JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 21, Number 11, 2014

Pp. 823–845
DOI: 10.1089/cmb.2014.0077

Emergent Protein Folding Modeled with Evolved Neural Cellular Automata Using the 3D HP Model

JOSÉ SANTOS, PABLO VILLOT, and MARTIN DIÉGUEZ

ABSTRACT

We used cellular automata (CA) for the modeling of the temporal folding of proteins. Unlike the focus of the vast research already done on the direct prediction of the final folded conformations, we will model the temporal and dynamic folding process. To reduce the complexity of the interactions and the nature of the amino acid elements, lattice models like HP were used, a model that categorizes the amino acids regarding their hydrophobicity. Taking into account the restrictions of the lattice model, the CA model defines how the amino acids interact through time to obtain a folded conformation. We extended the classical CA models using artificial neural networks for their implementation (neural CA), and we used evolutionary computing to automatically obtain the models by means of Differential Evolution. As the iterative folding also provides the final folded conformation, we can compare the results with those from direct prediction methods of the final protein conformation. Finally, as the neural CA that provides the iterative folding process can be evolved using several protein sequences and used as operators in the folding of another protein with different length, this represents an advantage over the NP-hard complexity of the original problem of the direct prediction.

1. INTRODUCTION AND PREVIOUS WORK

Proteins are Chains of amino acid residues that fold into native 3D structures under natural conditions, just after being synthesized in the ribosomes. The thermodynamic hypothesis states that this native conformation of the protein is the one with lowest Gibbs free energy. That native structure is independent of the starting conformation and determines its biological function. As experimental determination of the native conformation is still difficult—by time-consuming and relatively expensive X-ray crystallography or NMR spectroscopy—much work has been done to forecast the native conformation computationally. Along this line, there are numerous works on the direct prediction of the conformations of the final protein structure, both secondary (local regular elements such as helices and sheets) and tertiary structures.

In the case of the prediction of the final tertiary structure, such methods range from comparison methods with resolved structures to the ''ab initio'' prediction (Tramontano, 2006; Zaki and Bystroff, 2008). In the first case, in the homology modeling method, the search space is pruned by the assumption that the target

Department of Computer Science, University of A Coruña, A Coruña, Spain.

protein adopts a structure close to the experimentally determined structure of another homologous protein. The method of protein threading is based on the observation that many protein structures in the Protein Data Bank (PDB) are very similar. Thus, this approach to protein structure prediction tries to determine the structure of a new sequence by finding its best fit to some fold in a library of structures. So, these methods depend on the templates (or structures) of known proteins. Nevertheless, the experimental determination of the protein conformations remains far behind the rapid increase of the number of known protein sequences. The Critical Assessment of Techniques for Protein Structure Prediction (CASP, 2014) competition assesses the state of the art in protein structure modeling with different modalities in the tertiary structure predictions, including the free modeling or ab initio prediction. As also indicated in the proceedings of CASP9 (the last edition was CASP10 in 2012, the next one CASP11 in 2014), ''Methods for modeling protein structure continue to advance, although at a more modest pace than in the early CASP experiments'' (Moult et al., 2011).

The most difficult ab initio prediction is a challenge in computational biology. It uses only the information from the amino acid sequence of the primary structure (Tramontano, 2006). In such prediction there are models that simplify the complexity of the interactions and the nature of the amino acid elements, like the models that locate these in a lattice, or detailed atomic models like the Rosetta system (RS, 2014). In the first case, simplified or minimalist models are used. The use of a reduced alphabet of amino acids in minimalist models is based on the recognition that hydrophobic interactions are a dominant force in protein folding, and that the binary pattern of hydrophobic and polar residues is a major determinant of the folding of a protein. For instance, in the HP model (Dill, 1990) (detailed in section 2.1) the elements of the chain can be of two types: H (hydrophobic) and P (polar). To mimic hydrophobic interactions, each nearest-neighbor contact between two H monomers (not consecutive along the primary sequence) is assigned a favorable energy (-1), irrespective of whether the contact is in the native structure or not, while other contacts are modeled as neutral. Given a primary sequence, the problem is to search for the folding structure in the lattice that minimizes the energy.

Even with these simplifications, the complexity of the problem has been shown to be NP-hard (Hart and Istrail, 1997; Unger and Moult, 1993b; Berger and Leighton, 1998), so determining the native state conformations of HP sequences has become a computational challenge and, as Unger pointed out, ''minimal progress was achieved in the category of ab initio folding'' (Unger, 2004). Moreover, although the HP model is simple, it is nontrivial, captures many global aspects of real proteins and still remains the hardness features of the original biological problem (Dill et al., 1995). For this reason, many authors have been working on different local search methods like Tabu Search (Lesh et al., 2003; Blazewicz et al., 2004) or Large Neighborhood Search (Dotu et al., 2011), as well as several evolutionary algorithms (Garza-Fabre et al., 2012a; Patton et al., 1995; Santos and Diéguez, 2011; Unger, 2004; Unger and Moult, 1993b) or other natural computing algorithms (Cutello et al., 2007; Fidanova, 2006; Shmygelska and Hoos, 2005; Song et al., 2006; Zhao, 2008) on the direct prediction of the native conformations using the HP model, as we describe in greater detail in the next section.

We address here the problem of protein folding modeling, with the complex interactions between the amino acids of the primary structure, considering it as an emergent result of a dynamic process. The emergent behavior property was studied in Artificial Life (AL) with methods like Cellular Automata (CA) and Lindenmayer systems (Ilachinski, 2001; Langton, 1992). We will use CA to define the folding of a protein through time in the three-dimensional space of possible conformations imposed by simple lattice models like the HP model, following our initial work in the two-dimensional case (Santos et al., 2013). CAs have been the focus of attention because of their ability to generate a rich spectrum of complex behavior patterns out of sets of relatively simple underlying rules, and they appeared to capture many essential features of complex self-organizing cooperative behavior observed in real systems (Ilachinski, 2001).

So, unlike the focus of the vast research already done on the direct prediction of the secondary or tertiary structures of the final folded conformations, our interest here is different in that we will model the temporal and dynamic folding process. The AL methods will define how the amino acids interact through time to obtain a folded conformation. We will extend the classical CA models using artificial neural networks for their implementation, and we will use evolutionary computing to automatically obtain the models.

In the next two subsections of this introduction we summarize the previous work on the direct prediction of the final protein conformation using the HP model (section 1.1) and on protein folding modeling (section 1.2). In section 2 we present the methods used: The HP model for protein conformation representation; the neural-CA scheme used to model the folding process in the HP lattice, together with the evolutionary

algorithm (Differential Evolution) used to automatically obtain the CA models; and the benchmark protein sequences used to analyze the methodology. In section 3 we expose the setup for the folding with the details of a folding example, together with a comparison between the final results after the folding process using our methodology and previous works that obtained the final folding with search methods for the direct and final folded conformation. In section 3 we focus our analysis on the generalization results when several benchmark sequences are used for the training/evolution of a cellular automaton, whereas it is validated in different sequences. Finally, section 4 summarizes the conclusions with a short discussion of the ideas exposed.

1.1. Comments about previous work on the direct prediction of the final protein conformation

Many authors have been working on the use of search methods, especially evolutionary and natural computing algorithms, for determining the final protein conformation and using lattice models like the HP model. The folded sequences are typically encoded as a set of relative or absolute movements of the different amino acids in the lattice (section 2.1). As fitness function, most authors used the original definition of the HP model (Dill, 1990), which considers the local contacts between two hydrophobic amino acids. However, some authors included modifications in the usual fitness function. For instance, Custódio et al.'s modification (Custódio et al., 2004) was based on the assumption that it may be preferable for a hydrophobic monomer to have a polar neighbor than to be in direct contact with the polar solvent, whereas Lopes and Scapin (2006) introduced a term in the fitness function, called radius of gyration, which indicates how compact a set of amino acids is, thus rewarding more compact conformations. The work of Garza-Fabre et al. (2013a) analyzes several fitness functions used for the HP model.

The first works used a genetic algorithm (GA). For example, Unger and Moult (1993b) considered only feasible conformations in the genetic population, that is, embeddings that are self-avoiding paths on the considered lattice. In their work, when operators such as mutation and crossover were applied, the GA iterated until a ''nonlethal'' conformation was generated. Similarly, Rylance (2004) obtained the initial population using a recoil growth algorithm. Also, Song et al. (2005) applied the genetic operators used in Cox et al. (2004) for their GA, with six types of mutations. According to the authors, one of them (cornerchange, which mutates one shape to another shape) gives the conformation more biology significance.

On the contrary, Patton et al. (1995) allowed illegal conformations using a penalty in the fitness to avoid self-avoiding constraints, allowing the search to proceed through illegal states. Krasnogor et al. (1999) analyzed the impact of different factors when evolutionary algorithms are applied to the problem: the conformational representation, the energy formulation, and the way in which infeasible conformations are penalized. Their results supported the use of relative encoding. Although the crossover operator is used in the majority of works with GAs in the application, it must be noted that the crossover operator is not transferring building blocks to the offspring; as Krasnogor et al. (1998) commented, ''the problem with the crossover is closely related to the highly epistatic representation of individuals.''

Other authors used hybrid combinations of a local search with the evolutionary algorithm. For example, Cotta (2003) used a GA hybridized with a backtracking algorithm or repair procedure that maps infeasible solutions to feasible conformations. The combination provided slightly better results, with respect to a penalty-based approach and to a feasible-space approach. Krasnogor et al. (2002a) also used ''multimeme'' algorithms for the problem when a set of local searchers are used in the individuals of the genetic population. We previously used Differential Evolution (DE) for the problem, combining DE with methods to transform illegal protein conformations to feasible ones (Santos and Diéguez, 2011), showing better results of the hybridized DE with respect to previous works that used only DE (Lopes and Bitello, 2007) or the GA approach.

The multiobjective approach was also considered (Handl et al., 2008). Garza-Fabre et al. (2012a,b, 2013b) used the multiobjectivization of the HP model. The originally single objective problem was restated as a multiobjective one by decomposing the conventional objective function into two independent objectives, taking into account the odd or even location of the amino acids. Their proposed formulation increased the search performance of the implemented algorithms in most of the cases. In Garza-Fabre et al. (2012b) the authors also incorporated a fitness multiobjectivization based on the locality of amino acid interactions.

Other natural computing algorithms have also been considered. For instance, Cutello et al. (2007) proposed an immune algorithm with two special mutation operators, hypermutation and hypermacromutation, which were incorporated into the proposed algorithm to allow an effective searching. The authors also used an aging mechanism as a new immune-inspired operator to enforce diversity in the population during evolution.

Finally, Fidanova (2006), Song et al. (2006), and Shmygelska and Hoos (2005) used ant colony optimization (ACO) for the HP model. For instance, in the latter article (Shmygelska and Hoos, 2005), ants folded a protein adding one amino acid at a time based on the pheromone matrix value, which represents the experience of previous ants, together with the use of heuristic information. As the authors commented, their empirical results showed that their ''rather simple ACO algorithm scales worse with sequence length but usually finds that more diverse ensemble of native states.'' Also, Fidanova (2006) reported results with the same benchmark instances of length 48 for three-dimensional HP protein folding in a three-dimensional cubic lattice. With a change in the pheromone updating and the heuristic information, in most cases, the authors' results were better than the ones provided by other methods, and without the use of a local search as the previous authors. Song et al. (2006) also experimented with ACO in three-dimensional lattices, with a different updating method with dynamic pheromones and three local search methods.

1.2. Comments about previous work on protein folding modeling

Protein folding is the dynamic physical process by which a protein structure assumes its functional shape or conformation. Levinthal's paradox (Levinthal, 1968) postulates that it is too time-consuming for a protein to randomly sample all the feasible confirmation regions for its native structure. However, proteins in nature can still spontaneously fold into their native structures (the whole process typically takes only milliseconds or even microseconds to finish). So, the folding pathway of a protein is unclear, and a general assumption is that the lower a structure is in the energy landscape, the closer the folding is to the native state of the protein (Duarte et al., 2007).

As commented, in most of the previous work in the field, the different evolutionary methods worked only as search algorithms of structures of minimal energy configuration, ignoring the dynamic nature of the folding. In the HP model the energy of a protein conformation is minimized by maximizing the contacts between adjacent hydrophobic amino acids, as detailed next. Nevertheless, such works did not consider the folding dynamics through time. For instance, the energy component that minimizes the distance between H (hydrophobic) amino acids, used in Krasnogor et al. (1999), facilitates the minimization through time of the distance between such amino acids, thanks to the different genetic operators used. However, there was not an explicit definition of how an amino acid must move in each time instant, taking into account the positions of the neighbor amino acids. There are few previous works in this line.

Krasnogor et al. (2002b) used cellular automata and Lindenmayer systems to try to define the rules and dynamics of such processes, with very limited success. They used a one-dimensional cellular automaton with four states that correspond to the possible movements in two-dimensional lattices, and the rules of the cellular automaton were obtained with a genetic algorithm. In an extension of their work, the rules took into consideration the specific amino acids to which the rules were being applied, thus connecting the CA modeling with a particular primary sequence. For example, for a small sequence of 20 amino acids, only 50% of the runs led to a set of rules that allowed achieving the optimal configuration. For larger sequences, the results were even poorer. Their work with Lindenmayer systems was only focused on finding out sets of rules that captured a given folded structure but, again, without a connection between the rules and the nature of the amino acids of the primary sequence.

In an alternative work by Calabretta et al. (1995), the authors tried to establish the tertiary structure modeling the folding process through matrices of attraction forces of the 20 amino acids. The matrices of 20×20 components were obtained with a genetic algorithm, where each component represented the attraction or repulsion force between two amino acids in a given distance (100 Å). The model included rotations of the side chains of amino acids around the backbone and the possibility that a local portion of the backbone bent itself. The fitness function was measured taking into account the discrepancies of the alpha-carbon bend and the torsion angles between the real known structure and the artificial folded one. They used the methodology with a short fragment of crambin (13 amino acids) that resulted in an alpha-helix, as in the real protein.

More recently Danks et al. (2008) presented a Lindenmayer system model that used data-driven stochastic rewriting rules to fold protein sequences by altering the secondary structure state of individual amino acid residues. The state of each residue was rewritten in parallel across the whole protein. The change in a residue state depended on the amino acid type of such residue and the amino acid types and the current states of the neighboring residues on either side. Seven secondary structure states were employed, based on those used in the DSSP database of secondary structure assignments, as well as their probabilities.

Typical backbone torsion angles were obtained for each amino acid type in each of the seven states from the database and used to reconstruct the 3D structure of a protein at each derivation step. They showed results for four protein sequences from each major structural class. Local structure preference can be seen to emerge for some residues in a sequence. However, as indicated by the authors, the resulting structures did not converge to a preferred global compact conformation.

For the modeling of the folding process through time we propose here a new alternative that was not considered or studied previously in the literature. Our main goal is an attempt to model the temporal folding using CA-like systems, using evolutionary computing to automatically obtain the CA models that act over the multimodal energy landscape inherent to the protein folding problem (Zhao, 2008). The CA models will determine the movements of the amino acids through time, considering the restrictions of the HP model. Such CA will be implemented with artificial neural networks that take the input information directly from the energy landscape.

Our work is inspired by the work of Krasnogor et al. (2002b) in the use of CA for the folding modeling. Nevertheless, one of the main problems of the cellular automata used by Krasnogor et al. (2002b) is the number of possible transition rules (which define the CA) that can be defined when the considered neighborhood of a given amino acid is increased, far beyond the closest neighbors. The number of possible CA is $K^{(K^N)}$, where K is the number of possible states in each of the cells of the CA, and N is the considered neighborhood (Langton, 1992). For instance, in Krasnogor et al. (2002b) the authors considered $K = 4$ states in each cell (the four previous movements of amino acids in a two-dimensional lattice) and a neighborhood of five in a one-dimensional cellular automaton, using the two closest amino acids in both sides of the chain. Hence, there were $4^{(4^5)}$ possible CA, an enormous number of CA.

Moreover, we must take into account that now, in the intended modeling of the temporal folding, the evolutionary method obtains the optimized CA after a temporal iterative application, starting from an initial unfolded amino acid chain, until the application of the CA rules ends with a final and folded structure. Given these problems, we propose the use of structures that can adequately manage the input space of possible configurations at the same time that provide a good generalization capability. The neural structures can be the perfect ones, as we detail in the next section. Finally, as indicated, such input space the artificial neural network considers will correspond to the energy landscape.

2. METHODS

2.1. HP lattice model

As indicated previously, to reduce the complexity of the problem of protein folding, simplified models are used. In the HP model (Dill, 1990) the elements of the chain can be of two types: H (hydrophobic residues) and P (polar residues). The sequence is assumed to be embedded in a lattice that discretizes the space conformation and can exhibit different topologies such as two-dimensional (2D) square or triangular lattices, or three-dimensional (3D) cubic or diamond lattices. To mimic hydrophobic interactions, each nearest-neighbor contact between two H monomers (not consecutive along the primary sequence) is assigned a favorable energy (-1), irrespective of whether the contact is in the native structure or not, while other contacts are modeled as neutral. That is, the basic HP energy matrix only implies attractions (H with H), and neutral interactions (P with P and P with H). Given a primary sequence, in the intense research performed on the direct prediction of the final conformation, the problem is to search for the folding structure in the lattice that minimizes the energy.

A variant of this basic ''linear-chain'' HP model, which tries to include more realistic features of proteins, is the inclusion of a side bead representing a side chain of the amino acids (Bromberg and Dill, 1994; Benítez and Lopes, 2009). Therefore, a protein is represented by a backbone (common to all amino acids) and a ''side-chain,'' either hydrophobic (H) or polar (P). Nevertheless, we focus here on the linear HP model since it is not only the most popular but also the one used in our work.

As commented, the complexity of finding the folded protein with minimum energy is NP-hard (Hart and Istrail, 1997; Unger and Moult, 1993b). This problem has been shown to be NP-complete on the square lattice (Crescenzi et al., 1998) and the cubic lattice (Berger and Leighton, 1998). Therefore, determining the native state conformations of HP sequences has become a computational challenge. There are ''approximation'' algorithms for finding the minimum-energy folds with mathematically guaranteed error bound as well as ''exact'' algorithms (Giaquinta and Pozzi, 2013). An approximation algorithm for protein folding in an

HP-model is a polynomial-time algorithm that for every protein sequence outputs a fold of that protein whose number of contacts is provably near optimal (Istrail and Lam, 2009), like the first one published by Hart and Istrail (1995). The work by Istrail and Lam (2009) provides a survey focused on this type of algorithm.

As Istrail and Lam (2009) also comment, ''On the other hand, combinatorial optimization algorithms, despite being exponential, find exact optimal solutions; however, only for small to moderate size proteins (about 50 amino acids).'' For example, Giaquinta and Pozzi (2013) provide an exact method that devises upper bounds for the number of contacts achievable by an HP sequence, although only for standard HP sequences of size up to 80 elements in the 2D square lattice, and based on a heavily pruned exhaustive search.

Finally, ''heuristics'' or ''nonexact'' approaches are another category of combinatorial methods (Istrail and Lam, 2009; Giaquinta and Pozzi, 2013). This category has a large amount of literature and many authors have been working specially on several evolutionary and natural computing algorithms (Unger, 2004; Zhao, 2008) for the direct prediction of the native conformations using the HP model, as indicated in the previous section. Our focus is on these algorithms since our work, although modeling the temporal folding belongs to this type of solution.

Using evolutionary computing to determine protein conformations under the HP model, one of the main decisions is the genotype encoding of the protein conformation in the lattice. Three basic possibilities can be considered to the representation of the folded sequence in the lattice: Cartesian coordinates and two alternatives with internal coordinates. In the first one, the location of each amino acid is specified independently with its Cartesian coordinates. With the internal coordinates the embedding of the protein is specified as a sequence of movements taken on the lattice from one amino acid to the next. The first alternative with internal coordinates uses an absolute representation, and movements are specified with respect to it; for example, in the case of the cubic lattice: north, south, east, west, up, and down. A conformation is expressed as a sequence $\{N,S,E,W,U,D\}^{n-1}$, which is the genetic material in the individuals when this representation is used. In the relative representation, relative movements are considered. The reference system is not fixed and the next movement depends on the previous one. Now, in the same case as before, five movements are allowed: forward, turn up, turn down, turn left, and turn right. The conformations are expressed now as sequences $\{F, U, D, L, R\}^{n-2}$. This representation has the advantage of guaranteeing that all solutions are one-step self-avoiding (because there is no back movement).

Both alternatives were used in evolutionary computing works to encode the protein conformations. For example, Unger and Moult (1993b) used the absolute representation of the movements and Patton et al. (1995) used the relative movements to define the conformations. The results of Krasnogor et al. (1999) supported the use of the relative encoding when they analyzed the impact of different factors when evolutionary algorithms are applied to the problem. Nevertheless, there is not an ample study determining which representation is the best. We will use the relative representation of the movements, because of its interesting property of avoiding one-step self-conflicts.

2.2. *Differential evolution*

We used simulated evolution to optimize the CA models that define the folding process (next subsection). These CA models will be implemented by artificial neural networks (ANNs), and the genotypes of the genetic population encode the real-valued connection weights of the ANN. We used Differential Evolution (DE) (Price et al., 2005) since this algorithm is specially suited for optimization problems where possible solutions are defined by a real-valued vector (Feoktistov, 2006).

Additionally, we must take into account that the energy landscape in the determination of the protein-folded conformation presents a multitude of local energy minima separated by high barriers. As Zhao points out, ''there are many meta-stable states whose energies are very close to the global minimum and an exceedingly small number of global optimal states. Folding energy landscapes are funnel-like'' (Zhao, 2008). For this reason, a method like DE will be necessary, which offers fast convergence, robustness, conceptual simplicity, few parameters with an easy implementation, and with a reliable control in the balance between exploration and exploitation (Feoktistov, 2006).

Differential Evolution (Price et al., 2005) is a population-based search method. DE creates new candidate solutions by combining existing ones according to a simple formula of vector crossover and mutation, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand. The central idea of the algorithm is the use of different vectors for generating perturbations in a population of vectors. The basic DE algorithm is summarized in the pseudocode of Algorithm 2.1.

Differential Evolution needs a reduced number of parameters to define its implementation. The parameters are F or differential weight and CR or crossover probability. The weight factor F (usually in [0,2]) is applied to the vector resulting from the difference between pairs of vectors (x_2 and x_3). CR is the probability of crossing over a given vector of the population (x) and a candidate vector (y) created from the weighted difference of two vectors ($x_1 + F(x_2 - x_3)$). Finally, the index R guarantees that at least one of the parameters (genes) will be changed in such generation of the candidate solution.

Algorithm 2.1: DIFFERENTIAL EVOLUTION(*Population*)

```
for each Individual ∈ Population
  do { Individual ← INITIALIZERANDOMPOSITIONS()
repeat
  for each Individual x ∈ Population
    do { x1, x2, x3 ← GETRANDOMINDIVIDUAL(Population) //must be distinct from each other and x
         R ← GETRANDOM(1, n) //the highest possible value n is the dimensionality of the problem to
         //be optimized
         for each i ∈ 1 : n//Compute individual's potentially new position y = [y1 ..., yn]
           do { ri ← GETRANDOM(0, 1) //uniformly in open range (0, 1)
                do { if ((i = R)||(ri < CR))
                       yi = x1i + F(x2i − x3i)
                     else yi = xi
         if (f(y) < f(x)) x = y //replace x with y in Population
until TERMINATIONCRITERION()
return (GETLOWESTFITNESS(Population)) // return candidate solution
```

The usual variants of DE choose the base vector x_1 randomly (variant *DE/rand/1/bin*) or as the individual with the best fitness found up to the moment (x_{best}) (variant *DE/best/1/bin*). To avoid the high selective pressure of the latter, we used a tournament to pick the vector x_1, which also allows us to easily establish the selective pressure by means of the tournament size.

As Feoktistov (2006) indicates, the fundamental idea of the algorithm is to adapt the step length ($F(x_2 - x_3)$) intrinsically along the evolutionary process. At the beginning of generations the step length is large, because individuals are far away from each other. As the evolution goes on, the population converges and the step length becomes smaller and smaller, providing this way an automatic balance in the search.

2.3. Protein folding modeling with neural cellular automata

We used connectionist structures to implement the CA-like systems that provide the folding process (we named the scheme ''neural cellular automaton,'' or neural-CA). The artificial neural network (ANN) that implements the cellular automaton provides the next movement of an amino acid, whereas the inputs of the neural network are determined by the consequences of each possible movement of the current amino acid to which the neural model is applied. The same neural network is applied to each amino acid of the protein sequence sequentially, and this process is repeated through different temporal iterations or steps across the whole sequence of amino acids. Since we used the relative encoding of the movements to represent a protein chain, the neural network provides the relative movement for each amino acid position.

The individuals of the DE population encode the models that provide the folding process. The cellular automaton is implemented by means of a simple feedforward neural model. Hence, the individuals code the parameters that define the neural model. As we used a standard and fixed transfer function in the neural network nodes, every ANN is represented as its set of connection weights between the nodes of the layers.

2.3.1. Inputs of the neural network. The neural-CA is applied to each amino acid to decide its next movement. As inputs to the ANN we can consider the spatial configuration of the amino acid, taking into account the nearest neighbors and their states. For example, we can consider whether any neighbor position

in the lattice is occupied or empty, as well as whether the amino acid is H or P. Nevertheless, the ANN would have no information about whether the amino acids in the spatial neighborhood are neighbors in the primary sequence. Additionally, different rotations of the same protein conformation provide different spatial configurations. As an alternative, we could input the relative movements in both sides of the amino acid that moves, together with the H or P nature of the neighbor amino acids, but this implies a quite large number of possible different inputs to the ANN. Because of this, we applied another alternative that avoids these problems, taking into account the view of the energy landscape instead of the spatial information, considering the following inputs:

1. For each possible movement in the current position or amino acid i of the chain, we calculate the increase (positive or negative) of energy with respect to the current movement in such position. If a movement implies a collision in the amino acid i, then we consider a high increase of energy for such movement.
2. For every possible movement in position i, a greedy strategy several movements forward is applied. That is, the next movements are defined by those that provide the minimum energy. Once these posterior movements are applied, the increases in energy (with respect to the energy of the current conformation) are also provided as inputs to the neural network. Hence, for each possible movement in the current position or amino acid, the network receives as input what would be the increase of energy if the neural network decided a greedy strategy.

Obviously, the energy landscape is dynamic as a movement in an amino acid changes the energy landscape for the next amino acid, and the same through the different temporal steps when the ANN is applied to the whole chain.

2.3.2. Outputs of the neural network. The ANN has an output corresponding to the three possible relative movements $\{F, L, R\}$ (in 2D). In the 3D case, the network has five outputs corresponding to the five possible relative movements $\{F, U, D, L, R\}$. Figure 1 represents the process but with the 2D case to clarify the ANN use. The neural-CA decides which is the most appropriate movement in each situation.

For determining the inputs, as indicated before, all the possible movements are applied to the current position and the corresponding energy increases (with respect to the current movement) are calculated, being inputs to the ANN (obviously one of the inputs is 0). Moreover, once each one of the movements is applied, we follow a greedy strategy to change the protein chain in the next N movements (N being a parameter to tune). Again, the increase of energy of the resultant conformation, with respect to the current protein conformation, is an input to the ANN (one additional input for each one of the possible movements in position i).

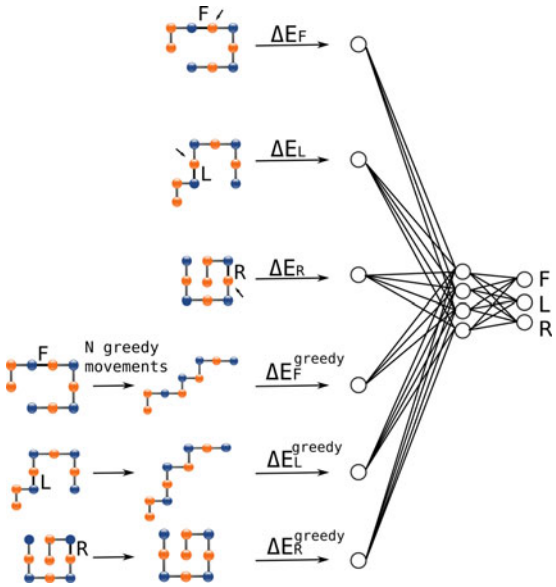

FIG. 1. Artificial neural network used for determining the next movement in each amino acid. The inputs correspond to the energy increases (with respect to the current energy) when the possible movements are applied in the amino acid position and greedy movements are considered after such possible movements.

This way, the ANN has a view of the energy landscape in order to decide the most appropriate movement in each situation, taking into account that the ANN can apply different movements than the greedy ones when it is applied to the next chain positions. Figure 1 shows a scheme of the feedforward ANN used, summarizing the input information the ANN receives, using a hidden layer and an output layer where each output node corresponds to each possible movement. The number of hidden nodes is selected between the number of inputs and outputs, trying to obtain a good generalization capability and sufficient capacity to learn the training patterns/proteins. The output node with the highest activation value determines the next movement.

2.3.3. Neighborhood considerations for energy calculation. In any case, for calculating the energy in each position or amino acid, we calculate it as the HP energy but considering only a spatial neighborhood around the current amino acid. This means that only the amino acid contacts that are below a given threshold distance to the current amino acid are considered (threshold distance D). Moreover, if the chain presents collisions after a movement, we only consider the amino acids of the chain until the point in which the first collision occurs. Hence, while moving all the amino acids of the chain, the protein conformation can pass through illegal states, and the ANN must learn to change the conformation in each amino acid position in such a manner that there are no collisions in the immediate neighborhood where the ANN is applied, in addition to trying to minimize the conformation energy.

Note also that this is the central idea of the use of cellular automata models for defining an emergent process, as the cellular automaton receives only the local information or environment of a single element, whereas the iteration of the cellular automaton, over all the elements and through time, provides the emergent phenomena, like in our case the emergent folding process.

Algorithm 2.2: FITNESS EVALUATION(*neural - CA*)

```
initial protein conformation ← unfolded // all amino acids in a straight line
for each s ∈ 1 : S // s - temporal step, S - maximum number of steps
do {
    n ← 0 //number of movements in each step
    for each i ∈ 2 : N //i − amino acid position, N-number of amino acids
    do {
        CALCULATE NEURAL-CA INPUTS IN POSITION i()
        CALCULATE NEURAL-CA OUTPUTS FOR AMINO ACID i()
        DETERMINE MOVEMENT FOR AMINO ACID i()
        // highest neural output
        APPLY MOVEMENT TO AMINO ACID i()
        if (applied movement != previous movement in the same position)
        n = n + 1
        if (illegal movmeent (conflict in amino acid or position i after the movement))
        return (NUMBER OF HH CONTACTS(folded conformation until the previous movement))
        //conformation without conflicts
    }
    if (n = = 0) //no movements applied in the complete step
    return (NUMBER OF HH CONTACTS (folded conformation))
}
return (NUMBER OF HH CONTACTS(folded conformation))
```

2.3.4. Fitness definition. Most of the previous works using the HP model on the direct prediction determined the goodness of the results taking into account the basic definition of the HP energy, that is, maximizing the number of HH contacts. Thus, we used the same fitness definition to access the goodness of a final folded conformation, taking into account that this is the number of final HH contacts after the emergent folding process driven by the evolved neural-CA.

In other words, the evolved neural-CA is applied to a protein chain, and we want that the iterative temporal folding (defined by the neural-CA) reaches a given conformation with a given and explicit fitness, such as the number of HH contacts in the basic HP model.

Table 1. Benchmarks Sequences of Length 27, for the Cubic Lattice, Used in the Experiments

	HP chain
273d.1	*PHPHPHHHPPHPHPPPPPPPPPPHHP*
273d.2	*PHHPPPPPPPPPPHHPPHHPPHPPHPH*
273d.3	*HHHHPPPPPHPPPPPHHHPPPPPPPPH*
273d.4	*HHHPPHHHHPPPHPHPPHHPPHPPPHH*
273d.5	*HHHHPPPPHPHHPPPHHPPPPPPPPPP*
273d.6	*HPPPPPPHPHHHPPHHPPPHPPPPHPH*
273d.7	*HPPHPHHPPPHPPPPPHPHHPHPHPHH*
273d.8	*HPPPPPPPPPPHPHPPPPPPPPHPHH*
273d.9	*PPPPPPPHHHPPPHPHHPPPHPPHPPP*
273d.10	*PPPPHHPHPHPHPHPPHHPHHPHPPP*

The process begins with the protein sequences unfolded (all amino acids in a straight line). Then, the neural-CA is applied to each of the amino acids of the protein chain, where the neural-CA determines the next movement of such amino acid. This procedure is repeated a given number of steps or until the neural-CA determines that no more movements are applied, so the fitness of such individual (encoded neural-CA) is given by the final HH contacts and returned to the evolutionary method. Moreover, in order that the neural network learns to avoid infeasible solutions (with conflicts), if the neural network determines a movement that generates a conflict in the position where the current amino acid is moved, then the individual has as fitness the energy accumulated until the previous situation without conflicts in the protein conformation. That is, if the neural-CA determines a movement that generates a conflict in the specific amino to which the neural-CA is applied (we call it ''illegal movement''), then the folding process is ended, returning the immediate configuration without conflicts. As explained previously, the ANN has the necessary information to avoid the illegal movements, since the inputs corresponding to such movements have a high energy increase. This decision to end the process, when an illegal movement is determined, helps that the evolutionary population has a great diversity of fitness and permits to refine the ANN so it does not determine movements with conflicts in the immediate neighborhood of the current position. The pseudocode of Algorithm 2.2 summarizes the process of fitness evaluation.

Note that the fitness definition (we used the basic HP energy matrix considering only the HH contacts in the final folded structure) can be different from the definition of the energy used to calculate the energy increases that are the inputs to the ANN. In this case, we also used the information of other contacts different to HH, as indicated in the Results section.

2.4. Benchmark protein sequences

We used benchmark sequences employed in different works such as Unger and Moult (1993b), Patton et al. (1995), Johnson and Katikireddy (2006), or, more recently, Mansour et al. (2011). The test data consisted of a series of 10 randomly produced 27-length sequences. Table 1 shows the sequences used for

Table 2. Benchmarks Sequences of Length 64, for the Cubic Lattice, Used in the Experiments

	HP chain
643d.1	*PPHHHHHPPPHHPPPPPHHPPPHPPPPPPHPHPPPHPPHPPHPPPPPPHPPPPHHPHHPPHPPHP*
643d.2	*PPHPHPPHPPHHHPHHHHPPHHHPPPPHPHPPPHPHPPPHPHPPPPPHPHPPHPHPPPHPPHPP*
643d.3	*HPHHPPHHPHPPPPPHHHPHHHHPPHPPHPHHPPPHPHPPHHHPHHPHPPPPPHHHHHHHHPPP*
643d.4	*HPPHHPPHPPHPHPPHPPPPHPPPPPPHPHPHHHPPHPHPPPHPHPPHHPPHPPHPHHHPH*
643d.5	*HPPPHHPPHPHPPPHPPPHPHHPPPHHPHPHHPHPPHPPPHPPHPHHHPPHPPHPPHHHPHHHH*
643d.6	*HPPHHPHHHHPPPPPPHHPPHPPPPHHPPPHPPHPHHPHPPPPHHPPPPHPPPPPPHPPPPHPHH*
643d.7	*PPPPHPPPHPPPHHHHPHHPPPPPHPPHPHHPHPHPPPPPHPPPPPPPPPPHHHHPPPPHHPPH*
643d.8	*PPPHHHPPHPHPPHPPHHPPPPHPPHPPHHPHPPPHPPPPPPPPHPHHHPHHHHHPPHHPPPHPPH*
643d.9	*HPPHPPHHHPPPPHPHPPPHPHHPHHHHHPPPPHPHPHPPPPHPHPPPHHPHPPPPHPPHHPHP*
643d.10	*PPHPPHPPHHHPPPHPHPPHPPHPPPPPPHPPHHHPPHPPHPPHPHPPPPPPHHHPPPPPHPHP*

the 3D cubic lattice, which were firstly published in the work of Unger and Moult (1993b). The global minimum energy value for these random configurations is unknown. Table 2 shows additional sequences for the 3D cubic lattice and 64 amino acids length, used also in some of the experiments reported in the next section. Again, the global minimum energy is unknown in these sequences.

3. RESULTS

3.1. Experimental setup and folding example

We first expose the setup for the folding process, including the details of a folding example provided by the neural cellular automata methodology. Figure 2 shows an example of the folding process provided by an evolved neural network model (10 inputs, 6 hidden nodes, and 5 output nodes) for a simple protein sequence used by Patton et al. (1995), which consists of eight hydrophobic residues evenly interspersed with two hydrophilic residues each (HPPHPPHPPHPPHPPHPPHPPH). In the minimal configuration, the hydrophobic residues form a cube with the vertices connected by hydrophilic ''loops.'' So, the optimized structure has an optimum with 12 HH contacts perfectly centered in the inner core of the final structure. The figure shows several intermediate configurations in the folding when movements in an amino acid and in different steps are applied, until the final folded conformation is obtained.

In this example and in all the runs presented in this section with the different benchmark sequences, the same ANN configuration was used. The number of posterior greedy movements applied after each of the

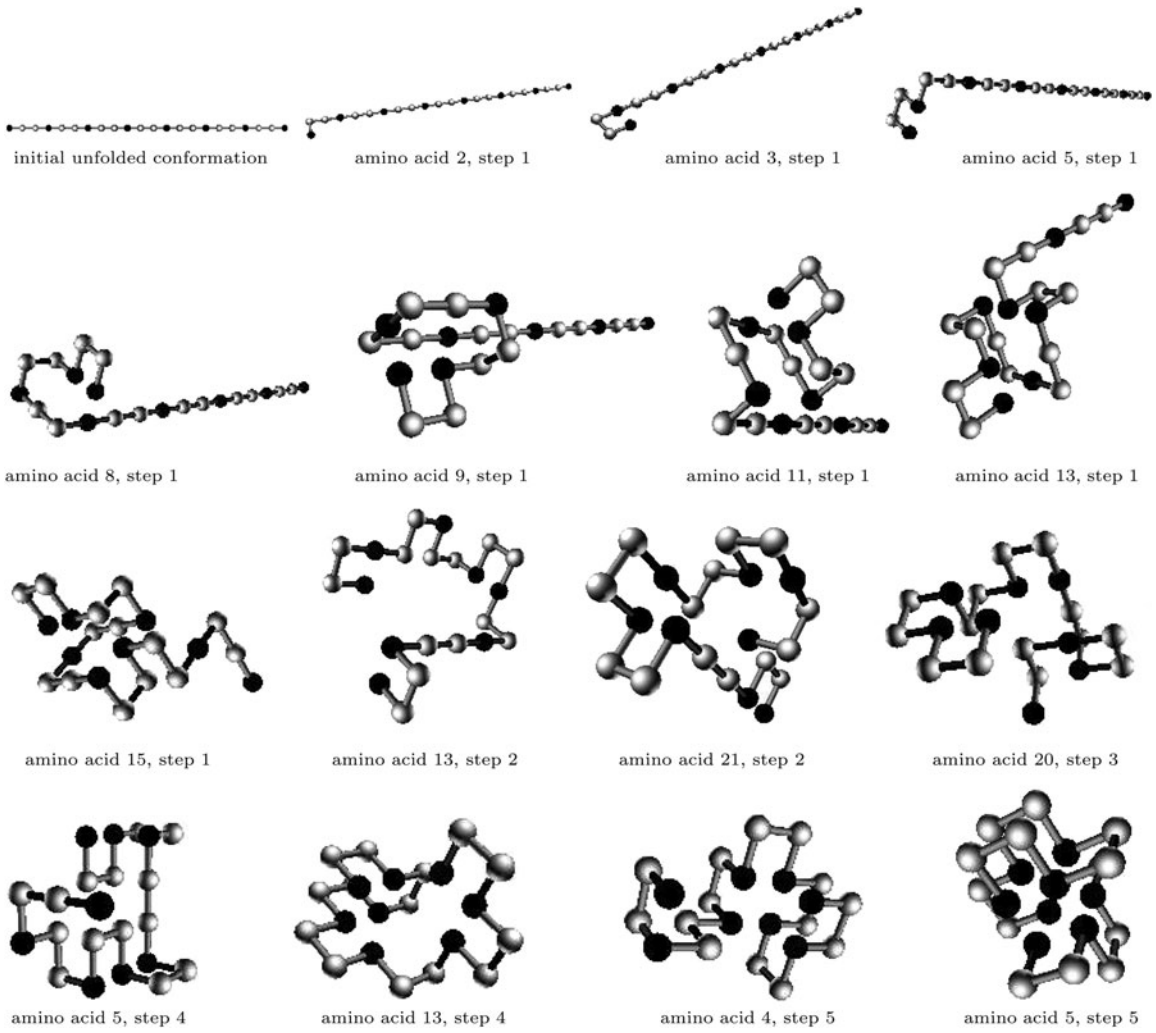

FIG. 2. Different temporal steps in the folding process with protein sequence HPPHPPHPPHPPHPPHPPHPPH.

five possible relative movements of the current amino acid was $N = 3$, in order to calculate the five energy increases that are inputs to the ANN when such posterior greedy movements are considered. This number provided us with the best results in most sequences (in terms of necessary evaluations as explained in the next subsection). Obviously, with a lower number of posterior greedy movements, the information about the fitness landscape is less detailed and, on the contrary, with a greater N, the ANN can learn exclusively the greedy path for the different movements.

For the calculation of energy increases, the local neighborhood was defined using a threshold distance $D = 5$, that is, including all the topological contacts between amino acids with Euclidean distance less than or equal to five with respect to the central amino acid. This central position is given by the previous amino acid of the one that is currently subject to move. This value of D allows taking into account the interactions between closer amino acids when the protein is partially folded or with other amino acids when the protein is becoming more compact. Finally, for each sequence, we allowed a maximum number of 10 folding steps, that is, the neural-CA is applied to all the amino acids of the protein chain a maximum number of 10 times, beginning with the first amino acid and applying the neural-CA sequentially. As we explained, if the neural-CA determines a movement that generates a conflict in the specific amino moved (illegal movement), or no movements are determined by the neural network in a complete step over the whole chain, then the process is ended providing the final conformation (without conflicts) defined by the neural-CA. For instance, in the example of Figure 2, the evolved neural-CA needed only five steps to complete the folding (without any illegal movement).

Once the neural-CA that performs the folding is evolved, then we can analyze the dynamic folding process, being this analysis is oriented to the restrictions of the HP model. A reasonable analysis is the test of the nature of the movements, analyzing the greediness of the movements. In this example, the ANN applied only an average of 20.7% of greedy movements in the whole process. Figure 3 shows the percentage of greedy movements, in different temporal steps of the folding process, with three different trained ANNs for the folding of the protein. The three ANNs obtain the optimal number of final HH contacts and can require different steps to obtain the final folding. The figure denotes that there is not a clear conclusion about whether the evolved neural cellular automata use more or less greedy movements depending on the temporal step, that is, at the beginning of the folding process or at the last steps when less movements without conflicts are available.

Regarding Differential Evolution, we used values, $CR = 0.9$ and $F = 0.9$, in the intervals suggested by Storn and Price (1997) ($F \in [0.5,1.0]$, $CR \in [0.8,1.0]$), whereas the size of the tournament to choose the base vector was 8% of the population, which implies a low selective pressure when choosing such vector to disturb. Since there is not a clear rule about what the best number of individuals is, we used, for each sequence, *population size = number of amino acids x 15* (as suggested in Lopes and Bitello, 2007). The DE individuals code the ANN weights in the range $[-1, 1]$, which are decoded multiplying the encoded value by a constant MAX_VALUE. We used MAX_VALUE = 2 since this value allows saturating the nodes using a standard sigmoid as transfer function of the ANN nodes.

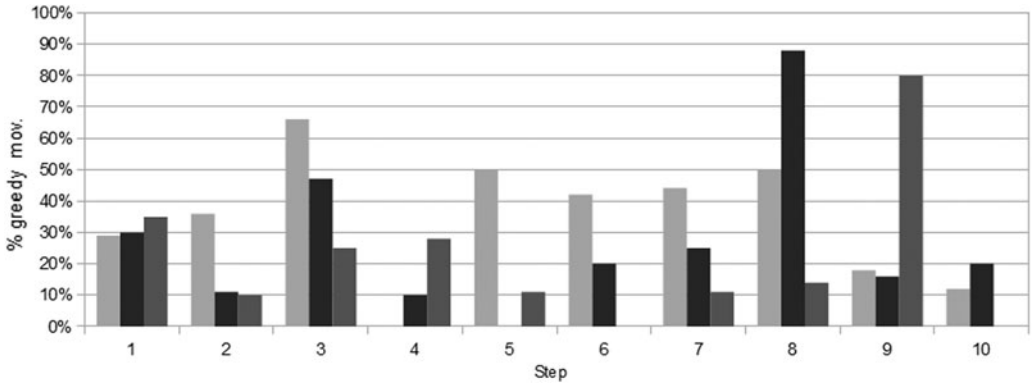

FIG. 3. Percentage of greedy movements applied by three different evolved artificial neural networks (ANNs) when folding the sequence HPPHPPHPPHPPHPPHPPHPPH and in the different folding steps (application of the ANN through all the amino acids of the chain).

3.2. Results with evolved neural-CA for the benchmark sequences; comparison with previous works on the direct prediction of the final folded conformation

As the folding process provides the final protein conformations, we can also test the capability of the neural-CA models to obtain the final HP optimal energies. In Table 3 we included the comparison of the HH contacts of our final conformations, after the iterative folding process, with respect to previous works that used search methods to predict directly the final conformations. Table 3 includes a comparison of our results with the ones in other works, such as the results of Unger and Moult using a GA hybridized with a Monte Carlo local search (which basically filters mutations) (Unger and Moult, 1993a) and Patton et al. (1995) with an improved GA using a relative encoding of the amino acid movements. Previous works summarize the results, taking into account independent runs for each protein sequence, and in each column it is specified the best energy value found in those different runs of the corresponding search algorithm. The energy values are reported using the basic HP energy function to allow direct comparison to previously published results. The best found solution for each sequence is highlighted in bold. The values in parentheses are the minimum number of conformations scanned before the lowest energy values were found in one of the runs. This is the case of the results of Unger and Moult (1993a) from five independent runs.

In the case of Patton et al. (1995), the authors do not specify how many different independent trials were run. Additionally, in our case, we included in Table 3 the average best energy in the different runs and the average number of evaluations (fitness calculation of individuals) for obtaining the best value (second values after the comma). We used 10 independent runs for each sequence. Note that the average number of evaluations can be lower with respect to the best case if the same best value was not obtained in all the runs (seq. 273d.6).

In the work of Unger and Moult (1993a), the authors used the absolute representation of the movements. In their work, the GA began with a population of identical unfolded configurations. In addition to a one-point crossover operator, in each generation a series of K mutations were applied to each individual in the population, being K equal to the length of the encoding. The mutations were filtered using a Monte Carlo acceptance algorithm that disallowed lethal configurations (those with collisions), always accepted mutations resulting in better energy, and accepted increased energy mutations based upon a threshold on the energy gain that becomes stricter over time. As in their work lethal configurations were rejected, the crossover operation was retried for a given pair of parents until a nonlethal offspring was generated. Finally, offspring were accepted using a second Monte Carlo filter, which accepted all reduced energy conformations and randomly accepted increased energy offspring, again using a cooling threshold on the energy gain. With all these considerations, as indicated in Table 3, the GA of Unger and Moult (1993a) operated on each of the 27-length sequences for roughly 1.2 million function evaluations. In their runs, they used a population size of 200, obtaining better comparison results in terms of performance, with respect to a Monte Carlo approach.

Regarding the work of Patton et al. (1995), the authors used the relative movements to define the conformations in the 3D lattice and, contrary to Unger and Moult (1993a), they allowed illegal conformations in the genetic population but penalized them for positional collisions. This essentially allows the search to proceed through illegal states. Patton et al. (1995) obtained results that compare quite favorably with those of Unger and Moult, as shown in the table. For example, as the authors remark on their work, for two of the 27-length samples, the minimal energies located by their algorithm were better and required only one tenth the number of energy evaluations.

In the work of Johnson and Katikireddy (2006), the authors reported fewer numbers of evaluations (the best value in five trials) to reach the best energy values, taking into account that their algorithm uses a backtracking procedure to resolve the positional collisions and illegal conformations that occur during the course of the genetic search, requiring a high number of applications of the backtracking routine, as detailed in their work. Also, in the work of Mansour et al. (2011), their GA was enhanced with heuristics that repair infeasible outcomes of the crossover operation and ensure that the mutation operation leads to fitter and feasible candidate solutions, obtaining again a decrease of necessary evaluations over the previous works (the authors do not specify how many runs were applied for each sequence).

In our case, regarding the inputs to the neural cellular automata, we considered two cases:

i) neural-CA(1), only HH contacts were considered in the calculations of energy increases; and
ii) neural-CA(2), HH and HP (or PH) contacts were considered in the energy increase calculations.

The results shown in Table 3, in this second alternative, were obtained weighting the HH contacts with -1 and penalizing the HP and PH contacts with a weight of 0.5, in a similar way to the HP Functional Model (Hirst, 1999), which considers all contact possibilities between amino acids.

Table 3. Comparison of Results with the Benchmark Sequences

Seq.	*Neural-CA (2)*	*Neural-CA (1)*	*M et al. GA*[a]	*J&K GA*[b]	*P et al. GA*[c]	*U&M GA*[d]
273d.1	**−9**, −9 (4015, 10380)	**−9**, −8.5 (12382, 14267)	**−9** (1450)	**−9** (15854)	**−9** (27786)	**−9** (1227964)
273d.2	**−10**, −10 (2766, 5020)	**−10**, −10 (8185, 13107)	**−10** (5473)	**−10** (19965)	**−10** (81900)	−9 (1225281)
273d.3	**−8**, −8 (1315, 4007)	**−8**, −8 (7398, 8168)	**−8** (1328)	**−8** (7991)	**−8** (16757)	**−8** (1247208)
273d.4	**−15**, 15 (4032, 5330)	**−15**, 15 (6752, 10795)	**−15** (5196)	**−15** (23525)	**−15** (85447)	**−15** (1207686)
273d.5	**−8**, −8 (1252, 3283)	**−8**, −8 (2851, 5360)	**−8** (1184)	**−8** (3561)	**−8** (8524)	**−8** (1118202)
273d.6	**−12**, −11.8 (6416, 8148)	**−12**, −11.6 (12322, 11344)	**−12** (18012)	−11 (14733)	−11 (44053)	−11 (1226090)
273d.7	**−13**, −13 (3391, 5514)	**−13**, −13 (11813, 16129)	**−13** (4920)	**−13** (23112)	**−13** (85424)	−12 (1239519)
273d.8	**−4**, −4 (1331, 6011)	**−4**, −4 (47973, 61029)	**−4** (654)	**−4** (889)	**−4** (3603)	**−4** (1248118)
273d.9	**−7**, −7 (2584, 3486)	**−7**, −7 (2030, 3682)	**−7** (1769)	**−7** (5418)	**−7** (10610)	**−7** (1198945)
273d.10	**−11**, −11 (1779, 3734)	**−11**, −11 (2895, 3458)	**−11** (3882)	**−11** (5592)	**−11** (16282)	**−11** (1174297)
Total eval.	28881, 59629	114601, 147339	43868	120640	380386	11113310

In each column of results is the best energy value (HH contacts) in different independent runs and, in parentheses, the minimum number of evaluations to find the best value. After the comma is the average best energy value in the different runs and average number of evaluations to find the best values in each run (in parentheses).

CA, cellular automata; GA, genetic algorithm.

[a]Mansour et al. (2011).

[b]Johnson and Katikireddy (2006).

[c]Patton et al. (1995).

[d]Unger and Moult (1993a).

For all sequences we allowed a maximum number of 30000 evaluations in the different runs (10). Note that now evaluations refer to the fitness calculation of the encoded neural cellular automata, and not to the evaluation of an encoded protein conformation as in the previous works commented here. That is, in all cases, the number of evaluations is the number of evaluated individuals of the population in order to obtain the best value. From the results shown in Table 3, several main aspects can be identified:

1. Our methodology with the evolved neural-CA (both configurations) obtains the same best values of the best search method for the direct prediction of the protein conformation (Mansour et al., 2011).
2. The number of necessary evaluations is reduced with respect to previous search methods for the direct prediction of the folded conformations.

   Using neural-CA(1), there is a decrease in the number of necessary genotypes that must be evaluated to obtain the best fitness value (maximum number of HH contacts). The exception is the reported number of evaluations in Mansour et al. (2011). These better values (in terms of number of evaluations) in most sequences in Mansour et al. (2011) are explained because of the filters applied by the authors to the outcome of the crossover and mutation operators in their enhanced GA. Also, in sequence 273d.8, using neural-CA(1), more evaluations were necessary to obtain the best value (-4) (with respect to Mansour et al., 2011; Johnson and Katikireddy, 2006; and Patton et al. 1995), because the few input information provided only by the low number of possible HH contacts (a maximum of 125,000 evaluations were applied to this sequence).

   When neural-CA(2) is considered, in many sequences the reduction is quite drastic. Even in most sequences there is a decrease with respect to the number of evaluations reported in Mansour et al. (2011).
3. Using the second configuration, neural-CA(2), weighting the HH and PH (or HP) contacts improves the results with respect to considering only the basic HP energy formulation, that is, only considering the HH contacts for calculating the energy increases before and after each possible movement [neural-CA(1)]. The parameters used (weighting HH contacts with -1 and penalizing the HP and PH contacts with a weight 0.5) provided us with the best results in most of the sequences. Penalizing the PP contacts had no significant effect.

In all the sequences (except 273d.9), the minimum number of evaluations to obtain the optimal energies, as well as the average number of individuals to obtain such optimum values (in the 10 independent runs) are better with respect to neural-CA(1). Even the average number of necessary evaluations to obtain the best values is lower with respect to, for example, the values published in Johnson and Katikireddy (2006) for their best case (except 273d.8). This is because this second configuration uses more information in the calculation of the energy increases (inputs to the neural-CA) than the basic information provided by the HH contacts, so it provides better results in terms of necessary evaluations to obtain the best values. There is a clear case in sequence 273d.8. This sequence has only six hydrophobic amino acids (Table 1) and a best known value of 4 HH contacts. Thus, if only the possible HH contacts are used for calculating the energy increases, the neural-CA has very little information to decide the best amino acid movement. Nevertheless, using also the information of the PH contacts, the neural-CA has more detailed information to decide the movement to apply in each situation. This explains the drastic reduction of the number of evaluations, using neural-CA(2) in that sequence, with respect to neural-CA(1). In other words, using neural-CA(2), the ANN has a more detailed view of how the immediate and dynamic energy landscape is when the protein is folding and in order to decide the next movement.

3.3. Generalization capability of neural cellular automata

In the previous subsection we used only one protein sequence to train each neural cellular automaton in order to test the capability of the evolved ANNs to obtain a folded structure that maximizes the HH contacts as well as the necessary evaluations of encoded solutions in the genetic population. But we can also train a neural-CA with several benchmark sequences (training set). The purpose of training a single neural-CA with a group of proteins is double: first to check the capability of an evolved neural-CA to define the folding of different sequences used during its ''training'' with the evolutionary algorithm. Second, to test the generalization capability of the evolved neural-CA for defining the folding of different protein sequences, proteins not used in the training. So, we used the classical methodology with ANNs, when an ANN is trained with a training set of examples and tested with a different set (test set). This represents one of the most important advantages of the methodology, since the evolved neural-CA can define the folding of different proteins, even with different lengths.

Table 4. Validation of Results with the Benchmark Sequences

Sequence	E_{min}	*Energy (HH contacts)*
Training sequences		
273d.1	−9	−5
273d.2	−10	−8
273d.3	−8	**−8**
273d.4	−15	**−15**
273d.5	−8	**−8**
Test sequences		
273d.6	−12	−8
273d.7	−13	−10
273d.8	−4	−3
273d.9	−7	−3
273d.10	−11	−6

The neural cellular automaton (neural-CA) is evolved with the training sequences and validated with the test sequences. E_{min}: maximum number of HH contacts known for each sequence (Mansour et al., 2011); last column: number of HH contacts once the folding process has ended.

When the neural cellular automata are trained with different sequences, each neural-CA, encoded in each individual of the genetic population, is applied independently to the different training sequences (initially unfolded). The fitness of each individual is the sum of the different fitness values (final HH contacts) in each of the final folding for each particular protein sequence used for the training. Obviously the order of the training sequences is irrelevant. In the validation process, this evolved neural-CA is applied independently to each of the benchmark sequences used for testing it.

In a first experiment we divided the benchmark sequences in two sets of equal number of protein chains. We considered two cases: i) The first five sequences of Table 1 were used for training the neural-CA, while the other five were used for its validation, and ii) the same but inverting the training and test sequences. We used only the previous configuration neural-CA(2). Since the results with this configuration point out that most of the best known values were obtained with less than 10,000 evaluations (Table 3), we decided that for all sequences such maximum number of 10,000 evaluations were allowed. The population size is maintained as *population size = number of amino acids × 15*.

Tables 4 and 5 show the results in both cases. When the best known value of energy was obtained, it is highlighted in bold. In the first case, Table 4 indicates that the evolved neural-CA performs optimally in

Table 5. Validation of Results Swapping the Training and Test Sets with Respect to the Previous Example and Table

Sequence	E_{min}	*Energy (HH contacts)*
Test sequences		
273d.1	−9	−6
273d.2	−10	**−10**
273d.3	−8	−6
273d.4	−15	−10
273d.5	−8	−6
Training sequences		
273d.6	−12	−9
273d.7	−13	−11
273d.8	−4	−3
273d.9	−7	**−7**
273d.10	−11	−10

three of the five training sequences, while worse in the test sequences. With the training sequences, the evolved neural-CA obtains 44 HH contacts of the 50 possible HH contacts (88.0%) in all the sequences (in terms of the best known values). With the test sequences, the evolved neural-CA obtains 30 HH contacts of the 47 possible HH contacts (63.8%). In the second case (Table 5), the results are more equally balanced between the training and test results. With the training, the evolved neural-CA obtains 40 HH contacts of the 47 possible HH contacts (85.1%), whereas with the test sequences the same evolved neural-CA obtains 38 HH contacts of the 50 possible HH contacts (76%).

We repeated the analysis using a cross-validation process, with the same aim of testing the generalization capability with sequences not used in the training as well as whether the neural-CA avoids overfitting. We used nine sequences for the training, whereas the evolved neural-CA is tested with the remaining sequence (10 cross-validation). Table 6 shows the results. The first column of Table 6 specifies the validation sequence, whereas the other nine (Table 1) are used for the training set. As in the previous case, a maximum number of 10,000 evaluations were allowed for each one of the training sequences. The second column of Table 6 shows the total number of HH contacts obtained in the training set (summing the HH contacts of each training sequence) and, in parentheses, the total number of possible HH contacts that could be obtained (considering the best known values for the different sequences), together with the percentage of HH contacts obtained with respect to the possible optimal number. The third column (E_{min}) specifies the best known value for the validation sequence, whereas the last column shows the final energy (number of HH contacts) after the folding process when the evolved neural-CA is used for that validation sequence, together with the corresponding percentage its final HH contacts obtained.

Considering the training sequences, the percentages of final HH contacts obtained are larger than 80% (average value of 85.7%), as in the previous case (training and test sets with five sequences) in which 88.0% and 85.1% of the possible final HH contacts were obtained. These must be considered as large percentages, taking into account that the same trained neural-CA is the one that folds the different training sequences. Considering always the validation of the sequences, some protein chains obtained better results (273d.3, 273d.4, 273d.6, 273d.9, 273d.10), while in others the results were worse (273d.2, 273d.5, 273d.7). Considering all the validation sequences, the average percentage of HH contacts obtained is 71.1%. So, in this case there is no noticeable improvement (with respect to the previous case shown in Tables 4 and 5) when considering more training sequences, and the average percentages of HH contacts with the training sequences (85.7%) and validation sequences (71.1%) points out that the evolved neural cellular automata are performing without overfitting.

It is important to emphasize that in all cases the final folding tends to locate the H amino acids in the inner area of the folding core. For example, Figure 4 shows two final folded configurations for sequence 273d.4. The one at the left corresponds to the optimal folding (best known number of final HH contacts),

Table 6. Results with a 10 Cross-Validation Process

Sequence	*HH contacts (training)*	E_{min} *(validation)*	*E (HH contacts, validation)*
273d.1	77 (88), 87.5%	−9	−6, 66.7%
273d.2	72 (87), 82.8%	−10	−7, 70.0%
273d.3	75 (89), 84.3%	−8	−7, 87.5%
273d.4	70 (82), 85.4%	−15	−11, 73.3%
273d.5	74 (89), 83.1%	−8	−5, 62.5%
273d.6	74 (85), 87.5%	−12	−9, 75.0%
273d.7	73 (84), 88.9%	−13	−7, 53.8%
273d.8	76 (93), 81.7%	−4	−3, 75.0%
273d.9	78 (90), 86.6%	−7	−4, 57.1%
273d.10	77 (86), 89.5%	−11	−10, 90.9%

First column: validation sequence (the other 9 [Table 1] are included in the training set). Second column: number of HH contacts obtained in the training set, total number of possible HH contacts that could be obtained (in parentheses), and percentage of HH contacts obtained. Third column (E_{min}): best known value for the validation sequence (Mansour et al., 2011). Last column: final number of HH contacts for that validation sequence and the percentage of HH contacts obtained.

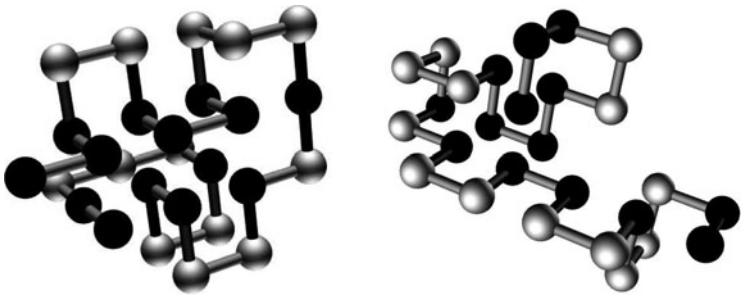

FIG. 4. (*Left*) Optimal configuration of sequence 273d.4 (15 HH contacts). (*Right*) Final folded configuration using the evolved neural-CA trained with the last five 27-length training sequences (Table 5) and 273d.4 as a test sequence.

corresponding to the result when such sequence was in the training set (Table 4), whereas the final folded conformation at the right corresponds to a suboptimal one when the sequence 273d.4 was used as a validation sequence for a neural-CA trained with the sequences indicated in Table 5. In this second case, the H amino acids tend to be located in the inner area, but a simple ''nonperfect'' movement in the folding (in the sense that it is not the best in terms of maximizing the final HH contacts) can drive the process toward a different folding path through time. The reason is in the hard constraints of the HP model, which allows very few movements available for every amino acid, so the posterior movements in the next amino acids cannot correct a ''nonperfect'' previous movement.

In the last analysis, we test the capability of evolved neural cellular automata to define the folding of proteins with different length to those used in the training, so now we repeated the process working with proteins of different length in the training and test sets. We trained the neural-CA folding model with the 10 sequences of Table 1 (length 27), whereas the protein sequences of Table 2 (length 64) were used for the validation process. Table 7 (left section) shows the results for each of the training sequences, whereas the section on the right shows the results when the validation sequences were folded with the trained neural-CA. With the training sequences the evolved neural-CA obtains 81 HH contacts in the final folded conformations out of 97 (summing all the best known values taken from Mansour et al., 2011), representing 83.5%. When the trained neural-CA is used in the validation process with the sequences of length 64, Table 7 (right section) shows that the trained CA obtains 197 out of 323 possible HH contacts (using again the best known values from Mansour et al., 2011). This represents 61.0%, a lower value since it must be taken into consideration the larger number of amino acids in those sequences.

Finally, we swapped the training and test sets, using the longest sequences of 64 amino acids for the training and the sequences of length 27 for the validation. Table 8 shows the results. The left part of Table 8 indicates that, for the training sequences, the evolved neural-CA obtains 264 out of 323 possible HH contacts, which represents 81.7% of the possible contacts. The right section of Table 8 shows the results when the 27-length amino acid sequences were used as test sets, obtaining in this case 70 out of 97 possible contacts, representing 72.2%, which is a more equal balance in the training and validation results as a consequence of the training with the longest sequences.

TABLE 7. RESULTS TRAINING THE NEURAL-CA WITH THE SEQUENCES OF LENGTH 27

Sequence	E_{min}	*HH Contacts*	*Sequence*	E_{min}	*HH Contacts*
273d.1	−9	−6	**643d.1**	−28	−14
273d.2	−10	−9	**643d.2**	−32	−15
273d.3	−8	**−8**	**643d.3**	−40	−27
273d.4	−15	−13	**643d.4**	−35	−25
273d.5	−8	−7	**643d.5**	−36	−16
273d.6	−12	−10	**643d.6**	−31	−24
273d.7	−13	−10	**643d.7**	−25	−15
273d.8	−4	−3	**643d.8**	−35	−21
273d.9	−7	−6	**643d.9**	−34	−26
273d.10	−11	−9	**643d.10**	−27	−14

Left: results with the training sequences. Right: results with the validation sequences.

TABLE 8. RESULTS TRAINING THE NEURAL-CA WITH THE SEQUENCES OF LENGTH 64

Sequence	E_{min}	*HH Contacts*	*Sequence*	E_{min}	*HH Contacts*
643d.1	−28	−26	**273d.1**	−9	−5
643d.2	−32	−23	**273d.2**	−10	−8
643d.3	−40	−37	**273d.3**	−8	−6
643d.4	−35	−28	**273d.4**	−15	−13
643d.5	−36	−32	**273d.5**	−8	−6
643d.6	−31	−21	**273d.6**	−12	−8
643d.7	−25	−19	**273d.7**	−13	−9
643d.8	−35	−27	**273d.8**	−4	−3
643d.9	−34	−29	**273d.9**	−7	−6
643d.10	−27	−22	**273d.10**	−11	−6

Left: results with the training sequences. Right: results with the validation sequences.

3.3.1. Results with real proteins. Previous works using the basic HP model (2D square or 3D cubic lattices) do not include real proteins to test the different search algorithms of final folded conformations, focusing the results on the algorithmic aspects and on different benchmark sequences. Nevertheless, we tested our methodology using proteins from the target list of the CASP9 competition, the same proteins used by Shatabda et al. (2012) and Rashid et al. (2013), although in their case they used the more detailed face-centered cubic (FCC) lattice model. To fit in the HP model, the CASP targets were converted to HP sequences based on the hydrophobic properties of the constituent amino acids.

Figure 5 includes two examples of two final folded conformations with proteins 3no6 (229 amino acids) and 3on7 (279 amino acids), when the best neural-CA evolved with the training set of Table 8 (proteins with 64 amino acids) is used to define the folding of these test proteins. The protein names correspond to the pdb code (PDB, 2014).

When the evolved neural-CA is applied to protein 3no6 the final folding presents 91 HH contacts in the 3D cubic lattice model used, whereas the final folded conformation has 97 HH contacts using protein 3on7 as the test sequence. In both cases, the final folding tends to locate the H amino acids in the inner areas of the folded conformation. The number of HH contacts obtained by Rashid et al. (2013) was 424 with the sequence 3no6 and 526 using the sequence 3on7, when they used a hybrid genetic algorithm that integrated a tabu-based local search within a genetic algorithm framework for the direct prediction of the final folded conformation. However, there is not a maximum known value of HH contacts using the 3D cubic HP model for those sequences, and we must take into account that the number of HH contacts obtained using the FCC lattice can be larger with respect to the use of the 3D cubic HP lattice (our case) for the same protein. So, despite the lower number of HH contacts with respect to the FCC lattice model and direct prediction methods, it must be emphasized, again, the advantage that an evolved neural-CA can define the folding of different proteins.

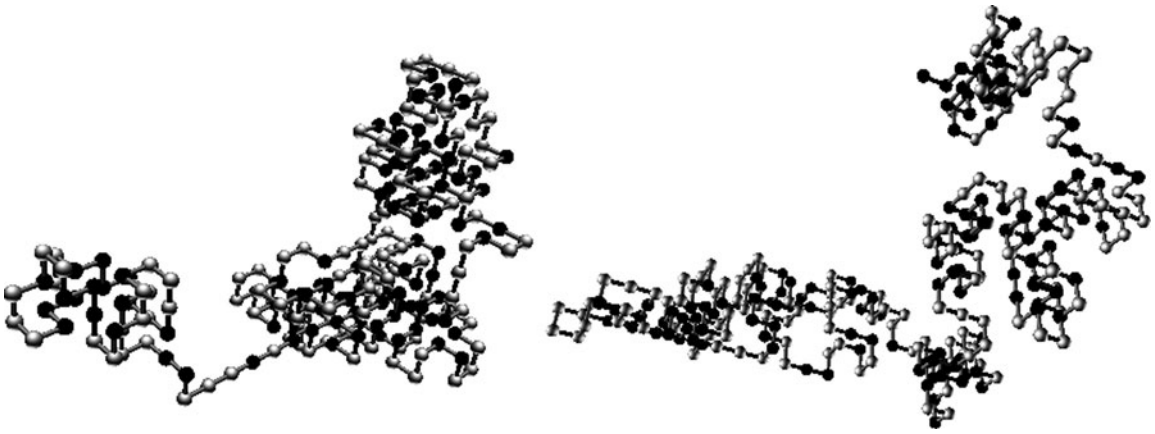

FIG. 5. (*Left*) Final folding using sequence 3no6 as test protein for an evolved neural-CA trained with the sequences of length 64 (Table 8). (*Right*) Final folding using sequence 3on7.

4. CONCLUSIONS

We defined an alternative methodology for modeling the protein folding process. We used cellular automata, implemented by means of connectionist systems (neural cellular automaton or ''neural-CA''), to define the protein folding process through time and use the HP lattice model. This new proposal is different from most of the previous work focused on the direct prediction of the final protein conformations.

The neural model acts like a classical CA system, in the sense that it is applied to all the simple elements or amino acids and through different temporal steps, deciding the next movement in each amino acid until a final folded conformation is reached. Nevertheless, contrary to classical CA models, the neural-CA model takes into consideration the vision of the immediate energy landscape, instead of the spatial neighborhood, to facilitate the decision of the amino acid movements. Thus, a neural cellular automaton has a view of the energy landscape that implies the HP lattice model (taking into account its constraints), so the evolved neural-CA learns how to move the amino acids through the energy or fitness landscape to obtain the optimal folding, using the appropriate movements (greedy or not) to optimize the final and emergent folded conformations. Thus, the process is treated now as an emergent and dynamic process.

The neural-CA models were automatically obtained by means of Differential Evolution. The fitness function was defined as the final number of HH contacts, as in the problem of the direct prediction of the final folded conformation. So, we used a fitness function with low ''level of canalization'' (typical of artificial life systems that study emergent processes) in the sense that we have not used more information related to intermediate steps in the folding. Nevertheless, the fitness used was sufficient to find out the optimized folded structures by the evolved neural-CA.

Finally, since the iterative modeling by a neural-CA obtains the final conformation using only the information of the primary structure, we can compare the results in terms of final HH contacts between our methodology and other search methods that obtain or predict directly the final folded conformation. The results with benchmark sequences showed that our method obtains the same best values as the best search methods of the direct and final protein conformation, and also needs fewer evaluations to obtain the best final conformations.

Additionally, we tested the generalization capability of the neural-CA methodology, when a neural-CA is evolved/trained with a subset of protein sequences and tested with different sequences that can also have different lengths. Thus, the evolved neural-CA is used as an operator that defines the emergent folding of any protein. This is a clear advantage over the NP-hard nature of the prediction problem of final protein conformations, in which a search algorithm has to be run for each individual protein sequence. The results indicated that our methodology obtains final folded protein conformations that tend to locate the H amino acids in the inner area of the folding cores, although in most cases the best known values of HH contacts were not obtained in the validation sequences. As indicated, the reason is in the hard constraints of the HP model, which impose very few possibilities of amino acid movements, so a future adaptation should extend the methodology to more detailed lattice models like the face-centered-cubic (FCC) lattice (Bagci et al., 2002; Shatabda et al., 2013; Hoque et al., 2009), which has the highest packing density compared to the other existing lattices (Hales, 2005) and allows more amino acid movements and more detailed spatial resolution of the conformations. Another aspect to consider is the use of a multiobjective optimization, simultaneously optimizing (maximizing) the final HH contacts and, for example, minimizing the number of steps that are necessary to obtain the final folded conformations.

ACKNOWLEDGMENTS

This work was funded by the Ministry of Economy and Competitiveness (project TIN2013-40981-R).

AUTHOR DISCLOSURE STATEMENT

The authors declare that no competing financial interests exist.

REFERENCES

Bagci, Z., Jernigan, R., and Bahar., I. 2002. Residue coordination in proteins conforms to the closest packing of spheres. Polymer 43, 451–459.

Benítez, C., and Lopes, H. 2009. A parallel genetic algorithm for protein folding prediction using the 3D-HP side chain model. Proceedings of IEEE Congress on Evolutionary Computation, 1297–1304.

Berger, B., and Leighton, T. 1998. Protein folding in the hydrophobic–hydrophilic (HP) model is NP-complete. *J. Comp. Biol.* 5, 27–40.

Blazewicz, J., Dill, K., Lukasiak, P., and Milostan, M. 2004. A tabu search strategy for finding low energy structures of proteins in HP-model. *Computational Methods in Science and Technology* 10, 7–19.

Bromberg, S., and Dill, K. 1994. Side-chain entropy and packing in proteins. Protein Science 3, 997–1009.

Calabretta, R., Nolfi, S., and Parisi, D. 1995. An artificial life model for predicting the tertiary structure of unknown proteins that emulates the folding process. Proceedings of the Third European Conference on Advances in Artificial Life - LNCS 929, 862–875.

CASP, 2014. Protein structure prediction center. http://predictioncenter.org/

Cotta, C. 2003. Protein structure prediction using evolutionary algorithms hybridized with backtracking. *Lecture Notes in Computer Science* 2687, 321–328.

Cox, G., Mortimer-Jones, T., Taylor, R., and Johnston, R. 2004. Development and optimization of a novel genetic algorithm for studying model protein folding. *Theoretical Chemistry Accounts* 112, 163–178.

Crescenzi, P., Goldman, D., Papadimitriou, C., Piccolboni, A., and Yannakakis, M. 1998. On the complexity of protein folding. *J. Comp. Biol.* 5, 423–466.

Custódio, F., Barbosa, H., and Dardenne, L. 2004. Investigation of the three dimensional lattice HP protein folding model using a genetic algorithm. *Genetics and Molecular Biology* 27, 611–615.

Cutello, V., Nicosia, G., Pavone, M., and Timmis, J. 2007. An immune algorithm for protein structure prediction on lattice models. *IEEE Transactions on Evolutionary Computation* 11, 101–117.

Danks, G., Stepney, S., and Caves, L. 2008. Protein folding with stochastic L-systems. Artificial Life XI: Proceedings of 11th Int. Conf. on the Simulation and Synthesis of Living Systems (MIT Press, Cambridge, MA), pp. 150–157.

Dill, K. 1990. Dominant forces in protein folding. Biochemestry 29, 7133–7155.

Dill K., et al. 1995. Principles of protein folding - a perspective from simple exact models. *Protein Science* 4, 561–602.

Dotu, I., Cebrián, M., Hentenryck, P. V., and Clote, P. 2011. On lattice protein structure prediction revisited. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8, 1620–1632.

Duarte, S., Becerra, D., Nino, F., and Pinzón, Y. 2007. A novel ab-initio genetic-based approach for protein folding prediction. Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'07, 393–400.

Feoktistov, V. 2006. *Differential Evolution: In Search of Solutions*. Springer, New York.

Fidanova, S. 2006. 3D HP protein folding problem using ant algorithm. Proceedings BioPS'06, 19–26.

Garza-Fabre, M., Rodriguez-Tello, E., and Toscano-Pulido, G. 2012a. Multiobjectivizing the HP model for protein structure prediction. Proceedings EvoCOP'12, Evolutionary Computation in Combinatorial Optimization - LNCS, 7245, 182–193.

Garza-Fabre, M., Rodriguez-Tello, E., and Toscano-Pulido, G. 2013a. Comparative analysis of different evaluation functions for protein structure prediction under the HP model. *Journal of Computer Science and Technology* 28, 868–889.

Garza-Fabre, M., Toscano-Pulido, G., and Rodriguez-Tello, E. 2012b. Locality-based multiobjectivization for the HP model of protein structure prediction. Proceedings of Genetic and Evolutionary Computation Conference - GECCO'12, 473–480.

Garza-Fabre, M., Toscano-Pulido, G., and Rodriguez-Tello, E. 2013b. Handling constraints in the HP model for protein structure prediction by multiobjective optimization. Proceedings of IEEE Congress on Evolutionary Computation - IEEE-CEC 2013, 2728–2735.

Giaquinta, E., and Pozzi, L. 2013. An effective exact algorithm and a new upper bound for the number of contacts in the hydrophobic-polar two-dimensional lattice model. J. Comp. Biol. 20, 593–609.

Hales, T. 2005. A proof of the Kepler conjecture. The Annals of Mathematics 162, 1065–1185.

Handl, J., Lovell, S., and Knowles, J. 2008. Investigations into the effect of multiobjectivization in protein structure prediction. *Parallel Problem Solving from Nature, Lecture Notes in Computer Science* 5199, 702–711.

Hart, W., and Istrail, S. 1995. Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal (extended abstract). Proceedings of 27th Annual ACM Symposium on Theory of Computation (STOC95), 157–168.

Hart, W., and Istrail, S. 1997. Robust proofs of NP-hardness for protein folding: General lattices and energy potentials. J. Comp. Biol. 4, 1–22.

Hirst, J. 1999. The evolutionary landscape of functional model proteins. Protein Engineering 12, 721–726.

Hoque, T., Chetty, M., and Sattar, A. 2009. Extended HP model for protein structure prediction. *J. Comp. Biol.* 16, 85–103.

Ilachinski, A. 2001. *Cellular automata. A discrete universe*. World Scientific.

Istrail, S., and Lam, F., 2009. Combinatorial algorithms for protein folding in lattice models: A survey of mathematical results. *Communications in Information and Systems* 9, 303–346.

Johnson, C., and Katikireddy, A. 2006. A genetic algorithm with backtracking for protein structure prediction. Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'06, 299–300.

Krasnogor, N., Blackburne, B., Burke, E., and Hirst, J. 2002a. Multimeme algorithms for protein structure prediction. *LNCS* 2439, 769–778.

Krasnogor, N., Hart, W., Smith, J., and Pelta, D. 1999. Protein structure prediction with evolutionary algorithms. Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'99, 1596–1601.

Krasnogor, N., Marcos, D., Pelta, D., and Risi, W. 1998. Protein structure prediction as a complex adaptive system. *In* Janikow, C. ed. Proceedings of Frontiers in Evolutionary Algorithms, 441–447.

Krasnogor, N., Terrazas, G., Pelta, D., and Ochoa, G. 2002b. A critical view of the evolutionary design of self-assembling systems. Proceedings of the 2005 Conference on Artificial Evolution, LNCS 3871, 179–188.

Langton, C. 1992. Life at the edge of chaos, 41–49. *In*: Langton, C.G., Taylor, C., Farmer, J.D., Rasmussen, S., eds. *Artificial Life II*. Addison-Wesley.

Lesh, N., Mitzenmacher, M., and Whitesides, S. 2003. A complete and effective move set for simplified protein folding. Proceedings of the seventh annual international conference on research in computational molecular biology - RECOMB'03, 188–195.

Levinthal, C. 1968. Are there pathways for protein folding? *J. Chim. Phys.* 65, 44–45.

Lopes, H., and Bitello, R. 2007. Differential evolution approach for protein folding using a lattice model. *Journal of Computer Science and Technology* 22, 904–908.

Lopes, H., and Scapin, M. 2006. An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic-polar model. *LNCS* 3871, 238–246.

Mansour, N., Kanj, F., and Khachfe, H. 2011. Enhanced genetic algorithm for protein structure prediction based on the HP model. *Search Algorithms and Applications, InTech*, 69–78.

Moult, J., Fidelis, K., Kryshtafovych, A., and Tramontano, A. 2011. Critical assessment of methods of protein structure prediction (CASP)-round IX. *Proteins: Structure, Function, and Bioinformatics* 79, 1–5.

Patton, W., Punch, W., and Goldman, E. 1995. A standard genetic algorithm approach to native protein conformation prediction. Proceedings of 6th International Conference on Genetic Algorithms, 574–581.

PDB. 2014. Protein data bank. Available at: www.wwpdb.org

Price, K., Storn, R., and Lampinen, J. 2005. *Differential Evolution. A practical approach to global optimization*. Springer, New York.

Rashid, M., Newton, M., Hoque, M., and Sattar, A. 2013. A local search embedded genetic algorithm for simplified protein structure prediction. Proceedings IEEE Congress on Evolutionary Computation - IEEE-CEC 2013, 1091–1098.

RS. 2014. Rosetta system. Available at: www.rosettacommons.org

Rylance, G. 2004. Applications of genetic algorithms in protein folding studies [PhD thesis]. University of Birmingham.

Santos, J., and Diéguez, M. 2011. Differential evolution for protein structure prediction using the HP model. *Lecture Notes in Computer Science* 6686, 323–323.

Santos, J., Villot, P., and Diéguez, M. 2013. Cellular automata for modeling protein folding using the HP model. Proceedings IEEE Congress on Evolutionary Computation - IEEE-CEC 2013, 1586–1593.

Shatabda, S., Newton, M., Pham, D., and Sattar, A. 2012. Memory-based local search for simplified protein structure prediction. Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine - BCB '12, 345–352.

Shatabda, S., Newton, M., Rashid, M., and Sattar, A. 2013. An efficient encoding for simplified protein structure prediction using genetic algorithms. Proceedings IEEE Congress on Evolutionary Computation - IEEE-CEC 2013, 1217–1224.

Shmygelska, A., and Hoos, H. 2005. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *Bioinformatics* 6, 30.

Song, J., Cheng, J., Zheng, T., and Mao, J. 2005. A novel genetic algorithm for HP model protein folding. Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies, 935–937.

Song, J., Cheng, J., Zheng, T., and Mao, J. 2006. Protein 3D HP model folding simulation based on ACO. Sixth International Conf. on Intelligent Systems Design and Applications (ISDA'06), 410–415.

Storn, R., and Price, K. 1997. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* 11, 341–359.

Tramontano, A. 2006. *Protein structure prediction*. Concepts and applications. Wiley-VCH.

Unger, R. 2004. The genetic algorithm approach to protein structure prediction. *Structure and Bonding* 110, 153–175.

Unger, R., and Moult, J. 1993a. A genetic algorithm for 3D protein folding simulations. Proceedings of the Fifth Annual International Conference on Genetic Algorithms, 581–588.

Unger, R., and Moult, J. 1993b. Genetic algorithms for protein folding simulations. *J. Mol. Biol.* 231, 75–81.
Zaki, M., and Bystroff, C. 2008. *Protein Structure Prediction*. (2nd Edition), Humana Press New York.
Zhao, X. 2008. Advances on protein folding simulations based on the lattice HP models with natural computing. *Applied Soft Computing* 8, 1029–1040.

Address correspondence to:
José Santos
Department of Computer Science
University of A Coruña
Campus de Elviña s/n
15071 A Coruña
Spain

E-mail: jose.santos@udc.es