Journal of Bioinformatics and Computational Biology
Vol. 15, No. 3 (2017) 1750009 (22 pages)
© World Scientific Publishing Europe Ltd.
DOI: 10.1142/S0219720017500093

NEAT-FLEX: Predicting the conformational flexibility of amino acids using neuroevolution of augmenting topologies

Bruno Grisci* and Márcio Dorn†

Institute of Informatics, Federal University of Rio Grande do Sul
Porto Alegre, 91501-970, RS, Brazil
**bigrisci@inf.ufrgs.br*
†mdorn@inf.ufrgs.br

Received 11 July 2016
Revised 1 March 2017
Accepted 8 March 2017
Published 12 April 2017

The development of computational methods to accurately model three-dimensional protein structures from sequences of amino acid residues is becoming increasingly important to the structural biology field. This paper addresses the challenge of predicting the tertiary structure of a given amino acid sequence, which has been reported to belong to the NP-Complete class of problems. We present a new method, namely NEAT–FLEX, based on NeuroEvolution of Augmenting Topologies (NEAT) to extract structural features from (ABS) proteins that are determined experimentally. The proposed method manipulates structural information from the *Protein Data Bank* (PDB) and predicts the conformational flexibility (FLEX) of residues of a target amino acid sequence. This information may be used in three-dimensional structure prediction approaches as a way to reduce the conformational search space. The proposed method was tested with 24 different amino acid sequences. Evolving neural networks were compared against a traditional error back-propagation algorithm; results show that the proposed method is a powerful way to extract and represent structural information from protein molecules that are determined experimentally.

Keywords: Neuroevolution; amino acid flexibility; three-dimensional protein structures; structural bioinformatics; machine learning in structural bioinformatics.

1. Introduction

Structural Bioinformatics deals with different problems where the rules that govern the biochemical processes and relations are partially known which makes it harder to design efficient computational strategies for these situations.[1] Among them, one of the main challenging problems is the three-dimensional (3D) protein structure prediction (PSP) problem. The PSP problem has over the past 40 years challenged biochemists, biologists, computer scientists, and mathematicians, comprising the

† Corresponding author.

domain of high-dimensional continuous optimization problems.[2] Nowadays, a wide range of optimization algorithms and metaheuristics are being applied in an attempt to find approximated solutions to predict the 3D structure of proteins only from its linear sequence of amino acids.[3,4] Metaheuristics developed for the PSP problem commonly incorporate knowledge from experimental protein data (NMR or X-Ray crystallography) to reduce the conformational search space or to better guide the search procedure[5] trying to overcome the problem of incomplete knowledge about the biological rules that makes the protein fold into a specific and functional shape. Despite the number and the variety of methods developed for structure prediction, no robust and general method exist to predict the correct 3D structure of proteins. In part, this is due to the absence of better methods and computational strategies to obtain and represent structural information from proteins that are determined experimentally.

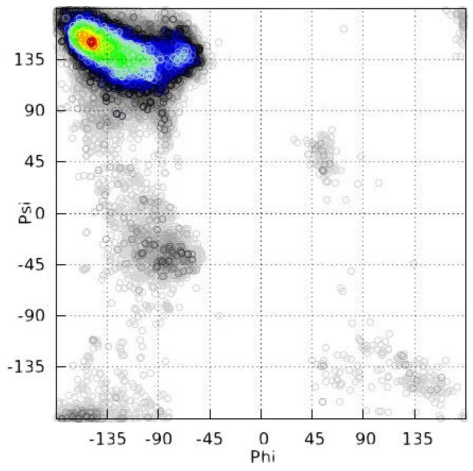
Proteins are not static and rigid macromolecules, the backbone and in particular the side-chain of a protein are always moving. This flexibility is essential so that the protein can perform its function. In nature, proteins fluctuate between alternative conformations[6] (folding and unfolding), and this motion play a critical role in many biological processes,[7] especially in ligand-protein binding[8] and fold recognition. The conformational changes of a protein depend on many environmental factors like interactions with other molecules, temperature, and pH. In PSP methods, one of the greatest challenges in using structural information is related to flexibility, which is not described in experimental data. Experimental protein structural data are commonly used as knowledge in prediction methods: *fragment libraries*,[9] *rotamer libraries*,[10] *conformational preferences of amino acids*,[11] and *coil libraries*.[12] These databases and libraries represent the experimental preferences of a single or segments of amino acids giving important details about the conformation to be assumed by a protein molecule. Nevertheless, none of them characterize, incorporate or represent the protein backbone flexibility.

Knowledge-based PSP methods commonly use different machine learning techniques to acquire and represent structural information of experimental protein structures. Artificial Neural Networks (ANNs),[13] Support Vector Machines (SVM)[14] are examples of the most used techniques in this field. Nevertheless, due to the complexity, size, and noise present in experimental data, determining, *a priori*, a general architecture, for example, for an ANN is an arduous task. Another challenge is associated with which type of structural information should be considered and how to characterize and represent the flexibility of protein as a way to better describe the natural protein motion. In this paper, we present NEAT-FLEX (Neuroevolution of Augmenting Topologies to predict FLEXibility), a method based on neuroevolution[15] to characterize and predict the backbone flexibility of target amino acid sequences. The proposed method takes advantage of experimental protein structures stored in the *Protein Data Bank* (PDB) to determine the conformational flexibility of amino acids in a protein backbone. This information may be used in the development of more robust and efficient computational strategies for the PSP problem. This paper is organized as follows. Section 2 presents fundamental concepts of proteins, conformational preferences of amino acids and flexibility, neuroevolution, and PSP methods. Section 3 describes the proposed method. Section 4 shows the computational experiments and discussion of results. Finally, the last section concludes the paper and points out future works.

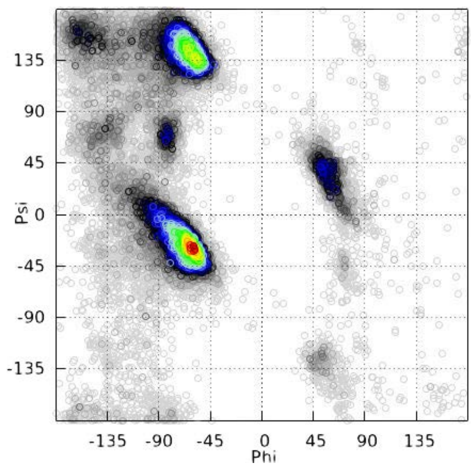
2. Material and Methods

X-ray crystallography is considered as the gold-standard method for solving the 3D structure of protein at an atomic level. From the electron density, the mean positions of the atoms in the crystal as well as their chemical bonds, their disorder, and various other information can be determined. These experimental data represent the structure of protein as a rigid model and are frequently used in knowledge-based prediction methods to restrict proteins motion and predict its native and functional state.[11] However, more and more emerging evidence show that protein structures are more complex with their internal dynamics (especially in disordered proteins), being a key determinant of their function. Crystallography data can present certain errors in measuring the intensities of atomic positions. To deal with the complexity, size, and noise present in experimental data, we combine concepts of neuroevolution with an unsupervised hierarchical clustering approach to extract conformational preferences of amino acids in protein structures that are determined experimentally. The main idea behind our method is to develop a computational strategy to learn the motion of amino acids in different proteins to further predict the structural flexibility of amino acid in sequences with unknown 3D structure. The rules that govern the biochemical processes are partially known what makes the development of a general approach for learning structural features a hard task. In this sense, we implement a NEAT[15] approach that uses a Genetic Algorithm to evolve the topology and link weights of an ANNs.

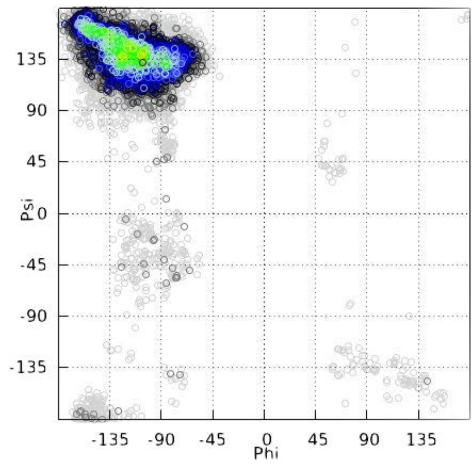
Conformational Preferences of Amino Acids in Proteins: Proteins are polymers formed by a sequence of 20 different possible amino acids that under physiological conditions fold into a precise shape known as its native state.[16] Each amino acid has an alpha carbon (CA) with bonds to amino (NH2) and carboxyl (COOH) groups and a variable side-chain (R) that determines the particular physicochemical properties of each residue. The interaction between amino acids in a protein causes the polypeptide chain to fold, usually in a proper configuration, as α-**helix**, β-**sheet**, **coil** or **turn**. These local folding patterns represent the secondary structure of a protein. The topology or fold is given by the succession of secondary structures connected in a 3D space. The specific characteristics of the peptide bond have significant implications for the 3D fold that can be adopted by polypeptides. The peptide bond (C-N) has a double bond, and does not allow rotation of the molecule around this bond. The rotation is only permitted around the bonds N–Cα and C_α–C. These bonds are known as phi (ϕ) and psi (ψ) dihedral angles and are free to rotate. The sequence of ϕ, ψ and ω angles of the residues in a protein defines the backbone conformation or its fold.[17] In this context, a protein structure can be represented by a set of dihedral angles, describing their internal rotations. The analysis of experimental protein structures (X-ray data) reveals that amino acid residues can assume many conformations (phi and psi angles, *Ramachandran plot*) in proteins.[11] Each amino acid has a set of physiochemical properties which contributes to its intrinsic conformational preference.[18] Amino acids in a secondary structure



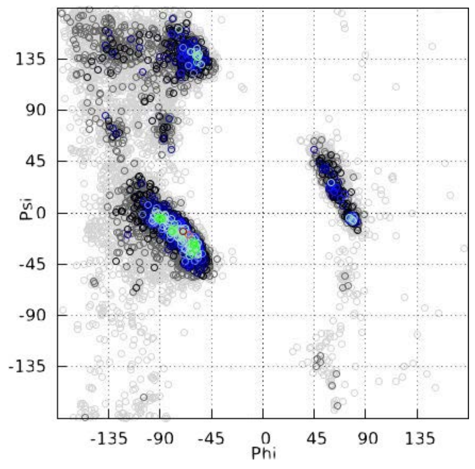
(a) Ala-β-sheet



(b) Ala-Turn



(c)Trp-β-sheet

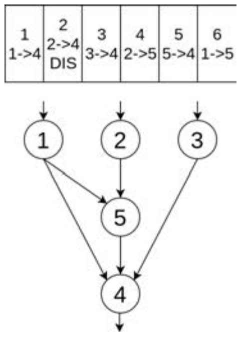


(d) Trp-Turn

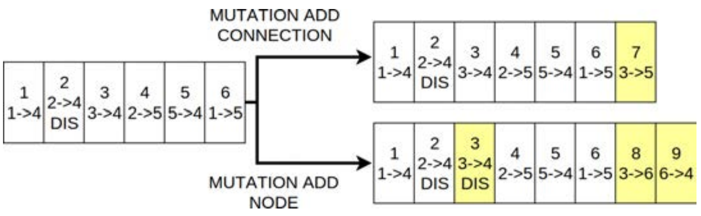
Fig. 1. Conformational preferences of amino acids in proteins depend on it secondary structure. The dark red color marks the most densely occupied regions. Ramachandran plots were generated by `NIAS-server` (*www.sbcb.inf.ufrgs.br/nias*).

usually adopt a particular set of backbone torsion angles (phi and psi).[11] For example, Fig. 1 shows the conformational preferences (*Ramachandran plot*) of Alanine (`Ala`) and Tryptophane (`Trp`) in β-`Sheet` and `Turn` secondary structures. In this paper, the conformational preferences (phi and psi) and secondary structure propensities of amino acids, obtained from experimentally determined proteins are taken into consideration to determine the conformational flexibility of amino acids in a target sequence. For a complete overview about conformational preferences of amino acids in protein, see Borguesan *et al.*[11,19] and Hovmoller *et al.*[17]

NeuroEvolution: ANNs are a group of machine learning models used to estimate or approximate functions.[20] In the recent years, they have shown to be good for different classification and biological problems in special with the PSP problem.[21] There are many different approaches to create and train ANNs, one of them being Neuroevolution, a family of methods that uses evolutionary algorithms. NEAT[15] uses a Genetic Algorithm (GA) to evolve the topology and link weights of ANNs (Fig. 2(a)). This ANN is adequate for problems for which a satisfactory network structure is unknown, especially in structural bioinformatics problems where the rules that govern the biological process are partially known. It starts with a random population of ANNs, all of them sharing the same basic topology, i.e. input neurons, output neurons and a link between the input neurons and output neurons with random weights. This minimalist start condition is important because it assures that useless complexity won't be added to the ANNs, since only improvements in the topology of the network that generated better results will be kept. If the method was initialized with random topologies, useless neurons or links might be present from the start, and it wouldn't



(a) NEAT genome



(b) Mutations in NEAT

Fig. 2. Schematic representation of NEAT. (a) Representation of a genome for an individual in a NEAT population. The first number in each gene is the historical marker used to identify each new structural transformation. The second information is the link between two nodes. The third is a disable bit that when active informs that the corresponding connection is ignored. (b) Representation of the two kinds of mutations present in NEAT. The first example adds a link with random weight between the nodes 3 and 5 and receives a new historical marker. The second example adds a node 6 between the nodes 3 and 4, so two new connections are created between 3 and 6 and 6 and 4, while the connection between 3 and 4 is disabled. Adapted from Stanley *et al.* (2002).

be possible to remove them, which may have a negative effect on the evolution. This also results in smaller, simpler final ANNs.

From this initial population, new populations are generated in an iterative fashion with traditional GA operators, namely the *crossover operation* (Fig. 3), that combines two individuals from the current population in order to generate a mixed new individual, and *mutation* (Fig. 2(b)), that can change the values of the weights of the links of an ANN, or add new hidden neurons or a new link between current neurons. Mutations in NEAT never remove an existing neuron or link for architectural reasons because it could create inconsistencies along the training. Alternatively, it presents an enable bit that can be set to ignore a link between neurons. A problem that arises from this method is that combining two ANNs with the crossover operation may result in a defective ANN since their topologies may not be compatible with a direct exchange of neurons and links.[15] That is why NEAT uses a historical marking, a numerical value that is assigned to each new piece of structure that appears through the structural mutations. The value of the historical marking is the order in which the structural transformation first appeared along the evolutionary process and is transferred without changes during the crossover. It allows the NEAT method to match perfectly the same pieces of the topology of two different ANNs, resulting in a functional ANN that respects the organization of its predecessors. Finally, NEAT uses speciation, also known as a niche, so individuals compete within similar groups of ANNs instead of the totality of the population.[15] This is useful because adding new structure to an ANN usually is disadvantageous without further adjustments, so the speciation gives time to useful topological innovations to evolve, not just discarding them when they first show up. The historical markings are used to determine which niche an individual belongs. The main advantage observed from this approach is that there is no need to project the network structure since it will arise by itself.

Hierarchical Clustering: The conformational preference of amino acid residues in proteins are described by data points in a two-dimensional (2D) angle space (Fig. 1). To analyze the conformational preferences of a residue regarding its secondary

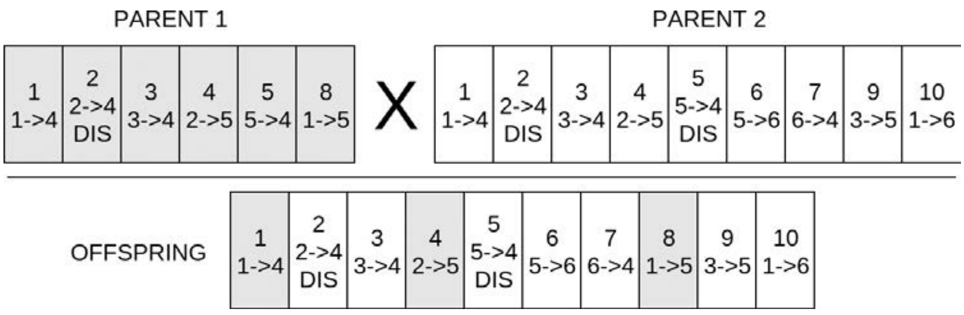


Fig. 3. Example of the *crossover* operation between two individuals. Their genes are aligned by historical marker in order to avoid structural inconsistencies. Image adapted from Stanley *et al.* (2002).

structure and neighbor amino acids, we partitioned the two-dimensional angle space by clustering the experimental data obtained from the PDB. Many significant clustering algorithms exist, each one having different characteristics caused by the use of various techniques, assumptions, and heuristics. Especially in protein experimental data, due to the problem particularities such as the delimitation of specific regions in the Ramachandran plot for amino acids in a regular (helices, sheets) or irregular (coil, turns) secondary structure state, a problem-oriented clustering strategy should be built to achieve better results. Many different clustering algorithms are available, some of them require the knowledge about the number of existing clusters. This class of clustering algorithms is not ideal for problems with unknown data distribution, like amino acids and its conformational preferences, since it would be impossible to predict the number and size of clusters that best fit the data. Some methods, like hierarchical clustering[22] and pvclust,[a] avoid this by having a distance threshold used to divide the clusters. In our method, we propose a hierarchical-based clustering method considering the problem particularities. We chose a distance threshold of 60° since this usually comprehended the distance between different structure regions[17] and generated the best general results. The metric distance used was the Euclidean distance, modified to take into account the cyclical property of amino acid in the Ramachandran plot (periodic boundary conditions). In hierarchical clustering, the iterative process starts considering each point of the dataset as being a group (line 2 in Algorithm 1) and then proceeds combining the nearer clusters (line 5 in Algorithm 1) until there is only one group containing every point. Using a distance threshold, it is possible to set a stop condition for the method, and it will stop when the distance between the clusters is larger than the threshold (line 3 in Algorithm 1), returning the current groups. Details about the clustering method developed will be

Algorithm 1. Pseudo-code for hierarchical clustering

```
Require: Matrix $D$ of pairwise distances between points, points $P$, threshold distance $t$
 1: Build graph $G$ assigning one vertex to each cluster;
 2: Form $\|P\|$ clusters with one element each;
 3: while there is more than one cluster and distance between clusters $> t$ do
 4:    Get the two closest clusters $C_A$ and $C_B$;
 5:    Merge $C_A$ and $C_B$ into new cluster C with $\|C_A\| + \|C_B\|$ elements;
 6:    Calculate the distance between C and all other clusters;
 7:    if the clusters are close then
 8:       Add new vertex C to G and connect it to vertices $C_A$ and $C_B$;
 9:       Remove the rows and columns of D corresponding to $C_A$ and $C_B$;
10:       Add a new row and column to D corresponding to cluster C;
11:    end if
12: end while
13: return G;
```

[a] http://www.sigmath.es.osaka-u.ac.jp/shimo-lab/prog/pvclust

explained in the next section. Hierarchical clustering is then able to classify our data points into an originally unknown number of clusters as desired. In the clustering step, the densely populated areas in the conformational space of an amino acid residue are identified and used with a supervised learning approach based on NEAT to predict the conformational flexibility of amino acids.

Related Work: Neural networks have been used over the last few years in the field of PSP. With good results, they were used, for example, to predict the dihedral angle probability distributions for protein coil residues[23] and secondary protein structures.[21] A previous approach (namely MOIRAE), to extract protein structural features from protein structure that was determined experimentally was proposed by Dorn and collaborators.[13] This method combines a clustering algorithm (*k-means*) with a *Multi-Layer Perceptron* (MLP) to retrieve structural information from the PDB and to predict the conformational preference of amino acids in a protein backbone. The primary goal of this strategy was to identify the degrees of flexibility that a segment of amino acids can assume. This information can help knowledge-based prediction methods to predict more accurate 3D protein structures. While MOIRAE uses *k-means* for clustering and MLP for training a neural network, our new method uses hierarchical clustering and NEAT[15] to autonomously train different neural networks better suited for the various data retrieved from the database.[24] Another fundamental difference between MOIRAE and our approach is that the first focuses on finding a single range of angles that comprehend the experimental values, while the second creates different intervals with probabilities. The capacity of NEAT finding optimal network topologies has already been shown to return better results than conventional MLP for the feature selection problem.[25]

3. The Proposed Method: NEAT-FLEX

The proposed method (NEAT-FLEX) uses structural information from the PDB to predict the flexibility of residues (ϕ and ψ torsion angles) in a target sequence. The goal is to find intervals inside the range -180° and 180° for each amino acid to restrict the conformational search space in 3D PSP methods. These intervals represent the conformational flexibility of each residue from a target sequence. This information may be used by different meta-heuristics such as *Genetic Algorithms* [11] or *Simulated Annealing*[26] to create a more efficient, less time-consuming search. Figure 4 summarizes the proposed method. It can be seen as a sequence of four steps: (1) Database search, (2) Data clustering, (3) Neural Network training, and (4) Intervals creation. The method input is a target amino acid sequence and its secondary structure, which can be obtained with accuracy from existing methods such as STRIDE.[27] The main idea behind the proposed method is to predict the range of expected values for backbone ϕ and ψ torsion angles by using information about the conformational preferences of the same amino acid with its neighborhood and secondary structure present in experimental proteins.

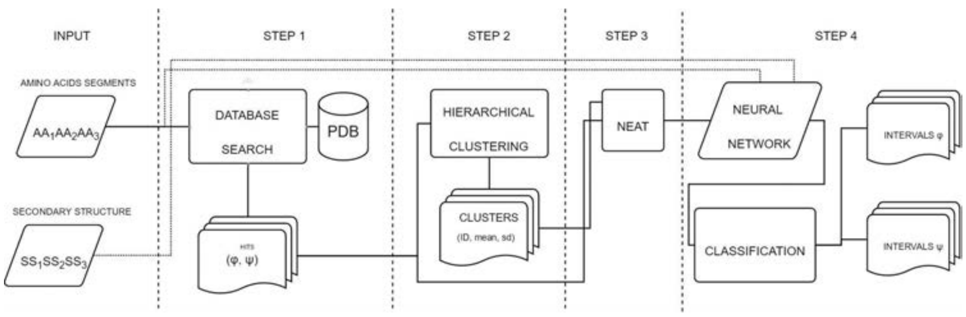


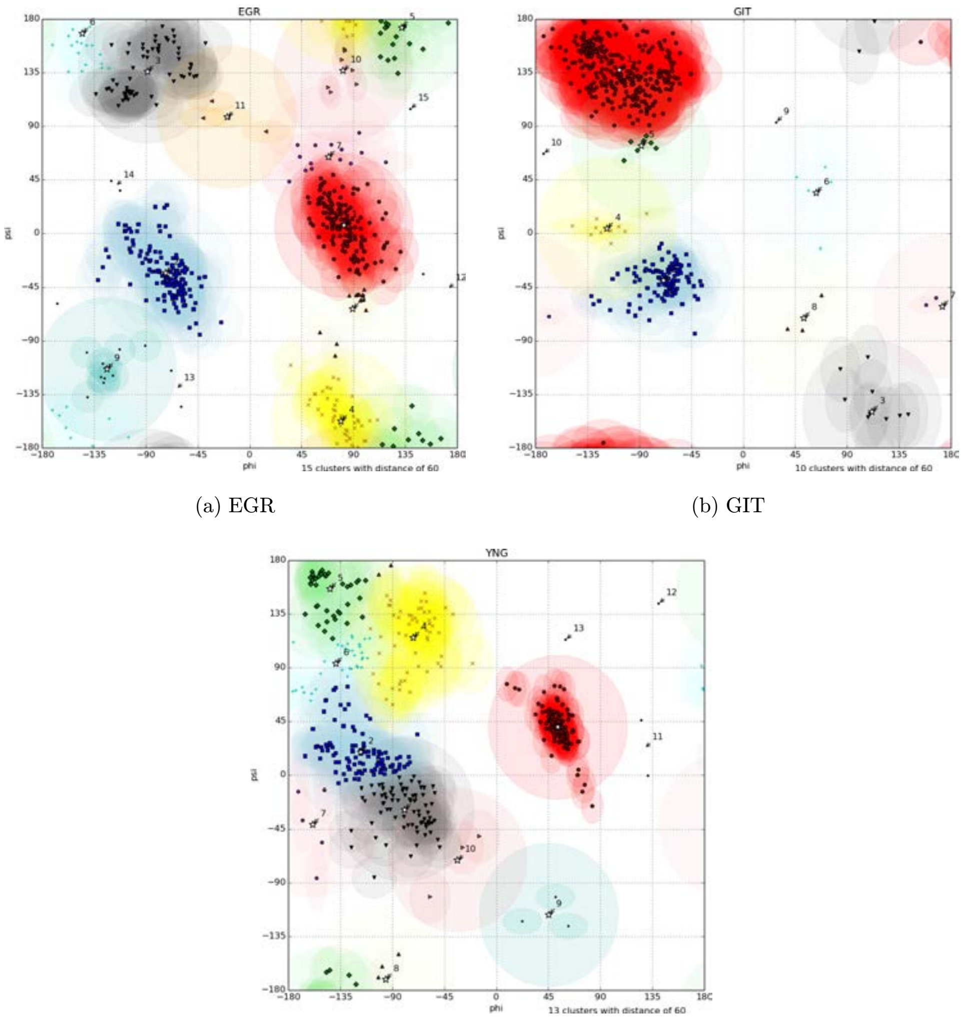
Fig. 4. Schematic representation of the proposed method when a segment of 3 amino acids is considered. In step 1, the PDB is searched for proteins that contain the segment in its chain, and the "hits" are saved. The secondary structure and the pair of ϕ and ψ torsion angles from the central amino acid of the segment are computed. In step 2, pairs of torsion angles belonging to an amino acid segment are clustered with a hierarchical clustering strategy, so each hit belongs to a cluster. In step 3, all information related to an amino acid segment is combined (amino acid segment + secondary structure assignment + cluster id) to form a training set. An evolved neural network using NEAT is trained to be able to predict a cluster from the segment of amino acids and secondary structure. In step 4, the final neural network receives as input the original amino acid and secondary structure segments and output the probabilities of it belonging to each existing cluster. The ϕ and ψ intervals for the segment are then created by attributing the probability of each cluster and adding and subtracting one and a half standard deviations to the ϕ and ψ means from each cluster.

Method Overview: The proposed method starts by dividing the target amino acid sequence into consecutive segments of size 3 and looking for *hits* in the PDB. A *hit* occurs when the segment is found in the amino acid sequence of a different protein stored in the PDB. With the list of proteins that contain the corresponding segment, the PDB files were downloaded, and STRIDE[27] assigns the secondary structures of them, and the phi and psi torsion angles from the segment are calculated. Then, each segment will have a list of pairs phi and psi from the different proteins in which it appears. These pairs are clustered. With each point assigned to a cluster, we create a dataset composed of patterns represented by the three amino acids of a segment, its secondary structure and the cluster it belongs. This is used as input for our neural network training that uses NEAT.[15] The neural networks are trained to learn how to classify the amino acids, and secondary structure of one of the segments in the clusters found previously. Once the training is complete, the neural network can classify new inputs with generalization. In the final step, the original target amino acid sequence and its secondary structure are submitted to the corresponding neural networks that output the probability of them belonging to the clusters for each one of them. We then create the intervals centering them in the mean values of the clusters and adding and subtracting one and a half standard deviation from it. The final method output is a set of intervals for the angles phi and psi of the target amino acids.

Structural Database Search: To find the needed information and limit the range of the final torsion angle intervals, the target sequence is divided into consecutive segments of size 3 and search the PDB for their occurrences. So, for a sequence of size n, we will have $n - 2$ segments of size 3. Each segment will later provide information for its middle residue, so we are considering not only the amino acid itself but also its neighbors as a form to learn the expected angle values from the database. The first (N-term) and last (C-term) amino acids of the target sequence are not represented by the edges of an amino acid sequence which are often unstable. For each segment, the PDB was searched using the PROTEIN BLAST tool,[28] looking for the occurrences of the segment in proteins that are determined experimentally. It returns a list of every protein that has a *hit* (meaning it contains the segment), the PDB file is downloaded and analyzed with STRIDE,[27] obtaining its corresponding secondary structure and experimental ϕ and ψ torsion angles for every amino acid. If a *hit* occurs at the start or the end of the protein, it is ignored because these regions are unstable. All segments of the original target sequence are matched to lists formed of equal segments from the proteins of the PDB and this information is saved.

Data Clustering: For each segment of 3 amino acids from the target amino acid sequence, there is a list of all the *hits* from proteins at the PDB, along with its corresponding secondary structures and the experimental phi and psi values for the central amino acid of the segments. The next step is to organize this data by the angle values in the form of clusters, to be able to generalize this information later on. Pairs phi and psi are treated as 2D points with a range of values from -180° to 180° in both axes, with phi being the X-axis and psi the Y-axis. Due to the cyclical nature of the angles, we need to consider the extremities values as being the same, i.e. the values -180° and 180° are two representations of the same angle value. From now on, we should consider it for every calculation, including the distance between points. Since the database search returns an unknown number of points distributed in a new fashion for each residue, one can't predict the number of clusters that best fit the data. For this reason, hierarchical clustering was chosen. For our method, the measured distance threshold was 60° since this usually comprehend the distance between different structure regions[17] and generated the best general results. The metric distance used was the *Euclidean* distance, modified to take into account the cyclical property already discussed. Hierarchical clustering is then able to classify our data points into an originally unknown number of clusters as desired. The clustering algorithm runs for all segments of the target amino acid sequence, and for all points, an integer value that corresponds to a cluster is assigned (Fig. 5). Clusters with few points are considered outliers and removed from the next steps.

Neural Networks Training: Once the segments from the database are classified in clusters, it is time to train neural networks to learn how to classify new inputs from which only the three amino acids and its secondary structures are known. A different neural network is needed for each of these segments since they all have a different set of points and clusters to be analyzed. A traditional method to perform this task would be to use ANN with a learning method such as *back-propagation*. But this approach trains ANNs to learn only the weights of the links between the neurons,



(a) EGR (b) GIT

(c) YNG

Fig. 5. Examples of hierarchical clustering for amino acids segments of size 3 (`EGR`, `GIT` and `YNG`) from the sequence `RGKWTYNGITYEGR` of the protein with `PDB ID1K43`. Each point is a pair ϕ, ψ for the central amino acid obtained from a *hit* returned from the `PDB` search in step 1. Each cluster is represented by a different color, the lighter colored shadow around the points representing the cluster standard deviation.

while the overall topology of the network, i.e. the number of neurons, links and which neuron is connected to another, remains fixed. But how to choose only one topology for the issue of classification of new segments of amino acids and secondary structures if each one of them has a unique, particular training set? To address this problem, we utilize a method of evolving `ANN`s called `NEAT`. A target amino acid sequence of length n originates $n-2$ segments of size 3, each one of them corresponding to the middle amino acid and with their set of clusters from the last step. So what we need is to evolve $n-2$ ANNs that are able to classify these segments in clusters using the information extracted from the database. These neural networks have seven neurons in the input layer that receive the three amino acids, the three secondary structures and also an extra input called bias that is always set to 1.0. Figure 6 illustrates the creation of the training set used for the training of the neural networks. To avoid over-fitting, only 80% of the data from the last step is used to build the training set, the remaining 20% being used as a testing set.

From the training set, there are sequences of amino acids and their secondary structures associated with the cluster to which they have been classified in the clustering step. What the neural networks need to learn is to output the correct `CLUSTER ID` given the input (target segment of amino acids and its secondary structure). The clusters are represented with one-hot encoding. If the information need to be classified between c different clusters, each cluster is encoded as an array of c bits set as 0 except the bit with an index corresponding to the `CLUSTER ID`. To be able to compare the cluster array from the training set with the array of real numbers received as output from the neural network, these values are given to a softmax function (1) that scales them between 0.0 and 1.0. These new values can be seen as the probability of the input of the neural network belonging to one cluster.

$$\mathbf{SOFTMAX}(X) = \mathrm{e}^{X_i} \Big/ \sum_{i=1}^{n} \mathrm{e}^{X_i} \tag{1}$$

where X is a vector of length n. `ANN`s work only with numerical inputs and outputs while our method has symbolical inputs and outputs (amino acids, secondary structures, and clusters). The clusters are defined as integer values that can be

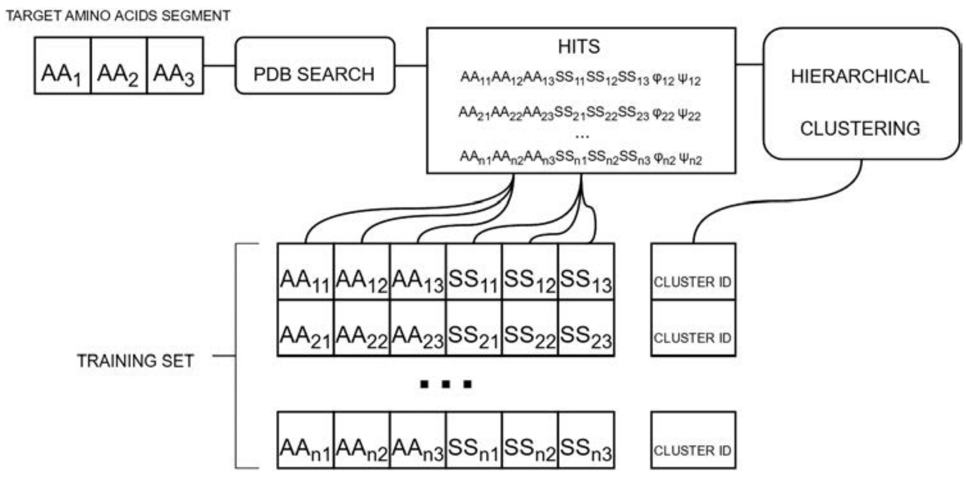

TARGET AMINO ACIDS SEGMENT
AA1 AA2 AA3
PDB SEARCH
HITS
AA11AA12AA13SS11SS12SS13 φ12 ψ12
AA21AA22AA23SS21SS22SS23 φ22 ψ22
...
AAn1AAn2AAn3SSn1SSn2SSn3 φn2 ψn2
HIERARCHICAL CLUSTERING
TRAINING SET
AA11 AA12 AA13 SS11 SS12 SS13
AA21 AA22 AA23 SS21 SS22 SS23
AAn1 AAn2 AAn3 SSn1 SSn2 SSn3
CLUSTER ID

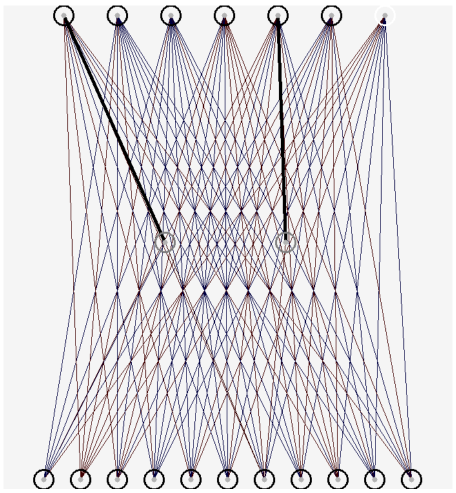
Fig. 6. Schematic representation of the creation of the training set. Each target segment of amino acid have a training set for the evaluation of the neural networks.

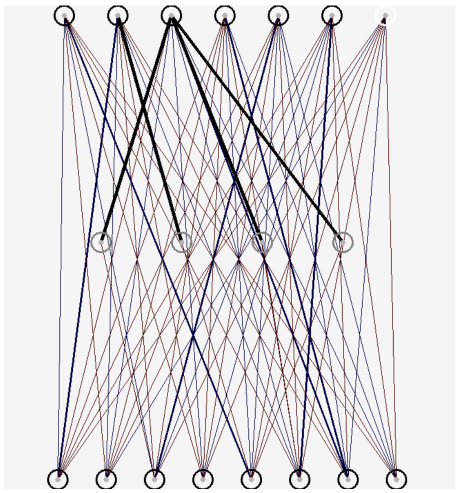
represented as one-hot encoded, but the way one converts the amino acids and secondary structures to numbers may affect the outcome of the learning phase of the ANN. Since if numerical values are arbitrarily assigned to each possible input, the algorithm may infer correlations and sorting that are not present in nature. Trying to avoid this problem, we looked for real world properties of amino acids and their secondary structures to make sense of a numerical conversion for the ANNs. For amino acids the hydrophobic parameters *pi* of amino acid side-chains from the partitioning of *N-acetyl-amino-acid amides*[29] were used. This procedure keeps close the amino acids that share the same natural groups, and for secondary structures, we assigned values that rank them by their structural similarity and complexity. Hydrophobic residues appearing periodically have already been used for predicting protein secondary structure since hydrophobic affects the stability of secondary structure.[30] Since NEAT uses genetic algorithms to evolve the ANNs, it needs a fitness function that evaluates each individual and ranks them by this value. For our method, this fitness is the loss of the cross entropy (Eq. (2)) of the output of the neural network with the *softmax function* applied and the original classification encoding in the training set. In the end, the individual with the lowest loss, i.e. the ANN that classified the largest quantity of inputs correctly, is selected. For each segment of the target amino acid sequence, a complete NEAT cycle is performed, using their training set, resulting in $n - 2$ different and independent ANNs. This procedure solves the original problem of guessing the best topology for each of the segments since with NEAT, it is possible to find different structures and link weights without human intervention. Figure 7 show examples of three fully evolved neural networks for three different segments of amino acids. Their final topologies are indeed very different, corroborating the idea that training ANNs with the same topology would not generate the best results.

$$\textbf{CROSS} - \textbf{ENTROPY}(A, B) = \sum_{i=1}^{n} (B_i \times \log_e A_i) \tag{2}$$

where A and B are vectors of same length n, A being the output of the neural network applied to the softmax function and B the one-hot encoding from the training set.

Intervals creation and conformational flexibility prediction: The last step of the proposed method produces torsion angle intervals for phi and psi of each amino acid from the target sequence. These intervals represent the conformational flexibility of the amino acid residue. For this, it is only needed to provide as input to the trained neural networks the amino acids and secondary structures from each target segments of size 3. Each segment has a matching neural network, and they will output a vector of probabilities of this segment belonging to each cluster. To create the intervals, for the angle phi of one particular amino acid, the mean values of the phi coordinate from the clusters are extracted. These values are the centers of the intervals. The boundaries are just the mean added and subtracted by one and a half standard deviations, respecting the cyclicity of the angles. Finally, the probability of


(a) EGR


(b) GIT

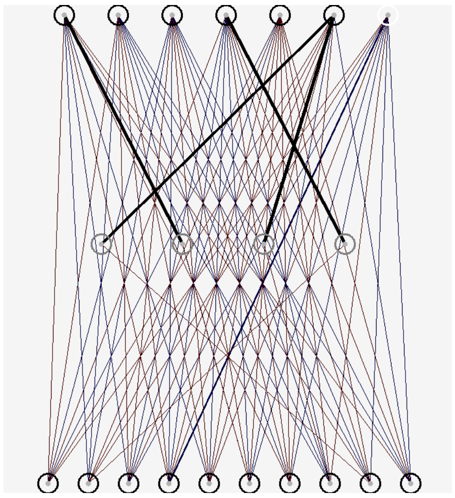

(c) YNG

Fig. 7. Examples of evolved neural networks with NEAT for the amino acids segments of size 3 from Fig. 5. The top nodes are the inputs (amino acids and secondary structures), the white node is the bias, the bottom nodes are the output (clusters) and the gray nodes the hidden layers. The thickness of the lines is proportional to the absolute value of the weight of a link.

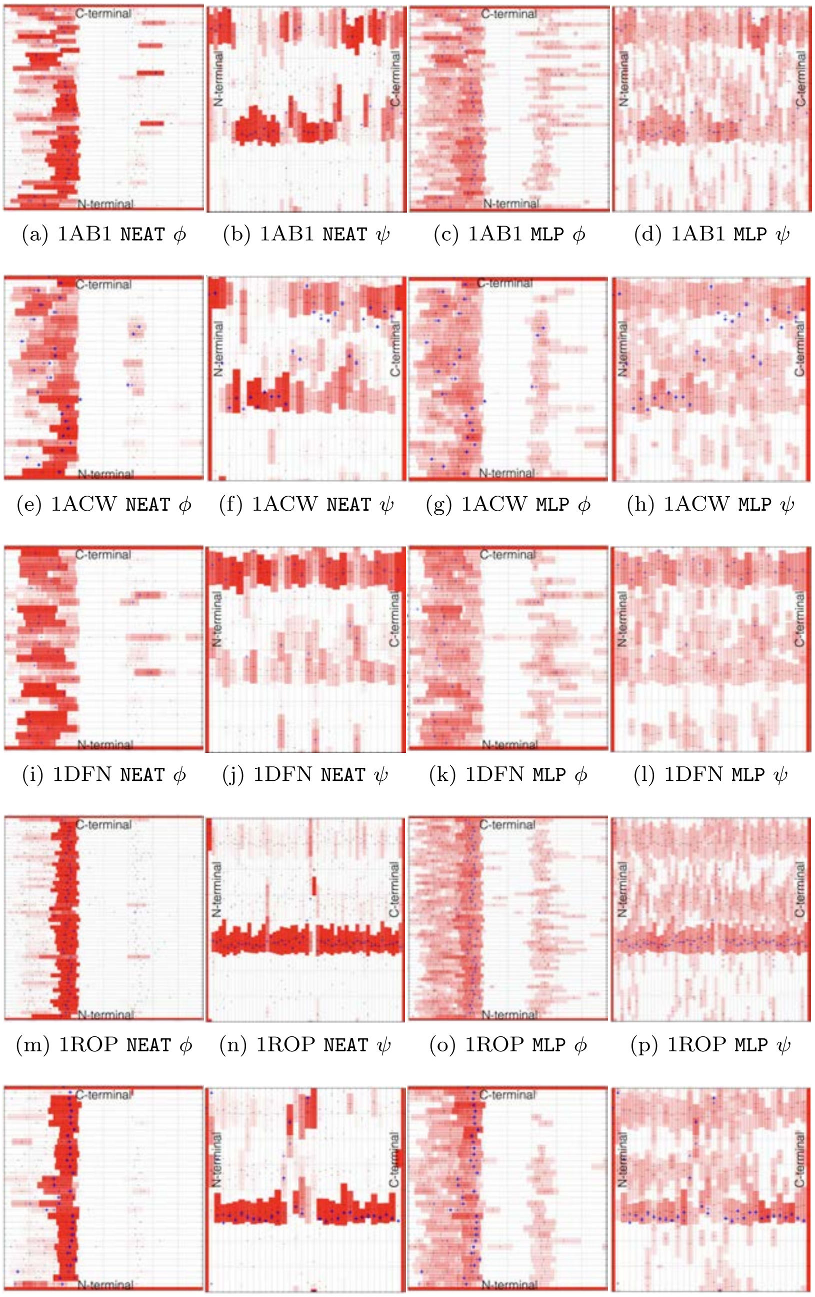
the angle being in one interval is defined by the probability of the cluster that originated the interval obtained from the neural network. The process to create the intervals for angle psi is the same, but uses the psi coordinate from the clusters. Repeating this for each one of the segments of the target amino acid sequence of length n will produce $n-2$ Phi sets of intervals and $n-2$ Psi sets of intervals, corresponding to all amino acids from the sequence except the first and last ones. In turn, this is not a problem since the amino acids positioned in the boundaries are unstable. For them, intervals with a range from -180° to 180° and probability of 1.0 is assigned.

4. Experiments and Results

The method (NEAT-FLEX) described was implemented and tested with 24 different amino acid sequences from proteins available in the PDB. These proteins were chosen so as to represent different sequence sizes, secondary structure content, and experimental methods. Target proteins are listed in Table 1 (columns 1 and 2). The neural networks were trained with both NEAT and MLP with error back-propagation algorithms for comparison, and for each method, the best neural network of 10 independent training runs with 100 iterations was chosen. The MLP networks trained have one hidden layer with five nodes and is fully connected. This topology is the same used in the neural networks of the MOIRAE method.[13] We used the Python packages

Table 1. Analysis of the neural networks. The values are the means with the standard deviation in parenthesis for each of the tested proteins. The success rate is the percentage of the test set correctly classified. The structural values presented (number of occult nodes, links and depth) are with regard to the NEAT networks since the MLP have a fixed topology with five occult nodes and one occult layer.

Pdb id	AA Size	N° occult nodes	N° links	Depth	Clas. NEAT	Clas. MLP
1AB1	46	1.84 (1.59)	46.29 (18.52)	1.86 (0.55)	0.75 (0.16)	0.77 (0.13)
1ACW	29	1.30 (1.18)	45.41 (11.47)	1.81 (0.67)	0.72 (0.16)	0.74 (0.15)
1DFN	30	1.64 (1.67)	44.93 (17.47)	2.21 (2.70)	0.78 (0.19)	0.79 (0.18)
1K43	14	2.75 (2.20)	56.17 (20.86)	2.17 (0.80)	0.73 (0.19)	0.72 (0.18)
1L2Y	20	1.56 (1.26)	46.89 (16.93)	4.33 (5.25)	0.78 (0.18)	0.80 (0.15)
1ROP	63	2.56 (2.04)	46.15 (10.37)	2.06 (0.60)	0.80 (0.12)	0.81 (0.11)
1ZDC	35	1.22 (1.43)	40.00 (14.79)	1.69 (0.68)	0.81 (0.13)	0.81 (0.12)
2PMR	87	1.96 (2.31)	45.37 (14.09)	2.63 (3.28)	0.80 (0.15)	0.82 (0.13)
1WQC	26	1.92 (1.63)	43.29 (12.35)	3.71 (4.67)	0.83 (0.10)	0.83 (0.11)
2MTW	20	2.17 (1.64)	51.28 (11.39)	2.78 (3.22)	0.73 (0.10)	0.73 (0.11)
3P7K	45	1.44 (1.28)	43.69 (12.50)	2.56 (3.17)	0.82 (0.11)	0.84 (0.12)
2P81	44	1.90 (2.00)	47.29 (14.49)	2.17 (2.25)	0.79 (0.13)	0.80 (0.12)
3V1A	48	2.22 (1.88)	45.30 (13.74)	2.80 (3.53)	0.79 (0.12)	0.81 (0.12)
1ENH	54	2.24 (1.70)	48.67 (13.19)	2.47 (2.79)	0.78 (0.12)	0.79 (0.12)
2F4K	35	1.81 (1.40)	48.55 (12.07)	2.74 (3.51)	0.76 (0.12)	0.79 (0.11)
2P6J	52	1.52 (1.55)	48.14 (11.56)	2.56 (3.46)	0.79 (0.09)	0.83 (0.09)
1AIL	73	2.40 (2.25)	48.76 (14.99)	3.43 (4.31)	0.76 (0.15)	0.77 (0.15)
2MR9	44	1.86 (1.96)	48.69 (16.77)	2.62 (3.09)	0.79 (0.13)	0.80 (0.14)
2JUC	59	2.58 (2.66)	46.68 (13.75)	2.70 (3.33)	0.76 (0.13)	0.77 (0.13)
1D5Q	27	2.52 (2.28)	52.76 (17.63)	2.68 (2.8)	0.68 (0.17)	0.70 (0.16)
1Q2K	31	1.90 (1.63)	46.31 (14.44)	1.90 (0.66)	0.74 (0.16)	0.77 (0.14)
2P5K	64	1.79 (1.61)	46.44 (12.11)	2.11 (1.88)	0.78 (0.14)	0.80 (0.13)
1CRN	46	1.57 (1.56)	44.89 (17.58)	2.36 (3.04)	0.77 (0.14)	0.79 (0.13)
1UTG	70	1.76 (1.64)	46.71 (13.08)	1.84 (0.50)	0.78 (0.11)	0.80 (0.10)



(a) 1AB1 `NEAT` ϕ (b) 1AB1 `NEAT` ψ (c) 1AB1 `MLP` ϕ (d) 1AB1 `MLP` ψ

(e) 1ACW `NEAT` ϕ (f) 1ACW `NEAT` ψ (g) 1ACW `MLP` ϕ (h) 1ACW `MLP` ψ

(i) 1DFN `NEAT` ϕ (j) 1DFN `NEAT` ψ (k) 1DFN `MLP` ϕ (l) 1DFN `MLP` ψ

(m) 1ROP `NEAT` ϕ (n) 1ROP `NEAT` ψ (o) 1ROP `MLP` ϕ (p) 1ROP `MLP` ψ

(q) 1ZDC `NEAT` ϕ (r) 1ZDC `NEAT` ψ (s) 1ZDC `MLP` ϕ (t) 1ZDC `MLP` ψ

Fig. 8. Visualization of some of the angle intervals ϕ and ψ created with the method using `NEAT` and `MLP` with error back-propagation. Each line (for ϕ) or column (for ψ) represents an amino acid. The columns for ϕ and lines for ψ represent the angle ranges from $-180°$ to $180°$. The intensity of the color is proportional to the interval probability. The crosses represent the angle values of the experimental structure.

SciPy[31] for the hierarchical clustering and `MultiNEAT`[32] for `NEAT`. The experiments were executed in a 64-bit Linux system with 16 GB RAM memory and a 2.80 GHz CPU with 8 cores. For the implementation of the `MLP` with error *back-propagation*, the library `PyBrain`[33] was used. All source codes and supplementary data are available at http://sbcb.inf.ufrgs.br/neatflex.

The comparison between `NEAT` and error *back-propagation* showed both methods have similar success rates for the classification, around 75% (Table 1). A perfect classification rate was not expected since the neural networks inputs are not perfectly homogeneous inside the clusters. `NEAT` managed to achieve similar classification rates with simpler and smaller networks and without the need of a known topology (Table 1). The comparison between the angle intervals provided by the two methods (Fig. 8) also indicates that `NEAT` was able to predict with more certainty the correct interval for each amino acid. Tables 2 and 3 provide numerical analysis of the created intervals. The enclosure is the percentage of experimental angle values contained in the interval with the largest probability or considering all intervals (in parenthesis) of each amino acid. In Table 3, columns 2–5, represent the mean size of the intervals.

Table 2. Analysis of the intervals illustrated in Fig. 8. Enclosures ϕ and ψ are the percentage of torsion angles values of the amino acid residue in the native state that are inside the predicted interval with higher probability and for "All", considering all the intervals. Higher percentages mean more reliable and accurate predictions. When available, it shows a comparison with the `MOIRAE` results.

Pdb id	Enclosure ϕ (All) %	Enclosure ψ (All) %	Enclosure ϕ `MOIRAE` %	Enclosure ψ `MOIRAE` %	`RMSD` `NEAT-FLEX` Å
1AB1	76.09 (93.48)	69.57 (93.48)	92.85	80.95	7.14
1ACW	55.17 (93.10)	51.72 (86.21)	68.00	36.00	13.27
1DFN	73.33 (90.00)	73.33 (90.00)	80.76	84.61	15.19
1K43	57.14 (71.43)	50.00 (78.57)	70.00	80.00	2.08
1L2Y	75.00 (90.00)	70.00 (90.00)	N/A	N/A	5.37
1ROP	96.43 (98.21)	92.86 (96.43)	98.07	94.00	19.81
1ZDC	85.29 (97.06)	82.35 (91.18)	N/A	N/A	3.62
2PMR	92.11 (98.68)	84.21 (93.42)	N/A	N/A	21.12
1WQC	65.38 (88.46)	50.00 (73.08)	54.54	50.00	6.58
2MTW	70.00 (90.00)	60.00 (85.00)	N/A	N/A	4.23
3P7K	95.56 (97.78)	93.33 (93.33)	N/A	N/A	3.27
2P81	81.82 (93.18)	68.18 (84.09)	N/A	N/A	5.12
3V1A	85.42 (97.92)	89.58 (95.83)	N/A	N/A	16.98
1ENH	90.74 (98.15)	92.59 (98.15)	98.00	96.00	10.76
2F4K	84.85 (87.88)	84.85 (87.88)	N/A	N/A	6.55
2P6J	76.92 (92.31)	71.15 (88.46)	N/A	N/A	8.61
1AIL	94.29 (98.57)	91.43 (95.71)	98.48	95.45	15.14
2MR9	72.73 (90.91)	68.18 (86.36)	N/A	N/A	4.74
2JUC	74.55 (85.45)	67.27 (85.45)	N/A	N/A	19.46
1D5Q	74.07 (96.30)	66.67 (85.19)	N/A	N/A	5.65
1Q2K	67.74 (87.10)	58.06 (77.42)	51.85	40.74	11.65
2P5K	90.48 (98.41)	82.54 (92.06)	N/A	N/A	6.76
1CRN	80.43 (97.83)	71.74 (93.48)	N/A	N/A	6.04
1UTG	84.29 (95.71)	85.71 (94.29)	N/A	N/A	13.96

Table 3. Analysis of the intervals illustrated in Fig. 8. Mean sizes ϕ and ψ are the mean values in degrees, with the standard deviation in parenthesis. The smaller the mean size, the better since it reduces the search space of the angles.

Pdb id	Mean size ϕ (sd)	Mean size ψ (sd)	Mean size MOIRAE ϕ	Mean size MOIRAE ψ
1AB1	51.69 (14.67)	50.25 (11.31)	24.60	29.63
1ACW	52.03 (17.20)	51.86 (12.80)	71.33	105.60
1DFN	64.08 (15.77)	54.13 (06.87)	16.16	17.13
1K43	52.13 (16.28)	51.49 (07.55)	02.40	6.51
1L2Y	43.11 (14.51)	46.94 (14.26)	N/A	N/A
1ROP	39.49 (10.09)	42.95 (09.48)	12.22	13.30
1ZDC	40.49 (12.12)	44.9 (12.72)	N/A	N/A
2PMR	47.11 (14.33)	46.25 (11.74)	N/A	N/A
1WQC	44.85 (17.11)	47.52 (10.97)	17.77	48.91
2MTW	46.12 (15.56)	49.50 (13.97)	N/A	N/A
3P7K	41.73 (07.96)	42.84 (07.51)	N/A	N/A
2P81	49.96 (17.15)	47.54 (09.75)	N/A	N/A
3V1A	43.27 (10.32)	46.12 (11.45)	N/A	N/A
1ENH	44.53 (13.41)	46.51 (09.43)	17.03	29.05
2F4K	45.47 (15.62)	50.44 (11.51)	N/A	N/A
2P6J	46.64 (16.03)	46.52 (11.00)	N/A	N/A
1AIL	46.78 (12.89)	49.90 (12.86)	12.97	15.6
2MR9	44.55 (15.31)	43.16 (09.29)	N/A	N/A
2JUC	49.76 (15.93)	51.17 (13.09)	N/A	N/A
1D5Q	52.65 (15.03)	54.62 (13.52)	N/A	N/A
1Q2K	55.10 (13.57)	54.55 (09.24)	45.98	37.05
2P5K	52.04 (18.21)	48.60 (10.73)	N/A	N/A
1CRN	49.35 (15.23)	49.35 (11.06)	N/A	N/A
1UTG	45.10 (11.65)	48.62 (11.06)	N/A	N/A

A small interval suggests that the corresponding residue has little flexibility, this information can be used by prediction methods as a way to reduce the protein search space. Figure 9 provides a visual analysis of the structural content of the created intervals by aligning the C_α of the experimental 3D structures and predict structures obtained by using the central value of the interval with largest probability of each amino acid as its angle value. The goal of this comparison was not to predict the 3D structure of the proteins, but to show the intervals contain structural information coherent with the experimental data. RMSD values are summarized in Table 2. The results of the cross-validation with both NEAT and MLP are in Table 1, with columns 6 and 7 showing the average neural networks success in classifying the testing set. Besides that, columns 1–4 from Table 2 have the neural networks classification results for a different set of proteins not used for training nor for testing.

The data about the enclosure and mean sizes of the MOIRAE was collected from its original paper,[13] so there is no available information for all the proteins tested in this research. What can be noted is that both methods have similar performances for proteins with high enclosures, but the proposed method obtained best results for the ones with low enclosure. Since MOIRAE creates only one interval by amino acid, the sizes of the intervals can be compared between the intervals from MOIRAE and the

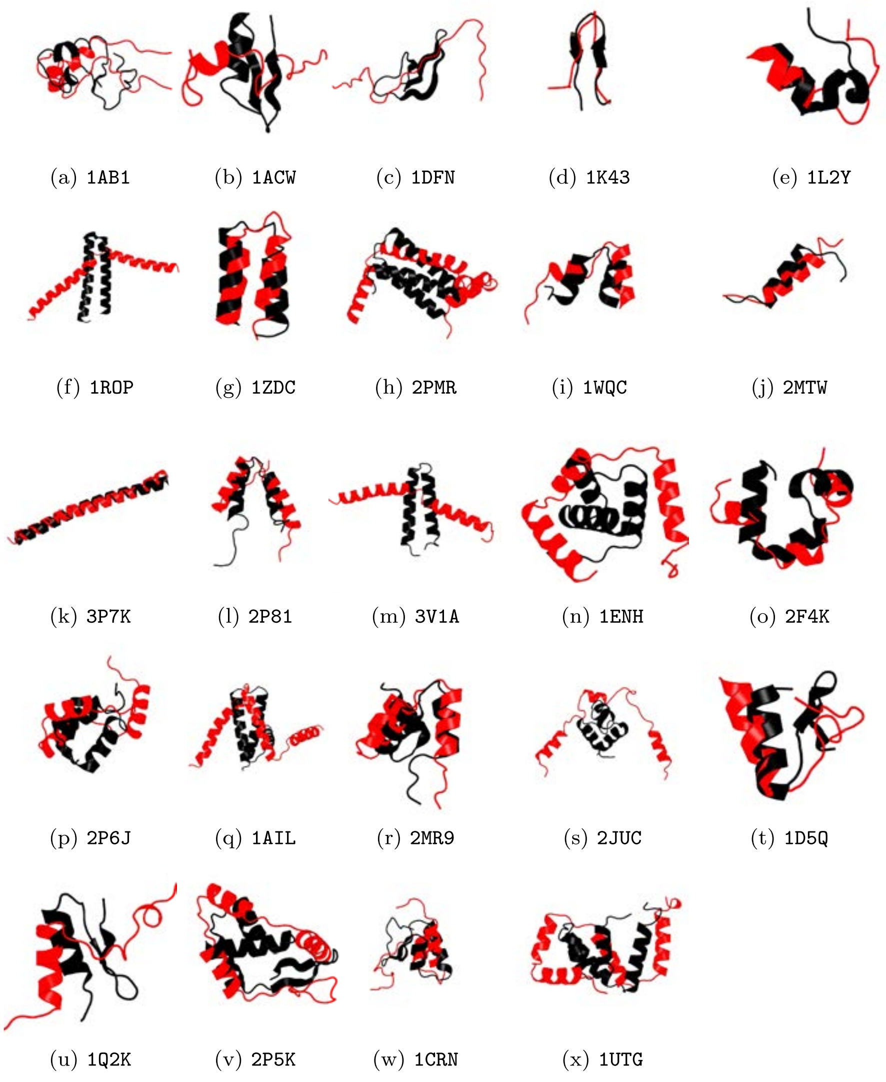


Fig. 9. Representation of 3D experimental structures (black) created with the central values from the intervals with largest probability obtained with NEAT (red). The C_α from the experimental and predict structures were aligned.

intervals with the largest probability from each amino acid. In general, MOIRAE has shorter intervals, which is expected since it uses segments of five amino acids instead of three. For this work, the size of 3 amino acids was chosen in order to avoid loss of information when performing the database search, since there is far fewer data available when searching for segments of five amino acids. In turn, this ends by creating larger intervals, but with better capacity for generalization.

5. Conclusions

The prediction of the 3D structure of proteins remains an important and arduous challenge. In this work, we propose a new method for the extraction of information from the PDB that can be used for reducing the problem search space and estimate the flexibility of the amino acids in a polypeptide chain. The use of algorithms such as hierarchical clustering and that NEAT avoids the need of previous knowledge of the data and allows the use of the method for any sequence of amino acids. The experiments results indicate that NEAT can be successfully applied to learn complex structural patterns from different protein data efficiently and the intervals created indeed contain relevant structural patterns coherent with the experimental data.

This work opens several interesting research avenues with a range of applications in computational biology and bioinformatics. For instance, one could apply the developed method to other classes of proteins; second, one could think of using search techniques such as PSO, Swarm Optimization, Simulated Annealing, GRASP or TABU search, which perhaps could lead to even more efficient algorithms for 3D protein structure prediction when flexibility of amino acid residues is considered. The search space is expected to be considerably reduced with the use of our method, thus enabling *ab initio* methods to reduce the computational time required to achieve more accurate polypeptide structures. This could, in turn, reduce the total time of *ab initio* methods which usually start from a fully extended conformation.

Acknowledgments

This research received funding from the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq); from the Fundação de Amparo a Pesquisa do Estado do Rio Grande do Sul (FAPERGS 002021-25.51/13, PRONUPEQ-16/2551-0000520-6) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior. This research is supported by Microsoft under a Microsoft Azure for Research Award.

References

1. Creighton TE, Protein folding, *Biochem J* **270**:1–16, 1990.
2. Crescenzi P, Goldman D, Papadimitriou CH, Piccolboni A, Yannakakis M, On the complexity of protein folding, *J Comput Biol* **5**(3):423–466, 1998.
3. Dorn M, Barbachan e Silva M, Buriol LS, Lamb LC, Three-dimensional protein structure prediction: Methods and computational strategies, *Comp Biol and Chem* **53**:251–276, 2014.
4. Dukka KC, Recent advances in sequence-based protein structure prediction, *Brief Bioinform,* bbw070, 2016.
5. Cozzetto D, Kryshtafovych A, Fidelis K, Moult J, Rost B, Tramontano A, Evaluation of template-based models in casp8 with standard measures, *Proteins: Struct Funct Bioinf* **77**(9):18–28, 2009.
6. Burnley B, Afonine PV, Adams PD, Gros P, Modelling dynamics in protein crystal structures by ensemble refinement, *eLife* , **1**:e00311, 2012.

7. Bornot A, Etchebest C, e Brevern AG, Predicting protein flexibility through the prediction of local structures, *Proteins* **79**(3):839–852, 2011.
8. Li J, Cai J, Su H, Du H, Zhang J, Ding S, Liu G, Tang Y, Li W, Effects of protein flexibility and active site water molecules on the prediction of sites of metabolism for cytochrome p450 2c19 substrates, *Mol Biosyst* **12**(3):868–878, 2016.
9. Li Y, Zhang Y, Atomic-level protein structure refinement using fragment guided molecular dynamics conformation sampling, *Structure* **19**(12):1784–1795, 2011.
10. Shapovalov MS, Dunbrack RL, A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions, *Structure* **19** (6):844–858, 2011.
11. Borguesan B, Barbachan e Silva M, Grisci BI, Inostroza-Ponta M, Dorn M, Apl: An angle probability list to improve knowledge-based metaheuristics for the three-dimensional protein structure prediction, *Comput Biol Chem* **59**(A):142–157, 2015.
12. Crasto CJ, Feng J, Sequence codes for extended conformation: A neighbor-dependent sequence analysis of loops in proteins, *Proteins* **42**(3):399–413, 2001.
13. Dorn M, Buriol LS, Lamb LC, Moirae: A computational strategy to extract and represent structural information from experimental protein templates, *Soft Computing* **18**(4):773–795, 2013.
14. Dong Q, Wang K, Liu B, Liu X, Characterization and prediction of protein flexibility based on structural alphabets, *BioMed Res Int* **4628025**:1–7, 2014.
15. Stanley KO, Miikkulainen R, Evolving neural networks through augmenting topologies, *Evol Comput* **10**(2):99–127, 2002.
16. Anfinsen CB, Principles that govern the folding of protein chains, *Science,* **181**(96):223–230, 1973.
17. Hovmoller TZ, Ohlson T, Conformation of amino acids in protein, *Acta Crystallogr* **58**(5):768–776, 2002.
18. Mathura VS, Kolippakkam D, Apdbase: Amino acid physicochemical properties database, *Bioinformation* **1**(1):2–4, 2005.
19. Borguesan B, Inostroza-Ponta M, Dorn M, Nias-server: Neighbors influence of amino acids and secondary structures in proteins, *J Comput Biol* , ahead of print:1–12, 2016.
20. Hornik K, Approximation capabilities of multilayer feedforward networks, *Neural Netw* **4**(2):251–257, 1991.
21. Wang S, Peng J, Ma J, Xu J, Protein secondary structure prediction using deep convolutional neural fields, ArXiv e-prints, 2015.
22. Johnson SC, Hierarchical clustering schemes, *Psychometrika* **32**(2):241–254, 1966.
23. Helles G, Fonseca R, Predicting dihedral angle probability distributions for protein coil residues from primary sequence using neural networks, *BMC Bioinformatics* **10**(1):338–346, 2009.
24. Grisci B, Dorn M, Predicting protein structural features with neuroevolution of augmenting topologies, In *2016 Int Joint Conf Neural Networks (IJCNN)*, pp. 873–880, July 2016.
25. Sohangir S, Rahimi S, Gupta B, Neuroevolutionary feature selection using neat, *J Softw Eng Appl* **7**(7):562–570, 2014.
26. Sakae Y, Hiroyasu T, Miki M, Okamoto Y, Protein structure predictions by parallel simulated annealing molecular dynamics using genetic crossover, *J Comput Chem* **32**(7):1353–1360, 2011.
27. Heinig M, Frishman D, Stride: A web server for secondary structure assignment from known atomic coordinates of proteins, *Nucleic Acids Res* **32**, 2004.
28. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ, Basic local alignment search tool, *J Mol Biol* **215**(3):403–410, 1990.

29. Fauchere J, Pliska V, Hydrophobic parameters pi of amino-acid side-chains from the partitioning of n-acetyl-amino-acid amides, *Eur J Med Chem* **18**:369–375, 1983.
30. Huang Y-F, Chen S-Y, Extracting physicochemical features to predict protein secondary structure, *Scientific World J* **2013**, 2013.
31. Jones E, Oliphant T, Peterson P *et al.*, SciPy: Open source scientific tools for Python, 2001. Accessed Online 2016-01-28.
32. Chervenski P and Ryan S, MultiNEAT, 2012. http://multineat.com/index.html. Accessed 28 January 2016.
33. Schaul T, Bayer J, Wierstra D, Sun Y, Felder M, Sehnke F, Rückstieß T, Schmidhuber J, *J Mach Learn Res* **11**(Feb):743–746, 2010.

Bruno Grisci graduated in Computer Science from the Federal University of Rio Grande do Sul, Brazil, in 2016. He is currently a Master's Student in Artificial Intelligence at the Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre, Brazil. His research interests include bioinformatics, machine learning and neural networks.

Marcio Dorn received the Ph.D. degree in computer science from the Federal University of Rio Grande do Sul, Brazil, in 2012. He is currently an associate professor with the Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre, Brazil. His research interests include bioinformatics, structural bioinformatics, machine learning, and meta-heuristics. He is currently a CNPq (Brazilian National Research Council) Advanced Fellow.