

# A Bi-Recursive Neural Network Architecture for the Prediction of Protein Coarse Contact Maps

Alessandro Vullo

Paolo Frasconi

*Department of Systems and Computer Science, University of Florence*  
Via di Santa Marta 3, I-50139 Firenze, Italy  
{vullo,paolo}@dsi.unifi.it

## Abstract

*Prediction of contact maps may be seen as a strategic step towards the solution of fundamental open problems in structural genomics. In this paper we focus on coarse grained maps that describe the spatial neighborhood relation between secondary structure elements (helices, strands, and coils) of a protein. We introduce a new machine learning approach for scoring candidate contact maps. The method combines a specialized noncausal recursive connectionist architecture and a heuristic graph search algorithm. The network is trained using candidate graphs generated during search. We show how the process of selecting and generating training examples is important for tuning the precision of the predictor.*

**Keywords:** Protein contact maps, machine learning, recursive neural networks.

## 1. Introduction

Protein folding is one of the most challenging open problems in computational molecular biology. Despite many decades of intensive research efforts, the problem has not a general solution yet. An evidence for this claim is the rapidly increasing *sequence-structure gap*. The number of proteins for which the sequences are known is about a million half [2], whereas the number of protein structures deposited in public databases after experimental determination is less than twenty thousand [6]. Excluding experimental difficulties, the reason for this impressive difference is largely due to our lack of a comprehensive theory of the folding. Protein organize themselves into a stable 3D conformation which is responsible of their biological functions. According to the experimentally confirmed Anfinsen's hypothesis [1], this tertiary structure depends only on lower order structures, plus the native solution environ-

ment. This means that the primary sequence contains all the information needed to reach the final stable conformation. The associated computational problem consists of predicting atom's 3D coordinates starting from the sequence.

Currently available classes of prediction methods include comparative or homology modeling, fold recognition, and ab initio approaches. The first two classes of methods assume some degree of similarity between the target sequences and the set of experimentally determined protein structures (at the level of sequence or at the level of fold category, respectively). In this scenario, a fourth class of methods based on machine learning has recently begun to emerge. Machine learning methods are known to be very effective in simpler but related sequential translation tasks, such as secondary structure prediction. Unfortunately, fold prediction can be hardly cast as a multivariate sequential regression task, mainly because protein structures are invariant under translations and rotations. For this reason, a straightforward approach like training feedforward neural networks to map a context-flanked residue into its main atom coordinates cannot succeed. A more reasonable perspective is to use learning algorithms to predict intermediate structural representations that are both translation and rotation invariant and constrain the space of conformations. One such description is the protein contact map, a constrained relaxed version of the distance matrix representing spatial neighborhood relationships between amino acids. The appeal of this representation in the context of folding is essentially due to the possibility of reconstructing 3D atom coordinates from contact maps using distance geometry and stochastic optimization algorithms (see e.g. [20, 18]).

Several algorithms have been proposed for prediction of contacts and contact maps [15, 7, 21, 8, 5]. At the last CASP competition [14], the best method reported a precision of 21% correct predictions [8]. More recent approaches report significantly improved precision (slightly over 50%) [5] using a two-dimensional generalization of bidirectional

recurrent neural networks [3]. In this paper we focus on a simplified topological representation that takes into account contacts at a coarse grained level of resolution, using secondary structure segments instead of the amino acids. This representation allows a significant dimensionality reduction but, at the same time, can provide useful constraints for reconstruction. Coarse maps are relatively small and can be conveniently thought of as undirected graphs. To the best of our knowledge, the literature does not report rigorous attempts for prediction of coarse contact maps although it is used as an evaluation criteria in CASP competition [14]. The method we propose in this paper is based on a heuristic graph-search algorithm, where search is guided by a connectionist architecture that assigns a score to each candidate map.

The paper is organized as follows: Section 2 introduces more formally the notion of contact map both at fine and coarse levels. In Section 3, we show how the contact maps prediction problem can be cast to learning graphs scoring functions and we introduce a recursive connectionist architecture that can perform regression tasks on any undirected graph. In Section 4 we demonstrate the viability of the approach applying our methodology to the prediction coarse contact maps for a non redundant set of proteins. Finally, Section 5 outlines possible extensions and improvements.

## 2. Contact Maps

The most accurate tools for doing protein folding are knowledge based methods, i.e. homology modeling and fold recognition. Homology modeling techniques assign a 3D structure to a novel protein searching for proteins with similar sequences. The basic assumption is that proteins with similar sequences adopt similar folds and functions. The unknown fold is then modeled using the structure of homologous as a template. However, it is frequently found that two proteins with low sequence identity have similar functions and three-dimensional structures. In this case, fold recognition techniques can be applied [9], which select the target structure as the most compatible one among those present in a library.

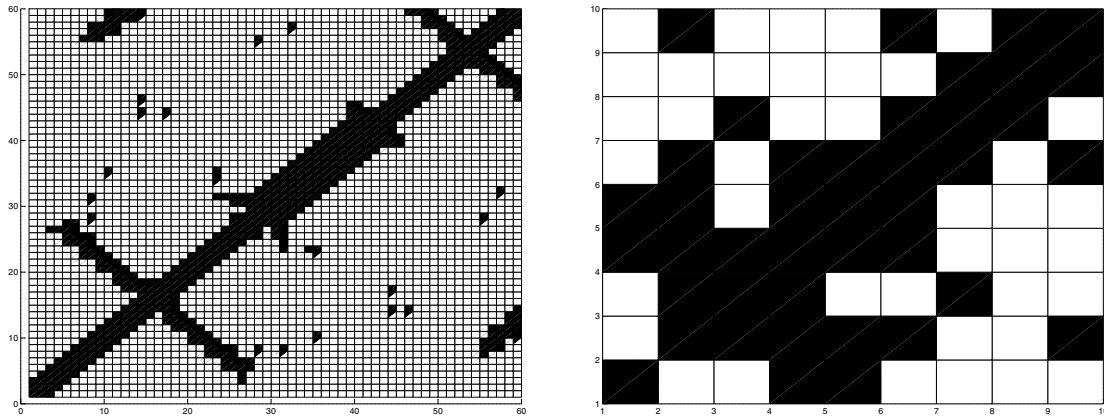
Overall, it is estimated that knowledge based methods can be applied only for about 20-30% of novel proteins. In the majority of cases, the structure of a novel protein must be assigned *ab initio*, not relying on the protein having a fold similar to a known one. Typical *ab initio* approaches compute 3D structure doing searches on the space of a protein allowable conformations. Unfortunately, exact numerical calculations are beyond the possibilities of actual and near future computers and results are obtainable only for small proteins. To overcome these difficulties, a strategy is to predict topological features to constrain the space of conformations. Such features are based on intermediate sim-

plified structural representations, such as the distance matrix that flattens the set of atom coordinates to a symmetric square matrix where the element in position  $(i, j)$  represents the distance among atoms  $i$  and  $j$ . Distance matrices have the important property of being independent of the coordinates frame. Moreover, 3D structure can be reconstructed even from partial knowledge of the matrix. Unfortunately, predicting distances is known to be very difficult and no method is available. For this reason, a further reduced representation has been investigated, known as the contact map. The contact map of a protein with  $n$  amino acids is defined as a symmetric  $n \times n$  matrix, with elements  $c_{ij}$  defined as:

$$c_{ij} = \begin{cases} 1 & \text{if amino acid } i \text{ and } j \text{ are in contact} \\ 0 & \text{otherwise} \end{cases}$$

The notion of contact is closely related to a relation of spatial proximity and many definitions are possible. Topically, two amino acids are said to be in contact if their distance is below a given threshold (in Amstrongs). Commonly used distance definitions are that between  $\alpha$ -helices (7-9 Å),  $\beta$ -strands (6.5-8 Å) atoms, or the minimal distance between two atoms belonging to the side-chain or backbone of the two residues (4.5 Å). Contact map representations can be derived not only at amino acid level, but also looking at secondary structure elements. Contacts in coarse maps are associated with the intuitive spatial neighborhood concept between two secondary structure elements. However, the spatial relation is not uniquely defined. Reasonable definitions could consider two elements in contact if the distance between their centers of gravity falls below a given cutoff or if there are any two atoms not belonging to the same element whose distance is below a given cutoff. Figure 1 shows an example of a protein fine-grained and coarse grained contact maps. In contact maps,  $\alpha$ -helices appear as thick bands of contacts along the diagonal (residues 28-41) and  $\beta$ -sheets as bands parallel or anti-parallel to the diagonal (residues 18-25 and 56-60).

Although a contact map is an approximate version of the distance matrix, it still conserves useful characteristics for the reconstruction process. Non zero elements in a contact map can be thought as spatial restraints and used in reconstruction algorithms based on distance geometry or Monte Carlo energy minimization approaches. Recently, Vendruscolo [19] has shown that even partial or noisy knowledge of the map may be sufficient for near exact reconstruction. Concerning coarse resolution maps, we argue that good predictions of secondary structure contacts could also be strongly informative about the shape of the unknown fold and usable to recover the structure too (Vendruscolo M., private communications). The main advantage of working at the segment level is to obtain a significant dimensionality reduction allowing better but more costly algorithms to be employed (Figure 1 gives an idea).



**Figure 1. Fine-grained (left) and coarse-grained (right) contact maps of streptococcal protein g (pdb code: 2igd).**

Here we propose a general architecture which can be used for the prediction of both fine-grained and coarse-grained contact maps. Our method looks at a contact map as an adjacency matrix of an undirected graph whose vertices are residues or secondary structure segments and whose edge set is the contact relation defined on pairs of residues or segments.

### 3. Methodology

The problem of predicting a protein contact map is formulated as inferring a good evaluation function whose purpose is to guide a heuristic graph search procedure mapping a sequence into an associated contact map. Let  $C = (V, E)$  denote the target contact map (an undirected graph). For every candidate map  $C' = (V, E')$ , let  $f(C' | C)$  be a scoring function taking values in  $[0, 1]$  and having the following properties:

1.  $f(C' | C) = 1$  iff  $C' = C$ ;
2.  $f(C' | C) = 0$  iff  $C'$  is the null graph
3. if  $C' \subset C$  and  $(\{e\} \cup C') \cap C = \{e\} \cup C'$ , then  $f(C' | C) > f(\{e\} \cup C' | C)$ .

If we assume that  $C$  is known, the correct contact map can be found by a simple procedure. The search algorithm takes the vertex set as input and starts by assigning the empty edge set to the candidate solution  $C'$ . The main loop essentially consists of three basic operations: generation (all  $C'$ 's successor are generated by applying allowable graph edit operators), evaluation ( $C'$ 's successors are scored using the function  $f(C' | C)$ ), and update ( $C'$  replaced by a new graph closer to the solution). The "monotonicity" assumption 3 above requires only an edit operator that adds

one edge  $(u, v)$  to  $C'$ , if  $(u, v)$  is not already in  $C'$ . Pseudo code version of the algorithm is given in Figure 2. Using the

```

SEARCH-CMAP(V)
1  E ← ∅
2  repeat
3    m ← 0
4    for each v in V
5      do for each u in V - {v}
6        do if (u, v) ∉ E
7          then a ← SCORE(E ∪ {(u, v)}, V)
8             if a > m
9                then m ← a
10                  e ← (u, v)
11      E ← E ∪ {e}
12  until m = 1
13  return E

```

**Figure 2. Pseudo code for finding the correct contact map with a perfect evaluation function.**

previous assumptions it can be easily shown that, when the algorithm terminates,  $C' = C$  so  $f(C' | C) = 1$ . To analyze the algorithm we observe that each iteration requires the evaluation of  $|V|$  for each candidate edge, i.e.  $(|V|)^2$  times. Overall,  $C'$  is evaluated  $(|V| \cdot |V|)^2$  times. Assuming that  $f(C' | C)$  takes linear time in  $|V|$  and that  $|V| = |E|$  (contact maps are usually sparse adjacent matrices), it follows that SEARCH-CMAP is a polynomial algorithm and takes time  $(|V|)^4$ . The apparent simplicity of the overall procedure for finding  $C$  is clearly due to the existence of an oracle that can compute  $f(C' | C)$  for every candidate map. In the following we first suggest a suitable scoring function and then we propose a neural network model capable of learn-

ing  $(\mathcal{C})$  from examples of successful searches.

### 3.1. Scoring functions

For each map  $\mathcal{C} = (\mathcal{C})$ , let us first introduce precision and recall of  $\mathcal{C}$  (relative to  $\mathcal{C}^*$ ) as follows:

$$p(\mathcal{C}) = \frac{|\mathcal{C} \cap \mathcal{C}^*|}{|\mathcal{C}|} \quad (1)$$

$$r(\mathcal{C}) = \frac{|\mathcal{C} \cap \mathcal{C}^*|}{|\mathcal{C}^*|} \quad (2)$$

Precision (Eq. 1) or specificity is the fraction of edges in the predicted set  $\mathcal{C}$  that are correctly assigned (they are also present in  $\mathcal{C}^*$ ). Recall (Eq. 2) or coverage is the fraction of edges in  $\mathcal{C}^*$  that have been correctly discovered, i.e. they are also present in  $\mathcal{C}$ . It can be easily verified that the function

$$f(\mathcal{C}) = \frac{2 \cdot p(\mathcal{C}) \cdot r(\mathcal{C})}{p(\mathcal{C}) + r(\mathcal{C})} \quad (3)$$

satisfies the first assumption and is therefore a suitable metric for guiding the heuristic search. Readers familiar in information retrieval may recognize the function in Eq. 3 as the  $F_1$  ( $\beta = 1$ ) metric.

### 3.2. Learning to predict the scoring function

The scoring function in Eq. 3 requires the knowledge of  $\mathcal{C}^*$ , which is unknown at prediction time. In realistic situations, the scoring function in algorithm SEARCH-CMAP must rely only on  $\mathcal{C}$  to give suitable scores to candidate solutions. In the case of fine-grained maps, elements in  $\mathcal{C}$  are amino acid symbols, whereas at coarse-grained level each secondary structure element can be described by a set of numerical and categorical attributes, such as length and position within the sequence, secondary structure category, and physico-chemical properties (e.g. related to the average exposure to solvent).

We now introduce a machine learning method that attempts to predict  $\mathcal{C}^*$  using only the information in  $\mathcal{C}$ . One fundamental issue is the generation of the training set. For each sequence of length  $|V|$  there are  $2^{O(|V|^2)}$  possible distinct contact maps. Clearly, using all these graphs as training examples is not realistic and a sub sampling strategy is required. The training set generation can be either *static* (i.e., examples are selected before training begins) or *dynamic* (i.e. examples are inserted and deleted as training proceeds). In both cases the correct  $\mathcal{C}^*$  is known during training.

In the static case, we note that random selection of graphs for each protein would bias the distribution of examples towards low score instances and would be extremely

unlikely to yield a balanced dataset (the probability of guessing a random graph with high score decreases exponentially with  $|V|$ ). A simple strategy that guarantees a reasonable balance between high and low score graphs is to run algorithm SEARCH-CMAP guided by  $\mathcal{C}^*$  and to collect as training examples all the  $(\mathcal{C} | \mathcal{C}^*)$  graphs that are generated during the search. One disadvantage of this static strategy is that after training, the learner is specialized in a relatively narrow region of the search space and, during prediction, repeated errors may drive the search algorithm far away from the goal. In order to mitigate this problem, the hill-climbing procedure in algorithm SEARCH-CMAP can be changed into a beam search. Unlike hill-climbing, that keeps at each stage the best candidate only, beam search maintains a bounded open list of size  $k$ . The open list is filled at each stage with the best  $k$  candidates, selected from all the possible successors of graphs in the open list at the previous stage.

In the dynamic case, ideas for exploring the state space can be borrowed from the reinforcement learning literature [17, 16]. During the learning phase, for a given sequence

$\mathcal{C}$  the learner is asked to follow a particular trajectory from the null graph to a final graph according to a given policy. A policy maps states to actions where a state is a candidate contact map and an action is an edit operator that adds one edge to a graph. Figure 3 summarizes the steps used to train a learning module in the dynamic case (using contact maps space exploration).

```

EXPLORE-CMAPSPACE( $V, L$ )
1  $E \leftarrow \emptyset, S \leftarrow (V, E)$ 
2 repeat
3    $L \leftarrow \text{LEARN}(L, S, s^*(E))$ 
4    $e \leftarrow \text{POLICY}(V, E)$ 
5    $S \leftarrow (V, E \cup \{e\})$ 
6 until  $\text{ISTERMINALSTATE}(S)$ 
7 return

POLICY( $V, E$ )
1  $\hat{E} \leftarrow \emptyset$ 
2 for each  $v$  in  $V$ 
3 do for each  $u$  in  $V - \{v\}$ 
4   do if  $\neg(u, v) \in E$ 
5     then  $a \leftarrow \text{SCORE}(E \cup \{(u, v)\}, V)$ 
6      $\hat{E} \leftarrow \hat{E} \cup (u, v, a)$ 
7 return  $e \in \hat{E}$ 

```

**Figure 3. Active learning by means of contact maps space exploration.**

EXPLORE-CMAPSPACE follows a trajectory from the null graph to a terminal graph for a given training sequence  $\mathcal{C}$ . During a training epoch, the algorithm runs for all the sequences in the training set. Three main steps are ex-

cuted in the main loop of EXPLORE-CMAPSPACE: (1) the learner is trained with the example given by the current candidate contact map and its score (line 3); (2) an action is selected in current state (line 4), resulting in the transition to a new state (line 5). The loop stops when a terminal state is reached (line 6). To avoid an endless trial, we consider to be in a terminal state when  $|E|$  transitions have been made (actually exploiting information given by the target contact map). The pseudo code at the bottom of Figure 3 define the steps used by a policy to select an action in a given state. POLICY maintains a labeled edge set  $\hat{E}$ , where each  $e \in \hat{E}$  is labeled with the score resulting from the adjoint of  $e$  to the edge set of the current graph.  $\hat{E}$  is built generating all possible successor graphs and computing their scores. In line 7, an edge from  $\hat{E}$  is returned, but the details on how to do this are not indicated, because this is what differentiates an exploration strategy from another one.

In our experiments, we have implemented various strategies, investigating the effect of the exploration-exploitation trade-off, which is well-known in reinforcement learning [17, 16]. In *random exploration*, we choose the successor graph randomly with uniform probability, whereas in *pure exploitation* the successor is always chosen to be the best possible one (according to the computed score). In *semi-uniform* exploration (also known as  $\epsilon$ -greedy policy), with probability  $\epsilon$  a successor is chosen uniformly from the set of successors and with the remaining probability  $1 - \epsilon$  we choose the best one. The experimental section gives details of the effects for the preference of a strategy with respect to the others. The choice that is made greatly affects the performance in terms of precision and recall (Eq. 1 and 2).

### 3.3. Bi-recursive neural network architecture

Given a candidate contact map  $C$ , the basic structure of the learning task is formulated as the problem of predicting  $(C)$ . As we have seen, the information we deal with is naturally described as a set of *structured* instances, i.e. modeled as graphs. By considering the application domain we come up with the notion of data structures. A data structure is a graph whose nodes are marked by sets of domain variables, called *labels*. Domain variables contained into labels are also called *attributes*, and may be either numerical or categorical. In this paper a data structure is considered to be a pair  $(G, L)$  where  $G = \{v_1, \dots, v_n\}$  is a set of labeled nodes or tuples and the edge set  $E \subset G \times G$  refers to pairs of elements (amino acids or secondary structure segments) whose distance  $d \leq d_{max}$  for some specified  $d_{max}$ . In the case of secondary structure segments, each tuple  $v_i$  contains attributes characterizing the  $i$ -th segment of the protein, such as type ( $\alpha$ -helix,  $\beta$ -strand or loop), amino acid frequencies within the segment, length in residues, average solvent exposure, and so on.

The supervised nature of the learning task (for each  $v_i$  in the training set,  $(C(v_i))$  is known) and its formulation in terms of a regression problem, suggests the use of connectionist models. Standard connectionist architectures, like feed-forward neural networks, can only deal with static (attribute-valued) data types and they are not the most appropriate choice when the learning domain presents additional information provided by the structured nature of data (modeling complex relations amongst atomic entities). These observations motivates the adaptation of a class of models for classification and regression on hierarchical data structures, called *Recursive Neural Networks* (RNN). RNNs [10] generalize the recursive state updating scheme of recurrent neural networks from sequences to directed acyclic graphs and rely on hidden state space representation. Likewise the sequential dynamics, where the state at a given time  $t$  is a summary of the information observed in time steps previous to  $t$ , in graphical dynamics the state at a given vertex  $v$  summarizes contextual information provided by the input substructure induced by descendants of  $v$ .

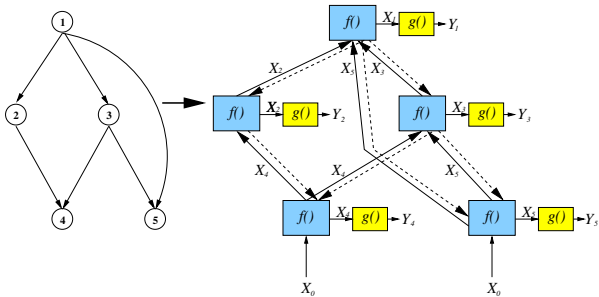
The adaptive processing of data structures implies the concept of transduction which is essentially a binary relation defined between an input structured space and an output one. Learning with data structures generally means inferring models (deterministic or probabilistic) of structural transductions which act as processors transforming structures in input to those in output. Learning general mappings is an open problem and models have been developed for the limited case of *IO-isomorph* transductions, that is functions in which the skeleton class of input and output data is the same. By skeleton class we mean a set of topological conditions satisfied by all instances in a given set. Here we are interested in data being in the class of bounded labeled positional DAGs (m-DPAGs) where for each labeled vertex  $v$ , a total order is defined on  $v$ 's children and  $(c_1) \leq (c_2)$  for some  $c \in C$ . Let  $ch[v]$  be the ordered  $m$ -tuple of  $v$ 's children, and  $(c)$  be the input and output labels attached to node  $v$ . Still, let  $I$  and  $O$  denote respectively the input and output label space. The graphical dynamics implied by the transduction is defined introducing, for each node  $v$  a state variable  $s_v \in S$  such that:

$$s_v = (c, ch[v]) \quad (4)$$

$$s_v = (c, s_{c_1}, \dots, s_{c_m}) \quad (5)$$

where  $f : S^m \times C \rightarrow S$  is the state transition function and  $g : S \times C \rightarrow O$  is the output function. In the above equation,  $(c, ch[v])$  is the  $m$ -tuple of state variables attached to the children of vertex  $v$ . If there are missing children, the corresponding tuples entries are replaced by a predetermined frontier state  $s_0 \in S$ . The frontier state plays the same role of the initial state for temporal transductions and it is associated with the base step of recursion.

A recursive neural network is a graphical formalism based on eq. 4 and 5 where the state transition function and the output function are approximated by feed-forward neural networks. In a typical connectionist realization of the state transition function the state variable is a continuous vector (i.e.  $\equiv^n$ ). The network realizing has  $|I| + |O|$  input units and  $|O|$  output units. Similarly, the network realizing has  $|I| + |I|$  input units and  $|O|$  output units. It is important to note that we are considering *stationary* RNNs, that is weights in the transition and output networks are independent of the node at which the functions are applied (all the nodes have the same replicas for transition and output function). This is interpretable as a form of weight sharing which influences the learning algorithms.



**Figure 4. Forward and backward propagation in a recursive neural network.**

Inference or propagation on a given m-DPAG is obtained by unrolling the state transition function according to a reverse topological order of its nodes. This guarantees that when a node is processed, all its successors have already been processed. For each node the previously computed representations of its children together with the encoding of its label are fed into the input layer of the network computing . The output of the network represents the state vector for current node which can be given as input to the's network together with the input label. At the end of the recursion, the output vector  $\in$  is computed for each node and the transduction is complete. The main idea is depicted in Figure 4. On the left, an example input DPAG is shown. On the right, the state transition network is unrolled bottom up (following solid lines) to match the topology of the input graph. Learning is accomplished by a special form of backpropagation referred to as *backpropagation through structure* (BPTS). The algorithm was first proposed in [11] for the limited case of labeled ordered -ary trees. To explain the algorithmic idea, first note that all the copies of transition and output networks linked according to the edge set of the input graph can be thought of as a global feed-forward neural network on which gradients can be computed and propagated. After forward propagation, output

is computed for each node . Then, the error for output is back-propagated through the network for each node . At this point, all nodes are processed according to a topological order (top-down in Figure 4, dashed lines). At each node , errors coming from its parents are back-propagated through the replica of in summed with error gradient contribution coming from current replica. Note that the stationarity assumptions implies that the overall gradient is obtained summing the error contributions from all the nodes.

The RNN architecture presented in [10] cannot be applied directly since it requires an input graph to be directed, ordered and acyclic, while contact maps are normally modeled as undirected, unordered and possibly cyclic graphs. The extension we propose here is based on forward-backward state space factorization to avoid directed cycles. Although is an undirected graph, a total (serial) order relation is naturally defined on its vertices allowing us to interpret as a pair of directed acyclic graphs: the forward graph (from N to C-terminus)  $f = (f)$  with  $f = \{(i, j) \in : \}$  and the backward graph (from C to N-terminus)  $b = (b)$  with  $b = \{(i, j) \in : \}$ . The novelty of the model, which we call *Bi-recursive Neural Network*, rely on the introduction of the following state space factorization:

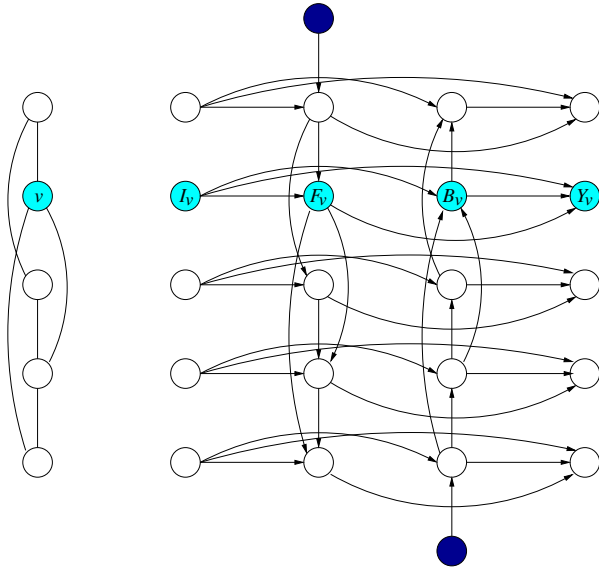
$$= (c [ ] ) \quad (6)$$

$$= (c_b [ ] ) \quad (7)$$

$$= ( ) \quad (8)$$

where is the forward state variable storing contextual information coming from the nodes at the beginning of the sequence to the nodes at the end.  $c [ ]$  is the -tuple of state variables attached to's children on graph  $f$ . Similarly, is the backward state variable storing contextual information coming from the nodes at the end of the sequence to the nodes at the beginning.  $c_b [ ]$  is the -tuple of state variables attached to children of on graph  $b$ . Now the output label at node depends on both the forward and backward states, together with the input label. The dependencies among random variables (Eq. 6, 7, 8) and implied by the graphical model are shown in Figure 5. A similar method was employed in [3] to realize a bi-directional recurrent neural network, a non-causal architecture that exploits upstream and downstream dependencies in sequence for the prediction of protein secondary structure.

The inference process in the Bi-recursive Neural Network model for biological sequences closely resemble that of RNNs for m-DPAGs. Note that in this domain, the topological order of the nodes is naturally dictated by the ordering of the sequence elements. Suppose an undirected graph  $= ( )$  ( $|I| =$ ) is given as input to the Bi-recursive model. The model applies the transformation



**Figure 5. Left: a sample graph ; right: graphical model for the bi-recursive network, consisting of input nodes, output nodes and two sets of hidden states linked according to forward and backward graphs  $f$  and  $b$ .**

$\rightarrow (f, b)$ . Then, starting with the node of the C-terminus and following the reverse sequence order, each state  $i$  is recursively updated according to the forward state transition function and the input (Eq. 6). Similarly, starting with the N-terminus node and following the sequence order, each state  $i$  is recursively updated according to the backward state transition function and the input (Eq. 7). At this point, for each node  $i$  in the sequence the output can be computed giving  $F_i$  and  $B_i$  as input to the function (Eq. 8). The learning algorithm is a natural extension of BPTS adapted for the double graphical dynamics. After a complete run of forward propagation, error for  $i$  is back-propagated through the replicas of network  $f$ , for each node  $j$ . Then, gradient contributions are propagated through replicas of  $f$  following the causal chain of states  $\{i-1, \dots, 1\}$  and through  $b$ 's replicas following the non-causal chain  $\{i+1, \dots, N\}$ . For each  $j$ , deltas coming from  $f$  network are stored so they can be properly summed with the delta contributions for  $j$ 's weights coming from  $b$ 's parents in  $f$  and error signals for  $j$ 's weights from  $b$ 's parents in  $b$ .

One issue remains to be addressed. Specifically, how the model predicts the global graph score  $G$ , given that it locally computes outputs for each single node  $i$  in  $G$ . Let  $E_f = \{(i, j) \in E : i \rightarrow j \text{ in } f\}$  and  $E_b = \{(i, j) \in E : i \rightarrow j \text{ in } b\}$  denote the subsets of edges incident on vertex

respectively in  $f$  and  $b$ . We introduce precision, recall and  $F_1$  measure of  $G$  (relative to  $G$ ) at the level of node  $i$ :

$$P(i) = \frac{|E_f(i)|}{|E_f(i)| + |E_b(i)|} \quad (9)$$

$$R(i) = \frac{|E_b(i)|}{|E_f(i)| + |E_b(i)|} \quad (10)$$

$$F_1(i) = \frac{2 \cdot P(i) \cdot R(i)}{P(i) + R(i)} \quad (11)$$

The following formula computes the score of  $G$  as a weighted sum of the local scores computed at node level:

$$S(G) = \frac{1}{|V|} \sum_{i=1}^N F_1(i) \quad (12)$$

We found that  $S(G)$  is a good locally based representation of  $G$ . Preliminary statistical studies have shown high correlation (0.9) between  $S(G)$  and  $G$  values. The inference process of our model computes the sequence of output values  $\{Y_1, \dots, Y_N\}$  and predicts  $G$ 's score as:

$$net(G) = \frac{1}{|V|} \sum_{i=1}^N Y_i \quad (13)$$

During backward propagation, learning takes place giving output networks the error signal  $Y_i - G(i)$  for  $i = 1, \dots, N$ .

#### 4. Implementation and Results

Our method has been extensively tested for the prediction of protein contact maps at the coarse level. All the experiments were carried out using a significant fraction of the current representative set of non homologous protein data bank chains (PDB Select, [12]). We extracted the chains in the file 2001\_Sep.25<sup>1</sup> listing 1544 proteins (1641 chains) with percentage of homology identity less than 25%. From this set we retained only high quality proteins on which the DSSP program [13] (CMBI version) does not crash, determined only by X-ray diffraction (not multiple NMR models), without any physical chain breaks and resolution threshold less than 2.5 Å. From the filtered set we retained the proteins with sequence length less than 300 amino acids, resulting in a database of 587 proteins. The DSSP program was also used to assign secondary structure categories. The automatic assignments were projected to the three secondary structure states alpha, beta and gamma, with the following criteria.: H maps to alpha, E maps to beta and the rest to gamma. Resulting segments with only one amino

<sup>1</sup>ftp://ftp.embl-heidelberg.de/pub/databases/

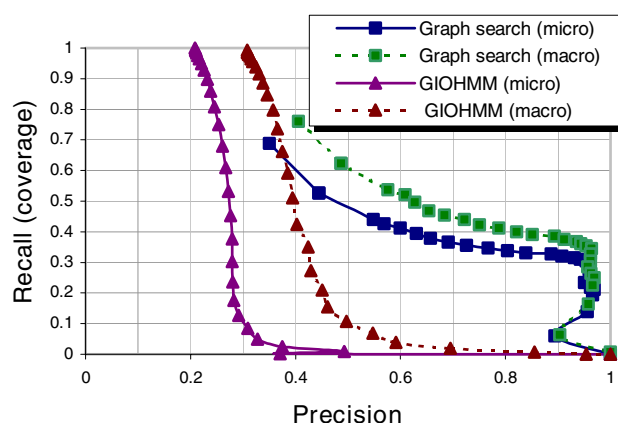
acid were discarded. As previously noted, working at the segment level results in significant dimensionality reduction. For example in a subset of 2000 PDB sequences with low similarity, the average segment length is 7.12 residues, thus the size of the coarse map is, on average, roughly 2% of the size of the residue resolution map.

An important issue is the definition of contact between secondary structure segments. In addition to the alternatives introduced in Section 2, we also tested another possible definition considering an average distance between the projections of the principal axes of secondary structure segments convex hulls. After preliminary attempts we did not find major differences. We also noted that a segment contact threshold fixed at 8Å is related to more than 20% of amino acids not in the same segment being in contact. In all the experiments reported here we adopted for the definition of contact as if the distance between any two atoms not in the same segment is less than 8Å. Eight numerical features encode the input label of each node and comprise one-hot encoding of secondary structure type; normalized linear distances from the N to C terminus; average, maximum and minimum hydrophobic character of the segment (based on the Kyte-Doolittle scale on moving 7-length window centered at all residues positions in the segment). Note that the network model in its present implementation uses as input a minimal amount of biologically significant information.

Several preliminary experiments were carried out to tune up the prediction system and choose the best architectural parameters. Splitting the proteins into training, test and validation sets we selected a Bi-Recursive NN architecture with state vectors of dimension five (both for forward and backward dynamics) and without hidden layers. By using hidden layers or a greater dimension for the state vectors we found the model particularly sensitive to the overfitting phenomenon. We then performed a set of experiments trying to figure out the effect of the static training set generation as explained in Section 3. For each protein we generated a sample of graphs by means of a search procedure towards the target graph guided by a perfect evaluation function (but collecting all the valid successors during the search) ending up in a total of 40,275 graphs. The above dataset was split in a training set of 300 proteins (19,574 graphs), a test set of 150 proteins (11,117 graphs) which was used to estimate the prediction accuracy, and a validation set of 137 (9,584) used for early stopping. The adapted version of BPTS for Bi-Recursive model was used as training procedure. The trained network replaced

( ) as a heuristic evaluation function in the subsequent topology search algorithm, which was based on a beam search procedure with beam size = 10. The search algorithm scales as  $(|V| \cdot |E|)^2$ , being the number of weights in the network. We compared our algorithm to the

model (GIOHMM) [4] that was used in [5] for prediction of contact maps at residue level. Shortly, a two-dimensional GIOHMM is a graphical model with nodes regularly arranged in one input plane, one output plane, and four hidden planes. In each plane, nodes are arranged on a square lattice. Each hidden plane contains directed edges associated with the square lattices, oriented in the direction of one of the four possible cardinal corners: NE, NW, SW, SE. The method does not perform graph search.



**Figure 6. Precision-recall plots comparing graph search to GIOHMMs.**

Results are summarized in Figure 6, where we plot macro- and micro-averaged precision vs. recall. Macro-averages are computed by averaging precision and recall over the set of proteins. This measure tends to weight more the performance on short sequences. Micro-averages are obtained by computing precision and recall over the flattened set of segment pairs. The precision of our search algorithm is consistently better than that of GIOHMM, although recall never reaches 100%.

As a final experimental step, besides prediction accuracy, we investigated the effect of dynamic sampling (i.e. exploration) during training procedure on network performances. We applied the three exploration strategies described in Section 3: random exploration, pure exploitation and semi-uniform exploration. The last one was applied trying to find the optimal balance between exploration and exploitation. The probability of uniform random exploration was set to  $p = 0.4$ . In addition, we also tried a fourth strategy in which the search proceeds greedily (i.e. the best successor is chosen at each step, as in pure exploitation), but this time the network being updated on a representative sub-sample of the successors of the current state. The main purpose of this exploration scheme is to guide gradient descent towards a region where parameters are optimized us-

Sampling strategy		( )		<sub>1</sub>		( )		<sub>1</sub>
Random exploration	.715	.769	.418	.518	.767	.709	.469	.574
Semi-uniform exploration	.454	.787	.631	.526	.507	.767	.702	.588
Pure exploitation	.431	.806	.726	.539	.481	.793	.787	.596
Hybrid	.417	.834	.79	.546	.474	.821	.843	.607

**Table 1. Training bi-recursive neural networks with dynamic sampling: summary of experimental results.**

ing the wide possible spectrum of values for candidate alternatives. In these experiments, we used a 5-fold cross validation procedure splitting a representative set of 370 proteins into 5 subsets and routinely testing on one subset and training on the remaining four. After cross validation for each exploration scheme we obtained the results indicated in Table 1. First column is labeled with the different updating schemes we applied. For each strategy we report performances measured with several indices: micro and macro-averaged precision ( , ), recall ( , ) and <sub>1</sub> measure ( <sub>1</sub>, <sub>1</sub>). Besides these, we also provide percentage of correct prediction for non-contacts averaged over the set of proteins ( ( )) and over the whole segments pairs ( ( )). Last row (Hybrid) provides the indices obtained with the example selection strategy described above.

## 5. Conclusions

We have presented a system which can be used for the prediction of protein coarse contact maps. The method is based on two architectural stages. First, a new connectionist model called Bi-recursive Neural Network is trained to predict real valued scores for candidate maps viewed as undirected graphs. Second, the trained model is employed as a heuristic evaluation function to guide search algorithms in undirected graphs space towards the goal map.

The experimental evaluation has shown how this architecture succeeds in prediction of coarse contact maps and how the training procedure can be tuned up biasing the predictor to favor precision and/or recall. Whether or not to improve precision vs. recall is a matter on how reconstruction algorithms are able to build a 3D structure from a corrupted contact map. Clearly, richer sequence descriptions than those we have used here can be integrated in order to further improve performance.

In principle our method can also deal with contact maps defined at the residue level, although experimental work in this direction is still to be done. In particular, our current implementation is still too computational demanding for long sequences and work is in progress to improve the algorithm efficiency for application at the residue-level resolution.

Finally we note that the approach presented here has a general applicability in all the contexts in which we can model data as undirected graphs. One possible direction is to apply the Bi-recursive model for the prediction of cysteines disulfide bonds connectivity in proteins.

## References

- [1] C. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.
- [2] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL. *Nucleic Acids Res.*, 28:45–48, 2000.
- [3] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri. Exploiting the past and the future in protein secondary structure predictor. *Bioinformatics*, 15:937–946, 1999.
- [4] P. Baldi and G. Pollastri. Generalized IOHMMs and recurrent neural network architectures. submitted.
- [5] P. Baldi and G. Pollastri. Prediction of contact maps by recurrent neural network architectures and hidden context propagation from all four cardinal corners. *Bioinformatics*, 2002. to appear.
- [6] F. Bernstein et al. The Protein Data Bank: a computer based archival file for macromolecular structures. *Journal of Molecular Biology*, 112:535–542, 1977.
- [7] P. Fariselli and R. Casadio. Neural network based predictor of residue contact in proteins. *Protein Engineering*, 12:15–21, 1999.
- [8] P. Fariselli, O. Olmea, A. Valencia, and R. Casadio. Prediction of contact maps with neural networks and correlated mutations. *Protein Engineering*, 14:835–843, 2001.
- [9] D. Fisher and D. Eisenberg. Fold recognition using sequence derived properties. *Prot. Sci.*, 5:947–955, 1996.
- [10] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Trans. on Neural Networks*, 9:768–786, 1998.
- [11] C. Goller and A. Kuchler. Learning task-dependent distributed structure-representations by backpropagation through structure. *IEEE Trans. on Neural Networks*, pages 347–352, 1996.
- [12] U. Hobohm, M. Scharf, R. Schneider, and C. Sander. Selection of representative data sets. *Prot. Sci.*, 1:409–417, 1992.
- [13] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

- [14] A. Lesk, L. L. Conte, and T. Hubbard. Assessment of novel fold targets in CASP4: predictions of three dimensional structures, function and genetics. *Proteins*, 2001.
- [15] I. Shindyalov, N. Kolchanov, and C. Sander. Can three-dimensional contacts of proteins be predicted by analysis of correlated mutations? *Protein Engineering*, 7:349–358, 1994.
- [16] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [17] S. Thrun. The role of exploration in learning control. In D. White and D. Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, Kentucky, 1992.
- [18] M. Vendruscolo. Protein folding using contact maps and contact vectors. In P. Frasconi and R. Shamir, editors, *Artificial Intelligence and Heuristic Methods in Bioinformatics*, NATO Science Series. IOS Press, 2002.
- [19] M. Vendruscolo, E. Domany, and K. Park. Towards an energy function for the contact map representation of proteins. *Proteins*, 40:237–248, 2000.
- [20] M. Vendruscolo, E. Kussell, and E. Domany. Recovery of protein structure from contact maps. *Folding and Design*, 2:295–306, 1997.
- [21] M. Zaki, S.J., and C. Bystroff. Mining residue contacts in proteins using local structure predictions. In *IEEE Int. Symposium on Bioinformatics and Biomedical Engineering*, pages 168–175, Washington, DC, 2000.