Graph Neural Networks for Metagenomic Binning

Andre Lamurias [1 2] **Alessandro Tibo** [1] **Katja Hose** [1 3] **Mads Albertsen** [1] **Thomas Dyhre Nielsen** [1]

Abstract

Most methods for metagenomic binning rely solely on the local properties of the individual contigs. Because of this, these techniques are unable to take advantage of the connections between contigs as established by the assembly graph. In this paper, we explore Graph Neural Networks (GNNs) to leverage the assembly graph when learning contig representations for metagenomic binning. We applied four different types of GNN architectures, comparing their results on real and synthetic datasets, demonstrating encouraging results and, therefore, a promising research direction to pursue and explore.

1. Introduction

Microbial communities have a direct impact on human health and our environment. Being able to explore the microbial potential for the general good does, however, require an astute understanding of the microbial world in terms of, among others, diversity and function. Metagenomics studies microbial communities at the DNA level, and in theory, it is possible to recover the genomes of all the microbes in a sample. However, this is a complex task since DNA sequencing technologies can only produce fragments of the full genome, and, due to the incompleteness of current reference databases, the full genome of most microbes in environmental samples remains unknown (Pasolli et al., 2019).

The process of recovering genomes from the fragmented sequencing data is called binning. Binning can be considered a two-step process, where the first step defines a notion of similarity between DNA sequences and the second step consists of grouping these sequences into clusters, which are referred to as bins. [1] The input to the binning process is a set of assembled contiguous DNA sequences (contigs). Contigs are obtained by representing the fragmented sequences as a graph, called an assembly graph, where each node represents a contig and the edges represent overlaps between contigs. Most binners (Yang et al., 2021) only use local features of the individual contigs, thus failing to take full advantage of the relational information embedded within the assembly graph. Since, by construction, connected contigs share similar DNA sub-fragments, we hypothesize that the assembly graph holds potentially important information that can be exploited during the binning process.

Deep learning approaches have also recently been exploited for metagenomic binning (Nissen et al., 2021; Lamurias et al., 2023). In particular, (Xue et al., 2021) explores GNNs, but fails to incorporate domain knowledge into the model, thereby limiting its applicability to real-world datasets. On the other hand, GraphMB (Lamurias et al., 2022) integrated the assembly graph into its graph deep learning model, exploring only one type of GNN.

In this paper, we present a comparative study of Graph Neural Networks (GNN) (Kipf & Welling, 2017; Hamilton et al., 2017; Velickovic et al., 2018) for metagenomic binning using assembly graphs. The contribution of this work is the novel framework combining the advantages of existing binning approaches based only on local representations with state-of-the-art GNN algorithms that can use the assembly graph to improve the local representations. We show how different types of GNNs perform on this task and on both simulated and real-world metagenomics datasets. The code and data used in the experiments are available at `https://github.com/MicrobialDarkMatter/vaegbin`.

2. Related Work

In recent years, several binners have been proposed based on k-mer composition and so-called abundance features (Lu et al., 2017; Yu et al., 2018; Yang et al., 2021). Two of the most well-established binners based on these features are MetaBAT2 (Kang et al., 2019) and MaxBin2 (Wu et al., 2016), where the SCGs associated with each contig are used

[1]Department of Computer Science, Aalborg University, Aalborg, Denmark [2]NOVA LINCS, NOVA School of Science and Technology, Lisbon, Portugal [3]Institute of Logic and Computation, TU Wien, Vienna, Austria. Correspondence to: Andre Lamurias <a.lamurias@fct.unl.pt>.

The 2023 ICML Workshop on Computational Biology. Honolulu, Hawaii, USA, 2023. Copyright 2023 by the author(s).

[1]In the remainder of this paper, the terms clusters and bins will be used interchangeably.

to estimate the number of bins. More recently, deep learning-based methods have been used to improve metagenomic binning. Deep learning models present an advantage over other statistical methods since the former types of models have the potential to learn more complex patterns in the data that would be difficult to model with other standard methods. An example is VAMB (Nissen et al., 2021), which is based on a variational autoencoder (Kingma & Welling, 2014), encoding k-mer composition and abundance features in a low dimensional embedding, which is subsequently used for clustering/binning.

Attempts have also been made to use the assembly graph to improve metagenomic binning. The common assumption is that contigs that are linked in the assembly graph should also be binned together. For example, GraphBin (Mallawaarachchi et al., 2020) refines bins from other tools using information from the assembly graph by adopting a label propagation scheme, and CCVAE (Lamurias et al., 2023) incorporates connectivity information from the assembly graph directly in the loss function of a variational autoencoder.

3. Methodology

In the following, we use x to denote vectors in $\mathbb{R}^n$ (including scalars) and $\mathcal{X}$ for sets. In this study, the data is always represented as an assembly graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of nodes and edges, respectively. Each node $u \in \mathcal{V}$ is associated with either a genome (categorical) label y^u or a set of single copy genes (SCGs) $\hat{\mathcal{Y}}(u)$ (up to 104) when genome labels are unavailable. SCGs appear only once in the complete genome and are crucial for the microbes' functioning and reproduction. These SCGs are identified using CheckM (Parks et al., 2015), a standard metagenomic evaluation tool. It is important to note that our framework remains entirely unsupervised in both scenarios regarding the genome labels, which are solely utilized for quantitative evaluations.

The set of edges $(u, v) \in \mathcal{E}$ represents pairs of nodes connected by the assembly graph. This more concise notation is adopted instead of a sparse adjacency matrix. The set of edges $(u, v) \in \mathcal{E}$ represents the pairs of nodes connected by the assembly graph. We adopt this more compact notation instead of a sparse adjacency matrix.

Due to sequencing errors and the presence of genomes with similar sequences, the existence of edges in the assembly graph does not necessarily indicate that the corresponding nodes should have the same label. This can lead to erroneous edges in the assembly graph. To address this concern, a weight $w(u, v) \in [0, 1]$ is assigned to each edge $(u, v) \in \mathcal{E}$, which signifies the multiplicity of the k-mer supporting that particular edge. This weight can be interpreted as the

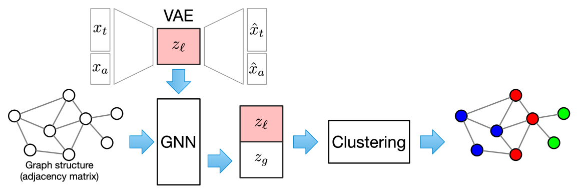

Figure 1. A Variational Autoencoder (top) is used to learn node representations z_ℓ. The graph structure and z_ℓ are fed into a graph neural network (bottom) which outputs features z_g depending on the graph structure. Finally, z_ℓ and z_g are concatenated and clustered.

confidence level of the edge. A value of 0 represents low confidence, while a value of 1 indicates high confidence. Nodes that are connected by edges with higher confidence levels are more likely to belong to the same genome. As a result, these nodes should share the same label.

Our framework, depicted in Figure 1, consists of a local and a global feature extractor for the nodes in $\mathcal{V}$. The local features (contig-specific representations) z_ℓ are learned with a Variational Autoencoder (VAE), while we adopt a graph neural network (GNN) approach for learning global features (graph representations). The GNN takes as input z_ℓ and G and produces a global representation for each node, z_g. Finally, z_ℓ and z_g are concatenated and used as the input for a clustering algorithm to discover the bins. Below we describe the graph representation learning, relying on a standard variational autoencoder for the contig-specific representations.

3.1. Graph representations

A GNN enables learning of node features that depend on node neighborhoods. In particular, GNNs aggregate the neighborhood information of a node through the following generic graph convolutional layer:

$$z_g^u = \alpha_{u,u}\Theta_1 z_\ell^u + \Theta_2 \sum_{v \in \mathcal{N}_G(u)} \alpha_{u,v} z_\ell^v, \qquad (1)$$

where z_ℓ^u and z_ℓ^v are the feature vectors produced by the VAE associated with nodes u and v, respectively. Θ_1 and Θ_2 are learnable parameterized matrices and $\alpha_{u,v} \in \mathbb{R}$ is a scalar for weighting the contribution of each node in the neighborhood. Note that multiple layers, as defined in Equation 1, can be stacked together in order to provide representations that depend on nodes at larger depths in the graph. Finally, each graph convolutional layer can also be intermixed with standard neural network layers. We note that our framework is generic with respect to the underlying GNN. In our experiments (see Section 4) we have evaluated three classical GNN architectures: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GAT (Velickovic et al., 2018).

The key to our framework is the loss function used to train the GNN, which is defined over pairs of GNN outputs, which we adapted from Lamurias et al. (2023):

$$\begin{aligned} J(z_g^u, z_g^v; \Theta) = {} & w(u,v)\log(\sigma(< z_g^u, z_g^v >)) \\ & + (1 - w(u,v))\log(1 - \sigma(< z_g^u, z_g^v >)) \\ & + \mathbb{I}[|\hat{\mathcal{Y}}(u) \cap \hat{\mathcal{Y}}(v)| > 0]\log(1 - \sigma(< z_g^u, z_g^v >)), \end{aligned} \tag{2}$$

where Θ are the GNN parameters, σ is the sigmoid function, $< \cdot, \cdot >$ denotes the scalar product, and $\mathbb{I}$ is the indicator function. The first two terms of the loss represent the weighted binary cross-entropy between connected and disconnected nodes in the assembly graph. The last term in the loss encourages different features for nodes with the same SCGs. This is desirable because nodes with the same SCGs should not be placed in the same clusters, since this would increase its contamination.

For the sake of simplicity, we consider all edges to have unitary weights. For GCNs, Equation 1 then becomes:

$$z_g^u = \frac{1}{d_u}\Theta z_\ell^u + \Theta \sum_{v \in \mathcal{N}_\mathcal{G}(u)} \frac{1}{\sqrt{d_u d_v}} z_\ell^v,$$

where $d_u = 1 + |\mathcal{N}_\mathcal{G}(u)|$, and $\Theta = \Theta_1 = \Theta_2$. For GraphSAGE, Equation 1 takes the form:

$$z_g^u = \Theta_1 z_\ell^u + \Theta_2 \frac{1}{|\mathcal{N}(u)|} \sum_{v \in \mathcal{N}_\mathcal{G}(u)} z_\ell^v.$$

Note that in our experiment, following (Hamilton et al., 2017), we also aggregate neighborhoods with LSTMs, which was already tested in (Lamurias et al., 2022). In Section 4 we denote with GRAPHSAGE-M and GRAPHSAGE-L the versions that use average and LSTM aggregations, respectively. For GATs, Equation 1 is specified as:

$$z_g^u = \alpha_{u,u}\Theta z_\ell^u + \Theta \sum_{v \in \mathcal{N}_\mathcal{G}(u)} \alpha_{u,v} z_\ell^v,$$

where

$$\alpha_{u,v} = \frac{\exp(\text{L-ReLU}(a^T(\Theta z_\ell^u || \Theta z_\ell^v)))}{\sum_{k \in \mathcal{N}_\mathcal{G}(u) \cup \{u\}} \exp(\text{L-ReLU}(a^T(\Theta z_\ell^u || \Theta z_\ell^k)))},$$

with a being a learnable parameter and L-RELU the leaky ReLU activation function.

3.2. Clustering and evaluation

We employ the clustering algorithm used in Nissen et al. (2021), which is an adapted form of the k-medoids algorithm. Notably, this modified version of the algorithm eliminates the need for an initial specification of the number of clusters.

Table 1. Datasets used in the experiments. STRONG100 is a simulated dataset, while the others are real-world datasets.

DATASETS	# NODES	# EDGES	AVG. DEGREE
STRONG100	852	1,952	2.291
AALE	45,831	33,173	0.724
MARI	41,559	35,001	0.842
DAMH	38,578	34,186	0.886

To evaluate the quality of the clusters, we adopt the completeness and contamination criteria, commonly used in metagenomic binning. Both criteria are domain-specific and indicate the quality of the clusters, according to the Minimum Information about a Metagenome-Assembled Genome (MIMAG) standard set by the Genomic Standards Consortium (Bowers et al., 2017). These two metrics are required to submit a genome to public databases and to report it in publications. Using these criteria, we can classify a bin as a High Quality (HQ) bin if completeness > 0.9 and contamination < 0.05, and as a Medium Quality (MQ) bin if completeness > 0.5 and contamination < 0.1[2].

For simulated datasets, the genomes in the dataset are known. It is therefore possible to map the node sequences to those genomes and obtain the ground truth genome label y^u of each node. We will therefore calculate the (average) precision, recall, and F1 score for these data sets. Similar to the previous criterion, we considered as HQ bins those with > 0.9 recall and > 0.95 precision, and as MQ bins those with > 0.5 recall and > 0.9 precision.

We followed the evaluation criteria for simulated datasets with ground truth labels as described in Meyer et al. (2018): using the AMBER evaluation tool, we evaluate precision and recall of each bin according to the labels of the nodes that constitute the cluster.

4. Experiments

In this section we present the experimental setup we used to evaluate our approach on simulated and real-world datasets, as well as the results obtained using the metrics previously introduced. The hyperparameters of the model are provided in the appendix. The VAE hyperparameters are based on the results obtained by Lamurias et al. (2023).

4.1. Data

We perform experiments on one simulated dataset and three Wastewater Treatment Plant (WWTP) datasets (Table 1). Since the benchmark simulated datasets used by

[2]HQ bins are also required to have the 5S, 16S and 23S rRNA genes and 18 tRNA genes, however, we did not check for these properties in this work.

Table 2. Results on the simulated dataset.

MODEL	AP	AR	F1	HQ	MQ
METABAT2	0.905	0.592	0.716	**26**	**37**
VAMB	0.969	0.755	0.849	**26**	34
MAXBIN2	0.818	0.765	0.791	14	23
GRAPHBIN	0.848	0.613	0.712	23	34
GCN	0.964	0.804	0.877	25±1	32±2
GRAPHSAGE-M	0.960	0.839	0.895	24±2	31±1
GRAPHSAGE-L	0.969	0.765	0.855	**26±1**	34±2
GAT	0.950	**0.863**	**0.904**	18±3	25±4

Table 3. Results on real-world datasets.

MODEL	AALE		MARI		DAMH	
	HQ	MQ	HQ	MQ	HQ	MQ
METABAT2	53	175	41	**155**	50	**219**
VAMB	42	160	34	135	31	132
MAXBIN2	20	60	20	70	21	82
GRAPHBIN	16	133	21	123	23	176
GCN	**55±1**	175±3	**46±1**	154±3	**54±1**	190±4
GRAPHSAGE-M	**55±0**	175±1	44±1	148±2	51±1	187±2
GRAPHSAGE-L	52±1	**184±4**	**46±2**	147±3	51±1	190±4
GAT	53±1	174±3	45±1	147±2	50±1	184±3

other binners do not include the assembly graph, we simulated a new dataset (Strong100). The simulated dataset was produced using the badread (Wick, 2019) tool (v0.2.0), where we generated reads according to the methodology proposed in (Quince et al., 2021); we simulate reads from 100 strains, corresponding to 50 species, with randomly generated abundances. We then assemble the contigs with the metaflye (Kolmogorov et al., 2020) tool (v2.9).

The WWTP datasets come from a previous study (Singleton et al., 2021), where we have access to four samples for each waste water treatment plant. Recall that the sample of each treatment plant is associated with a set of contigs, hence the abundance vector of each contig is of length four with one entry for each sample.

4.2. Results

We compare the results of the GNNs with four competitors (see Section 2) on the same datasets, using the default values specified in the corresponding papers. All the methods take as input the contig sequences and their abundances, except for GraphBin, which takes into account the assembly graph.

4.2.1. SIMULATED DATA

Table 2 shows the results obtained on the simulated dataset, where the metrics are calculated on the ground truth labels of the contigs using the AMBER evaluation tool (Meyer et al., 2018). In this scenario, the graph-based methods outperform the established binners on almost all metrics. Note that for downstream analyses, only the HQ bins can be considered recovered genomes, while the others do not have enough quality to be analyzed, because they are too incomplete or too contaminated.

4.2.2. REAL-WORLD DATA

As shown in Table 3, we can see that most of the GNNs outperform the other methods in terms of HQ bins recovered. By combining a VAE with a GNN, we can consistently obtain more HQ bins than all other baseline methods. In particular, in terms of HQ bins, we outperform both VAMB and MetaBAT2, both of which only rely on local contig features and thus fail to take advantage of the relational contig information embedded within the assembly graph. In terms of MQ bins, we obtain a higher or comparable number of bins relative to the baselines on two out of the three datasets. Different instantiations of the GNN model have been tested on all three datasets, with the GCN approach obtaining the largest number of high-quality bins. The other instantiations obtain similar results on some datasets, but not consistently. We hypothesize that this may partly be due to the loss function not being a good proxy for the quality metrics being used during the evaluation, hence more complex models, such as the GAT, may fail to bring consistent improvements. Also, the GraphSAGE-L model imposes a specific order in the neighbors of a node, which does not exist naturally.

5. Conclusion

This paper reports on interdisciplinary research between data science and bioinformatics, addressing the problem of metagenomic binning of contiguous DNA fragments (contigs). We have studied the effectiveness of multiple GNN architectures in leveraging the assembly graph for metagenomic binning. We observed that GCN tends to perform better on real-world datasets. Datasets with higher edge density seem to benefit the most from the GNN, as is the case of Mari and DamH, which had larger improvements compared to the other methods than AalE.

This work represents an exploration of graph learning methods for metagenomic binning and we believe that there are several promising directions for further work. An end-to-end approach that incorporates both representation learning and clustering could bring further improvements to this task. We expect that these approaches will have impact within both the machine learning and the bioinformatics communities.

Acknowledgments

This work was partially funded by VILLUM FONDEN under grant agreement number 34299 and the Poul Due Jensen Foundation. Alessandro Tibo is now working at AstraZeneca AB R&D, Gothenburg.

References

Bowers, R. M., Kyrpides, N. C., Stepanauskas, R., Harmon-Smith, M., Doud, D., Reddy, T., Schulz, F., Jarett, J., Rivers, A. R., Eloe-Fadrosh, E. A., et al. Minimum information about a single amplified genome (misag) and a metagenome-assembled genome (mimag) of bacteria and archaea. *Nature biotechnology*, 35(8):725–731, 2017.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, pp. 1025–1035, 2017.

Kang, D. D., Li, F., Kirton, E., Thomas, A., Egan, R., An, H., and Wang, Z. Metabat 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies. *PeerJ*, 7:e7359, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Kolmogorov, M., Bickhart, D. M., Behsaz, B., Gurevich, A., Rayko, M., Shin, S. B., Kuhn, K., Yuan, J., Polevikov, E., Smith, T. P., et al. metaflye: scalable long-read metagenome assembly using repeat graphs. *Nature Methods*, 17(11):1103–1110, 2020.

Lamurias, A., Sereika, M., Albertsen, M., Hose, K., and Nielsen, T. D. Metagenomic binning with assembly graph embeddings. *Bioinformatics*, 38(19):4481–4487, 2022.

Lamurias, A., Tibo, A., Albertsen, M., Hose, K., and Nielsen, T. D. Metagenomic binning using connectivity-constrained variational autoencoders. In *International Conference on Machine Learning*. PMLR, 2023.

Lu, Y. Y., Chen, T., Fuhrman, J. A., and Sun, F. Cocacola: binning metagenomic contigs using sequence composition, read coverage, co-alignment and paired-end read linkage. *Bioinformatics*, 33(6):791–798, 2017.

Mallawaarachchi, V., Wickramarachchi, A., and Lin, Y. Graphbin: refined binning of metagenomic contigs using assembly graphs. *Bioinformatics*, 36(11):3307–3313, 2020.

Meyer, F., Hofmann, P., Belmann, P., Garrido-Oter, R., Fritz, A., Sczyrba, A., and McHardy, A. C. Amber: assessment of metagenome binners. *GigaScience*, 7(6):giy069, 2018.

Nissen, J. N., Johansen, J., Allesøe, R. L., Sønderby, C. K., Armenteros, J. J. A., Grønbech, C. H., Jensen, L. J., Nielsen, H. B., Petersen, T. N., Winther, O., et al. Improved metagenome binning and assembly using deep variational autoencoders. *Nature biotechnology*, pp. 1–6, 2021.

Parks, D. H., Imelfort, M., Skennerton, C. T., Hugenholtz, P., and Tyson, G. W. Checkm: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome research*, 25(7):1043–1055, 2015.

Pasolli, E., Asnicar, F., Manara, S., Zolfo, M., Karcher, N., Armanini, F., Beghini, F., Manghi, P., Tett, A., Ghensi, P., et al. Extensive unexplored human microbiome diversity revealed by over 150,000 genomes from metagenomes spanning age, geography, and lifestyle. *Cell*, 176(3): 649–662, 2019.

Quince, C., Nurk, S., Raguideau, S., James, R. S., Soyer, O. S., Summers, J. K., Limasset, A., Eren, A. M., Chikhi, R., and Darling, A. E. Strong: metagenomics strain resolution on assembly graphs. *Genome Biol*, 2021.

Singleton, C. M., Petriglieri, F., Kristensen, J. M., Kirkegaard, R. H., Michaelsen, T. Y., Andersen, M. H., Kondrotaite, Z., Karst, S. M., Dueholm, M. S., Nielsen, P. H., et al. Connecting structure to function with the recovery of over 1000 high-quality metagenome-assembled genomes from activated sludge using long-read sequencing. *Nature communications*, 12(1):1–13, 2021.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *ICLR*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Wick, R. R. Badread: simulation of error-prone long reads. *Journal of Open Source Software*, 4(36):1316, 2019.

Wu, Y.-W., Simmons, B. A., and Singer, S. W. Maxbin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics*, 32(4): 605–607, 2016.

Xue, H., Mallawaarachchi, V., Zhang, Y., Rajan, V., and Lin, Y. Repbin: Constraint-based graph representation learning for metagenomic binning. *arXiv preprint arXiv:2112.11696*, 2021.

Yang, C., Chowdhury, D., Zhang, Z., Cheung, W. K., Lu, A., Bian, Z., and Zhang, L. A review of computational tools for generating metagenome-assembled genomes from metagenomic sequencing data. *Computational and Structural Biotechnology Journal*, 2021.

Yu, G., Jiang, Y., Wang, J., Zhang, H., and Luo, H. Bmc3c: binning metagenomic contigs using codon usage, sequence composition and read coverage. *Bioinformatics*, 34(24):4172–4179, 2018.

A. Hyperparameters

Both the encoder and decoder of the VAE consist of two hidden layers with 512 nodes and leaky ReLU activations. μ_z and $\log \sigma_z^2$ have size 32 for the simulated and 64 for the real-world datasets. The VAE is trained using gradient descent for 500 epochs with a learning rate of $1e^{-3}$. We use GNNs with three graph convolutional layers for the real-world datasets and one graph convolutional layer for the simulated dataset. In both cases, the hidden layers consist of 128 nodes and the output z^u has 64 nodes. The learning rate was set to $1e^{-2}$ and we performed 500 epochs of training.