ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

Graph Neural Networks for the Prediction of Protein–Protein Interfaces

Niccolò Pancino[1,2] Alberto Rossi[1,2] Giorgio Ciano[1,2]
Giorgia Giacomini[1] Simone Bonechi[1] Paolo Andreini[1]
Franco Scarselli[1] Monica Bianchini[1] Pietro Bongini[1,2,c]

1 – SAILAB - University of Siena - via Roma 56, 53100, Siena, Italy
2 – DINFO - University of Florence - via di Santa Marta 3, 50139, Florence, Italy
c – Corresponding author, email: pietro.bongini@gmail.com

Abstract. Binding site identification allows to determine the functionality and the quaternary structure of protein–protein complexes. Various approaches to this problem have been proposed without reaching a viable solution. Representing the interacting peptides as graphs, a correspondence graph describing their interaction can be built. Finding the maximum clique in the correspondence graph allows to identify the secondary structure elements belonging to the interaction site. Although the maximum clique problem is NP-complete, Graph Neural Networks make for an approximation tool that can solve the problem in affordable time. Our experimental results are promising and suggest that this direction should be explored further.

1 Introduction

Proteins are fundamental molecules for life. They are involved in any biological process that takes place in living beings, carrying out a huge variety of different tasks. In these molecules, functionality and structural conformation are strictly correlated [8]. Therefore, analyzing structural features of proteins is often useful in understanding which biological processes they are involved in, which ligands they bind to and which molecular complexes they form.
The structure of a protein can be described at three different levels: the *primary* structure corresponds to the sequence of amino acids it is composed of; the *secondary* structure corresponds to the local conformation of the peptide chain, in the shape of α*–helices*, β*–sheets* or *coils*; the *tertiary* structure represents the three–dimensional configuration of the molecule. Often, two or more molecules bind together to form a protein complex, whose shape goes under the name of *quaternary* structure. *Dimers* are the simplest protein complexes, as they are composed of just two *monomers*.
To form such complexes, *monomers* interact through specialized parts of their surface, called *binding sites* or *interfaces*. These interactions can be studied with the help of *graph theory*. Indeed, each *monomer* can be represented as a graph, with nodes corresponding to secondary structure elements (SSEs), while edges stand for spatial relationships between adjacent SSEs, which can be parallel, anti–parallel or mixed. Using graphs of two different *monomers*, a *correspondence graph* can be built, whose nodes describe all the possible couples of SSEs from the two different subunits [7]. Based on the correspondence graph, identifying binding sites on protein surfaces can be reformulated as a maximum clique search problem [5].
The *maximum clique problem* is known to be an NP–complete problem, meaning that,

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

except for very small graphs, traditional operations research algorithms [3] will employ a prohibitive amount of time before solving it. From this consideration stemmed the idea of using a machine learning method to solve the problem with reasonable computational costs. In particular, Graph Neural Networks (GNNs) [16] look like the perfect model, with their ability to process graph–structured inputs. GNNs have seen many recent advances and have become a leading tool in graph–based applications [11, 18, 13, 15, 2].
The maximum clique problem consists in a binary classification between the nodes which belong to the maximum clique and those which do not. Clique detection was already addressed with GNNs in the seminal work [6], and, more recently, also in the transductive learning framework [14]. Finally, this strategy was also further refined by exploiting the deeper version of GNNs, the Layered Graph Neural Networks (LGNNs) [1]. In this model, each layer is a standalone GNN which is trained separately, using always the same target. The solution proposed by the previous layer is integrated to the input of each layer after the first, significantly addressing the long–term dependency issue.
The rest of the paper is organized as follows: Section 2 describes the data acquisition and processing operations; Section 2.3 illustrates the experimental setup; Section 3 presents the results of our work, which are discussed in Section 4.

2 Materials and Methods

In this section, we will describe the data and the experimental methodology used in this work. In 2.1, we will describe the dataset and how it was built. In 2.2 we will describe the GNN model. In 2.3, we will explain our experimental procedures.

2.1 Dataset Construction

To collect our dataset, heterodimers (i.e. dimers formed by two different monomers) characterized by the absence of disulfide bridges, the presence of salt bridges, and protein–protein interaction sites were searched in the Protein DataBank in Europe (PDBePISA). We obtained a database of 6,695 known proteins for a total of 160,680 monomeric interfaces. To guarantee biological significance, some criteria were enforced: an area of at least 200 A^2, $\langle x, y, z\rangle$ symmetry, and only two interacting protein molecules. After this operation, we obtained a set of 12,455 interfaces. For every interface, two protein graphs were built, representing two polypeptide chains which interact on the binding site.
The monomeric graphs were built using VPLG [17], with *PDB* and *DSSP* [9] files representing the whole protein. Each node v is labeled with a feature vector $l(v)$ which consists of: an ID number, the SSE type, the number of occurrences of cysteine and that of the aromatic amino acids (tyrosine, tryptophan and phenylalanine), the percentage of amino acids taking part in the interface and the overall hydrophobicity [12], the charge and Accessible Surface Area (ASA) of the SSE, respectively as the sum of hydropathic indexes, charges and accessible surface areas of each amino acid at pH 7.
Once the graph has been produced for both monomers, it is possible to build the correspondence graph [7, 5]. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be the graphs representing two protein chains and $G = (V_G, E_G)$ be the correspondence graph of G_1 and G_2. Let $v_i, u_i \in V_i$ be two generic nodes in G_i with $i = 1, 2$. Therefore, two nodes $(v_1, v_2), (u_1, u_2) \in V_G$ are connected by an edge $e \in E_G$ if and only if $\exists(v_1, u_1) \in E_1$ and

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

$\exists(v_2, u_2) \in E_2$. The edge label $w\{e\}$ is a one–hot representation of the spatial relationship $w'\{e\}$ between two adjacent nodes in G, which depends on the labels $w\{(v_1, u_1)\}$ and $w\{(v_2, u_2)\}$, so that $w'\{e\}$ is the same edge label if both the edge labels in G_1 and G_2 are equal, *mixed* otherwise. The label of node $(v_1, v_2) \in V_G$ consists of: an ID number, a one–hot representation of the SSE type, the differences in the occurrences of cysteine and the aromatic amino acids, the arithmetic mean of the two hydrophobicity values, the minimum of the ASAs and the sum of the charges of the two SSEs. In particular, the SSE type of the node $v \in V_G$, which represents $v_1 \in V_1$ and $v_2 \in V_2$, is the same as that of the nodes v_1 and v_2 if both belong to the same SSE class, while it is defined as *mixed* if they belong to different SSE classes.
The node targets were generated with the Bron and Kerbosch algorithm [4], which identified the cliques within each correspondence graph, with a minimum size of three nodes. Subsequently, these cliques were analyzed, in order to determine whether or not they were biologically significant. In this context, a clique is defined as *positive* or *biologically significant* if and only if all the nodes belonging to that clique represent pairs of SSEs of different monomeric graphs, that contain both at least one residue that is part of the interface. Hence, the target attached to each node is a two–dimensional vector containing a one–hot coding of the two classes: *positive* if the node belongs to a biologically significant clique, *negative* otherwise.
We obtained 512 correspondence graphs, each containing at least one biologically significant clique (and any number of negative cliques) composed of three or more nodes. These were not completely connected, often being made of multiple separated connected components. Since many connected components did not include cliques, a pruning strategy was adopted, in order to clean the dataset. The correspondence graphs were split, obtaining a graph for each connected component. We kept only those which contain at least one clique, whether positive or not. This operation produced the final dataset of 1044 connected graphs, 537 of which contain a positive clique, while the remaining 507 contain only negative cliques. Table 1 provides numerical information on the dataset, before and after the pruning process.

Dataset	Graphs	Edges	Nodes	Nodes0	Nodes1	%Nodes1
Before Pruning	512	441.203	328.629	325.798	2.831	0.86 %
After Pruning	1.044	274.608	166.424	163.593	2.831	1.7 %

Table 1: Dataset statistics before and after data cleaning (Nodes0/1 represent negative/positive nodes)

2.2 Graph Neural Networks

Graph Neural Networks (GNNs) can process graph–structured data, calculating an output at each node, or on any subset of relevant nodes. GNNs create an *encoding network*, an architecture that replicates the structure of the input graph, using two Multi–Layer Perceptrons (MLPs) as building blocks. A state x_n is associated to each node n. One MLP calculates the state transition function f_w at each node, the other computes the output function g_w. The network unfolds itself in time and space, respectively by replicating the MLP units on each node of the input graph, and by iterating the state computations until a stable point is reached. At each iteration t, for each node n, the state $x_n(t) \in R^s$ is updated by f_w, depending on the node label and on the states of its neighbors at $t-1$. The information associated to each node can thus be propagated through the whole graph in a sufficient number of iterations. The output

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2.
Available from http://www.i6doc.com/en/.

function g_w, taking in input the final node state x_n^*, is then computed on the nodes for which output is required. More formally, the state updating process is described by Eq. (1), where $l_n \in R^q$ is the label attached to node n.

$$x_n(t) = \sum_{(n,v)\in E} f_w(l_n, x_v(t-1), l_v), \quad o_n = g_w(x_n^*, l_n) \tag{1}$$

As previously stated, the computation takes into account the neighborhood of n, defined by its edges $(n, v) \in E$. The summation in Eq. (1) allows us to deal with any number of neighbors without needing any order relationship among them.
A Layered Graph Neural Network (LGNN) is a stacked architecture in which each layer consists of a GNN. The first layer is a standard GNN operating on the original input graphs. Each layer after the first is trained on an enriched version of the graphs, in which the information obtained from the previous layer is concatenated to the original node labels. This additional information consists in, either, the node state, the node output, or both. Formally, the label of node n in the i–th layer, $i > 1$, is $l_n^i = [l_n, x_n^{i-1}]$ or $l_n^i = [l_n, x_n^{i-1}]$ or $l_n^i = [l_n, x_n^{i-1}, o_n^{i-1}]$, where x_n^{i-1}, o_n^{i-1} are, respectively, the state and the output of node n at layer $i-1$ of the cascaded architecture. This schema encourages each layer to refine the solution provided by the previous layers, improving the performance with respect to a single–layered GNN. LGNNs are trained step by step, one layer after the other, from the first to the last. Each layer is trained using the same original targets.

2.3 Experimental Setup

We developed a binary GNN classifier for the detection of maximum cliques in the correspondence graphs, which addresses the problem as a node–focused classification task. Usually, in a classification task, the performance is measured in terms of accuracy. This metric, though, is not precise on unbalanced datasets. Therefore, we decided to evaluate the performances of our model using the *F1–Score*, which combines precision and recall to provide a balanced measure.
The architecture of the MLP module dedicated to the output function g_w was kept fixed, using a single level MLP and the softmax activation function. On the contrary, a 10–fold cross–validation was performed in order to determine the best hyperparameters for the MLP implementing the state transition function f_w. According to the cross–validation results, the MLP architecture with better performance has got a single hidden–layer with logistic sigmoid activation functions. This setup was used also to test a 5–layered GNN network, where each GNN layer shares the same architecture.
In order to evaluate the performances of the LGNN, a 10–fold cross–validation was carried out again. The LGNN is composed of 5 GNN layers, with state dimension equal to 3. The state is calculated by a 1–layer MLP with logistic sigmoid activations, while the output is calculated with a 1–layer MLP with softmax activation. Since the negative/positive examples ratio is quite large, the weight of positive examples is fixed to the 10% of this ratio, against a weight of 1 for negative examples, in order to balance the learning procedure. The model is trained with Adam optimizer [10] and cross–entropy loss function.

3 Experimental Results

The best performance is obtained with LGNNs integrating only the state in the node labels. There are slight improvements in precision and more tangible improvements

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

in recall, which gains more than 10 percentage points in the second GNN level, and then continues to grow and stabilize in the following levels, as shown in Fig. 1. This architecture is the only one in which we observe a significant increase of the F1–Score, getting more than 6 percentage points from nearly 35% of the first GNN level to more than 40% in the final GNN level, as reported in Table 2.
Contrariwise, integrating in the node labels only the output or both the state and the output, the F1–score decreases through the LGNN layers. The other parameters remain almost stable, except for recall, which slightly increases among the LGNN strata. However, the standard deviation of the recall tends to grow, suffering from a marked dependence on the initial conditions of the experiment. The results confirm the expectations based on biological data and show good performances in determining the interaction sites, recognizing on average about 60% of the interacting nodes.

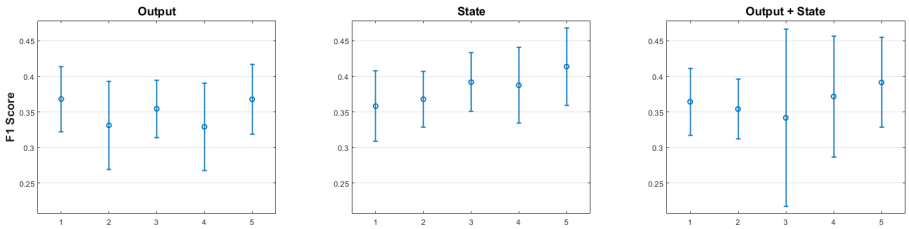

Fig. 1: 5 levels LGNN 10–fold cross validation results: F1–score

Output	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Precision	0.319(0.069)	0.271(0.058)	0.287(0.049)	0.266(0.046)	0.295(0.07)
Recall	0.455(0.048)	0.447(0.111)	0.476(0.061)	0.446(0.101)	0.517(0.059)
F–Score	0.368(0.046)	0.331(0.062)	0.354(0.04)	0.329(0.062)	0.368(0.049)

State	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Precision	0.31(0.061)	0.279(0.045)	0.322(0.052)	0.295(0.053)	0.328(0.061)
Recall	0.436(0.063)	0.558(0.056)	0.524(0.087)	0.585(0.08)	0.571(0.067)
F–Score	0.358(0.05)	0.368(0.039)	0.392(0.041)	0.387(0.053)	0.414(0.055)

Both	**Level 1**	**Level 2**	**Level 3**	**Level 4**	**Level 5**
Precision	0.308(0.063)	0.273(0.052)	0.261(0.106)	0.301(0.064)	0.296(0.06)
Recall	0.46(0.056)	0.544(0.109)	0.52(0.185)	0.518(0.168)	0.597(0.096)
F–Score	0.364(0.047)	0.354(0.042)	0.342(0.125)	0.372(0.085)	0.392(0.063)

Table 2: Results obtained with three different LGNN settings: propagating the output, the state or both from one layer to the next

4 Conclusions

We addressed the problem of protein–protein binding site detection as a search for the maximum clique in the interface correspondence graph. Although the problem is NP–complete, our method, based on GNNs, can find the maximum clique in affordable time. The performances of the model were measured in terms of F1–score and show that our approach is very promising, though it can be further improved. One key idea in this direction is that of using graphs in which the nodes correspond to single

ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 2-4 October 2020, i6doc.com publ., ISBN 978-2-87587-074-2. Available from http://www.i6doc.com/en/.

amino acids, rather than to SSEs. Although this latter approach would increase the complexity of the problem, it would avoid the loss of information we encounter in compressing amino acid features into SSE nodes. Moreover, predictions obtained in this setting would be more accurate describing the binding site at the amino acid level.

References

[1] N. Bandinelli, M. Bianchini, and F. Scarselli. Learning long-term dependencies using layered graph neural networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010.

[2] P. Battaglia et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[3] I. M. Bomze et al. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[4] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, Sept. 1973.

[5] E. J. Gardiner, P. J. Artymiuk, and P. Willett. Clique-detection algorithms for matching three-dimensional molecular structures. *Journal of Molecular Graphics and Modelling*, 15(4):245–253, 1997.

[6] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, July 2005.

[7] H. M. Grindley et al. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *Journal of Molecular Biology*, 229(3):707 – 721, 1993.

[8] H. Hegyi and M. Gerstein. The relationship between protein structure and function: a comprehensive survey with application to the yeast genome (edited by g. von heijne). *Journal of Molecular Biology*, 288(1):147 – 164, 1999.

[9] W. Kabsch and C. Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 1983.

[10] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[11] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[12] J. Kyte and R. Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157(1):105–132, 1982.

[13] Y. Li et al. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[14] A. Rossi et al. Inductive–transductive learning with graph neural networks. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 201–212. Springer, 2018.

[15] A. Santoro et al. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[16] F. Scarselli et al. The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.

[17] T. Schäfer, P. May, and I. Koch. Computation and Visualization of Protein Topology Graphs Including Ligand Information. In S. Böcker et al., editors, *German Conference on Bioinformatics 2012*, volume 26 of *OpenAccess Series in Informatics (OASIcs)*, pages 108–118, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[18] P. Veličković et al. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.