considered with the complete protein chain). As the effect of the perturbations caused to calculate the energy increases (inputs to the ANN) are considered with the complete protein chain, consequently the evolved ANN does not consider changes that, for example, produce conflicts in areas away from the angle that is modified. This is the main limitation of working with the (knowledge-based) Rosetta energy model, so the folding provided by the evolved ANN must act very conservatively, restricting the possibilities of determining other folding pathways.

If the energy changes were considered by defining a neighborhood (e.g., a sphere) around the angle to which the artificial neural network is applied (similarly to classical CA), this problem could be overcome with more possibilities in the definition of folding pathways and, consequently, with more flexibility in folding pathways for large proteins. The chosen proteins correspond to different topologies and are sufficient to observe the behavior of the evolved CA when modeling the interactions of the different and common elements of the secondary structure. Therefore, the current study demonstrates the possibility of modeling the folding process with the proposed approach, but with the main restriction of modeling with short proteins. Thus, the proteins considered in the experiments are not exhaustive, and further testing is needed. For this, it will be necessary to extend the methods to energy models without the aforementioned restriction, which will allow the same modeling with longer proteins.

Furthermore, since neural-CA were evolved for a particular protein, generalizability could be tested when these evolved neural-CA are applied to different proteins. That is, a neural cellular automaton was evolved to provide the folding process for a single protein, but the same ANN can be evolved/optimized to provide the folding process for a set of different proteins (training set), while validating the ANN processing with a different set of proteins not used in the optimization process (validation or test set). This would require energy normalization for different proteins. However, it is worth noting the importance of this property of optimized neural-CA, as an optimized ANN could be considered as a “folding operator” that can provide the folding process of any protein and independently of its length.

Finally, an interesting idea is that our modeling of the dynamic folding process can provide transitions of a protein between different structures at equilibrium, e.g., transitions between healthy and pathogenic variants of proteins involved in diseases. In previous approaches, for example Sapin et al. [35], an evolutionary algorithm was used to evolve path representations of structural transitions in proteins. Contrary to this approach of finding a particular transition path between each pair of protein variants at equilibrium, our modeling of the folding process can provide these structural transitions in a more direct way, if the neural-CA model is evolved to provide such transitions through the energy landscape.

Therefore, the ideas developed in this study are not intended to compete with PSP, but rather to integrate the ideas used in Artificial Life in the development of final complex phenotypes and in the modeling of emergent processes for the modeling of

the folding process of proteins. The ideas presented serve to lay the foundations for a suitable methodology when using machine learning to automatically obtain a model of the folding process, one that would be valid for any protein.

Acknowledgements This study was funded by the Xunta de Galicia and the European Union (European Regional Development Fund - Galicia 2014-2020 Program), with grants CITIC (ED431G 2019/01), GPC ED431B 2019/03 and IN845D-02 (funded by the “Agencia Gallega de Innovación”, co-financed by Feder funds), and by the Spanish Ministry of Science and Innovation (project PID2020-116201GB-I00).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. C. Anfinsen, Principles that govern the folding of proteins. *Science* **181**(96), 223–230 (1973)
2. R. Calabretta, S. Nolfi, D. Parisi, An artificial life model for predicting the tertiary structure of unknown proteins that emulates the folding process, in Proceedings Third European Conference on Advances in Artificial Life - LNCS Vol 929, (1995), pp. 862–875
3. E. Callaway, The protein-imaging technique taking over structural biology. *Nature* **578**, 201 (2020)
4. G. Danks, S. Stepney, L. Caves, Protein folding with stochastic L-systems, in Artificial Life XI: Proceedings of 11th International Conference on the Simulation and Synthesis of Living Systems (MIT Press, 2008), pp. 150–157
5. S. Das, P. Suganthan, Differential evolution: a survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **15**(1), 4–31 (2011)
6. S. Englander, L. Mayne, The nature of protein folding pathways. *Proc. Natl. Acad. Sci.* **111**(45), 15873–15880 (2014)
7. M. Feig, V. Mirjalili, Protein structure refinement via molecular-dynamics simulations: what works and what does not? *Proteins Suppl* **1**, 282–292 (2016)
8. V. Feoktistov, *Differential evolution: In search of solutions* (Springer, Berlin, 2006)
9. M. Garza-Fabre, S. Kandathil, J. Handl, J. Knowles, S. Lovell, Generating, maintaining, and exploiting diversity in a memetic algorithm for protein structure prediction. *Evol. Comput.* **24**(4), 577–607 (2016)
10. A. Hagler, S. Lifson, Energy functions for peptides and proteins, II: the amide hydrogen bond and calculation of amide crystal properties. *J. Am. Chem. Soc.* **96**, 5319–5327 (1974)
11. A. Ilichinski, *Cellular automata. A discrete universe* (World Scientific, Singapore, 2001)
12. K. Kaufmann, G. Lemmon, S. DeLuca, J. Sheehan, J. Meiler, Practically useful: what the Rosetta protein modeling suite can do for you. *Biochemistry* **49**, 2987–2998 (2010)
13. S. Kmiecik, D. Gront, M. Kolinski, L. Wieteska, A. Dawid, A. Kolinski, Coarse-grained protein models and their applications. *Chem. Rev.* **116**, 7898–7936 (2016)
14. N. Krasnogor, G. Terrazas, D. Pelta, G. Ochoa, A critical view of the evolutionary design of self-assembling systems. Proceedings of the 2005 Conference on Artificial Evolution, LNCS **3871**, 179–188 (2006)
15. S. Kriegman, N. Cheney, J. Bongard, How morphological development can guide evolution. *Sci. Rep.* **8**, 13934 (2018)

16. V. Krishnan, B. Rupp, *Macromolecular structure determination: comparison of X-ray crystallography and NMR spectroscopy* (Wiley, Hoboken, 2012). <https://doi.org/10.1002/9780470015902.a0002716.pub2>
17. J. Lee, S. Wu, Y. Zhang, *Ab initio protein structure prediction, in From Protein Structure to Function with Bioinformatics* (Springer, London, 2009), pp. 3–25
18. C. Levinthal, Are there pathways for protein folding? *J. Chim. Phys.* **65**, 44–45 (1968)
19. A. Márquez-Chamorro, G. Asencio-Cortés, C. Santiesteban-Toca, J. Aguilar-Ruiz, Soft computing methods for the prediction of protein tertiary structures: a survey. *Appl. Soft Comput.* **35**, 398–410 (2015)
20. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**(6), 1087–1092 (1953)
21. J. Moulton, K. Fidelis, A. Kryzhtafovich, T. Schwede, A. Tramontano, Critical assessment of methods of protein structure prediction: progress and new directions in round XI. *Proteins: Struct., Funct., Bioinform.* **84**(1), 4–14 (2016)
22. P. Narloch, M. Dorn, A knowledge based differential evolution algorithm for protein structure prediction, in *Proceedings International Conference on the Applications of Evolutionary Computation*, pp. 343–359 (2019)
23. F. Noé, G. De-Fabritiis, C. Clementi, Machine learning for protein folding and dynamics. *Current Opin. Struct. Biol.* **60**, 77–84 (2020)
24. B. Olson, K. De-Jong, A. Shehu, Off-lattice protein structure prediction with homologous crossover, in *Proceedings Conference on Genetic and Evolutionary Computation - GECCO 2013* (2013) pp. 287–294
25. S. Patodia, A. Bagaria, D. Chopra, Molecular dynamics simulation of proteins: a brief overview. *J. Phys. Chem. Biophys.* **4**(6), 166 (2014)
26. Protein Data Bank. <http://www wwPdb.org>
27. K. Price, R. Storn, J. Lampinen, *Differential evolution. A practical approach to global optimization* (Springer, Berlin, 2005)
28. PSIPRED protein sequence analysis workbench. <http://bioinf.cs.ucl.ac.uk/psipred/>
29. C. Rohl, C. Strauss, K. Misura, D. Baker, Protein structure prediction using Rosetta. *Methods Enzymol.* **383**, 66–93 (2004)
30. Rosetta system. <http://www.rosettacommons.org>
31. J. Santos, M. Diéguez, Differential evolution for protein structure prediction using the HP model. *Lecture Notes Comput. Sci.* **6686**, 323–323 (2011)
32. J. Santos, P. Villot, M. Diéguez, Cellular automata for modeling protein folding using the HP model, in *Proceedings IEEE Congress on Evolutionary Computation - IEEE-CEC 2013* pp. 1586–1593 (2013)
33. J. Santos, P. Villot, M. Diéguez, Protein folding with cellular automata in the 3D HP model, in *ACM Proceedings International Workshop on Evolutionary Computation in Bioinformatics - BIO 2013 - Genetic and Evolutionary Computation Conference (GECCO 2013)*, pp. 1595–1602 (2013)
34. J. Santos, P. Villot, M. Diéguez, Emergent protein folding modeled with evolved neural cellular automata using the 3D HP model. *J. Comput. Biol.* **21**(11), 823–845 (2014)
35. E. Sapin, A. Shehu, K. De Jong, An evolutionary algorithm to model structural excursions of a protein, in *Proceedings of the ACM Workshop Evolutionary Computation in Computational Biology, GECCO-Genetic and Evolutionary Computation Conference (2017)*, pp. 1669–1673
36. A. Shmygelska, M. Levitt, Generalized ensemble methods for de novo structure prediction. *PNAS* **106**(5), 1415–1420 (2009)
37. Software to model the protein folding process with Rosetta. <https://github.com/danielvarela/ProteinFoldCA>
38. A. Tramontano, *Protein structure prediction. Concepts and applications* (Wiley, Hoboken, 2006)
39. R. Unger, The genetic algorithm approach to protein structure prediction. *Struct. Bond.* **110**, 153–175 (2004)
40. Universal protein resource (uniprot). <https://www.uniprot.org>
41. D. Varela, J. Santos, Protein folding modeling with neural cellular automata using Rosetta, in *GECCO 2016 ACM Proceedings Companion, Workshop Evolutionary Computation in Computational Structural Biology* pp. 1307–1312 (2016)
42. D. Varela, J. Santos, A hybrid evolutionary algorithm for protein structure prediction using the Face-Centered Cubic lattice model, in *Proceedings International Conference on Neural Information Processing - ICONIP 2017, Lecture Notes in Computer Science 10634* pp. 628–638 (2017)

43. D. Varela, J. Santos, Automatically obtaining a cellular automaton scheme for modeling protein folding using the FCC model. *Nat. Comput.* **18**, 275–284 (2019)
44. D. Varela, J. Santos, Protein structure prediction in an atomic model with differential evolution integrated with the crowding niching method. *Nat. Comput.* (2020). <https://doi.org/10.1007/s11047-020-09801-7>
45. P. Wolynes, J. Onuchic, D. Thirumalai, Navigating the folding routes. *Science* **267**, 1619–1620 (1995)
46. X. Zhao, Advances on protein folding simulations based on the lattice HP models with natural computing. *Appl. Soft Comput.* **8**, 1029–1040 (2008)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.