Graph Neural Networks for Microbial Genome Recovery

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

[1]Department of Computer Science, Aalborg University, Aalborg, Denmark
[2]Center for microbial communities, Aalborg University, Denmark
{andrel,alessandro,khose,tdn}@cs.aau.dk, ma@bio.aau.dk,

Abstract

Microbes have a profound impact on our health and environment, but our understanding of the diversity and function of microbial communities is severely limited. Through DNA sequencing of microbial communities (metagenomics), DNA fragments (reads) of the individual microbes can be obtained, which through assembly graphs can be combined into long contiguous DNA sequences (contigs). Given the complexity of microbial communities, single contig microbial genomes are rarely obtained. Instead, contigs are eventually clustered into bins, with each bin ideally making up a full genome. This process is referred to as metagenomic binning.

Current state-of-the-art techniques for metagenomic binning rely only on the local features for the individual contigs. These techniques therefore fail to exploit the similarities between contigs as encoded by the assembly graph, in which the contigs are organized. In this paper, we propose to use Graph Neural Networks (GNNs) to leverage the assembly graph when learning contig representations for metagenomic binning. Our method, VAEG-BIN, combines variational autoencoders for learning latent representations of the individual contigs, with GNNs for refining these representations by taking into account the neighborhood structure of the contigs in the assembly graph. We explore several types of GNNs and demonstrate that VAEG-BIN recovers more high-quality genomes than other state-of-the-art binners on both simulated and real-world datasets.

1 Introduction

Microbial communities have a direct impact on human health and our environment and they play an essential role in achieving the sustainable development goals [Akinsemolu, 2018; Timmis and others, 2017], in particular *good health and well-being* (SDG-3), *life below water* (SDG-14), and *life on land* (SDG-15), to name a few. 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 and others, 2019].

The process of recovering genomes from the fragmented sequencing data is called binning. In general, binning is 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. 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 and others, 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.

With the recent successes of applying deep neural networks to various problems, there has also been an increasing focus on adapting such approaches to graph data structures. Graph Neural Networks (GNNs) take advantage of the connectivity information in a graph and can be used to perform node, edge, and graph-level tasks. Several types of GNNs have been proposed, such as Graph Convolutional Networks (GCN) [Kipf and Welling, 2017], GraphSAGE [Hamilton *et al.*, 2017], and Graph Attention Networks [Velickovic *et al.*, 2018]. Concurrently with the present work, GNNs have also been used for metagenomics binning, showing promising results [Xue *et al.*, 2021; Lamurias *et al.*, 2022].

In this paper, we present VAEG-BIN, a binning approach based on Graph Neural Networks (GNN), integrating local features obtained through a Variational Autoencoder (VAE) [Kingma and Welling, 2014] with global features learned from the assembly graph. We compare VAEG-BIN to existing state-of-the-art binning techniques on real-world and simulated datasets and demonstrate a significant improve-

ment compared to state of the art using standard genome-recovery evaluation metrics. The code and data used in the experiments will be made available upon acceptance.

2 Domain background

The genome of an organism is the collection of all its genetic information, represented in the form of a sequence of DNA bases. In an environmental sample, we encounter a combination of genomes from multiple individuals. The general metagenomic workflow starts then by extracting and sequencing DNA fragments from an environmental sample. High-throughput sequencing produces a raw electrical signal that is then converted into a sequence consisting of the four DNA bases (ATCG). This procedure generates millions to billions of reads, which may originate from any of the genomes of all the organisms in the sample. Reads can have variable lengths, and depending on the technology used, they are classified as short reads (100-150 bases) or long reads (2-30k bases). While longer read lengths are preferable to fully reconstruct the genome, up until recently long reads were also more prone to errors [Sereika and others, 2021].

To obtain full microbial genomes, which are in the order of millions of bases, we need to combine these reads into longer sequences. As one sample may contain numerous identical copies of a microbial species, the reads will be a collection from these organisms starting at random points of the genome, and hence have partial overlaps if enough reads are sampled. The process of combining these reads is called assembly and it involves finding overlaps between reads to obtain contiguous sequences, called contigs. Specifically, reads (through k-mers) are encoded in a de Brujin graph [Compeau *et al.*, 2011] that serves as a generator, where each walk of the de Brujin graph corresponds to a contig. By finding sub-sequence overlaps (k-mers) within the reads, an assembly graph is generated, where each node corresponds to a contig and an edge represents a possible continuation of that contig in the genome. The number of reads that overlap on the same position is called coverage or depth. Figure 1 shows an example assembly graph generation starting from the reads.

Since the genome of each organism will be split into several contigs, advanced methods are required to recover high-quality genomes from a set of contigs. These methods are referred to as binners since they partition contigs into different bins. As reads correspond to actual DNA sequences present in the sample, the read coverage of a contig will be correlated to the number of organisms in the sample. This property is called abundance and is a useful feature to bin contigs since contigs from the same genome should have similar abundances [Albertsen and others, 2013]. Another useful property is the k-mer frequencies of a contig, generally of size 3 or 4, which should also be similar for contigs from the same genome (also known as k-mer composition) [Burge and others, 1992]. An important set of genes are the Single Copy Genes (SCG), which occur only once in the full genome but which are essential for the functioning and reproduction of the microbes. Information about the single copy genes can be incorporated into the binning process, since two contigs with the same SCG must belong to different genomes and should therefore appear in different bins. Therefore, the aim of the binning task is to partition contigs into bins that contain a single copy of all the genes in the set of SCGs.

3 Related Work

In recent years, several binners have been proposed based on k-mer composition and abundance features [Yang and others, 2021]. One of the best-performing binners based on these features is MetaBAT2 [Kang and others, 2019]. MetaBAT2 uses these two features to compute a pairwise distance matrix for all contig pairs, calculated with a k-mer frequency distance probability and abundance distance probability. The former is based on an empirical posterior probability obtained from a set of reference genomes. MaxBin2 [Wu *et al.*, 2016] is another method that uses an Expectation-Maximization algorithm to estimate the probability of a contig belonging to a particular bin. The SCGs associated with each contig are used to estimate the number of bins. Although more k-mer composition and abundance methods have been proposed [Lu *et al.*, 2017; Yu and others, 2018], MetaBAT2 and MaxBin2 are the most established and commonly used ones.

More recently, deep learning-based methods have been used to improve metagenomic binning. Deep learning models present an advantage over other statistic methods since these types of models have the potential to learn complex patterns in the data that would be difficult to model with other methods. VAMB [Nissen and others, 2021] is a binner based on a variational autoencoder that encodes k-mer composition and abundance features in a low dimensional embedding that can lead to improved binning results. However the usage of deep learning for metagenomics is still in its early stages and very few works have explored how to adapt existing algorithms for these problems, in particular for the most recent sequencing technologies that produce longer reads [Sereika and others, 2021].

Some recent works have attempted 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. Specifically, GraphBin navigates the assembly graph using a label propagation algorithm and refines clusters that were separated in the binning process, but which nevertheless contain contigs that are linked in the assembly graph. However, GraphBin uses the assembly graph only as a post-processing step, and does not integrate it into the full binning process. By relying on the assembly graph only as a last step of the binning process, errors can potentially be introduced if the relational structure in the assembly graph is not carefully used, e.g., contigs may be incorrectly assigned to bins due to misleading or erroneous links in the assembly graph. This is more likely to occur in complex samples, where variants of the same species (strains) exist, thereby making it more likely that the assembly graph contains links between contigs even if these contigs belong to different genomes.

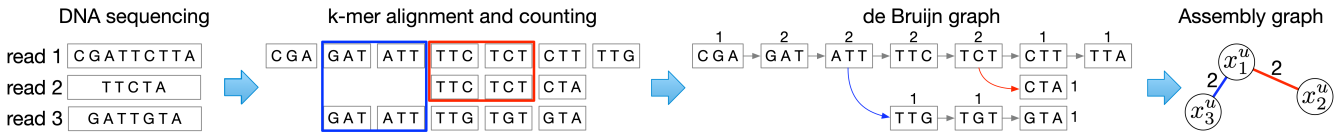

Figure 1: Assembly graph generation. The DNA sequences are read from an environmental sample, converting the raw signal to one of four bases. While finding the best alignment, the reads are broken into k-mers (k= 3 in this example, but usually much larger), and matching k-mers are aligned. The overlapping k-mers are aggregated and organized in a de Brujin graph. Here, each path from the root (C G A) to an end node ((T T A), (C T A), (G T A)) generates a contig. For example, the sequence C G A T T G T A is a contig. The integer numbers reported in the de Brujin graph corresponds to the number of times k-mers overlap for different readings. Finally, each contig is associated with a node in the assembly graph. The edge weights are the fraction of reads that overlap at the intersection of the contig pair, in this case, two reads align to both edges.

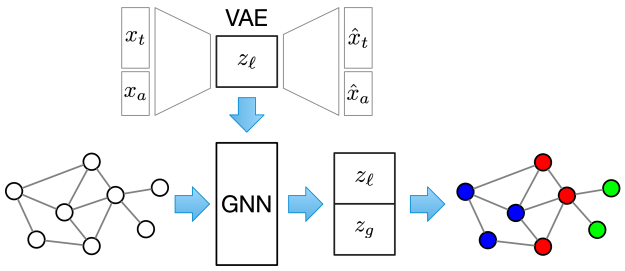

Figure 2: A Variational Autoencoder (top) is used to learn node representations z_ℓ. The graph structure and z_ℓ are fed into a graph neural network (bottom) which outputs features z_g depending on the graph structure. Finally, z_ℓ and z_g are concatenated and clustered.

4 Methodology

In the following, we denote with x vectors in $\mathbb{R}^n$ (including scalars) and $\mathcal{X}$ for sets. In VAEG-BIN, 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 a sequence of length $\ell(u) \in \mathbb{N}$, but it is represented as a tuple of features $x^u = (x_t^u \in \mathbb{R}^{n_t}, x_a^u \in \mathbb{R}^{n_a})$, where x_t^u represents the k-mer frequencies, and x_a^u represents the relative abundances. In all our experiments, we consider x^u as the concatenation of x_t^u and x_a^u which has size $n_t + n_a$. The dimensionalities n_t and n_a of both vectors depend on the specific datasets. Each node $u \in \mathcal{V}$ is either associated with a genome (categorical) label y^u or a set of SCGs $\hat{\mathcal{Y}}(u)$ (up to 104) when genome labels are not available. The SCGs are predicted by CheckM [Parks *et al.*, 2015], a standard metagenomic evaluation tool. Note that in both scenarios VAEG-BIN remains completely unsupervised with respect to the genome labels, which are only used in the quantitative evaluations. In contrast to classical graph problems, the set of edges in the assembly graph may contain several false positives. To mitigate this issue, each edge $(u, v) \in \mathcal{E}$ is assigned a weight $w(u, v) \in [0, 1]$, which represents the fraction of reads that overlap with both nodes of that edge and can thus be seen as an edge confidence. Here, 0 and 1 mean low and high confidences, respectively.

The VAEG-BIN framework, depicted in Figure 2, 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 fed into a clustering algorithm to discover the bins. In the following sections, the terms clusters and bins are used interchangeably. Recall that we aim at determining the clusters assignments in which each cluster contains as many unique SCGs nodes as possible. While our approach remains completely unsupervised, our aim is reflected in Equation 2 as described below.

4.1 Contig-specific representations

We first generate contig-specific representations by encoding k-mers x_t and relative abundances x_a with a VAE (see Figure 2). A VAE consists of an encoder E, paramaterized by θ_E and a decoder D, parameterized by θ_D. Each x_t is normalized to have zero mean and unitary variance, while each component of x_a is normalized to have a sum equal to 1 across all the relative abundances. The loss function used to train the VAE is adopted from [Nissen and others, 2021] and consists of three components[1] :

$$\begin{aligned} J(x_t, x_a; \theta_E, \theta_D) = w_a \, x_a^T \log(\hat{x}_a + \epsilon) \\ + w_t \, \|x_t - \hat{x}_t\|^2 \\ - w_{kl} \, D_{KL}(\mathcal{N}(\mu_z, \sigma_z) || \mathcal{N}(0, I)), \end{aligned}$$

where D_{KL} is Kullback-Leibler divergence, ϵ is a small constant, $(\mu_z, \log \sigma^2) = E((x_t, x_a); \theta_E)$, and $(\hat{x}_t, \hat{x}_a) = D(z; \theta_E)$. Thus, the reconstruction error is separated into two terms capturing the k-mer compositions and abundances of the contigs, respectively. Note that here z is sampled by using the reparametrization trick on μ_z and $\log \sigma^2$. Finally, we use $z_\ell = \mu_z$ produced by E as node features in the following sections; in preliminary experiments we found that σ attains very small values and is therefore not included in the feature representation.

4.2 Graph representations

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

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

[1] In all of our experiments, $w_a = (1 - \alpha) \log(n_a + 1)^{-1}$, $w_t = \alpha / n_t$, and $w_{kl} = (n_z \beta)^{-1}$, where n_z is the dimension of μ_z, $\alpha = 0.15$ and $\beta = 200$. See also [Nissen and others, 2021].

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 remark that our framework, VAEG-BIN, is generic with respect to the GNN. In our experiments (see Section 5) we have evaluated VAEG-BIN on three classical GNN architectures: GCN [Kipf and Welling, 2017], GraphSAGE [Hamilton *et al.*, 2017], and GAT [Velickovic *et al.*, 2018].

The key to VAEG-BIN is the loss function used to train the GNN, defined on pairs of GNN outputs.

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

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

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

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

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

Note that in our experiment, following [Hamilton *et al.*, 2017], we also aggregate neighborhoods with LSTMs. In Section 5 we denote with GRAPHSAGE-M and GRAPHSAGE-L the versions that use average and LSTM aggregations, respectively. For GATs, Equation 1 is specified as:

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

where

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

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

4.3 Clustering and evaluation

For the sake of consistency, we adopt the same cluster algorithm used in [Nissen and others, 2021], a modified version of the k-medoids algorithm, which does not require an initial number of clusters. The clustering algorithm receives as input the concatenation of the contig-specific and graph representations, i.e., $z^u = (z_\ell^u, z_g^u)$. This algorithm consists of a three-step process: it first finds a seed medoid by picking a random z^u associated with a node and calculates the cosine distance to all other z^v. If any node has more neighbors than the current medoid within a small radius, that one is picked as the new medoid. The second step consists in determining the cluster radius. The distance from the chosen medoid to all other nodes is calculated, and the algorithm tries to find an optimal distance threshold that includes most of the nearby nodes, but small enough to exclude distant nodes, which should correspond to a local minimum in a histogram plot of the distances. The third step consists in removing the nodes within that threshold from the list of nodes to cluster and returning to step one until no more unclustered nodes are left. A more detailed description of the algorithm can be found in [Nissen and others, 2021].

To evaluate the quality of the bins (clusters), we adopted the completeness (see Equation 3) and the contamination (see Equation 4) criteria. Both criteria are domain-specific and indicate the quality of the cluster, according to the Minimum Information about a Metagenome-Assembled Genome (MIMAG) standard set by the Genomic Standards Consortium [Bowers and others, 2017]. Completeness indicates whether the genome is suitable for a specific downstream analysis, while contamination indicates the fraction of the genome that might be contaminated with sequences from other genomes. 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 High Quality (HQ) if completeness > 0.9 and contamination < 0.05, and as Medium Quality (MQ) if completeness > 0.5 and contamination < 0.1[2].

The recommended way of calculating these metrics is to use the list of SCGs as ground truth (recall that these genes are present only once in the genomes of nearly all bacteria). Some SCGs are collocated, meaning that they are in close proximity in the DNA, and so their occurrences are not fully independent. For this reason, the ground truth is defined in terms of a set of sets of SCGs, $\mathcal{G}_M$, where each set of SCGs represents a group of collocated SCGs.

The completeness of a bin is given by:

$$\text{COMP}(\mathcal{G}_M, \hat{\mathcal{Y}}) = \frac{1}{|\mathcal{G}_M|} \sum_{\mathcal{G} \in \mathcal{G}_M} \frac{|\mathcal{G} \cap \hat{\mathcal{Y}}|}{|\mathcal{G}|}, \quad (3)$$

where $\hat{\mathcal{Y}}$ represents the multiset of SCGs associated with the nodes of a single bin. The completeness takes value 1 (the maximum) when all genes from $\mathcal{G}_M$ are identified in the bin. Completeness can be associated with the concept of recall, since it measures the fraction of retrieved genes in the bins.

[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.

The contamination of a bin is defined as

$$\textsc{Cont}(\mathcal{G}_M, \hat{\mathcal{Y}}) = \frac{1}{|\mathcal{G}_M|} \sum_{\mathcal{G} \in \mathcal{G}_M} \frac{1}{|\mathcal{G}|} \Big(\sum_{g \in \mathcal{G}} \Big(\sum_{y \in \hat{\mathcal{Y}}} \mathbb{I}[g = y] \Big) - 1 \Big), \tag{4}$$

where $\mathbb{I}$ is the indicator function which is 1 if g is equal to y, and 0 otherwise. Here, we assume that if $g \notin \hat{\mathcal{Y}}$ the inner most summation in Equation 4 is 0. There is no maximum value of contamination, since it will depend on the number of times an SCG is duplicated, i.e., a value of 1 means that on average all genes from $\mathcal{G}_M$ are duplicated once, and 2 means that all genes have two additional copies on average.

For simulated datasets, the genomes in the dataset are known. Therefore, it is possible to map the node sequences to those genomes and obtain the ground truth genome label y^u of each node. We followed the evaluation criteria for simulated datasets with ground truth labels as described in [Meyer and others, 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. If a bin contains all the nodes associated with one label, then that bin will have a recall of 1, and if it does not contain nodes of any other labels, it will have a precision of 1. In these metrics, we also take into account the length of the nodes, because longer nodes will have a bigger impact on recovering the genome sequence than smaller nodes.

Average precision (AP), average recall (AR), and F1 are thus defined as follows:

$$\text{AP} = \frac{1}{K} \sum_{k=1}^{K} \frac{TP_k}{TP_k + FP_k} \quad \text{AR} = \frac{1}{K} \sum_{k=1}^{K} \frac{TP_k}{TP_k + FN_k}$$

$$\text{F1} = \frac{2 \cdot \text{AP} \cdot \text{AR}}{\text{AP} + \text{AR}},$$

where K is the number of clusters and

$$TP_k = \sum_{u \in C_k} \ell(u) \mathbb{I}[y^k = y^u] \quad FP_k = \sum_{u \in C_k} \ell(u) \mathbb{I}[y^k \neq y^u]$$

$$FN_k = \sum_{u \notin C_k} \ell(u) \mathbb{I}[y^k = y^u].$$

Here, y^k is the label associated with the cluster C_k, calculated as the majority label of the node labels belonging to C_k. 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.

5 Experiments

Data We perform experiments on one simulated dataset and three Wastewater Treatment Plant (WWTP) datasets (Table 1). Since the benchmark simulated datasets used by 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 and others, 2021]; we simulate reads from 100 strains, corresponding to 50 species, with randomly generated abundances. Badread is a read simulator developed specifically for long-reads, taking into consideration the error rate of these technologies. In this way, we also generate a dataset that is up-to-date with the current state of DNA sequencing technologies, where longer reads can be obtained, leading to longer contigs, as well. We then assemble the contigs with the metaflye [Kolmogorov and others, 2020] tool (v2.9) and ran other binners for comparison. The WWTP datasets come from a previous study [Singleton and others, 2021]. For the WWTP datasets, we have access to four samples for each WWTP. Recall, that each WWTP is associated with a set of contigs. Therefore, for each contig, the abundance values are stored in the entries of a vector of size four (one for each sample). While the simulated dataset has ground truth labels, mapping each node to a specific genome, for the real-world datasets we do not have access to this information and we instead follow common practice and estimate the quality of the binning results in terms of the number of high and medium quality bins (see Section 4.3). The details of the graphs of each dataset are reported in Table 1.

Table 1: Datasets used in the experiments. STRONG100 is a simulated dataset, while the others are real-world datasets. n_t is the dimension of the k-mer frequency features and n_a is the dimension of the abundance features.

DATASETS	# NODES	# EDGES	n_t	n_a
STRONG100	852	1,952	136	1
AALE	45,831	33,173	136	4
MARI	41,559	35,001	136	4
DAMH	38,578	34,186	136	4

Parameters The input dimensions of each dataset are specified in Table 1. The n_t value is the same for all datasets as we used k-mers of size 4 and aggregated k-mers that were the same as their reverse complement. 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 are trained by 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.

5.1 Results

We compare the results of VAEG-BIN with four competitors 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. We compare against MetaBAT2 [Kang and others, 2019] and MaxBin2 [Wu *et al.*, 2016], which are generally considered state-of-the-art [Yue and others, 2020; Vosloo and others, 2021]. We also compare against VAMB [Nissen and others, 2021] and GraphBin [Mallawaarachchi *et al.*, 2020], the former because it is the only published binner that uses deep learning methods, and the latter because it also takes the assembly graph as input. GraphBin runs on top of another binner, so it requires the output of another binner as input. We used MetaBAT2 as the

Table 2: Results on the simulated dataset. AP and AR denotes the average precision and recall over all bins. The F1 score is calculated by considering the average precision and recall. Finally, HQ and MQ refer to the number of High-quality and Medium-Quality bins.

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. HQ and MQ refer to the number of High-quality and Medium-Quality bins.

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

input to GraphBin because it obtained the highest results of the three other binners we considered. We present the results of the simulated and real-world datasets separately due to the different metrics used. We evaluate each of the four binners as well as VAEG-BIN with the four considered GNNs. To show the stability of VAEG-BIN, we ran the experiments ten times.

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 and others, 2018]. These results indicate how the methods work in a scenario where the original genome of each contig is known. In this scenario, the graph-based methods outperform the established binners on almost all metrics. In terms of F1-score, GAT achieves the best balance, obtaining however a low number of HQ and MQ bins. The GraphSAGE-L variant obtained a higher number of HQ bins, at the expense of a lower F1-score. While the F1-score takes into account the precision and recall of all bins, the HQ and MQ values exclude the lowest quality bins. Hence, we can have many bins with low F1-score, without affecting the HQ and MQ values. Although MetaBAT2 obtained the second lowest F1-score, it had the same number of HQ bins as VAMB and GraphSAGE-L, which is the main quality criterion for metagenomic applications. 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.

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 explored for 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 may fail to bring consistent improvements.

6 Conclusion

This paper reports on interdisciplinary research between data science and bioinformatics, addressing the problem of metagenomic binning of contiguous DNA fragments (contigs). This activity is key for understanding the diversity and function of microbial communities, which have a direct impact on both health and the environment and thus play a vital role in addressing the sustainable development goals. We have proposed VAEG-BIN, a novel methodology for learning feature representations for contigs, combining local feature representations (obtained through a variational autoencoder) with global features learned using a GNN based on the assembly graph in which the contigs are organized.

We have compared VAEG-BIN with other state-of-the-art metagenomic binning methods on both simulated and real-world datasets. We observe that by leveraging the relational information in the assembly graph, we can significantly increase the number of high-quality genomes recovered during the subsequent binning process as compared to the state-of-the-art baseline methods.

This work represents an initial step in the exploration of graph learning methods for metagenomic binning and we believe that there are several promising directions for further work. For instance, we plan to refine the clustering step in order to better take into account the distribution of single copy genes over the different clusters. This will involve refining the loss function to promote high completeness and low contamination of the clusters. Additionally, an end-to-end approach that incorporates both representation learning and clustering could bring further improvements to this task. We expect that the challenges presented by this task will lead to more solutions that benefit both the Artificial Intelligence field and progress on the SDGs.

Acknowledgments

The study was funded by research grants from VILLUM FONDEN (34299, 15510) and the Poul Due Jensen Foundation (Microflora Danica)

References

[Akinsemolu, 2018] Adenike A Akinsemolu. The role of microorganisms in achieving the sustainable development goals. *Journal of cleaner production*, 182:139–155, 2018.

[Albertsen and others, 2013] Mads Albertsen et al. Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes. *Nature biotechnology*, 31(6):533–538, 2013.

[Bowers and others, 2017] Robert M Bowers 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.

[Burge and others, 1992] Chris Burge et al. Over-and under-representation of short oligonucleotides in dna sequences. *Proceedings of the National Academy of Sciences*, 89(4):1358–1362, 1992.

[Compeau *et al.*, 2011] Phillip EC Compeau, Pavel A Pevzner, and Glenn Tesler. How to apply de bruijn graphs to genome assembly. *Nature biotechnology*, 29(11):987–991, 2011.

[Hamilton *et al.*, 2017] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1025–1035, 2017.

[Kang and others, 2019] Dongwan D Kang et al. Metabat 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies. *PeerJ*, 7:e7359, 2019.

[Kingma and Welling, 2014] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Kolmogorov and others, 2020] Mikhail Kolmogorov et al. metaflye: scalable long-read metagenome assembly using repeat graphs. *Nature Methods*, 17(11):1103–1110, 2020.

[Lamurias *et al.*, 2022] Andre Lamurias, Mantas Sereika, Mads Albertsen, Katja Hose, and Thomas Dyhre Nielsen. Metagenomic binning with assembly graph embeddings. *bioRxiv*, 2022.

[Lu *et al.*, 2017] Yang Young Lu, Ting Chen, Jed A Fuhrman, and Fengzhu Sun. Cocacola: binning metagenomic contigs using sequence composition, read coverage, co-alignment and paired-end read linkage. *Bioinformatics*, 33(6):791–798, 2017.

[Mallawaarachchi *et al.*, 2020] Vijini Mallawaarachchi, Anuradha Wickramarachchi, and Yu Lin. Graphbin: refined binning of metagenomic contigs using assembly graphs. *Bioinformatics*, 36(11):3307–3313, 2020.

[Meyer and others, 2018] Fernando Meyer et al. Amber: assessment of metagenome binners. *GigaScience*, 7(6):giy069, 2018.

[Nissen and others, 2021] Jakob Nybo Nissen et al. Improved metagenome binning and assembly using deep variational autoencoders. *Nature biotechnology*, pages 1–6, 2021.

[Parks *et al.*, 2015] Donovan H Parks, Michael Imelfort, Connor T Skennerton, Philip Hugenholtz, and Gene W Tyson. Checkm: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome research*, 25(7):1043–1055, 2015.

[Pasolli and others, 2019] Edoardo Pasolli 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 and others, 2021] Christopher Quince et al. Strong: metagenomics strain resolution on assembly graphs. *Genome Biol*, 2021.

[Sereika and others, 2021] Mantas Sereika et al. Oxford nanopore r10. 4 long-read sequencing enables near-perfect bacterial genomes from pure cultures and metagenomes without short-read or reference polishing. *bioRxiv*, 2021.

[Singleton and others, 2021] Caitlin M Singleton 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.

[Timmis and others, 2017] Kenneth Timmis et al. The contribution of microbial biotechnology to sustainable development goals, 2017.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*. OpenReview.net, 2018.

[Vosloo and others, 2021] Solize Vosloo et al. Evaluating de novo assembly and binning strategies for time series drinking water metagenomes. *Microbiology spectrum*, 9(3):e01434–21, 2021.

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

[Wu *et al.*, 2016] Yu-Wei Wu, Blake A Simmons, and Steven W Singer. Maxbin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics*, 32(4):605–607, 2016.

[Xue *et al.*, 2021] Hansheng Xue, Vijini Mallawaarachchi, Yujia Zhang, Vaibhav Rajan, and Yu Lin. Repbin: Constraint-based graph representation learning for metagenomic binning. *arXiv preprint arXiv:2112.11696*, 2021.

[Yang and others, 2021] Chao Yang et al. A review of computational tools for generating metagenome-assembled genomes from metagenomic sequencing data. *Computational and Structural Biotechnology Journal*, 2021.

[Yu and others, 2018] Guoxian Yu et al. Bmc3c: binning metagenomic contigs using codon usage, sequence com-

position and read coverage. *Bioinformatics*, 34(24):4172–4179, 2018.

[Yue and others, 2020] Yi Yue et al. Evaluating metagenomics tools for genome binning with real metagenomic datasets and cami datasets. *BMC bioinformatics*, 21(1):1–15, 2020.