Protein Folding with Cellular Automata in the 3D HP Model

José Santos
Department of Computer Science
University of A Coruña
Campus de Elviña s/n, 15071
A Coruña (Spain)
santos@udc.es

Pablo Villot
Department of Computer Science
University of A Coruña
Campus de Elviña s/n, 15071
A Coruña (Spain)
pablo.villot@udc.es

Martín Diéguez
Department of Computer Science
University of A Coruña
Campus de Elviña s/n, 15071
A Coruña (Spain)
martin.dieguez@udc.es

ABSTRACT

In the difficult *ab initio* prediction in protein folding only the information of the primary structure of amino acids is used to determine the final folded conformation. The complexity of the interactions and the nature of the amino acid elements are reduced with the use of lattice models like HP, which categorizes the amino acids regarding their hydrophobicity. On the contrary to the intense research performed on the direct prediction of the final folded conformation, our aim here is to model the dynamic and emergent folding process through time, using the scheme of cellular automata but implemented with artificial neural networks optimized with Differential Evolution. Moreover, 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.

Categories and Subject Descriptors

F.1 [**Theory of Computation**]: Computation by abstract machines—*Unbounded-action devices*; I.2.6 [**Artificial Intelligence**]: Learning—*Connectionism and neural nets*; J.3 [**Computer Applications**]: Life and medical sciences—*Biology and genetics*

General Terms

Theory

Keywords

Protein folding, cellular automata, differential evolution

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 and time consuming, 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. In the first case the search space is pruned by the assumption that the target protein adopts a structure close to the experimentally determined structure of another homologous protein. Nevertheless, the experimental determination of the protein conformations remains far behind the rapid increase of the number of known protein sequences.

Thus, 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 [26]. 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 [25]. 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 (as in the HP lattice model [4]) is a major determinant of the folding of a protein.

In these lattice models, given a primary sequence, the problem is to search for the folding structure in the lattice that minimizes the energy. The complexity of the problem has been shown to be NP-hard [10, 28] and the progress was slow; as Unger points out "minimal progress was achieved in the category of ab initio folding" [27]. Along this line, many authors have been working on several evolutionary algorithms [9, 20, 22, 27, 28] or other natural computing algorithms [2, 8, 23, 24, 29] on the direct prediction of the native conformations using the HP model.

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 [12, 16]. We will use CA to define the folding of a protein through time in the 3 dimensional space of possible conformations imposed

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

by simple lattice models like the HP model. CA 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 [12].

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 neural networks for their implementation and we will use evolutionary computing to automatically obtain the models.

Comments on previous works in protein folding modeling

Levinthal's paradox [17] 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 [6].

As commented, in most of the previous work in the field, the different evolutionary methods worked only as search algorithms of structures of minimal energetic configuration. 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 energetic component which minimizes the distance between H (hydrophobic) amino acids, used in Krasnogor et al. [14], 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. [15] used cellular automata and Lindenmayer systems to try to define the rules and dynamics of such process, with a very limited success. They used a one-dimensional cellular automaton with four states that correspond to the possible movements in 2D 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 the rule was being applied to, 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. The work with Lindenmayer systems was only oriented to find 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. [1], the authors tried to establish the tertiary structure modeling the folding process through matrices of attraction potentials among the 20 amino acids. The matrices of 20x20 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 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 obtained success only in chains with very few amino acids (13 in the example of their paper).

Also, more recently Danks et al. [3] presented a Lindenmayer system model which 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 that 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 could 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 which was not considered or studied previously in the literature. Our main goal is the 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 [29]. The CA models will determine the movements of the amino acids through time and considering the restrictions of the HP model, and they will be implemented with artificial neural networks using input information directly from the energy landscape.

Our work is inspired by the work of Krasnogor et al. [15] in the use of CA for the modeling of the folding. Nevertheless, one of the main problems of the cellular automata used by Krasnogor et al. [15] 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 of 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 [16]. For instance, in [15], the authors considered $K = 4$ states in each cell (the 4 previous movements of amino acids in a 2D lattice) and a neighborhood of 5 in a one-dimensional cellular automaton, using the two closest amino acids in both sides of the chain. Hence, there was $4^{(4^5)}$ 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 with 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.

2. METHODS

2.1 HP lattice model

As indicated, to reduce the complexity of the problem of protein folding, simplified models are used. In the HP model [4] 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 2D square or triangular lattices, or 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. As commented, the complexity of the problem is NP-hard [10, 28], so determining the native state conformations of HP sequences has become a computational challenge. Although the HP model is simple, it is non-trivial, captures many global aspects of real proteins and still remains the hardness features of the original biological problem [5]. For this reason, many authors have been working on several evolutionary algorithms [27, 29] for the direct prediction of the native conformations using the HP model, as indicated in the previous section.

Using evolutionary computing to determine the protein conformations under the HP model, one of the main decisions is the genotypic 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 moves are allowed: Forward, Turn Up, Turn Down, Turn Left, Turn Right. The conformations are expressed now as sequences $\{F, U, D, L, R\}^{n-2}$. That representation has the advantage of guaranteeing that all solutions are 1-step self-avoiding (because there is no back move).

Both alternatives were used in evolutionary computing works to encode the protein conformations. For example, Unger and Moult [28] used the absolute representation of the movements and Patton et al. [20] used the relative movements to define the conformations. The results of Krasnogor et al. [14] supported the use of the relative encoding when they analyzed the impact of different factors when evolutionary algorithms are used to the problem. Nevertheless, there is not an ample study that determined which representation is the best. We will use the relative representation of the movements, because of the interesting property of avoiding 1-step self-conflicts.

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)
               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
```

2.2 Differential Evolution

Differential Evolution (DE) [21] 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 difference vectors for generating perturbations in a population of vectors. This algorithm is specially suited for optimization problems where possible solutions are defined by a real-valued vector. The basic DE algorithm is summarized in the pseudo-code 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 over 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.

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 [7] 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", 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 sequence sequentially, and this process is repeated through different temporal iterations or steps across the full sequence of amino acids. As 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 code 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, the parameters are the weights between the nodes of the neural network layers.

Inputs of the neural network

1. For each possible movement in current position or amino acid i of the chain, we calculate the increment (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 increment of energy for such movement.

2. For each of the possible movements 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 increments in energy (with respect to the energy with 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 increment of energy if the neural network decided a greedy strategy.

Outputs of the neural network

Figure 1 represents the process but with the 2D case to clarify the ANN use. The ANN has an output corresponding to the 3 possible relative movements $\{F, L, R\}$ (in 2D). In the 3D case the network has 5 outputs corresponding to the 5 possible relative movements $\{F, U, D, L, R\}$. The neural-CA decides which the most adequate movement in each situation is.

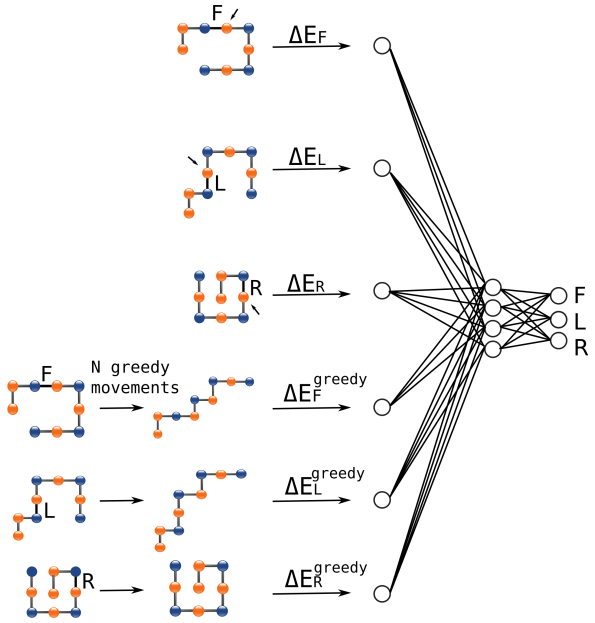

Figure 1: Artificial neural network used for determining the next movement in each amino acid. The inputs correspond to the energy increments (with respect to the current energy) when the possible movements are applied in the amino acid position and when greedy movements are considered after such possible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

For determining the inputs, as indicated before, all the possible movements are applied in the current position and the energy increments with respect to the current movement are calculated, being inputs to the ANN (obviously one 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 (being N a parameter to tune). Again, the increment of energy of the resultant conformation, with respect to the current protein conformation, is an input to the ANN (1 additional input corresponding to each one of the possible movements in position i).

Note that this way, the ANN has a view of the energy space in order to decide the most adequate 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 output node with the highest activation value determines the next movement.

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 nodes of the chain, the protein conformation can pass through illegal states, and it is one of the ANN responsibilities to try 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 try 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.

Fitness definition

We use simulated evolution to optimize a given neural-CA. This 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.

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 until a final conformation is reached, where 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. This helps that the evolutionary population has a great diversity of fitness, which permits to refine the ANN to not determine movements with conflicts in the immediate neighborhood of the current position.

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 to the definition of the energy used to calculate the energy increments 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 Mould [28] and Patton et al. [20]. The test data consisted of a series of 10 randomly produced 27 length sequences. Table 1 shows the sequences used for the 3D cubic lattice, which were firstly published in the work of Unger and Moult [28]. The global minimum energy value for these random configurations is unknown.

	HP Chain
273d.1	*PHPHPHHHPPHPHPPPPPPPPPPPHHP*
273d.2	*PHHPPPPPPPPPPHHPPHHPPHPPHPH*
273d.3	*HHHHPPPPPHPPPPPHHHPPPPPPPPH*
273d.4	*HHHPPHHHHPPPHPHPPHHPPHPPPHH*
273d.5	*HHHHPPPPHPHHPPPHHPPPPPPPPPP*
273d.6	*HPPPPPPHPHHHPPHHPPPHPPPPHPH*
273d.7	*HPPHPHHPPPHPPPPPHPHHPHPHPHH*
273d.8	*HPPPPPPPPPPPHPHPPPPPPPPHPHH*
273d.9	*PPPPPPPHHHPPPHPHHPPPHPPHPPP*
273d.10	*PPPPPHHPHPHPHPHPPHHPHHPHPPP*

Table 1: Benchmarks sequences, for the cubic lattice, used in the experiments.

3. RESULTS

First we can see an example of the folding 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, 5 output nodes) for a simple protein sequence used by Patton et al. [20], which consists of 8 hydrophobic residues evenly interspersed with two hydrophilic residues each (HPPHPPHPPHPPHPPHPPH). 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 interior 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 5 possible relative movements of the current amino acid was $N = 3$, in order to calculate the 5 energy increments that are inputs to the ANN when such posterior greedy movements are considered. For the calculation of energy increments, 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 or equal than 5 with respect to the central amino acid. Such central position is given by the previous amino acid of the one that is currently subject to move. Finally, for each sequence we allowed a maximum number of 10 folding steps, that is, the neural-CA is applied over all the amino acids of the protein chain a maximum number of 10 times, beginning with the first amino acid an applying the neural-CA sequentially. If 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 defined by the neural-CA. For instance, in the example of Figure 2, the evolved neural-CA needed only 5 steps to complete the folding. Moreover, the ANN applied only 20.7% of greedy movements.

Regarding Differential Evolution, we used standard values: $CR = 0.9$ and $F = 0.9$, 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 [21] and [18]). The DE individuals code the ANN weights in the range [-1,1] and decoded multiplying the encoded value by a constant MAX_VALUE. We used MAX_VALUE=2 since this value allows to saturate the

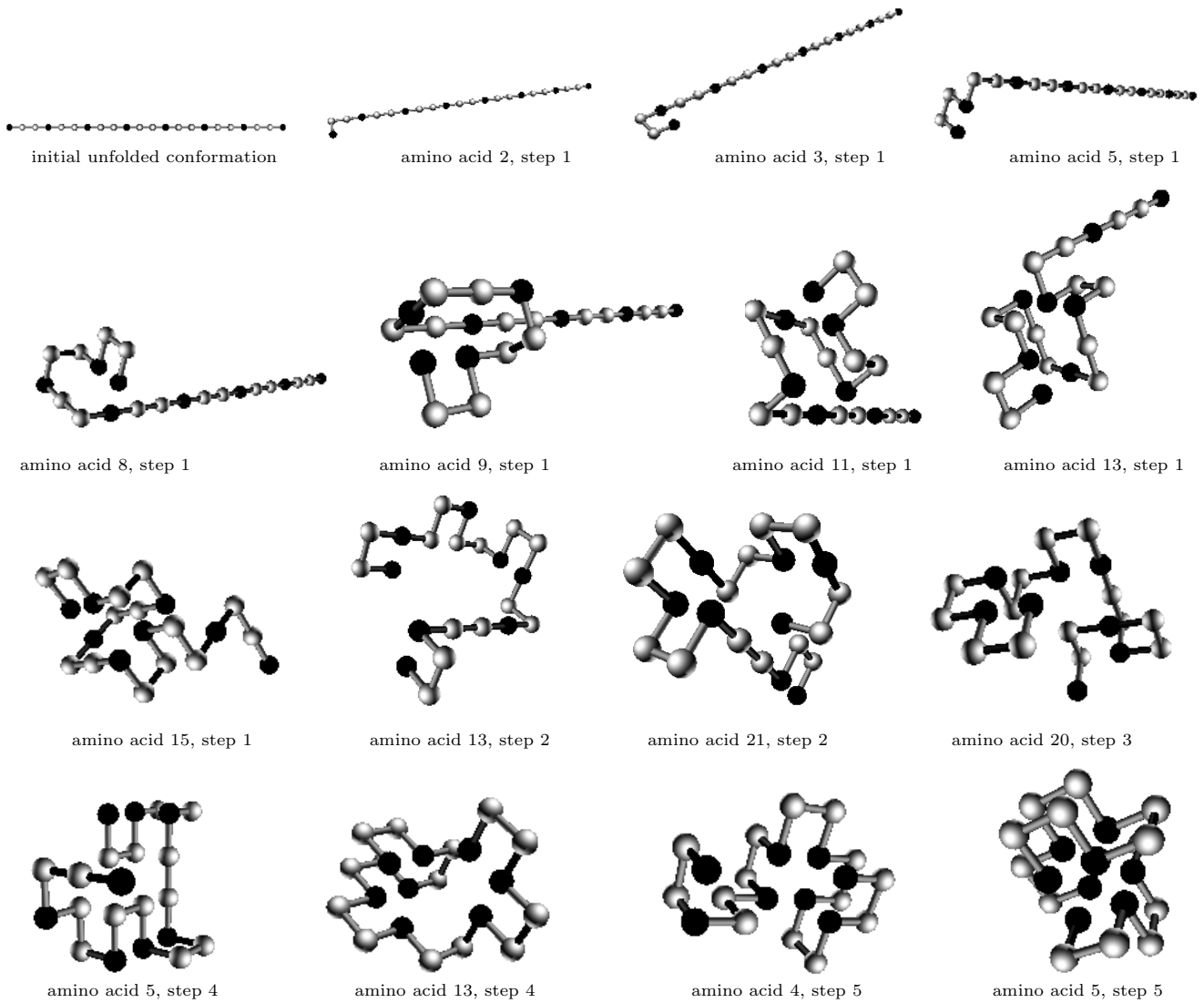

Figure 2: Different temporal steps in the folding process with protein sequence HPPHPPHPPHPPHPPHP-PHPPH.

Seq.	neural-CA (2)	neural-CA (1)	M et al GA [19]	J&K GA [13]	P et al GA [20]	U&M GA [28]
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

Table 2: Comparison of results with the benchmark sequences. In each column of results: Best energy value (HH contacts) in different independent runs and, in parentheses, the minimum number of evaluations to find the best value. After ",": Average best energy value in the different runs and average number of evaluations to find the best values in each run (in parentheses).

nodes using a standard sigmoid function as transfer function of the ANN nodes.

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 energy optima. In Table 2 we included the comparison of the HH contacts of our final conformations after the iterative folding process, with respect to previous works which used search methods to predict directly the final conformations. Table 2 includes the comparison of our results with the ones in other works, such as the results of Unger and Moult using a Genetic Algorithm (GA) hybridized with a Monte Carlo local search [28] and Patton et al. [20] 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 the 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 indicated 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 [28] from 5 independent runs. In the case of Patton et al. [20] the authors do not specify how many different independent trials were run. Additionally, in our case, we included in the Table 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 ","). 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 [28], 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 which 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 which become 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. With all these considerations, as indicated in the table, the GA of Unger and Moult [28] operated on each of the 27 length sequences for roughly 1.2 million function evaluations, and obtaining better comparison results of performance with respect to a Monte Carlo approach.

Regarding the work of Patton et al. [20], the authors used the relative movements to define the conformations in the 3D lattice and, on the contrary to Unger and Moult [28], they allowed illegal conformations in the genetic population, but penalizing them for positional collisions. This essentially allows the search to proceed through illegal states. Patton et al. [20] obtained results which compare quite favorably with those of Unger and Moult, as shown in the table. For example, as the authors remark in 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 [13], the authors report less number of evaluations (the best value in five trials) for obtaining 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. [19], 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: neural-CA (1), only HH contacts were considered in the calculations of energy increments; neural-CA (2), HH contacts and HP (or PH) contacts were considered in the energy increment calculations. The results shown in Table 2 in this second alternative were obtained weighting the HH contacts with -1 and penalizing the HP and PH contacts with a weight 0.5, in a similar way to the HP Functional Model [11] which considers all contact possibilities between amino acids. These parameters provided us with the best results in most of the sequences.

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. As shown in the table, the second configuration, neural-CA (2), which uses more information in the calculation of the energy increments (inputs to the neural-CA) than the basic information provided by the HH contacts, provided better results in terms of number of necessary evaluations to obtain the best values. This is because the ANN has more detailed information to decide the movement to apply in each situation. Or, in other words, the ANN has a more detailed view of how is the immediate and dynamic energy landscape when the protein is folding and in order to decide the next movement.

We obtained in all cases the best value reported in the previous works. In sequence 273d.8, using neural-CA (1), more evaluations were necessary to obtain the best value (-4), and because the few input information provided only by the short number of possible HH contacts (a maximum of 125000 evaluations were applied in this case). Additionally, using neural-CA (2), in most of the sequences, fewer individual evaluations were required to obtain the best values. Even the average number of necessary evaluations to obtain the best values is lower with respect to, for example, the values published in [13] for their best case (except 273d.8). The better values (in terms of number of evaluations) in some sequences in [19] are explained because of the filters applied by the authors to the outcome of the crossover and mutation operators in their enhanced GA.

4. CONCLUSIONS

We used cellular automata, implemented by means of connectionist systems, to define the protein folding process through time and using the HP lattice model. This new proposal is

different from most on the previous work focused on the direct prediction of the final protein conformations. The connectionist 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. The neural model acts like a CA-like system, over all the simple elements or amino acids and through different temporal steps until a final folded conformation is reached. Thus, the process is treated as an emergent and dynamic process.

Finally, as the iterative modeling by the 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 also needs fewer evaluations to obtain the best final conformations.

5. ACKNOWLEDGMENTS

This work was funded by the Ministry of Science and Innovation of Spain (project TIN2011-27294).

6. REFERENCES

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

[2] V. Cutello, G. Nicosia, M. Pavone, and J. Timmis. Immune algorithm for protein structure prediction on lattice models. *IEEE T. Evol. Comp.*, 11(1):101–117, 2007.

[3] G. Danks, S. Stepney, and L. Caves. Protein folding with stochastic L-systems. In *Artificial Life XI: Proc.. of 11th Int. Conf. on the Simulation and Synthesis of Living Systems (MIT Press)*, pages 150–157, 2008.

[4] K. Dill. Dominant forces in protein folding. *Biochemestry*, 29:7133–7155, 1990.

[5] K. Dill et al. Principles of protein folding: a perspective from simple exact models. *Protein. Science*, 4(3):561–602, 1995.

[6] S. Duarte, D. Becerra, F. Nino, and Y. Pinzón. A novel ab-initio genetic-based approach for protein folding prediction. In *Proc. GECCO '07*, pages 393–400, 2007.

[7] V. Feoktistov. *Differential Evolution: In Search of Solutions*. Springer, NY, 2006.

[8] S. Fidanova. 3D HP protein folding problem using ant algorithm. *BioPS'06*, pages 19–26, 2006.

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

[10] W. Hart and S. Istrail. Robust proofs of NP-hardness for protein folding: General lattices and energy potentials. *Journal of Computational Biology*, 4(1):1–22, 1997.

[11] J. Hirst. The evolutionary landscape of functional model proteins. *Protein Engineering*, 12(9):721–726, 1999.

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

[13] C. Johnson and A. Katikireddy. A genetic algorithm with backtracking for protein structure prediction. In *Proc. GECCO'06*, pages 299–300, 2006.

[14] N. Krasnogor, W. Hart, J. Smith, and D. Pelta. Protein structure prediction with evolutionary algorithms. In *Proc. GECCO'99*, pages 1596–1601, 1999.

[15] N. Krasnogor, G. Terrazas, D. Pelta, and 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, 2002.

[16] C. Langton. Life at the edge of chaos. In *Artificial Life II, C.G Langton, C. Taylor, J.D. Farmer, S. Rasmussen (Eds.), Addison-Wesley*, pages 41–49, 1992.

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

[18] H. Lopes and R. Bitello. Differential evolution approach for protein folding using a lattice model. *Journal of Computer Science and Technology*, 22(6):904–908, 2007.

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

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

[21] K. Price, R. Storn, and J. Lampinen. *Differential Evolution. A Practical Approach to Global Optimization*. Springer - Natural Comp. Series, 2005.

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

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

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

[25] R. System. `http://www.rosettacommons.org`.

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

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

[28] R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231(1):75–81, 1993.

[29] X. Zhao. Advances on protein folding simulations based on the lattice HP models with natural computing. *Applied Soft Computing*, 8:1029–1040, 2008.