2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

DMNP: A Deep Learning Approach for Missing Node Prediction in Partially Observed Graphs

Faezeh Faez[†], Ali Akhoondian Amiri[‡], Mahdieh Soleymani Baghshah[§] and Hamid R. Rabiee[¶]
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
Email: [†]faezeh.faez@gmail.com, [‡]akhoondian@ce.sharif.edu, [§]soleymani@sharif.edu, [¶]rabiee@sharif.edu

***Abstract*—Missing data is unavoidable in graphs, which can significantly affect the accuracy of downstream tasks. Many methods have been proposed to mitigate missing data in partially observed graphs. Most of these approaches assume they have complete access to graph nodes and only focus on recovering missing links, while in practice a part of the graph nodes can also be out of access. This work presents Deep Missing Node Predictor (DMNP), a novel deep learning-based approach to recovering missing nodes in partly observed graphs. Our proposed approach does not rely on additional information that in many cases does not exist. We compare our model with graph completion and deep graph generation baselines. The experimental results show that the DMNP model outperforms previous state-of-the-art approaches.**

***Index Terms*—Missing Node Prediction, Partially Observed Graphs, Deep Graph Generation**

I. Introduction

Graph data structures are ubiquitous in many data analysis areas, making it possible to take into account the relationships between data entities. Almost all graph processing approaches need complete data to perform at their best. However, in most cases, parts of the data might be missing. Therefore, many methods have been proposed to recover missing parts of partially observed graphs. The majority of these methods can only predict missing links between fully observed nodes. However, an intrinsically more complicated challenge arises when some graph nodes are also missing.

Graph generation refers to a research field that aims to generate new graph samples, which dates back to several decades ago, and has recently been resurrected thanks to the advancement of deep learning techniques. Modern deep graph generators, in contrast to the traditional approaches, can learn the graph generation process directly from data, making them more effective to capture complicated dependencies in graphs.

In this paper, we propose a novel deep learning model named Deep Missing Node Predictor (DMNP) to predict missing nodes in partially observed graphs. The core idea of DMNP is to train a deep autoregressive graph generative model that takes an initial graph as input and generates new nodes and their associated edges on top of it. To do so, we first propose the DMNP-S model that adds one new node to the initial partially observed graph. The DMNP-M model then generalizes the DMNP-S to recover an arbitrary number of missing nodes. This paper brings the following main contributions:

- It proposes a novel deep graph generation model named DMNP to predict missing nodes in partially observed graphs.
- The DMNP approach only makes use of structural information of partially observed graphs (i.e., how the available nodes are linked) and does not rely on the existence of additional information, which causes it to be applicable in cases where such information is not available.
- Experiments are conducted on graph completion, as well as deep graph generation baselines. The results demonstrate the superiority of our proposed DMNP model against competitors in both categories.

II. Related Work

Here, we review the previous methods in two different research fields.

Graph Reconstruction. Most of the existing methods can broadly be classified into two categories: link prediction and missing/hidden node detection. A significant number of proposed approaches belong to the former category, making the simplistic assumption that they have access to all the graph nodes and they only intend to recover the missing links. The latter category, on the other hand, is relatively less explored due to its complexity. Kim and Leskovec [1], Sina et al. [2], and Masrour et al. [3] propose approaches addressing the problem of missing node prediction. However, most of them rely heavily on the existence of additional information besides the adjacency matrix, making them less efficient or even useless in the absence of such information. In contrast, our proposed DMNP method completes the network structure without any prerequisite for the existence of side information.

Graph Generation. Modern graph generation approaches, unlike their traditional counterparts, can learn the generation process directly from data [4]. We divide these approaches into non-sequential and sequential categories. The methods belonging to the first category generate the entire graph simultaneously. Simonovsky and Komodakis [5] and Li et al. [6] have proposed prominent examples of such methods. The methods falling under the second category, on the other hand, adopt stepwise generation strategies. These approaches are the most relevant to our research. DeepGMG [7] is one of the initial steps in this direction, which is followed by more mature approaches. In this regard, GraphRNN [8] is an autoregressive model that maximizes a lower bound of the data distribution

IEEE/ACM ASONAM 2022, November 10-13, 2022
978-1-6654-5661-6/22/$31.00 É 2022 IEEE

using recurrent neural networks. GRAN [9] tries to better capture the autoregressive conditioning between the already-generated and to-be-generated parts of the graph using GNNs with attention. More recently, CCGG [10] makes the GRAN model class-conditional, enabling it to generate graphs of desired classes.

III. Problem Definition

An initial graph is represented by $g_0 = (v_0, e_0)$, where v_0 and e_0 are its node and edge sets, respectively. Given the g_0, we intend to generate graph $g_n = (v_n, e_n)$ by adding n new nodes and their corresponding edges to the initial graph such that $e_0 \subset e_n$, $v_0 \subset v_n$, and $|v_n - v_0| = n$. To do so, we first estimate the conditional distribution $p(G_n|g_0)$ in a stepwise manner, where in each step a new node is added to the graph. In this regard, the conditional distribution $p(G_n|g_0)$ can be expanded as follows:

$$\begin{aligned} p(G_n|g_0) &= p(N_1|g_0) \times p(N_2|g_0, N_1) \times ... \times p(N_n|g_0, N_1, ..., N_{n-1}) \\ &= p(N_1|g_0) \times \prod_{i=2}^{n} p(N_i|g_0, N_1, ..., N_{i-1}) \end{aligned} \tag{1}$$

where $G_n = g_0 \cup N_1 \cup N_2 \cup ... \cup N_n$, and N_i denotes the random variable corresponding to the i-th new node and its connections to the graph generated at the $(i-1)$-th step. At the inference time, we construct graph g_n on top of the initial graph g_0 by sampling from the estimated distribution $p(G_n|g_0)$. An example showing the generation procedure of our proposed DMNP method is presented in Fig. 1.

Here we use capital letters to denote random variables. Small letters, on the other hand, represent the realization of random variables. For example, G_n is the random variable corresponding to the graphs obtained after n steps of graph generation, and g_n is a realization of G_n.

IV. DMNP: Deep Missing Node Predictor

To address the problem, we first propose DMNP-S ("S" for "single node") to solve the problem when $n = 1$. Then we generalize DMNP-S by proposing DMNP-M ("M" for "multiple nodes") for an arbitrary value of n.

DMNP-S. We explain the training workflow of the DMNP-S approach in the following steps. Moreover, Fig. 2 (a) provides an overview of it.

Step 1. We randomly select a node from the input graph as the missing node and retain its true links to be used later as the ground truth labels for computing the loss function.

Step 2. As an initial guess, we add all the potential links between the missing node and the existing ones to build an augmented network, making it possible to attain an initial representation of the missing node. This initial guess will be refined in the following steps.

Step 3. The augmented graph is passed into a 2-layer Graph Convolutional Network (GCN) to get representations of its nodes.

Step 4. The representation of the newly added (missing) node is concatenated with each of the remaining nodes.

Step 5. As the refinement of our initial guess, the concatenated representations are passed into a multilayer perceptron (MLP) to determine which links genuinely exist.

Step 6. To train the model parameters, we use the binary cross-entropy loss function, as formulated bellow:

$$\mathcal{L}(\theta) = -\frac{1}{M-1} \sum_{i=1}^{M-1} y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \tag{2}$$

where θ denotes the model parameters, M is the maximum graph size in the dataset, p_i indicates the probability of edge existence between the missing node and the i-th remaining node, and $y_i \in \{0, 1\}$ is the corresponding ground truth label.

Fig. 2 (b) illustrates the model at inference time, where it adds one new node to the input graph.

DMNP-M. We further solve the problem when $n > 1$. In this case, for model training, we should consider the fact that any order of inserting new nodes that leads to generating a

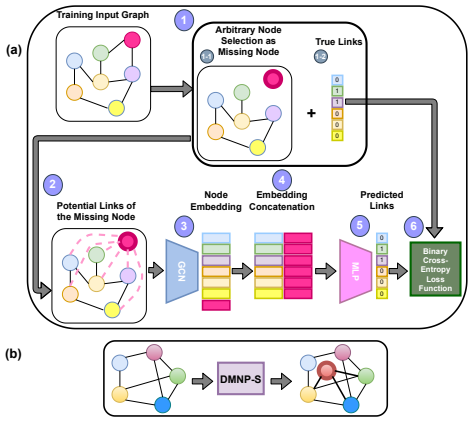

Fig. 2: An overview of the proposed DMNP-S method. (a) Training phase. (b) Inference phase.

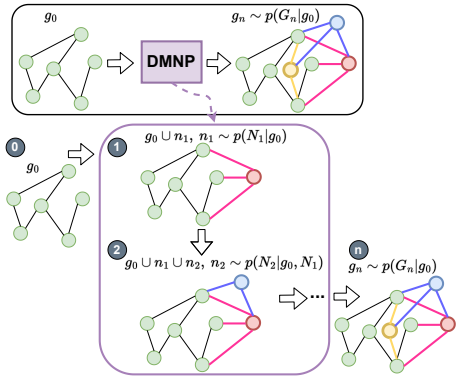

Fig. 1: An example illustrating the input, the output, and the stepwise generation procedure of our DMNP model.

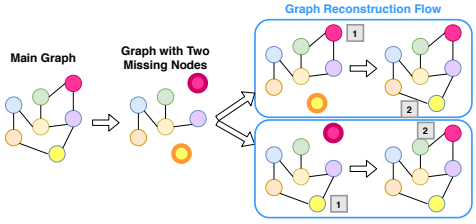

Fig. 3: An example showing the order of inserting new nodes.

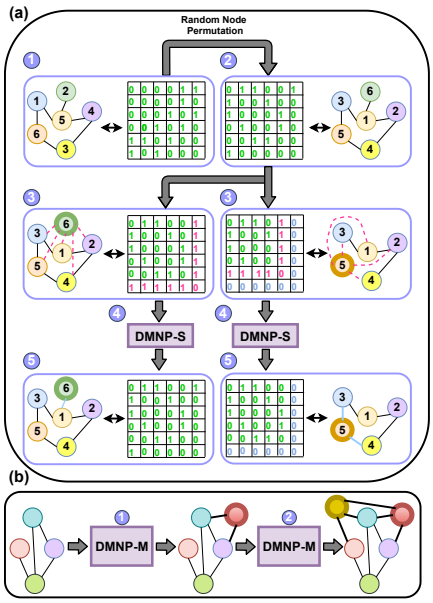

Fig. 4: An example of the DMNP-M model. (a) Training phase. 1) Input graph. 2) Random node permutation. 3) In this example $n = 2$, and we train the model to add the last two nodes: on the right, the model learns to add the 5-th node, and on the left the same process is in progress for the 6-th node. 4) The augmented matrices are passed into the DMNP-S. 5) The resulting graphs. (b) Inference phase.

valid graph is acceptable. In other words, the model should not be penalized for not following a specific node insertion ordering. Fig. 3 exemplifies this, where whether the model first inserts the red node and then the yellow one, or it does the reverse, both lead to the same resulting graph. To generalize the DMNP-S, we proposed a novel training technique called DMNP-M. Fig. 4 depicts an example of the DMNP-M in the training and inference phases. First, to train the model, we randomly permute the node ordering of the input adjacency matrix. This way, all the graph nodes get the chance to be in the set of the last nodes. Then, we randomly pick a number i between 1 and n. Next, the value of matrix cells with the column or row indices greater than $N - i + 1$ is set to zero, where N is the number of nodes in the graph g_n. It is done as we want to teach the model how to generate the (i)-th new node. Thus we connect this node to all other nodes and pass the resulting matrix to the DMNP-S model to refine our initial guess. By training the model in this way, it would be able to generate new nodes in a stepwise manner at the inference time.

V. EXPERIMENTS AND RESULTS

We compare our proposed DMNP model with both graph completion and deep graph generation baselines on the task of missing node prediction in partially observed graphs.

A. Datasets

We perform experiments on five datasets including three social network datasets (COLLAB, IMDB-BINARY, and IMDB-MULTI), as well as two bioinformatics datasets (ENZYMES and PROTEINS) [11]. Since some of the baselines are limited by the size of graphs, we performed the experiments on three subsets of datasets whose statistics are listed in Table I. In this regard, Dataset 1 is the most comprehensive one. In contrast, graphs of Dataset 2 are limited to those with smaller sizes (i.e., number of nodes), making it possible to compare with the baseline that suffers from scalability issues. Moreover, Dataset 3 consists of graphs with sizes that are powers of two.

B. Baseline Methods and Evaluation Metrics

We assess the performance of our model against one well-known graph completion approach, namely the KronEM [1]. As this method can only generate graphs whose number of nodes is a power of two, to ensure a fair comparison, we use Dataset 3 for comparing our method and other baselines with KronEM. Deep graph generators are the second category of methods with which we compare our model. More specifically, we compare against GraphRNN-S [8], GraphRNN [8], and DeepGMG[7]. Considering that the last approach suffers from scalability constraints, we use Dataset 2, as its graphs are relatively small. Furthermore, to assess how well the missing nodes' connections (i.e., their edges) are recovered, we report the results in terms of precision, recall, F1-score, AUC-ROC, and MAE.

C. Implementation Details

In all experiments, we use a batch size of 32 and train our model using the Adam optimizer with a learning rate of 0.01 for 200 epochs. We also use a 2-layer GCN with its first and second embedding dimensions set to 32 and 16, respectively. A 2-layer multilayer perceptron followed by a sigmoid activation function is also used to predict links for each new node. Moreover, 80% of the data is used for model training and the remaining 20% constitutes the test set.

D. Results

We perform several experiments for different values of the parameter n. Regarding this, the experimental results of our proposed DMNP model against all state-of-the-art deep graph generation baselines on Dataset 2 are reported in Table II, where it is evident that DMNP outperforms baselines in most cases. This is because our proposed model, unlike these baselines, is specifically designed to solve the problem of missing node prediction in partially observed graphs. Besides, the obtained results in terms of the F1-score and as a function of the parameter n are reported in Figures 5, 6, and 7 for Datasets 1, 2, and 3, respectively. These results also indicate

TABLE I: Datasets Statistics

		Range # Nodes	Avg. # Nodes	Std. # Nodes	# Graphs	Avg. Sparsity
Dataset 1	IMDB-BINARY	12-40	18.41	6.59	963	0.5
	IMDB-MULTI	7-40	12.25	6.54	1471	0.3
	COLLAB	42-51	46.05	2.74	992	0.46
	ENZYMES	10-40	26.63	8.35	426	0.83
	PROTEINS	4-40	20.49	9.82	794	0.75
Dataset 2	IMDB-BINARY	12-15	13.28	1.18	446	0.43
	IMDB-MULTI	7-15	9.43	2.37	1160	0.26
	ENZYMES	10-20	15.82	3.06	109	0.74
	PROTEINS	4-15	9.9	3.32	280	0.58
Dataset 3	IMDB-BINARY	16-64	18.62	7.83	61	0.61
	IMDB-MULTI	8-32	10.23	5.95	240	0.22
	COLLAB	32-128	37.4	14.38	296	0.38
	PROTEINS	8-64	20.68	14.76	77	0.74

TABLE II: Results on Dataset 2 for $n = 1$.

		Precision	Recall	F1	AUC-ROC	MAE
IMDB-BINARY	GraphRNN-S	0.027	0.016	0.02	0.687	0.51
	GraphRNN	0.007	0.006	0.007	0.66	0.525
	DeepGMG	0.234	0.256	0.244	0.537	0.472
	DMNP-S	**0.638**	**0.792**	**0.707**	**0.739**	**0.329**
IMDB-MULTI	GraphRNN-S	0.003	0.002	0.002	0.87	0.733
	GraphRNN	0.008	0.007	0.008	0.872	0.733
	DeepGMG	0.454	0.342	0.391	0.625	0.514
	DMNP-S	**0.84**	**0.977**	**0.903**	**0.911**	**0.114**
ENZYMES	GraphRNN-S	0.26	0.194	0.222	0.587	0.256
	GraphRNN	0.231	0.174	0.199	0.584	0.253
	DeepGMG	**0.564**	0.451	0.501	0.672	**0.172**
	DMNP-S	0.396	**0.842**	**0.539**	**0.838**	0.311
PROTEINS	GraphRNN-S	0.275	0.117	0.164	0.673	0.356
	GraphRNN	0.425	0.391	0.407	0.678	**0.256**
	DeepGMG	0.494	0.386	0.433	0.622	0.302
	DMNP-S	**0.527**	**0.911**	**0.668**	**0.883**	0.268

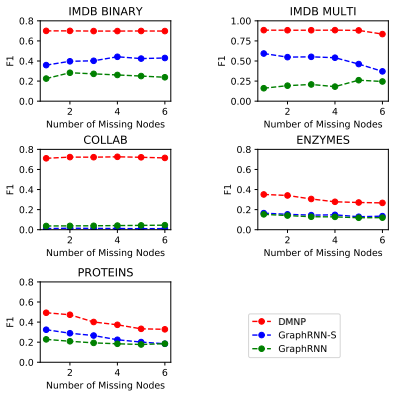

Fig. 5: Results on Dataset 1 as a function of the parameter n.

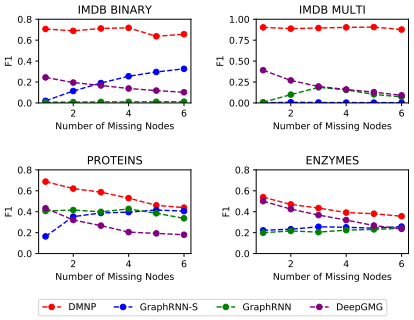

Fig. 6: Results on Dataset 2 as a function of the parameter n.

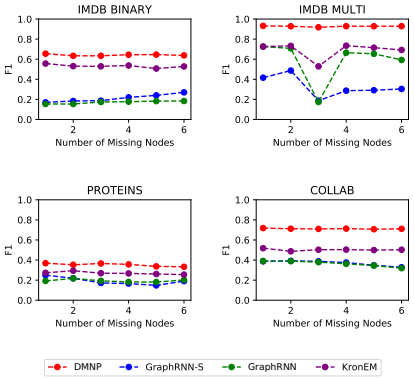

Fig. 7: Results on Dataset 3 as a function of the parameter n.

that the DMNP model outperforms both graph completion and deep graph generation competitors. Due to space limitations, here we only report the results in terms of the F1-score. The full results can be accessed via this link.

VI. CONCLUSION AND FUTURE WORK

In this work, we have addressed the problem of missing node prediction in partially observed graphs by proposing a deep learning-based graph generative approach named DMNP. Our model has two versions, namely DMNP-S and DMNP-M, where the former solves the problem when n (i.e., number of missing nodes) is set to 1, while the latter generalizes the DMNP-S model for an arbitrary value of the parameter n. Experimental results demonstrate the superiority of our approach over both graph completion and deep graph generation baselines. As a future extension to this work, we plan to add an attention mechanism to the DMNP-S model to make its initial guess more precise. We also intend to enhance the model architecture in a way that the DMNP-M can better capture the generating history, thereby increasing its effectiveness.

ACKNOWLEDGMENT

The authors thank Maryam Ramezani and Mahsa Ghorbani for their support and helpful comments on the paper.

REFERENCES

[1] Myunghwan Kim and Jure Leskovec. The network completion problem: Inferring missing nodes and edges in networks. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 47–58. SIAM, 2011.

[2] Sigal Sina, Avi Rosenfeld, and Sarit Kraus. Sami: an algorithm for solving the missing node problem using structure and attribute information. *Social Network Analysis and Mining*, 5(1):54, 2015.

[3] Farzan Masrour, Iman Barjesteh, Rana Forsati, Abdol-Hossein Esfahanian, and Hayder Radha. Network completion with node similarity: A matrix completion approach with provable guarantees. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 302–307. ACM, 2015.

[4] Faezeh Faez, Yassaman Ommi, Mahdieh Soleymani Baghshah, and Hamid R Rabiee. Deep graph generators: A survey. *IEEE Access*, 9:106675–106702, 2021.

[5] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

[6] Jia Li, Jianwei Yu, Jiajin Li, Honglei Zhang, Kangfei Zhao, Yu Rong, Hong Cheng, and Junzhou Huang. Dirichlet graph variational autoencoder. *Advances in Neural Information Processing Systems*, 33:5274–5283, 2020.

[7] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10(1):33, 2018.

[8] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5694–5703, 2018.

[9] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems*, pages 4257–4267, 2019.

[10] Yassaman Ommi, Matin Yousefabadi, Faezeh Faez, Amirmojtaba Sabour, Mahdieh Soleymani Baghshah, and Hamid R Rabiee. Ccgg: A deep autoregressive model for class-conditional graph generation. In *Companion Proceedings of the Web Conference 2022*, pages 1092–1098, 2022.

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