VARIATIONAL RECURRENT NEURAL NETWORKS FOR GRAPH CLASSIFICATION

Edouard Pineau[⋆†] **& Nathan de Lara**[⋆]
Telecom ParisTech[⋆], Safran[†]
{edouard.pineau,nathan.delara}@telecom-paristech.fr

ABSTRACT

We address the problem of graph classification based only on structural information. Inspired by natural language processing techniques (NLP), our model sequentially embeds information to estimate class membership probabilities. Besides, we experiment with NLP-like variational regularization techniques, making the model predict the next node in the sequence as it reads it. We experimentally show that our model achieves state-of-the-art classification results on several standard molecular datasets. Finally, we perform a qualitative analysis and give some insights on whether the node prediction helps the model better classify graphs.

1 INTRODUCTION

Many natural or synthetic systems have a natural graph representation where entities are described through their mutual connections: chemical compounds, social or biological networks, for example. Therefore, automatic mining of such structures is useful in a variety of applications.

Graph classification raises several difficulties to leverage standard machine learning algorithms. Indeed, most of these algorithms take vectors of fixed size as inputs. In the case of graphs, usual representations such as edge list or adjacency matrix do not match this constraint. The size of the representations is graph dependent (number of edges in the first case, number of nodes squared in the second) and these representations are index dependent: up to indexing of its nodes, a same graph admits several equivalent representations. In a classification task, the label of a graph is independent from the indices of its nodes, so the model used for prediction should be invariant to node ordering as well. Handling discrete inputs with variable size and ordering is a well known problem in natural language processing (NLP). This is why we adapt NLP techniques to tackle graph classification.

In this paper, we propose a method to sequentially embed graph information in order to perform classification. By construction, this recurrent graph classifier overcomes the common difficulties listed above. Besides, we propose to use an additional node prediction block to help the model to capture the intrinsic structure of the graphs. The complete model is denoted *variational recurrent graph classifier* (VRGC). Experiments show that this leads to better classification results for larger datasets. For clarity of the contribution of our work, we use neither node attributes nor edge attributes.

Related works use either graph kernels (Nikolentzos et al., 2017a;b; 2018; Neumann et al., 2016; Shervashidze et al., 2011; Yanardag and Vishwanathan, 2015), sequential methods (Callut et al., 2008; Xu et al., 2012; Jin and JaJa, 2018; You et al., 2018) or graph features (Barnett et al., 2016; Narayanan et al., 2017; Gomez et al., 2017; Dutta and Sahbi, 2017).

2 MODEL

We propose to use a sequential approach to embed graphs with a variable number of nodes and edges into a vector space of a chosen dimension. This latent representation is then used for classification. Node index invariance is approximated through specific pre-processing and aggregation.

Let $G = (V, E)$ be an undirected and unweighted graph with V a set of nodes and E a set of edges. The graph G can be represented, modulo any permutation π over its nodes $\{n_i\}_{i=1}^{|V|}$, by its boolean

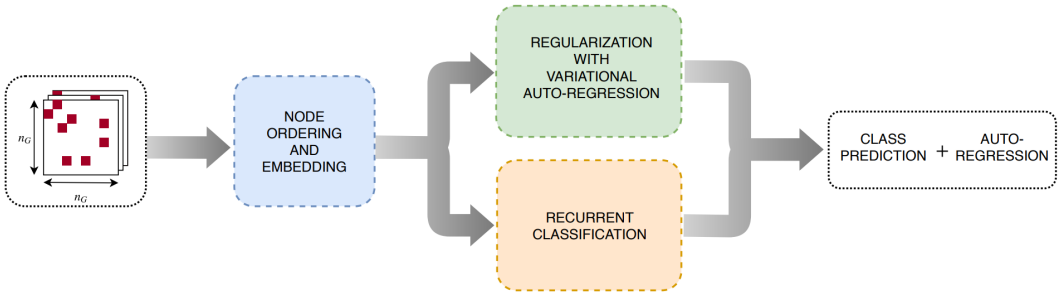

Figure 1: Macroscopic representation of VRGC.

adjacency matrix A^π such that $A^\pi_{ij} = 1$ if nodes indexed by i and j are connected in the graph and $A^\pi_{ij} = 0$ otherwise. We use this adjacency matrix as a raw representation of the graph.

Our VRGC is composed of three main parts: node ordering and embedding, classification and regularization with variational auto-regression (VAR). See Figure 1 for an illustration.

Node ordering and embedding Before being processed by the neural network, the adjacency matrix of a graph is transformed on-the-fly (You et al., 2018). First, a node is selected at random and used as root for a breadth first search (BFS) over the graph. The rows and columns of the adjacency matrix are then reordered according to the sequence of nodes returned by the BFS. Next, each row i (corresponding to the i^{th} node in the BFS ordering) is truncated to keep only the connections of node n_i with the $\min(i, d)$ nodes that preceded in the BFS. This way, each node is d-dimensional, and each truncated matrix is zero-padded in order to have dimensions $(|V|_{max}, d_n)$. Throughout the rest of the paper, we use the notation n_G for $|V|_{max}$.

After node ordering and pre-embedding, each graph is processed as a sequence of d-dimensional nodes by a gated recurrent unit (GRU) neural network (Cho et al., 2014). The GRU is a special RNN able to learn long term dependencies by solving vanishing gradient effect[1]. In order to help the recurrent network training, we propose to add a simple fully connected network between pre-embedding and recurrent embedding. Therefore, the node will be presented to the GRU in the shape of continuous vectors instead of binary adjacency vectors.

Finally the GRU sequentially embeds each node n_i by using n_{i-1} and information contained in a memory cell h_{i-1} that theoretically embeds all previously seen information. The embedded node sequence $\{h_i\}_{i=1}^{n_G}$ then feeds both the VAR and the classifier as discussed in subsequent sections. See top line of Figure 2.

Classification After the embedding step, we use an additional GRU dedicated to classification that takes $\{h_i\}_{i=1}^{n_G}$ as input. Its last memory cell, denoted $\tilde{h}_{n_G}$, feeds a softmax multilayer perceptron (MLP) which performs class prediction. Formally, let c be the class index, the classifier is trained by minimizing the cross-entropy loss between ground-truth and $\hat{p}(G, r)$ the softmax class membership probability vector for a given graph G that has been sorted by a BFS rooted with node r. We call this objective term $\mathcal{L}_{classif}$. As discriminating patterns might be spread across the whole graph, the network is required to model long-term dependencies. By construction, GRUs have such ability. See middle line of Figure 2.

Regularization with variational auto-regression As the structure of a graph is the concatenation of the interactions between all nodes and their respective neighbors, learning a good representation without using node attributes requires for the model to capture the structure of the graph while classifying. Accordingly, we add an auto-regression block to our model: at each node, the network makes a prediction for the next node adjacency. Multi-task learning is a powerful leverage to learn rich representation (Sanh et al., 2018) in NLP. In particular, such representation for sequence classification has already been used for sentiment analysis (Latif et al., 2017; Xu et al., 2017).

[1]The choice of GRU over Long Short Term Memory networks is arbitrary as they have equivalent long-term modeling power (Chung et al., 2014).

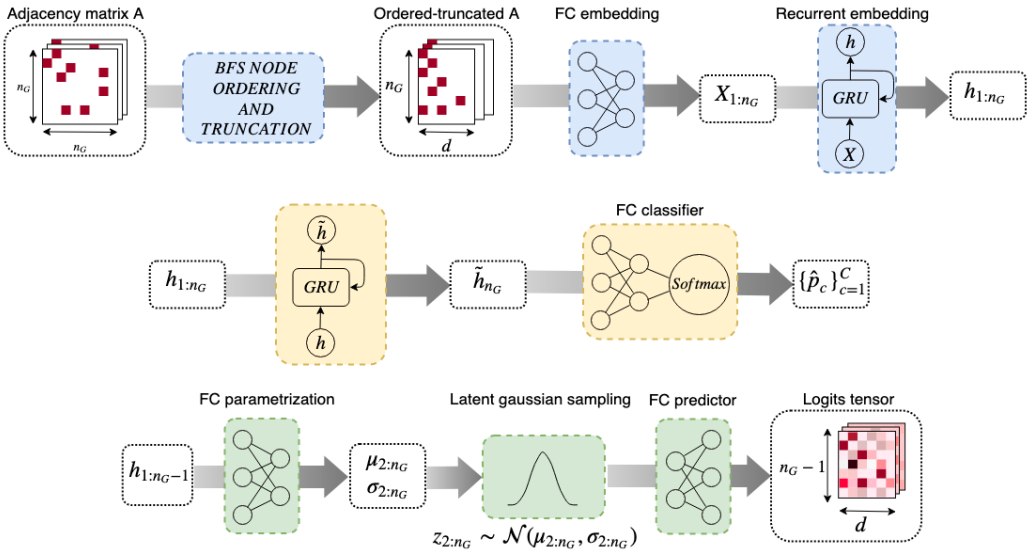

Figure 2: Architecture for VRGC. Top: node ordering and embedding. Middle: classification. Bottom: regularization with VAR plus final aggregation. FC stands for fully-connected.

We use a variational auto-encoder (VAE) (Kingma and Welling, 2013) to learn a representation of each node n_i given h_{i-1}. The first layers of the encoder are shared with the classifier and corresponds to the graphs preprocessing (blue part in Figures 1 and 2). The subsequent encoder layers, the latent sampling and the decoder constitute the VAR. For each graph G with embedded nodes $\{n_i\}_{i\in[\![1,n_G]\!]}$, the fully connected variational auto-encoder takes $h := \{h_i\}_{i\in[\![1,n_G-1]\!]}$ as input. Let $\{z_i\}_{i\in[\![1,n_G]\!]}$ be the latent random variables for the following model

$$p(G,z|h) = p(n_1,z_1)\prod_{i=2}^{n_G} p_\theta(n_i|z_i)q_\phi(z_i|h_{i-1}).$$

In practice, p_θ and q_ϕ are modelled by neural networks parametrized by θ and ϕ, which require differentiable functions for training. However, $p_\theta(n_i|z_i)$ models a binary adjacency vector representing the connections between node n_i and previously visited nodes $n_{j<i}$. Therefore, we use sigmoid continuous relaxation to train our model, and hard binary sampling at test time. We use a Gaussian variational posterior distribution. Training is done by maximizing the variational lower bound of the log-likelihood of the observation as in Kingma's VAE. The exact loss is displayed in Appendix A and denoted $\mathcal{L}_{pred}$.

The regularization part is illustrated in the bottom line of Figure 2. In the end, the model is trained by minimizing the total loss $\mathcal{L} = \mathcal{L}_{classif} + \alpha\mathcal{L}_{pred}$, where α is a hyper-parameter.

Aggregation of the results at test time The node ordering step introduces randomness to our model. On the one hand, it helps learn more general graph representations during the training phase, but on the other hand, it might produce different outputs for the same graph during the testing phase, depending on the root of the BFS. In order to counter this side effect, we add the following aggregation step for the testing phase. Each graph is processed N times by the model with N different roots for BFS ordering. The N class membership probability vectors are extracted and averaged. The average score vector is noted $\bar{p}$ and computed as follows with an element-wise sum:

$$\bar{p}(G) = \frac{1}{N}\sum_{\substack{i=1\\ r\sim\mathcal{U}([\![1,n_G]\!])}}^{N} \hat{p}(G,r).$$

	MT	EZ	PF	NCI1
EMD	86.1	36.8	-	72.7
PM	85.6	28.2	-	69.7
FB	84.7	29.0	70.0	62.9
DyF	86.3	26.6	73.1	66.6
SGE	87.3	40.7	71.9	-
TLS	88.4	43.7	73.6	75.2
FGSD	**92.1**	-	73.4	79.8
RGC	89.5	**48.7**	72.5	78.1
VRGC	86.3	48.4	**74.8**	**80.7**

Table 1: Experimental results of different models plus our own on four standard molecular datasets. RGC stands for recurrent graph classifier, VRGC for variational RGC. All other acronyms are defined in section 3.

This soft vote is repeated K times resulting in K probability vectors $\{\bar{p}_{.,k}(G)\}_{k=1}^{K}$ for each graph G. The final class attributed to a graph corresponds to the highest probability among the K vectors. This second hard vote enables to choose the batch of votes for which the model is the most confident.

$$\hat{c}(G) = \arg \max_{c \in [\![1,C]\!]} \{||\bar{p}_{c,.}(G)||_\infty\}$$

3 EXPERIMENTS

Datasets and results We evaluated our model against four standard datasets from biology: Mutag (MT), Enzymes (EZ), Proteins Full (PF) and National Cancer Institute (NCI1) (Kersting et al., 2016). A detailed description of each dataset in provided in Appendix B.

We compare our results to those obtained by Earth Mover's Distance (Nikolentzos et al., 2017b) (EMD), Pyramid Match (Nikolentzos et al., 2017b) (PM), Feature-Based (Barnett et al., 2016) (FB), Dynamic-Based Features (Gomez et al., 2017) (DyF), Stochastic Graphlet Embedding (Dutta and Sahbi, 2017) (SGE), Truncated Laplacian Spectrum (TLS) (de Lara and Pineau, 2018) and Family of Graph Spectral Distances (FGSD) (Verma and Zhang, 2017). All values are directly taken from the aforementioned papers as they use a setup similar to ours. For algorithms presenting results with and without node features, we reported the results without node features. For those presenting results with several sets of hyper-parameters, we reported the results for the parameters that performed best on the largest number of datasets. Results are reported in Table 1. We obtain state-of-the-art results on three out of the four datasets used for this paper and the second best result on the fourth one.

Node indexing invariance Our model is designed to be independent from node ordering of the graph with respect to different BFS roots. Inputs representing the same graph (up to node ordering) should be close from one another in the latent embedding space. As the preprocessing is performed on each graph at each epoch, a same graph is processed many times by the model during training with different embeddings. This creates a natural regularization for the network. Indeed, as illustrated in Figure 3, the projections corresponding to the same graphs form a heap in the low dimensional representation of the latent space.

Contribution of the VAR to classification The variational regularization term seems to help the model finding a more meaningful latent representation for classification while graph dataset becomes larger. Note that the extra cost of training the VAR is marginal with respect to the training of the RNNs. We provide an illustration of the output of VAR block in Figure 4.

Conclusion and room for improvement This paper proposed a recurrent graph classifier with variational regularization. The invariance to node indexing is greedy learned from numerous iterations on randomly rooted BFS-ordered graph. For future work, we should investigate the impact VAR capacity (number and size of hidden layers) on classification accuracy or generalization.

Acknowledgments We would like to thank Thomas Bonald and Sebastien Razakarivony for their comments and help. We also would like to thank NVIDIA and its GPU Grant Program for providing the hardware we used in our experiments. This work is supported by the company Safran through the CIFRE convention 2017/1317.

REFERENCES

I. Barnett, N. Malik, M. L. Kuijjer, P. J. Mucha, and J.-P. Onnela. Feature-based classification of networks. *arXiv preprint arXiv:1610.05868*, 2016.

J. Callut, K. Françoisse, M. Saerens, and P. Dupont. Classification in graphs using discriminative random walks, 2008.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

N. de Lara and E. Pineau. A simple baseline algorithm for graph classification. *Relational Representation Learning Workshops (NIPS 2018)*, 2018.

A. Dutta and H. Sahbi. High order stochastic graphlet embedding for graph-based pattern recognition. *arXiv preprint arXiv:1702.00156*, 2017.

L. G. Gomez, B. Chiem, and J.-C. Delvenne. Dynamics based features for graph classification. *arXiv preprint arXiv:1705.10817*, 2017.

Y. Jin and J. F. JaJa. Learning graph-level representations with gated recurrent neural networks. *arXiv preprint arXiv:1805.07683*, 2018.

K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann. Benchmark data sets for graph kernels, 2016. `http://graphkernels.cs.tu-dortmund.de`.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

S. Latif, R. Rana, J. Qadir, and J. Epps. Variational autoencoders for learning latent representations of speech emotion. *arXiv preprint arXiv:1712.08708*, 2017.

A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs. *CoRR*, abs/1707.05005, 2017. URL `http://arxiv.org/abs/1707.05005`.

M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

G. Nikolentzos, P. Meladianos, A. J.-P. Tixier, K. Skianis, and M. Vazirgiannis. Kernel graph convolutional neural networks. *arXiv preprint arXiv:1710.10689*, 2017a.

G. Nikolentzos, P. Meladianos, and M. Vazirgiannis. Matching node embeddings for graph similarity. In *AAAI*, pages 2429–2435, 2017b.

G. Nikolentzos, P. Meladianos, S. Limnios, and M. Vazirgiannis. A degeneracy framework for graph similarity. In *IJCAI*, pages 2595–2601, 2018.

A. Paszke, S. Gross, S. Chintala, and G. Chanan. Pytorch, 2017.

V. Sanh, T. Wolf, and S. Ruder. A hierarchical multi-task approach for learning embeddings from semantic tasks. *arXiv preprint arXiv:1811.06031*, 2018.

N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

S. Verma and Z.-L. Zhang. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, pages 88–98, 2017.

W. Xu, H. Sun, C. Deng, and Y. Tan. Variational autoencoder for semi-supervised text classification. In *AAAI*, pages 3358–3364, 2017.

X. Xu, L. Lu, P. He, Z. Pan, and C. Jing. Protein classification using random walk on graph. In *International Conference on Intelligent Computing*, pages 180–184. Springer, 2012.

P. Yanardag and S. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM, 2015.

J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec. Graphrnn: A deep generative model for graphs. *arXiv preprint arXiv:1802.08773*, 2018.

A VAR LOSS

The VAE-like loss for VAR regularization is the following:

$$\mathcal{L}_{pred} = \mathbb{E}_{p_d(G)}\left[\sum_{i=2}^{n_G} \mathrm{KL}\left(q_\phi(z_i|h_{i-1})||q(z_i)\right)\right] - \mathbb{E}_{p_d(G)}\left[\sum_{i=2}^{n_G} \mathbb{E}_{q_\phi(z_i|h_{i-1})}\left[\log p_\theta(n_i|z_i)\right]\right],$$

which is a lower bound of the negative marginal log-likelihood $\mathbb{E}_{p_d(G)}\left[\log p_\theta(G)\right]$. p_θ and q_ϕ are the respective densities of $n|z$ and $z|h$, whose distribution are parameterized by θ and ϕ respectively. KL denotes the Kullback-Leibler divergence, p_d is the empirical distribution of G and $q(z_i)$ is the density of the prior distribution of latent variables $\{z_i\}_{i=2}^{n_G}$. We chose the standard Gaussian prior for $q(z_i)$.

B DATASET CHARACTERISTICS

All graphs represent chemical compounds, nodes are molecular substructures (typically atoms) and edges represent connections between these substructures (chemical bound or spatial proximity). In MT, the compounds are either mutagenic or not mutagenic. EZ contains tertiary structures of proteins from the 6 Enzyme Commission top level classes; it is the only multiclass dataset of this paper. PF is a subset of the Dobson and Doig dataset representing secondary structures of proteins being either enzyme or not enzyme. In NCI1, compounds either have an anti-cancer activity or do not.

	MT	EZ	PF	NCI1
# graphs	188	600	1113	4110
# classes	2	6	2	2
bias	0.66	0.17	0.60	0.5
avg. |V|	18	33	39	30
min |V| / max |V|	10/28	2/125	4/620	3/106
avg. |E|	39	124	146	64.6

Table 2: Basic characteristics of the datasets. Bias indicates the proportion of the largest class.

C FEATURES OF NETWORK ARCHITECTURE

MT, EZ, PF and NCI1 are respectively divided into 10 folds such that the class proportions are preserved in each fold for all datasets. These folds are then used for cross-validation i.e., one fold serves as the testing set while the other ones compose the training set. Results are averaged over all testing sets. Our model is implemented in Pytorch (Paszke et al., 2017) and trained with the Adam stochastic optimization method (Kingma and Ba, 2014) on a NVIDIA TitanXp GPU.

The input size d_n of the recurrent neural network is chosen for each dataset according to the algorithm described in (You et al., 2018), namely 11 for MT, 25 for EZ, 80 for PF and 11 for NCI1. α is set to 0.1. For training, batch size is set to 64, and the learning rate to 10^{-3}, decreased by 0.3 at iterations 400 and 1000. We use the same hyper-parameters for every dataset.

D ILLUSTRATION OF EXPERIMENTAL RESULTS

The following table presents the hyperparametrization of our model, i.e. the neural network architectures for each part presented in Section 2.

Step	**Architecture**
BFS embedding	1-layer FC. $d_n \times 64$ 2-layer GRU. 64×128
VAR	Encoder 1-layer FC. $128 \times 2 \times 8$ Gaussian sampling Predictor 2-layer ReLU FC. $8 \times d_n$
Classifier	2-layer GRU. 128×128 + DP(0.25) 2-layer ReLU FC. $128 \times C$ + SF

Table 3: Generic architecture used in our experiments. ReLU FC stands for fully-connected network with ReLU activation. DP stands for dropout. SF stands for softmax.

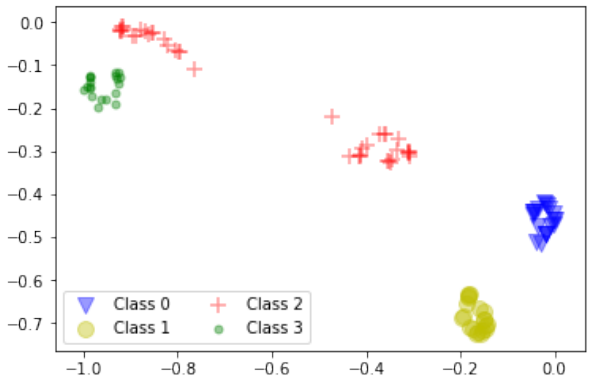

Figure 3: TSNE projection of the latent state preceding classification for five graphs of EZ each initiated with 20 different BFS. Colors and markers represent the respective classes of the graphs.

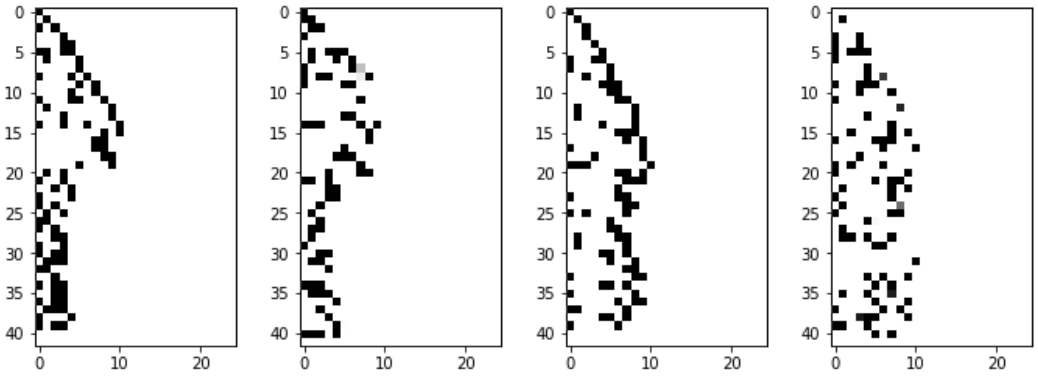

Figure 4: Left: Representation of the same graph after two differently rooted BFS ordering and truncation. Right: corresponding auto-regressions.