The Australian National University
2600 ACT | Canberra | Australia

Australian National University

School of Computing

College of Engineering and Computer Science (CECS)

Metagenomic Binning via Graph Representation Learning and Clustering

— 24 pt Honours project (S1/S2 2022)

A thesis submitted for the degree
Bachelor of Advanced Computing

By:
Wei Zhou

Supervisor:
Dr. Yu Lin

November 2022

Declaration:

I declare that this work:

- upholds the principles of academic integrity, as defined in the University Academic Misconduct Rules;
- is original, except where collaboration (for example group work) has been authorised in writing by the course convener in the class summary and/or Wattle site;
- is produced for the purposes of this assessment task and has not been submitted for assessment in any other context, except where authorised in writing by the course convener;
- gives appropriate acknowledgement of the ideas, scholarship and intellectual property of others insofar as these have been used;
- in no part involves copying, cheating, collusion, fabrication, plagiarism or recycling.

November, Wei Zhou

Acknowledgements

I would like to express my greatest thanks to my supervisor Dr Yu Lin from the Research School of Computer Science of the Australian National University for the patient support and guidance on algorithm development, problem-solving, and presentation in my honours year.

Many thanks to my brother Ziwen Zhou for his encouragement and unconditional support throughout this year.

I would like dsay thank you to my friend Yizhen Zhang for her support and understanding throughout this year.

Abstract

Metagenomics study the genetic content obtained directly from various natural environments. The area of metagenomics provides valuable insights of the structure, function, and diversity of microbial communities. The recent advancement of high-throughput sequencing technologies allows metagenomics to analyze environmental samples, unlike the early approaches that have to rely on culture-based methods. Metagenomic binning is a process of clustering the sequenced environmental samples into bins. Each bin represents a taxonomic group such as species, genera, etc. The early approach of binning is to bin the raw reads before assembling the reads into longer contigs. However, the reads are too short to generate accurate binning results as the short reads may not contain enough genetic information. As a result, the conventional procedure for metagenomics analysis is to use assembly graph to join short reads into longer contigs. Then place the resulting contigs into bins. The majority of the existing metagenomic binning methods developed based on the composition information and abundance information which make it challenging to bin short contigs with similar composition and abundance information.

An Assembly graph is used to join the short reads into longer contigs, thus containing valuable connectivity information. But the majority of the existing binning tools ignore the assembly graph. In this thesis, we propose MixBin, a stand-alone metagenomic binning tool that uses both assembly graph and composition information.

Table of Contents

Chapter 1

Introduction

Metagenomics has opened up fresh views of the microbial world. Traditionally, microbiology has relied on pure cultures that are amenable to be grown in the laboratory. However, it is typically challenging to cultivate most microorganisms in the laboratory. Only about 0.01-1% of the microorganisms that are observed under the microscope could be isolated and grown in the laboratory (Garza and Dutilh, 2015). Microorganisms fail to be grown in the laboratory due to many reasons such as (1) inadequate incubation periods (Pulschen et al., 2017) (2)use of wrong pH of medium during and culture process (Köpke et al., 2005) (3) not supplying the correct nutrients (Davis et al., 2011). Therefore, researchers have been ignorant of the existence of such microorganisms which skewed and limited our view of microbial diversity and how microbial communities function. To overcome the issues of cultivating microorganisms, culture-independent methods are developed to investigate microbial communities. Metagenomics is one of the culture-independent methods to analyze microbial samples.

Metagenomics study the genetic materials that are collected directly from microbial communities, such as soil, seawater, fermented food, and human distal gut. Metagenomic samples are first collected from various environments, then those samples are separated to obtain the *metagenomes*. After that, the metagenomes are sequenced to obtain the exact sequences of the nucleotide information. The nucleotide information contains the genetic information of how a microorganism function and behavior. By analyzing the nucleotide information, different microorganisms in the microbial communities can be characterised. Metagenomics allows us to explore and investigate microbial communities and their relations.

Metagenomics is one of the most rapidly developed research areas today. Since the term *metagenomics* was coined by (Handelsman et al., 1998) in 1998, great progress has been made. With the development of the high-throughput sequencing (HTS) platforms, both the time costs and financial costs of sequencing are reduced significantly. This led to the

generation of more and more complex sequence data sets that are derived from various environments, such as coastal ocean (Mou et al., 2008), human distal gut (Qin et al., 2010), glacier ice (Simon et al., 2009) and fermented food (Chen et al., 2016). HTS approaches allow genomic analyses ideally not only for cultures that are amenable to grow in the laboratory but also for all microbes in a sample. This led to the development of bioinformatic tools of analyzing these data sets.

Metagenomic binning is one of the methods to do the taxonomic analysis for large metagenomic data sets. Binning is the process of grouping sequences based on taxonomic groups. Associating the organisms with the genomic sequences is highly desirable for researchers to understand the taxonomic structure of the organisms. Note that the samples are collected from different environments, which means the samples are mixed together. It is difficult for scientists to distinguish which species each contig belongs to in the laboratory. This is the problem we address in this thesis. We aim to produce an efficient and accurate metagenomic binning tool.

A typical metagenomics analysis workflow contains sampling, sequencing, assembly and binning. The workflow starts with collecting samples from various environments. The samples are processed to obtain metagenomes, then sequenced by HTS platforms to produce short sequences called *reads*. During assembly, those short reads are joined based on the overlapped regions to form longer sequences called *contigs*. Finally, these assembled contigs are placed into imaginary bins that each bin represent a taxonomic group (i.e, specie).

1.1 Contribution

Most of the existing metagenomoic binning tools either use the composition and abundance information of contigs (Alneberg et al., 2014; Bowers et al., 2017; Nissen and et al., 2021) or use assembly graph for binning (Xue et al., 2022; Mallawaarachchi et al., 2020a). We use both composition information of contigs and assembly graph to learn the global representation of contigs, and use bipartite graph matching to label the constrained contigs. Then take the labeled constrained contigs as labeled training data to label the rest contigs with semi-supervised GCN-based label propagation. The experiment result shows our approach archives state-of-the-art performance on most of datasets.

The contribution of this thesis can be summarized as:

- We propose a graph-based binning approach, called MixBin, which explicitly incorporate homophily relationships of the contigs (assembly graph) and the sequence information (composition features).
- We propose a bipartite graph matching algorithm to mine the implicit information among constraints derived from the single-copy marker genes, followed by a constraints-based label propagation approach.
- We benchmark the performance of the proposed MixBin model on five simulated

data sets against state-of-the-art metagenomic binning tools, graph clustering algorithms and graph learning methods. Experimental results demonstrate the superior of the proposed MixBin model.

1.2 Thesis outline

In this thesis, the introduction of this thesis is provided in Chapter 1. The research background of metagenomic binning and related state-of-art metagenomic binning tools are reviewed in the Chapter 2. The methodology of MixBin is presented in the Chapter 3, includes the representation learning, bins initialization and GCN-based label propagation. The experiments of benchmarking Mixbin with baselines, machine learning models and metagenomic binning tools is presented in the Chapter 4. Finally, we summarize our thesis and discuss the possible future work in Chapter 5.

Chapter 2

Background and Related Work

This chapter provides the research background of metagenomic binning, specifically explain what is genome sequencing, metagenome sequencing, genome assembly, metagenome assembly and metagenomic binning and the differences between them along with some related popular tools. In addition, composition information, abundance information and single-copy marker genes are explained. Some popular state-of-the-art mategenomic binning tools are reviewed and analyzed. Finally, the challenges of metagenomic binning are discussed.

2.1 Research Background

2.1.1 Genome Sequencing

Genome is made of DNA. It is the genetic material of an organism. DNA (deoxyribonucleic acid) is the hereditary material in almost every organisms. DNA is made up of four nucleotides which are adenine(A), thymine(T), cytosine(C), guanine(G). Those bases form pair with each other, A pairs with T, C pairs with G. The sequence of these bases contains the genetic information of organisms.

Genome sequencing is the process of idenitifying the exact sequence of the nucleotides in DNA. After collecting the sample from the environment, the obtained DNA fragments are put in the high-throughput sequencing(HTS) platform to determine the sequence of them. The output sequences are known as reads. The reads are strings that made up with the four chemical bases A, T, C, G.

The Sanger sequencing method (Sanger et al., 1977) also known as the first generation sequencing can produce reads of length ranging from 400 to 900 base pairs(bp) with extreme low error rate of about 0.001%. However, the costs for Sanger sequencing to produce a base pair are ranging from a hundred to a thousand times of HTS platform

which cause low throughput of sequences. Hence, HTS platform is known for its low costs and high throughput. It almost replace the Sanger sequencing. Although the HTS platform costs much lower than Sanger sequencing, it produces much shorter reads that ranging from 100bp to 300bp with higher error rate of about 0.1%. In addition, the short reads lead to problems when assembling genomes.

2.1.2 Genome Assembly

Genome assembly refers to the process of assembling short reads fragments into longer non-overlapping contigs based on their genomes origins. After the reads are obtained from genome sequencing, the reads are too short to do metagenomic binning. The reads need to be joined by overlaps to form non-overlapping longer contigs. It is done by finding the overlaps between the reads and joining two reads based on the overlapped sequences. However, it is not as simple as it sounds, one of the challenge is the existences of large numbers of repeats which are the duplicate identical sequences. As aforementioned, the sequences are made up with four chemical bases A, T, C, G, hence there exist many repeats. The length of these repeats can vary from a few hundred to a few thousand base pairs. It causes the problems when assemble such repeats because the HTS platform outputs short reads that are not long enough to find the overlaps between the reads and repeats

The workflow of genome assembly start by identifying the repeats from the reads and get rid of the duplicate reads, then identify the overlaps between the reads. The reads are joined based on the overlaps to form the initial assembly graph. Then the assembly graph is simplified by joining the overlaps into contigs. The nodes of resulted final assembly graph are the contigs.

There are two categories of the recent genome assemblers which are the De-Bruijn-graph(DBG) and the overlap-layout-consensus(OLC).

The OLC assemblers first find the overlaps from all the obtained reads and the layout of the reads are carried out afterward. Then build a graph with the reads as the nodes and the overlaps as the edges. Finally, the consensus sequences can be inferred from the built graph (hence the term *overlap-layout-consensus*). Some early developed OLC assemblers are XBAP (Gleeson and Staden, 1991), TIGR (Sutton et al., 1995). Some recent developed OLC assemblers are SGA (Simpson and Durbin, 2012) and Canu (Koren et al., 2017).

The DBG assemblers first split the reads into k-mers. k-mers is the short sequences with length k. Then build a De Bruijn graph of the k-mers. The genome can be inferred from the De Bruijn graph. Idbury and Waterman (Idury and Waterman, 1995) first developed the DBG algorithm in the 1995. EULER (Pevzner et al., 2001) was the first assembler that use this algorithm in 2001. There are many popular assemblers that use the DBG approach such as Velvet (Zerbino and Birney, 2008), SPAdes (Bankevich et al., 2012), MEGAHIT (Li et al., 2015), and Flye (Kolmogorov et al., 2019). Nowadays, DBG assemblers are more commonly used because they are less time consuming than the OLC

assemblers. It is very time consuming to identifying the overlaps among the reads for OLC assemblers because it requires all-vs-all mapping of the reads. This thesis make use of the DBG assemblers.

2.1.3 Metagenome Assembly

Metagenome assembly is almost the same as genome assembly, except the metagenome is collected from various environments. The genome assembly approaches are explicitly designed for the single genome. Hence, it cannot be directly used for metagenomic data. Metagenomic data contains a mixture of genomes that come from different organisms. Different organisms have different levels of abundance and relatedness with others. Such characteristics make the assembly process more complicated. As aforementioned, the genome assembly faces challenges if there exist large numbers of repeats. For single organism, the problem can be solved by assuming an uniform sequencing process. Then such repeats can be simply treated as anomalies in the depth of coverage. The problem become more severer for metagenomic data because metagenomic data consists with many different organisms. Simple coverage statistics use for metagenomic data because different organisms in the metagenomic data have uneven representations such that some genomes are sequenced into higher depth of coverage than others. In addition, two unrelated genomes can contain almost identical repeats, and multiple close-related genomes can have tiny genetic differences. It is difficult to distinguish such genetic differences from the sequencing error which make it difficult to decide whether to keep or ignore such differences when assembling metagenomes. Therefore, it is challenge for metagenomic assemblers to produce complete/nearly-complete metagenomes.

Recently, metagenomic assemblers such as metaFlye (Kolmogorov et al., 2020) and metaSPAdes (Nurk et al., 2017) are developed specifically to assemble metagenomic data sets. Recent results from the seconed round of Critical Assessment of Metagenome Interpretation challenge (Meyer et al., 2022) revealed that metaSPAdes and metaFlye performed well on metagenomic data. These metagenomic assemblers are favoured for generating assembly graph. This thesis use metaSPAdes to generate the assembly graph as one of the input for our model MixBin.

metaSPAdes is a software to do metagenomic assembly by first using SPAdes assembly toolkit (Bankevich et al., 2012) to create the De Bruijn graph of all the reads, then simplify it into assembly graph. In the assembly graph, each node denotes a contig and each edge denotes homophily information between two nodes(contigs). The connected nodes in the assembly graph are highly like to belong to the same specie because the assembly graph disconnect those weak edges if the coverage of the contig that connected by the edge is significantly lower than another contig.

2.1.4 Metagenomic Binning

Metagenomic samples contain mixed samples of different microbial genomes. As aforementioned, it is challenge to produce complete or nearly-complete metagenomes. Metage-

nomic binning are developed to overcome this issue. Metagenomic binning is the process of placing the assembled metagenome into different bins, each bin represents a specie. After the contigs are separated into groups, the results can be used to construct metagenome-assembled genomes and reconstruct accurate genome (Frank et al., 2016). Most of the metagenomic binning tools developed based on genetic information such as composition information, abundance information and single-copy marker genes.

Composition information Composition information refers to the characteristics that can be derived directly from the nucleotide sequences such as oligonucleotide frequencies and GC content(guanine-cytosine content). Previous research has demonstrated that the oligonucleotide frequencies (k-mers) contains species-specific signals such that the contigs come from the same specie have shown high similarity in composition information (Karlin and Ladunga, 1994). In addition, it is possible to compare the GC content to distinguish the species. (Land et al., 2015) proved that the GC content is different among different unrelated species. As a result of this finding, composition-based binning tools have been introduced. (Dick et al., 2009) perform binning by comparing the normalised frequencies of oligonucleotides. (Saeed et al., 2012) perform binning by comparing the GC content.

Since oligonucleotide frequencies are made of characters, it is necessary to first convert it to an appropriate numerical feature vector (refer to Figure 2.1). k is often set to four, then all the sub-sequences with window size of four are extracted. The corresponding frequencies of them are counted(refer to the blue shaded box in Figure 2.1). Finally, transform them to a numerical feature vector with 136 dimensions. Typically, the result vector should be in high dimensional space with 256 dimensions. In this thesis, the composition information is obtained with CONCOCT (Alneberg et al., 2014) which reduce the dimensionality of feature vector by considering the palindromic k-mers.

Abundance information One challenge that composition-based binning method would be faced is cluster the species with low abundance. Such species are easily misclassified into the larger bins that contain higher abundance species because low abundance sequences usually form smaller indistinct clusters. This problem can be solved by abundance-based binning method.

Abundance information refers to the coverage of contigs. Coverage is the average number of sequences that align or "cover" to the reference sequences of the contig (refer to Figure 2.2 below). As show in the Figure 2.2, the coverage of sequence 5 is 4 for the reference sequence 3 because there 4 sequences that are align to the reference sequence.

Abundance-based binning methods are developed based on the assumption that the coverage profiles of contigs should be either strongly correlated across multiple samples (Sedlar et al., 2017) or follow the LanderWaterman(Lander and Waterman, 1988) distribution.

Single-copy marker genes

The single-copy marker genes refer to the special genes that occurred once and only once

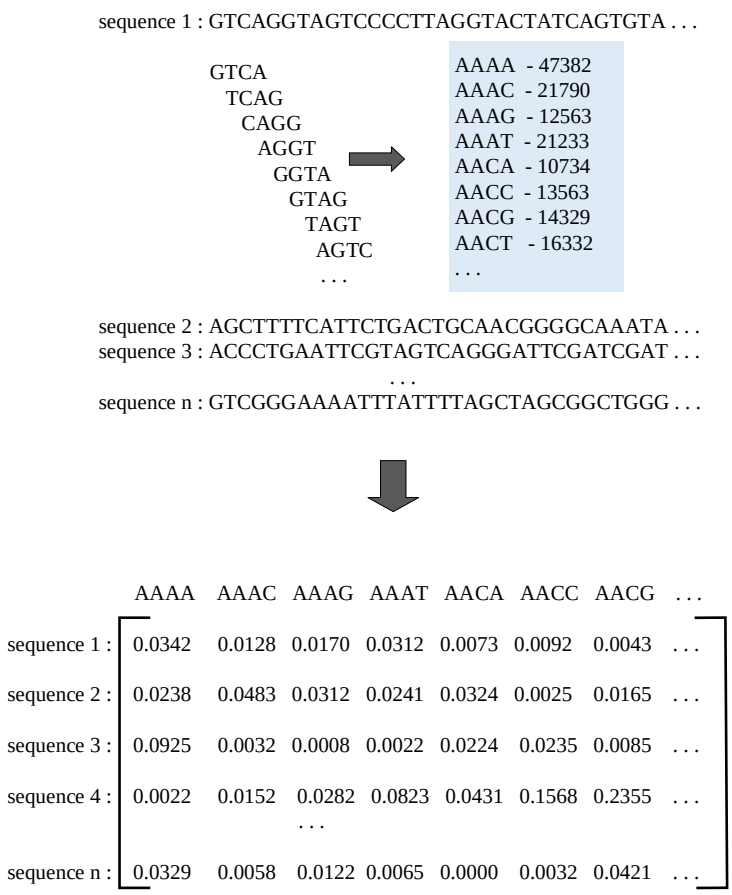

Figure 2.1: Construct composition information vector

in every genome. For example, if two contigs contain the same single-copy marker gene, then two of them are not belong to the same genome. A contig contain a single-copy marker gene only if it aligns to more than 50% of this marker gene length. The unique characteristic is helpful for metagenomic binning. Hence, many metagenomic binning methods are developed based on this fact.

MaxBin (Wu et al., 2014) and Maxbin 2.0 (Wu and et al., 2015) initialize the number of bins by using the marker genes. MyCC (Lin and Liao, 2016) use the marker genes to improve the final binning result. In this thesis, the single-copy marker genes are obtained with HMMER (Eddy, 2011) and FragGeneScan(Rho et al., 2010).

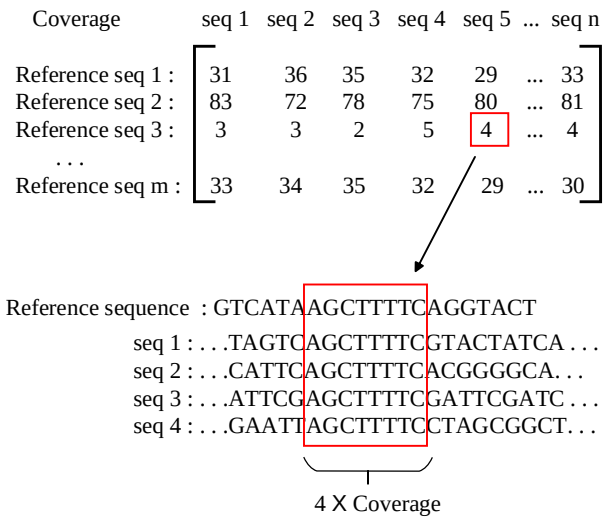

Figure 2.2: Construct Abundance information vector

2.2 Related Work

There are mainly two categories of metagenomic binning which are reference-based binning and reference-free binning. The reference-free binning can be broadly divided into two categories which are assembly graph-based binning and composition and abundance-based binning (refer to Figure 2.3 below).

2.2.1 Reference-based Binning

Reference-based binning tools requires a reference database of microbial genomes. It compares the sequences with the reference microbial genomes to obtain similarity values. The genome of the sequence is determined by the one with the highest similarity value. Then the sequences are assigned to corresponding taxonomic groups. Popular reference-based binning tools are MetaABC (Su et al., 2011), MetaPhlAn (Segata et al., 2012), Kraken (Wood and Salzberg, 2014) and Kraken2 (Wood et al., 2019).

For the sequences that come from known microbial genomes(included in the existing reference database), reference-based tools are able to archive high accuracy. However, such tools are severely limited when classify sequences that come from unknown microbial genomes(not included in the existing reference database).

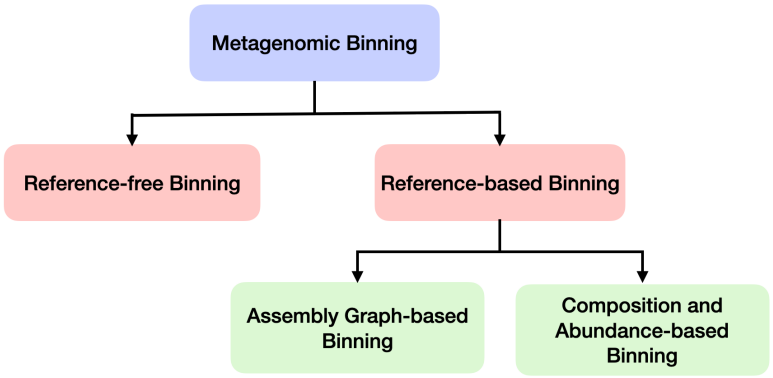

Figure 2.3: Metagenomic Binning categories

2.2.2 Reference-free Binning

As the name implies, the reference-free binning methods do not require any reference database to classify the sequences into different taxonomic groups (i.e species). They cluster the sequences into unlabeled bins based on their genetic information such as composition and coverage information. Reference-free binning methods can be broadly divided into two categories which are assembly graph-based binning and composition and abundance-based binning.

Assembly Graph-based Binning
As the name implies, assembly graph-based binning methods use assembly graph to do metagenomic binning. The recent state-of-art approaches are GraphBin, (Mallawaarachchi et al., 2020a), GraphBin2 (Mallawaarachchi et al., 2020b) and RepBin (Xue et al., 2022).

GraphBin use assembly graph and label propagation to improve the exist metagenomic binning tools. GraphBin is the first tool that use such approach in the metagenomic binning. GraphBin first assemble the short reads into longer non-overlapping contigs to obtain the assembly graph. Then GraphBin use existing binning tools to cluster the contigs and label the nodes(contigs) in the assembly graph. After that, GraphBin use the assembly graph to refine the labels of the contigs based on the fact that the connected and close nodes(contigs) in assembly graph are highly likely belong to the same species. Finally, GraphBin use semi-supervised label propagation to label the rest unlabeled contigs.

The experiment is done by comparing the performance of the original binning tools :

MetaWatt (Strous et al., 2012), MaxBin2 (Wu and et al., 2015), BusyBee Web (Laczny et al., 2017), MetaBAT2 (Kang and et al., 2019) and SolidBin (Wang et al., 2019) with the combination of GraphBin with these tools. The results shows significant improvement on all four evaluation metrics, precision, recall, F1-score and ARI for different combinations of GraphBin and other binning tools.

GraphBin2 is an advanced version of GraphBin. It also use assembly graph to refine the metagenomic binning results of other existing binning tools. The differences between GraphBin and GraphBin2 are : (1) GraphBin2 supports overlapped binning. (2) GraphBin2 can apply to long-read assembler. (3) GraphBin2 refined the algorithm that based on assembly graph and coverage information instead of only assembly graph.

GraphBin2 is motivated by the fact that different species may have the same sequences in the genomes. Hence, a contig contains such sequences should belong to multiple species. However, the existing binning tools only assign each contig to at most one bin. GraphBin2 is the first one consider and address the overlapped binning problem.

The preprocess step of GraphBin2 is similar to GraphBin. GraphBin2 obtain the reads with HTS platform (Illumina HiSeq 2000). Then assemble the reads into longer contigs with assembly graph. There are two categories of assemblers which are DBG assemblers and OLC assemblers. GraphBin2 selected a representative assembler from each of the category. For DBG, metaSPAdes (Nurk et al., 2017) is selected to construct assembly graph. For OLC, SGA (Simpson and Durbin, 2012) is selected to construct assembly graph. GraphBin2 obtain the initial binning results with CONCOCT (Alneberg et al., 2013), MaxBin2 (Wu and et al., 2015) and SolidBin (Wang et al., 2019). Then the assembly graph is labeled by the initial binning results.

There are four steps in GraphBin2 : (1) Removing labels of unsupported nodes which are the nodes are neither are isolated, nor directly connect to a node with the same label, nor connect to a node with the same label through a path. (2) Correcting labels of inconsistent nodes by maximizing the labeled score. (3) Applying label propagation for unlabeled nodes based on the intuition of nodes that are close to each others in the assembly graph and nodes with similar abundance information are highly possible to have the same label. (4) Inferring all the multi-labelled nodes.

GraphBin2 shows significant improvement on recall value for all three binning tools MaxBin2, CONCOCT and SolidBin. In addition, GraphBin2 reduced the running times and memory usage by further advance the algorithm of GraphBin. Although GraphBin and GraphBin2 show significant improvement. They are not stand-along binning tools. They have to be used with other binning tools. Hence, the number of bins are determined by those binning tools which can not be dynamically changed during the binning process. Hence, some possible incorrect binning results would be propagated during the label propagation step.

In contrast, RepBin is a stand-along binning tool. RepBin use single-copy marker genes as constraints and assembly graph to do metagenomic binning on both real world data

and simulated data. RepBin first perform constraint-based representation learning with assembly graph and single-copy marker genes to capture the global graph structure while satisfying the constraints of single-copy marker genes. Then perform k-means algorithm with the constrained contigs to obtain initial clusters. Finally, RepBin performs GCN-based(Graph Convolution Network) label propagation to label the rest unlabeled contigs. RepBin creatively designed constraint-based representation learning and constraint-based binning. The experiment results shows RepBin significantly outperform the state-of-art binning tools (i.e Metawatt (Strous et al., 2012), CONCOCT (Alneberg et al., 2013), MaxBin (Wu et al., 2014), BusyBeeWeb(Laczny et al., 2017), SolidBin (Wang et al., 2019)) on both real world data and simulated data.

However, there exist two limitations of RepBin : (1) The constraints are used implicitly by incorporating in the loss function of representation learning which the constraints may not be fully satisfied. (2) The number of the bins need to be predefined to perform the k-means algorithm which means we have to know how many species exist in the given samples. (3) RepBin not utilize the composition or abundance information, it relies on the assembly graph and single-copy marker genes only.

Composition and Abundance-based Binning

Composition and abundance-based binning methods are developed based on both oligonucleotide frequencies and abundance information (Refer to Figure 2.4). As aforementioned, oligonucleotide frequencies and abundance information contain species-specific signals that helpful for metagenomic binning. These binning methods usually outperform composition-based and abundance-based binning methods. Composition and abundance-based binning method first assemble short reads into loner contigs, then make use of both composition and abundance information to analyze the metagenomic data. Finally, it place the contigs into different bins which represents a taxonimic group (i.e specie). These binning methods have been developed by using techniques such as principal component analysis (PCA) (Alneberg et al., 2013), probabilistic models (Wu and et al., 2015), expectation maximisation (EM) (Wu et al., 2014; Wu and et al., 2015), algorithms and machine learning models.

2.2.3 Challenges in Metagenomic Binning

The existing binning tools have problems when (1) bin sequences that shorter than 1,000 base pairs. (2) bin sequences by using only composition and abundance information, ignore the homophily information of assembly graph. (3) incorporate single-copy marker genes during the process of contigs binning.

- **Short Sequences** The composition and abundance information are unreliable for short sequences due to the low resolution of them. The composition information of them may highly deviated form their origin genomes. It is challenge to bin such short sequences. Hence, the short sequences are discarded by majority of the existing binning tools. For example, CONCOCT (Alneberg et al., 2013), MaxBin2 (Wu and et al., 2015) and SolidBin (Wang et al., 2019) ignore the contigs that

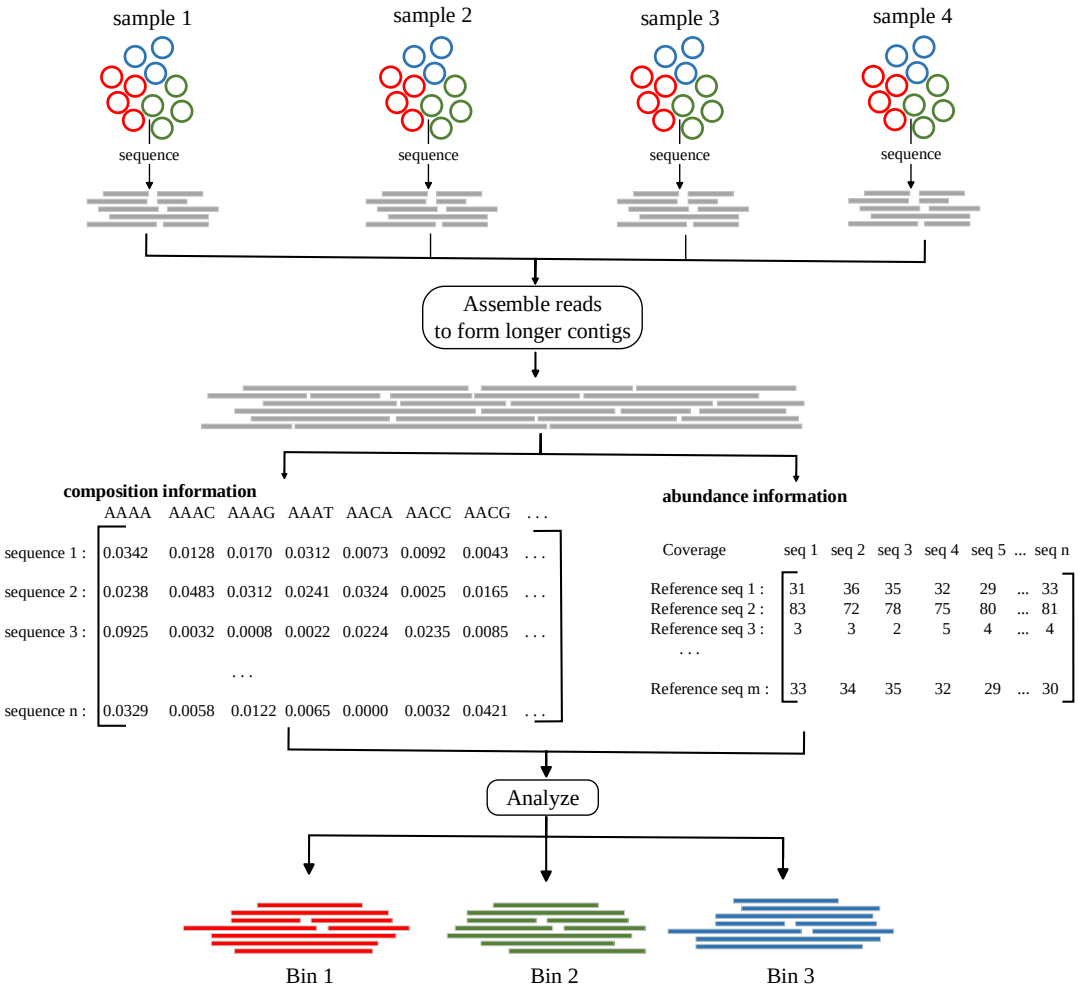

Figure 2.4: Composition and abundance-based binning

shorter than 1,000 base pairs, MetaBAT2 (Kang and et al., 2019) ignore the contigs that shorter than 1,500 base pairs and BusyBeeWeb ignore the contigs that shorter than 500 base pairs (refer to table 2.1 below). Although ignore such short contigs can improve the precision, it also make those tools suffer from low recall values.

- **Ignore homophily information** As aforementioned, the popular method to assemble the metagenome is to use assembly graph to assembly the short read into longer contigs. The contigs that are connected in the assembly graph are almost certainly belongt to teh same species. However, the existing metagenmoic binning tools overlook the homophily information of assembly graph and use only composition or coverage information. For example, CONCOCT, MetaWatt, MaxBin 2.0,

Table 2.1: Comparison of different metagenomic binning tools

Bining Tool	Composition	Coverage	Single-copy marker genes	Assembly Graph	Minimum contigs length (bp)
MetaWatt (Strous et al., 2012)	✓	✓	✗	✗	N/D
CONCOCT (Alneberg et al., 2013)	✓	✓	✗	✗	1,000 bp
Maxbin 2.0 (Wu and et al., 2015)	✓	✓	✓	✗	1,000 bp
BusyBeeWeb (Laczny et al., 2017)	✓	✗	✗	✗	500 bp
SolidBin (Wang et al., 2019)	✓	✓	✓	✗	1,000 bp
MetaBAT 2 (Kang and et al., 2019)	✓	✓	✗	✗	1,500 bp
GraphBin (Mallawaarachchi et al., 2020a)	✗	✗	✗	✓	depend on the used binning tools
VAMB (Nissen and et al., 2021)	✓	✓	✗	✗	100 bp
RepBin (Xue et al., 2022)	✗	✗	✓	✓	any length

SolidBin and MetaBAT 2 are use composition and coverage information only and not use the connectivity information for assembly graph.

- **Single-copy marker genes** As show in the table 2.1, the majority of the binning tools not use the constraints of single-copy marker genes. However, the unique characteristic of single-copy marker genes that each of them appears once and only once in every genome is extremely useful for metagenomic binning. As show in the table 2.1, only MaxBin 2.0, SolidBin and RepBin make use of the single-copy marker genes.

  Although MaxBin 2.0 make use of the single-copy marker genes, it only use it to determine the number of the target bins before binning contigs. MaxBin 2.0 do not further explore and utilize the single-copy marker genes during binning process. It is similar for SolidBin, SolidBin only use single-copy marker genes to estimate the bin number. In contrast, RepBin implicitly incorporate the single-copy marker genes as constraints in the representation learning step. The RepBin model learn the vector representations for each contig by pushing the contigs that

contain the same single-copy maker genes further away from each others in the representation space, and vice versa. Then RepBin use the resulted representations for subsequent contigs classification task. The problem with RepBin is RepBin implicitly incorporate the constraints in the loss function of representation learning which the constraints may not be fully satisfied.

2.3 Chapter Summary

This chapter provides metagenomic binning related terminology and a big picture of metagenomic binning along with some state-of-art metagenomic binning methods.

The workflow of sequencing genome, sequencing metagenome, assembling genome, assembling metagenome are explained in the section 2.1 along with some related tools and terminology of genetic information (i.e composition information, abundance information, single-copy marker genes). The differences and advantages between reference-based and reference-free binning tools are discussed in section 2.2.1 and section 2.2.2.

As aforementioned in the section 2.2.1, the reference-based binning is limited by the reference database. In contrast, the reference-free binning method able to break such limitation which make it more favoured. Among the reference-free binning methods, the assembly-based binning tools are more favoured because it contains valuable connectivity information of contigs. Three related work of assembly graph-based binning methods, GraphBin, GraphBin2 and RepBin are reviewed and analyzed in details in section 2.2.2. After analyzing these three metagenomic binning tools, several shortcomings are identified which provide some directions for this thesis. Therefore, we propose MixBin, an assembly graph-based stand-along metagenomic binning tools which explicitly incorporate with the single-copy marker genes and automatically determine the size of bins.

Chapter 3

Methodology

This chapter provides the methodology of our model MixBin, specifically about the representation learning, bins initialization and GCN-based label propagation. The following figure 3.1 is the general framework of our model MixBin.

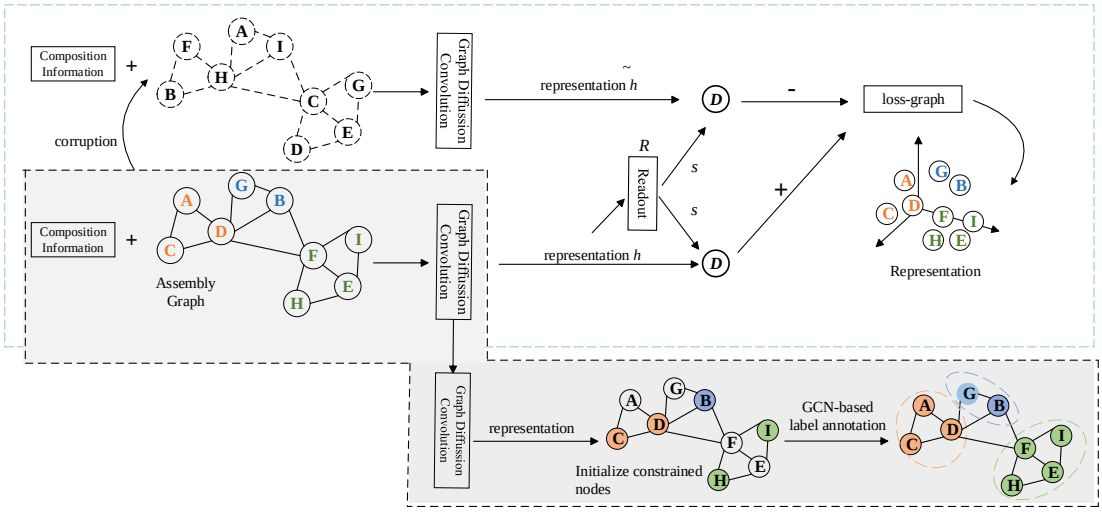

Figure 3.1: MixBin framework

3.1 Preprocessing

The inputs for MixBin are assembly graph, composition information and single-copy marker genes for each contig.

During the prepocessing step, the reads are first assembled into longer contigs with assembly graph. We choose to use the DBG assembler metaSPAdes (Nurk et al., 2017) because DBG assmblers are less time consuming than the OLC assemblers. In addition, metaSPAdes is a representative assembler among all DBG assemblers. After obtained the assembly graph, the isolated nodes are removed since they do not provides any useful connectivity information.

The composition information is obtained with CONCOCT (Alneberg et al., 2013). The k-mer frequency is summarized in the composition vector. CONCOCT filters contigs that shorter than 1000 base pairs before the prepocessing step. In this thesis, ther is no such restriction, we use any length of the contigs and k-mer frequency of 4. The contigs are indexed by $i = 1, ..., n$ where n is the number of contigs. The composition vector has dimension of 136, we denote the dimension $v = 1, ..., 136$. We denote the composition vector as $\mathbf{C}_i = C_{i,1}, ..., C_{i,v}$ for each contig. To remove the non-zero entries, a small pseudo-count is added to each k-mer $C'_{i,j} = C_{i,j} + 1$. Then the composition is normalized as follows,

$$p_{i,j} = \frac{C'_{i,j}}{\sum_{k=1}^{v} C'_{i,k}} \tag{3.1}$$

Then apply log-transformed over the normalized composition vector. The log-transformed normalised composition vector $\mathbf{x_i}$ for ith contig is formulated as follows,

$$\begin{aligned} \mathbf{x_i} &= [\log(p_{i,1}), ..., \log(p_{i,v})] \\ &= [\log(\frac{C'_{i,1}}{\sum_{k=1}^{v} C'_{i,k}}), ..., \log(\frac{C'_{i,v}}{\sum_{k=1}^{v} C'_{i,k}})] \end{aligned} \tag{3.2}$$

The log-transformed vectors gives better results than both the non-transformed vector and square root-transformed vector.

The single-copy marker genes are extracted with FragGeneScan(Rho et al., 2010) and HMMER (Eddy, 2011). FragGeneScan is a software to predict genes of genomic sequences or assembled contigs. FragGeneScan generates four files. The third file contains amino acid sequences correspond to the putative genes in the first file. Then we make use of the amino acid sequences form the third file with HMMER to extract the single-copy marker genes in the sequences.

Preliminaries

Consider the assembly graph $G = (V, E)$ where V denotes the contigs and E denotes the edges that connect two contigs. We denote the adjacency matrix as $A \in \mathbb{R}^{N \times N}$ where N is the number of contigs. The adjacency matrix A is a symmetric square matrix that used to represent the assembly graph.

We denote composition information as matrix $C \in \mathbb{R}^{N \times 136}$ where N is the number of contigs, 136 is the dimension of composition information for each contig.

3.2 Representation learning on Assembly Graph

In the representation learning step, we aim to learn both the graph structure of the assembly graph and composition information of contigs. This part contains two components, graph diffusion convolution and contrastive graph learning.(Refer to Figure 3.2 below).

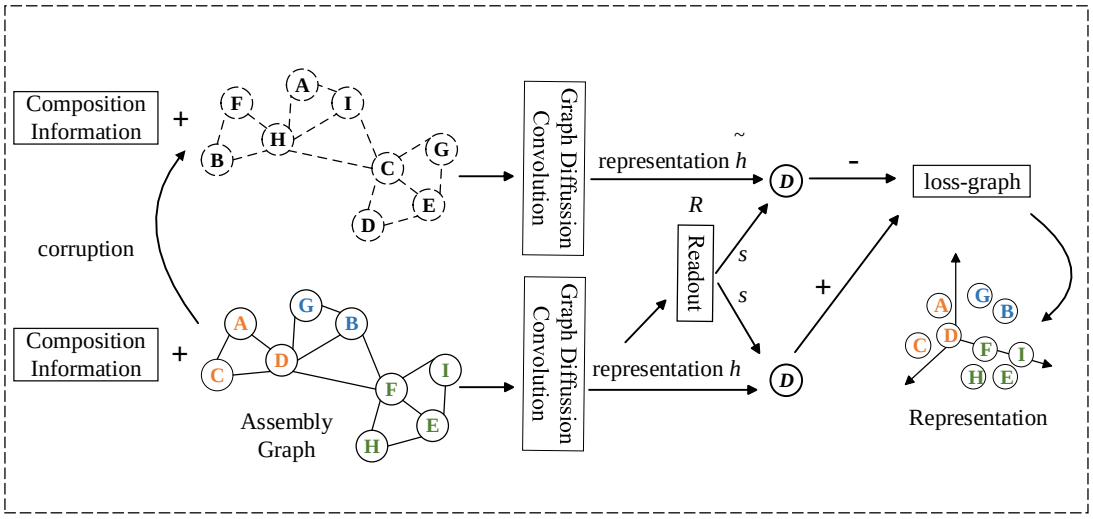

Figure 3.2: Representation learning on Assembly Graph

3.2.1 Contrastive graph learning framework

The framework contains encoder model, negative sampling, readout function and discriminator model. The objective is to learn an contrastive encoder to encodes the assembly graph and composition information into vector representations for each contig such that the contigs that are close in the assembly graph and have similar composition information are also close in the representation space. Here we use one-layer graph diffusion convolution as the encoder. The graph diffusion convolution will be explained in the next section 3.2.2. By employing the equation 3.7 of the graph diffusion convolution, we can obtain the representations of contigs $H = \{h_1, h_2, ..., h_n\}$.

It is important to train an contrastive encoder that captures the statistical dependencies between the original samples and negative samples. Here the negative samples are generated by applying corruption function on the original samples such that randomly shuffle the original feature matrix by row-wise. Then the encoder model (graph diffusion convolution) is used to encodes the original and negative samples to obtain the representations H and $\tilde{H}$ for each contigs.

Then readout function is used to map the node-level representation to a fixed size vector representation. It is formulated as follows,

$$s = R(H) = \frac{1}{n}\sum_{i=1}^{n} h_i \tag{3.3}$$

It maps the node-level representations H to global representations s by taking the average of the local-level representations H.

The underlying intention of employing a contrastive encoder is to maximize the mutual information between the output summary vector s and the representations H. Hence the discriminator model D is used to maximize the mutual information, it is used to discriminate the original samples (h_i, s) from its negative samples $(\tilde{h}_j, s)$.

The standard binary cross-entropy loss (equation 3.4) is selected to maximizing the mutual information between the positive examples and the negative examples), same approach as DGI (Veličković et al., 2019). The following objective function can maximize the mutual information between h_i and s effectively.

$$\mathcal{L} = -\frac{1}{2n}[\sum_{i=1}^{n} \log \mathcal{D}(\mathbf{h_i}, \mathbf{s}) + \sum_{j=1}^{n} \log(1 - \mathcal{D}(\tilde{\mathbf{h}}_\mathbf{j}, \mathbf{s}))] \tag{3.4}$$

3.2.2 Graph diffusion convolution

Most of the Graph Convolution Networks(GCNs) as known as message passing neural networks(MPNNs) only pass message to direct(one-hop) neighbors in each layer. Then, those messages are aggregated at each node to form a new representation for the next layer. Although MPNNs explores high-order neighbors in further layers, it limits the exploration of high-order neighbors in each layer. In contrast, graph diffusion convolution(GDC) (Klicpera et al., 2019) breaks such limitation by aggregating messages from further neighbors instead of direct neighbors. Hence, GDC shows excellent performance on graphs with high homophily score which is suitable for the assembly graph.

The primary distinction between the assembly graphs and arbitrary graphs is the high homophily score of the assembly graph. A high homophily score indicates that the most of connected contigs in the assembly graph are belong to the same species. Hence, GDC operator is employed in our model to preserve high-order graph structure of assembly graph.

A well-known example of leveraging graph diffusion is the personalized PageRank(PPR)(Page et al., 1999). (Klicpera et al., 2018) proposed an adaptation of PPR that using the recurrent equation as follows,

$$\mathbf{PPR}(x_i) = (1 - \alpha)\hat{\mathbf{T}}\mathbf{PPR}(x_i) + \alpha x_i \tag{3.5}$$

where $\hat{\mathbf{T}} = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}$ is the symmetric transition matrix, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ is the adjacency matrix with self-loops , $\hat{\mathbf{D}}$ is the diagonal degree matrix such that $\hat{\mathbf{D}}_{ii} = \sum_{j=1}^{N} \hat{\mathbf{A}}_{ij}$, x_i is teleport vector of node i, $\alpha \in (0, 1]$ is the teleport probability. Then solve the equation 3.2, we can obtain $\mathbf{PPR}(x_i) = \alpha(\mathbf{I}_n - (1-\alpha)\hat{\mathbf{T}}^{-1}x_i)$. By substituting the teleport vector x_i with identity matrix $\mathbf{I}_n$, we can obtain a fully personalized PageRank matrix $\mathbf{P} = \alpha(\mathbf{I}_n - (1-\alpha)\hat{\mathbf{T}}^{-1})$. Finally, we normalize the matrix $\mathbf{P}$ along each columns to obtain normalized graph diffusion matrix $\mathbf{N}$ as follows,

$$\mathbf{N}_{Pij} = \frac{\mathbf{P}_{ij}}{\sum_i \mathbf{P}_{ij}} \tag{3.6}$$

The graph diffusion matrix $\mathbf{N}$ is used in the contrastive graph learning framework to obtain the representation vector by capture the graph structure. The graph diffusion convolution operator is formulated as follows,

$$\mathbf{H} = \sigma(\mathbf{N}_P \cdot \mathbf{X}\mathbf{\Theta}) \tag{3.7}$$

where $\mathbf{\Theta} \in \mathbb{R}^{N \times d}$ is a trainable matrix which usually initialized with random values, $\mathbf{X} \in \mathbb{R}^{(N+136)\times N}$ is the input feature matrix which is the concatenation of assembly graph and composition information, σ denotes a nonlinear activation function, here we use Parametric Rectified Linear Unit(PReLU) (He et al., 2015) which formulate as follows,

$$\sigma(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases} \tag{3.8}$$

where a_i is the trainable coefficient to control the slope when the input y is negative.

Finally, the vector representation $\mathbf{H}$ is obtained with the GDC operator by using assembly graph and composition information as inputs. The representation is used in the later steps to cluster the contigs.

3.3 Bipartite graph matching for constrained nodes

3.3.1 Order single-copy marker genes

Single-copy marker genes are the that marker genes usually appear just one time in a bacteria genome. Two contigs with the same marker genes are highly likely belong to the different species. Given single-copy marker genes, we know the corresponding contigs that contain them. Then order the marker genes by the descending order of the total number of single-copy marker genes that its related contigs contain (refers to step 1 and step 2 of Figure 3.3). The first single-copy marker gene indicates the related contigs

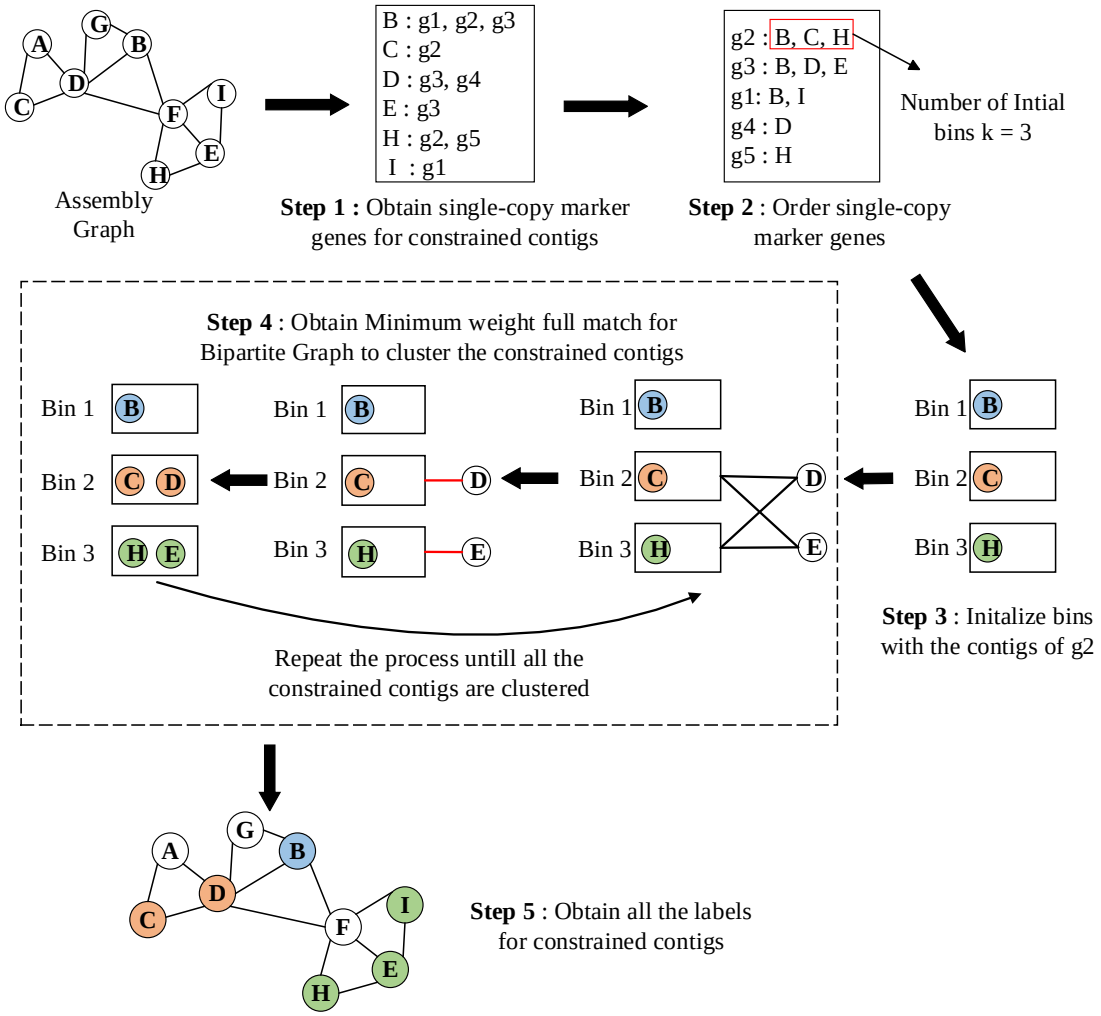

Figure 3.3: Bipartite graph Matching workflow for constrained nodes

contain the highest total number of single-copy marker genes. For example, as show in the step 2 of Figure 3.3, the first single copy marker gene is g2 and three contigs contain g2. The three contigs are B, C ,H which contain 4 single-copy marker genes(g1,g2,g3,g5) in total. The ordered marker genes are used to initialize bins.

3.3.2 Initialize bins

The contigs with constraints are clustered first. A contig is constrained refers to that it contains the same single-copy marker gene with the other contigs. Note that the constrained contigs have to satisfy the constraints which means each contig can not be

put in the same bin with the other contigs that contain the same single-copy marker gene with it. The rule of cluster the contigs are (1) Satisfy the constraints. (2) Cluster the contigs with the most similar representations together. The contigs with similar representations shows they share similar composition information and similar homophily information in assembly graph.

Then the single-copy marker genes are ordered in descending order to find the first marker gene. Then denote the size of first marker gene as k (refer to step 2 in Figure 3.3). Then initialize k empty bins and put each of those contigs in each of the empty bin (refer to step 3 in Figure 3.3).

3.3.3 Bin contigs with bipartite graph matching

To cluster the rest constrained contigs, we construct a bipartite graph G. Bipartite graph is a graph with two independent and disjoint sets U and V that the vertices in U and V are connected by one or more edges with or without weight. A matching exists in bipartite graph if there exist one edge connects one vertex in U and one vertex in V.

To obtain a minimum weight full match in bipartite graph if each vertex in U connects to one distinct vertex in V when $|U| = |V|$ and sum of the edges weights are minimised. Every vertices in U and V are connected by one distinct global minimum weight edges. If $|U| < |V|$, each vertex in U need to connect to one distinct vertex in V that minimise the weight to obtain minimum weight full match, and vice versa.

The bipartite graph is suitable for the data structure of the constraints because each set of constrained contigs(i.e contigs that contain the same single-copy marker gene) can not be put in the same bin. It follows the matching rule of the bipartite graph that each vertex in one set can only match to distinct vertex in the other set. We can treat one set of bipartite graph as the bins and another set as the constrained contigs to obtain a match between the bins and constrained contigs with the minimum edge weight. Then the contigs are put in the matched bins respectively.

A bipartite graph is constructed for each set of constrained contigs and targeted bins. If the given contig not conflict with the contigs in the bin, we connect this contig and the bin with a weighted edge. The edge weight between a bin and contig is calculated by the Euclidean distance between the two representation vectors of bin and contig. The representation vector of the contig is learned before. The representation vector of the bin is calculated by the average of the representation vectors of all the contigs in this bin. Then obtain a minimum weight full matching for the bipartite graph to place each contig into the closest bin that satisfy the constrains (refer to step 4 in Figure 3.3).

We repeat the process of constructing the bipartite graph with bins and constrained contigs until all the contigs are placed in bins. This approach can avoid any conflict between the constrained contigs. The contigs with the same single-copy marker gene are put in the different bins. All the contigs that are share the similar representations are put in the same bin. We use the Euclidean distance to determine if two contigs are

similar.

3.4 GCN-based label propagation

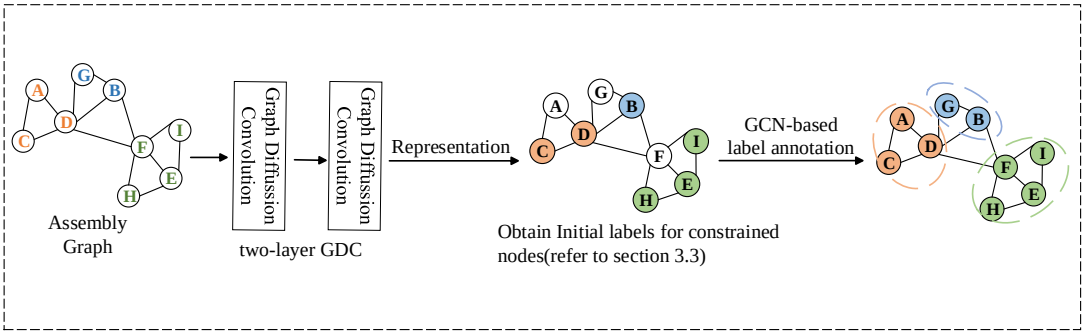

Figure 3.4: GCN-based label propagation

After handling the constrained nodes, we obtained initial labels for constrained contigs. The rest unconstrained nodes are labeled by two-layer graph diffusion convolutional operator (refer to Figure 3.4). As aforementioned, GDC aggregates the messages from the nearby neighbors of each node which essentially is a form of Laplacian smoothing. Such smoothing operation makes the node representations in the same cluster similar to each others which makes the subsequent classification task easier. However, it may also causes the potential over-smoothing problem. If the GDC with many convolution layers, the output representations maybe over-smoothed. A deep GDC also requires a substantial amount of labeled training data and training time. Therefore, here we choose to use two-layer graph diffusion convolution layer operator which is similar to the equation 3.7, it is formulated as follows:

$$\begin{aligned} \mathbf{Z} &= \sigma(\mathbf{N}_S \cdot \mathbf{H}\Theta^{'}) \\ &= \sigma(\mathbf{N}_S \cdot (\sigma(\mathbf{N}_S \cdot \mathbf{X}\Theta))\Theta^{'}) \end{aligned} \tag{3.9}$$

The loss is evaluated with cross-entropy function over all initialed labeled contigs which is formulated as follows,

$$\mathcal{L} = -\sum_{l \in Y_m} \sum_{k=1}^{K} Y_{lk} \ln \mathbf{Z}_{lk} \tag{3.10}$$

where Y_m is the set of labeled constrained contigs, k is the bin size. Then optimize it with back propagation.

Finally, all the labels are obtained by propagating the known labels to the neighbors as show in the last step of Figure 3.4.

3.5 Chapter Summary

In this chapter, the detailed approach of MixBin is explained and general framework is provided in Figure 3.1, where the four steps are summaised as follows:

1. **Preprocessing :** load the composition information, single-copy marker genes and assembly graph as inputs to MixBin. The get rid of the isolated nodes in assembly graph.
2. **Representation Learning :** train a contrastive encoder to encodes the assembly graph and composition information into vector representations.
3. **Bipartite graph matching for constrained nodes :** use the single-copy marker genes and learned vector representations to obtain the initial labels of the constrained nodes
4. **GCN-based label propagation :** propagate the known labels to the unlabelled neighbors.

Chapter 4

Experiments

4.1 Datasets

To evaluate the our model `MixBin`, Five simulated datasets are used in our experiments, which are generated based on the species that are found in the simMC+ dataset (Wu et al., 2014; Wu and et al., 2015). The detailed information of the datasets are collected in Table 4.1. We simulated the paired-end reads with 300 bp mean read length using the tool InSilicoSeq (Gourlé et al., 2019). metaSPAdes (Nurk et al., 2017) is used to assemble these simulated reads. We use FragGeneScan (Rho et al., 2010) and HMMER (Eddy, 2011) to identify the contigs that contain single-copy marker genes. The Sim-5G contains 5 species, Sim-10G contains 10 species, the rest data sets are in the similar fashion.

Table 4.1: Information of five simulated datasets.

Datasets	Read length (bp)	Number of paired-end reads	Number of assembled contigs	Mean contigs length (bp)	Number of links	Number of constraints	Number of species in ground truth
Sim-5G	300	2,000,000	519	51,723	2,488	91	5
Sim-10G	300	6,999,998	920	47,279	4,210	67	10
Sim-20G	300	15,000,001	1,452	48,021	6,531	75	20
Sim-50G	300	20,477,955	5,088	28,680	18,808	85	50
Sim-100G	300	51,167,221	15,729	19,978	62,518	85	100

4.2 Evaluation metrics

To evaluate the model, we applied four common evaluation metrics which are precision, recall, F1-score, Adjusted Rand Index(ARI) and Normalized Mutual Information (NMI). These four metrics are commonly used in previous binning approaches (Alneberg et al., 2014; Mallawaarachchi et al., 2020a; Xue et al., 2022) Following are the equations and definitions of the five metrics.

Precision. For each bin k, we obtain the species s by the maximum number of contigs in this bin and denote it as $max_s\{a_{ks}\}$. Then, we sum the $max_s\{a_{ks}\}$ of each bin k and divide it by the total number of contigs that are assigned in all the bins. The resulting value is precision. The equation of precision is defined as follows. Precision represents the fraction of the contigs of the same specie are assigned in the same bin.

$$Precision = \frac{\sum_k max_s\{a_{ks}\}}{\sum_k \sum_s a_{ks}} \tag{4.1}$$

Recall. For each specie k, we obtain the bin k by the maximum number of contigs in this specie and denote is as $max_k\{a_{ks}\}$. Then, we sum $max_k\{a_{ks}\}$ of each species and divide it by the total number of the contigs(binned and unbinned contigs). The resulting value is recall and it is defined as follows. Recall represents the fraction of the contigs that are assigned in the same bin are belong to the same specie.

$$Recall = \frac{\sum_s max_k\{a_{ks}\}}{(\sum_k \sum_s a_{ks} + U)} \tag{4.2}$$

F1-score. The F1-score is the harmonic mean of precision and recall. It is calculated as follows.

$$F1 = 2 \times \frac{Recall \times Precision}{Recall + Precision} \tag{4.3}$$

Adjusted Rand Index (ARI). The Adjusted Rand Index (ARI) is a measure of similarity between the binning result and its actual grouping. It is calculated as follows.

$$ARI = \frac{\sum_{k,s} \binom{a_{ks}}{2} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3} \tag{4.4}$$

$$where\ t_1 = \sum_k \binom{\sum_s a_{ks}}{2},\ t_2 = \sum_s \binom{\sum_k a_{ks}}{2},\ and\ t_3 = \frac{t_1 t_2}{\binom{N}{2}}$$

Normalized Mutual Information (NMI). The Normalized Mutual Information calculate the mutual information to evaluate the performance of contigs binning. The equation can be calculated as follow:

$$NMI = \frac{2 \times \sum_{k,s} A_{k,s} \log(\frac{N \times A_{k,s}}{A_{k,.} \times A_{.,s}})}{\sum_s A_{.,s} \log \frac{A_{.,s}}{N} + \sum_k A_{k,.} \log \frac{A_{k,.}}{N}} \tag{4.5}$$

We use the F1-score, ARI, and NMI to evaluate the machine learning-based baselines to evaluate the performance of `MixBin`. We use the Precision, Recall, and F1-score to

evaluate the binning results. As aforementioned, existing binning tools usually discard many short contigs. Hence, those contigs are ignored during the binning process. ARI and NMI are not used as evaluation metrics for experiments of binning tools to avoid the possible biases of binned contigs. The number of bins that the binning tools predicted are also included in the results Table 4.4.

4.3 Baselines.

4 unsupervised Graph Neural Networks models, 4 graph clustering models, and 8 metagenomic binning tools are selected as baselines to evaluate the performance of our proposed `MixBin` model.

i) Graph Neural Networks (GNN): These GNN models can only capture the graph structure of the assembly graph in an unsupervised manner.

- GraphSAGE (Hamilton et al., 2017) iteratively samples and aggregates messages from neighbors, which generates representations for nodes in an inductive manner.
- GAT (Veličković et al., 2018) aggregates messages from neighbors with different weights by employing a masked self-attention mechanism in the learning process.
- DGI (Veličković et al., 2019) is a contrastive graph learning model. It generates low-dimensional features for nodes by maximizing the mutual information between local-level and global-level features to capture the graph structure.
- VGAE (Kipf and Welling, 2016) learns the latent representation for nodes by minimizing the error between reconstructed and original graph, which is a generative graph learning model.

ii) Graph Clustering: These models can simultaneously learn the structural information of the graph and cluster graph-structural data.

- O2MAC (Fan et al., 2020) is a joint training model which mainly contains a generative graph learning and self-training clustering model.
- AGC (Zhang et al., 2019b) captures the high-order structural information of the graph by an adaptive graph convolutional learning model and clusters nodes using a spectral clustering algorithm.
- CSC (Wang et al., 2014) introduces constraints into the optimization process of spectral clustering and proposes a constraint-based spectral clustering algorithm.
- DCC (Zhang et al., 2019a) is a deep constrained clustering model which combines a deep learning model with constrained clustering together.

iii) Metagenomic Contigs Binning: These methods are state-of-the-art metagenomic binning models which use the information of sequence and the assembly graph.

- MetaWatt (Strous et al., 2012) combines the tetra-nucleotide frequencies and interpolated Markov models together to bin contigs, which is a multivariable statistics model.
- CONCOCT (Alneberg et al., 2013) proposes a mixture model to learn both composition of sequence and coverage across multiple samples to improve binning results.
- MaxBin2 (Wu and et al., 2015) encodes the tetra-nuncleotide frequencies and coverage information as features and cluster contigs by using an Expectation Maximization algorithm.
- BusyBeeWeb (Laczny et al., 2017) is a online web application which can bin and annotate metagenomic data in a bootstrapped supervised manner.
- SolidBin (Wang et al., 2019) proposes a semi-supervised spectral normalized cut algorithm by integrating pairwise constraints to cluster metagenomic contigs.
- MetaBAT2 (Kang and et al., 2019) iteratively partition contigs by integrating distances of genome abundance and normalized tetra-nuncleotide frequencies.
- GraphBin (Mallawaarachchi et al., 2020a) creatively introduces the assembly graph into the contigs binning and uses a label propagation algorithm to further improve the binning results of existing sequence-based binning tools.
- VAMB (Nissen and et al., 2021) is a deep learning-based binning method which learn the coabundance of sequence and k-mer distribution with a variational autoencoder. Then bin contigs with a iterative clustering model.
- RepBin (Xue et al., 2022) proposes a contrastive graph representation learning model, which can learn both the structural information of assembly graph and the prior information of the single-copy marker genes, and a GCN-based label propogation model to annotate contigs.

4.4 Experimental setup

Both MixBin and baselines are repeatedly run five times to avoid any experimental biases. We report the binning results by the mean and standard deviation. Three categories of baselines are employed in our paper, unsupervised graph neural network models, graph clustering approaches, and metagenomic binning tools. For these selected GNNs, we ran these baselines in an unsupervised manner to learn the topological information of the assembly graph. After obtaining the low-dimensional features for each contig, we run K-Means algorithms directly to get the final binning results. For Graph clustering approaches, we implement these models directly to obtain final results of binning contigs.

Detailed parameters. We need initial number of bins for ablation study and MixBin-learning. The initial number of bins is different for different data sets (5 bins for Sim-5G, 10 bins for Sim-10G, 20 bins for Sim-20G, 50 bins for Sim-50G, and 100 bins for Sim-

100G). The dimension of the latent representation is 32, which is set according to our experience. The parameter α in the graph diffusion operator range from 0.005 to 0.1. We also analyze the robustness of the parameter of the dimension d and probability of transferring α in the parameter analysis part.

Running environment. MixBin is implemented in Pytorch 1.8 and Python 3.6 with the Linux server with Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz and NVIDIA Tesla V100 SXM2 32 GB.

4.5 Benchmarking against Machine Learning Models.

Results against Graph Neural Networks.
We benchmark the results of MixBin with 4 graph Neural network models (i.e Graph-SAGE, GAT, DGI, VGAE). Both MixBin-*Learning* and GNN models cluster the contigs with K-Means algorithm after learning the representations for nodes. Table 4.2 shows the MixBin-*Learning* significantly outperform all the GNN baselines. The highest score archived by GNN baselines are highlighted by underline. We use F1-score, NMI and ARI to evaluate the performance. MixBin-*Learning* archives the highest score across all the metrics and data sets. Take Sim-50G as an example, MixBin-*Learning* achieves 69.38% for F1, 53.68% for ARI and 79.48% for NMI, the best performed GNN model(GAT) achieves 64.3% for F1, 44.7% for ARI and 75.9% for NMI. The differences results between GNN models and MixBin-*Learning* shows the strength of our model in graph structure learning.

Results against Graph Clustering.

We benchmark the results of binning contigs of MixBin with 4 graph clustering methods (O2MAC, AGC, CSC, DCC). The graph clustering methods can simultaneously learn the structural information of the graph and cluster graph-structural data. Table 4.3 shows the MixBin significantly outperform all the graph clustering methods. The highest score archived by graph clustering methods are highlighted by underline. We use F1-score, Ari and NMI to evaluate the performance. For Sim-20G, MixBin achieves 97.78% for F1, 96.20% for ARI and 97.06% for NMI, the best results achieved by clustering method are 83.0% for F1, 75.9% for ARI and 83.1% for NMI. The two constrain-based graph clustering methods (CSC,DCC) have lower results than MixBin due to fail to capture graph structure well.

Table 4.2: The results of MixBin and GNN models on five simulated datasets for contigs binning (%).

Datasets		Graph Neural Networks				MixBin-*Learning*
		GSAGE	**GAT**	**DGI**	**VGAE**	
Sim-5G	F1	88.0±0.6	94.5±1.6	79.9±3.4	85.7±1.4	95.04±0.09
	ARI	72.9±0.9	86.9±1.7	54.6±6.8	70.1±2.6	91.00±0.17
	NMI	81.6±0.8	87.7±0.7	68.2±3.3	80.1±3.2	90.92±0.42
Sim-10G	F1	76.6±0.2	73.7±0.5	68.1±1.9	71.4±1.9	92.23±2.19
	ARI	59.3±0.7	54.0±2.6	39.3±2.4	46.0±2.6	84.65±5.56
	NMI	75.9±0.6	74.3±0.4	61.8±2.4	68.9±1.1	91.17±2.14
Sim-20G	F1	77.4±0.6	79.8±1.0	63.9±2.5	72.2±1.7	85.19±2.40
	ARI	61.5±1.8	65.4±1.3	40.1±2.3	54.8±2.1	73.20±3.74
	NMI	81.5±0.3	83.6±0.7	65.8±2.5	75.9±1.2	86.37±1.77
Sim-50G	F1	58.8±0.1	64.2±0.8	51.4±0.3	62.1±1.2	69.38±3.27
	ARI	40.9±1.7	44.7±1.8	37.1±1.1	41.1±0.5	53.68±3.44
	NMI	72.6±0.2	75.9±0.2	64.2±0.7	68.7±0.8	79.48±1.98
Sim-100G	F1	46.7±0.5	50.1±0.1	20.8±0.5	37.6±1.1	54.10±2.23
	ARI	31.2±0.8	26.9±0.3	21.7±0.8	25.8±0.6	37.90±2.88
	NMI	68.2±0.2	66.8±0.8	51.6±0.4	62.3±0.9	71.70±1.30

Table 4.3: The results of MixBin and graph clustering methods on five simulated datasets for contigs binning (%).

Datasets		Graph Clustering				MixBin
		O2MAC	**AGC**	**CSC**	**DCC**	
Sim-5G	F1	74.9±3.5	80.9±0.4	96.7±0.0	90.9±0.0	99.69±0.18
	ARI	63.6±2.1	92.7±0.8	87.9±0.0	94.0±1.0	99.11±0.42
	NMI	72.5±3.1	90.4±0.5	91.5±0.0	88.6±1.1	98.75±0.68
Sim-10G	F1	65.8±1.4	78.3±0.3	90.9±1.3	92.1±2.9	99.55±0.00
	ARI	53.5±2.1	87.9±0.3	77.9±4.2	83.3±3.0	99.39±0.08
	NMI	69.1±1.7	89.6±0.7	85.1±1.7	77.3±2.3	99.20±0.05
Sim-20G	F1	61.0±3.4	67.0±0.2	83.0±1.4	82.1±1.9	97.78±0.05
	ARI	52.0±2.7	75.9±1.1	63.2±4.3	65.3±2.8	96.20±0.11
	NMI	71.1±1.8	82.0±0.5	83.1±1.1	75.1±3.3	97.06±0.02
Sim-50G	F1	37.1±0.5	54.9±0.2	86.8±0.8	64.4±5.7	87.62±1.91
	ARI	16.5±0.3	44.4±0.8	77.0±2.6	48.7±4.1	85.49±2.83
	NMI	59.8±0.8	79.3±0.1	90.3±0.5	51.3±4.0	91.54±0.99
Sim-100G	F1	29.0±2.0	50.5±0.4	72.5±0.7	57.1±3.3	71.97±0.73
	ARI	8.5±0.1	21.1±0.5	37.5±7.9	40.9±7.9	58.58±0.80
	NMI	58.9±0.1	72.0±0.2	81.0±1.7	55.4±0.9	82.51±0.37

4.6 Benchmarking against Metagenomic Binning Models

Table 4.4 shows the contigs binning results of MixBin, GraphBin and 6 other stand-along binning tools on 5 data sets. As GraphBin has to be used with other binning tools, we provide the results of GraphBin combined with 6 other stand-along binning tools on Sim-20G data set.

Table 4.4 shows MixBin outperform MetaWatt,CONCONCT, MaxBin2, BusyBeeWeb, MetaBTA2, SolidBin and VAMB on all data sets. Take Sim-50G as an example, the highest recall and F1-score is archived by RepBin(90.59% and 84.55%) which is lower than MixBin (92.49% for recall, 87.62% for F1-score). The highest precision (83.24 %) is archived by MixBin which is slight lower than VAMB(84.22% for precision). MixBin has significantly higher recall and F1-score than VAMB (39.32% for recall, 55.37% for F1-score). MixBin has slightly lower recall and F1-score than the combination of GraphBin and RepBin on Sim-20G data sets.

MixBin outperform RepBin on all the data sets except the Sim-100G data set. RepBin archives 66.42% for precision, 83.79% for recall, 74.08% for F1-score which about 3% higher than MixBin (64.39% for precision, 81.63% for recall, 71.97% for F1-score). This is because MixBin rely on the quality of single-copy maker genes. The single-copy marker genes of Sim-100G indicate there are 91 species in the data sets. Hence, MixBin cluster contigs into 91 bins. However, there are 100 species based on the ground truth. Overall, MixBin beats the majority of the state-of-art models.

4.7 Visualization of the Assembly Graph.

The python-iGraph package is used to visualize the binning results in the assembly graph of the Sim-10G data set. There are visualisation of ground truth and 8 other stand-along binning tools (refer to Figure 4.1). Different colours represents different species(bins) that clustered by the correspondent tool. The grey nodes are the contigs that are either discarded or not identified during the binning process. k represents the number of species(bins) that corresponding tool able to identify and the numbers of colour in the graph. The grey edges represent the constraints of the single-copy marker genes. The black edges represent the homophily information of the assembly graph. As shown in the Figure 4.1, MixBin able to achieve almost the same label as the ground truth. CONCOCT, MaxBin2, MetaBAT2, MetaWatt and VAMB identify more bins than the ground truth. MetaBAT2 fail to label almost half of the contigs because it discard the contigs that shorter than 1,500 base pairs. It is same for BusyBeeWeb, CONCONCT, MaxBin2, MetaBAT2, SolidBin and VAMB as they all discard the contigs that are shorter than certain length. MixBin able to bin the contigs with any length, thus majority of the contigs are labeled by MixBin.

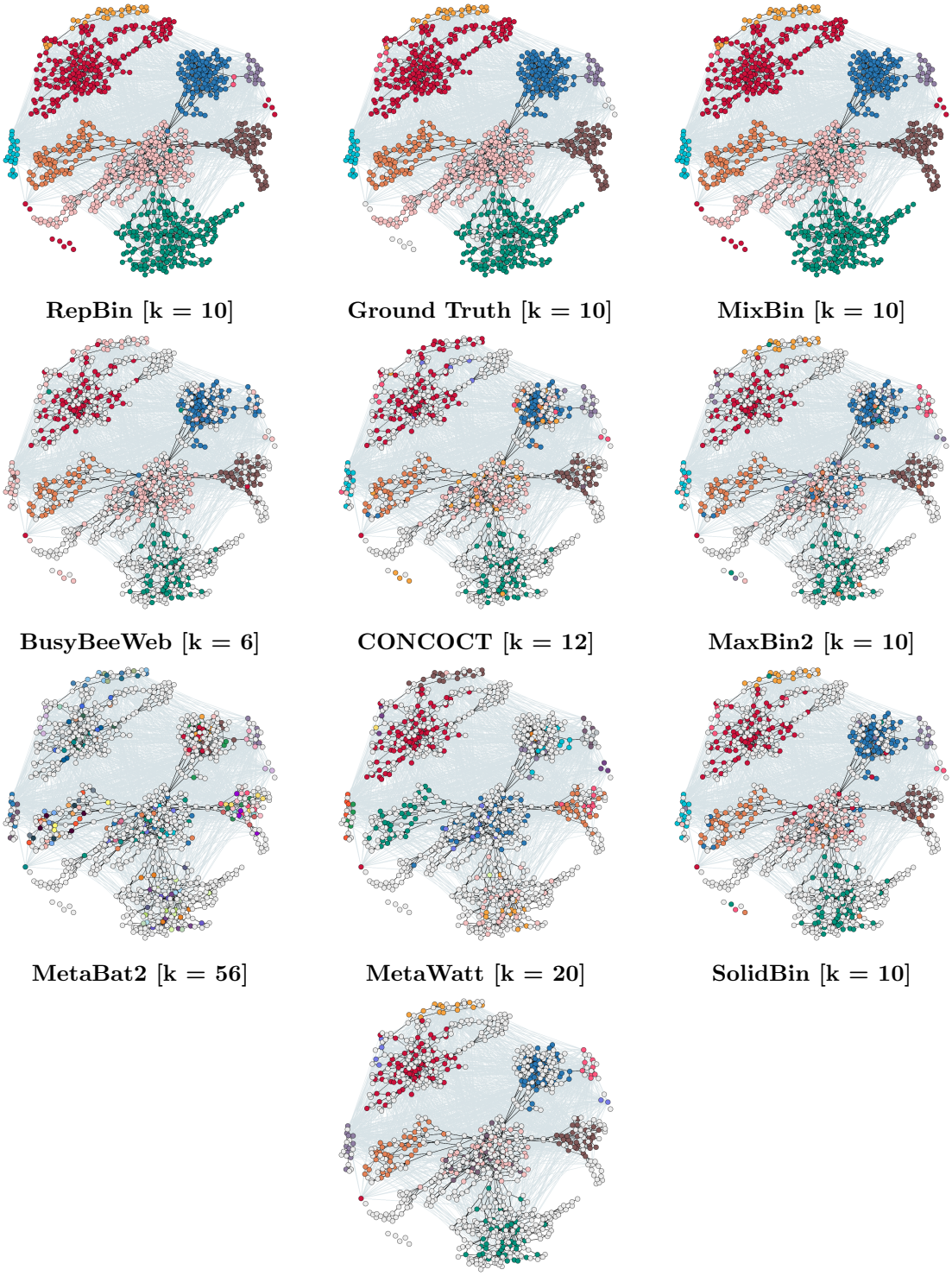

Figure 4.1: Sim-10G Assembly Graph Visualization.

4.8 Further Analysis

Ablation Study

To evaluate the effectiveness of utilising constraints in bipartite graph matching and the effectiveness of the combination of bipartite graph matching and GCN-based label propagation. A ablation study is conducted to examine the variant of the MixBin. *MixBin without matching* is the variant of MixBin that use K-Means algorithm to initialize the bins in stead of incorporating constraints in bipartite graph matching to initialize the bins. *MixBin-learning* is another variant of the MixBin that cluster the all contigs use K-Means algorithm after learning the representations for nodes. Figure 4.2 and Figure 4.3 shows incorporate constraints in bipartite graph matching in MixBin improves the binning results (the red bar is higher than the yellow bar). Besides, incorporate bipartite graph matching and GCN-based label propagation in the MixBin significantly improves the binning results compare to use K-Means to cluster the contigs(read bar is much higher than the blue bar).

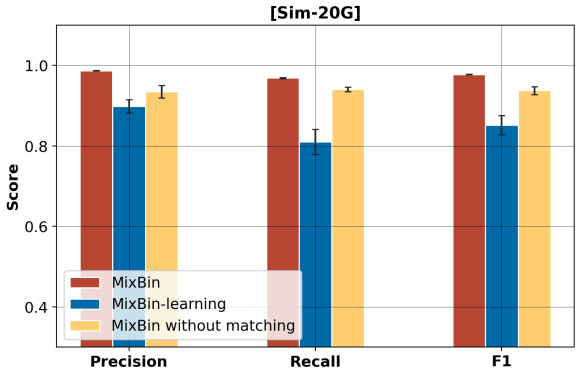

Figure 4.2: Results of MixBin, MixBin-learning, and MixBin without matching

Parameters Analysis

The parameter α is the teleport probability that used in the graph diffusion convolution (equation 3.5) to adjust the probability of teleporting to another node. We experiment with parameter α change from 0.005 and 0.1 for the Sim-20G data set to see how does parameter a affect the the binning results. We evaluate the binning results with precision, recall and F1-score. As shown in the Figure 4.4, MixBin is not sensitive to the changes of alpha.

The parameter d represents the hidden dimension of the graph diffusion convolution of the representation learning part. We experiment with parameter d that ranges from 16, 32, 64 to 128 dimensions on Sim-20G data set. As shown in Figure 4.5 below, the binning results are steady. We can see that MixBin is not sensitive to the changes in dimension parameter.

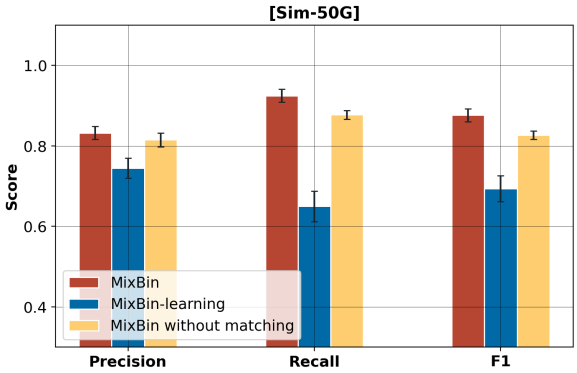

Figure 4.3: Results of MixBin, MixBin-learning, and MixBin without matching

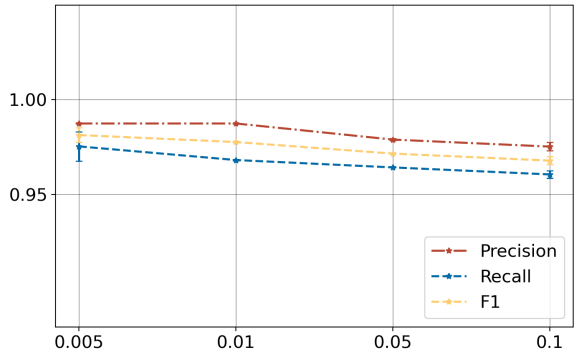

Figure 4.4: Alpha parameter of MixBin.

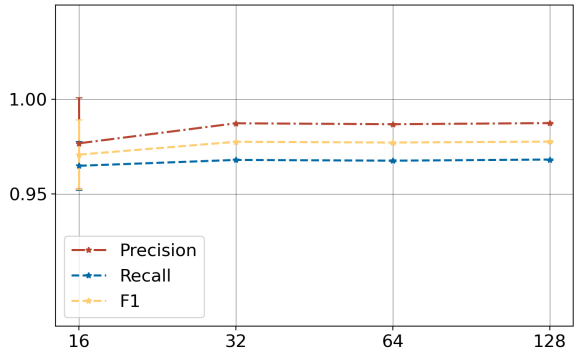

Figure 4.5: Dimension parameter of MixBin.

Running Time
We collect running time of the MixBin and baselines to cluster the contigs of Sim-20G data set. Figure 4.6 shows the CPU running time of the MixBin and baselines. To be noted, MixBin, RepBin, and VAMB are implemented on the GPU server, and other baselines are CPU-based algorithms. GraphBin is a refinement binning tool that is used based on the other binning tools. MixBin is a relatively fast tool with 25.99 seconds. Although MixBin is not the fastest tool, it has better performance than MetaBAT2, CONCONCT, SolidBin, VAMB and MetaWatt.

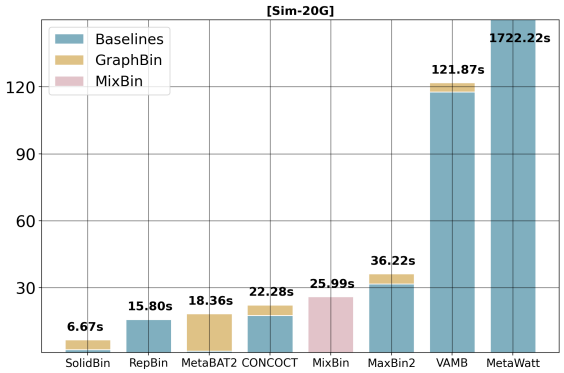

Figure 4.6: The running time of Sim-20G dataset.

4.9 Chapter Summary

In this chapter, we conduct extensive experiments of benchmarking MixBin and state-of-art metagenomic binning tools.

In section 4.1, we describe five data sets that are used in the experiments. In section 4.2, we describe and formulate five evaluation metrics(precision,recall,F1-score,ARI,NMI) that used to evaluate the binning results. In section 4.3, we list all the baselines along with the brief explanation of their key methods. In section 4.4, we describe the running environment of the experiments.

In section 4.5, we benchmark the MixBin against the machine learning models(GNN, graph clustering methods) on five simulated data sets. The results show MixBin has the best performance among all GNN models on all the data sets, and slightly lower F1-score than the DCC on the Sim-100G data set. MixBin has the highest score among the rest graph clustering models on all the data sets.

In section 4.6, we benchmark the MixBin against the state-of-art metegenomic binning tools. The results show the superior of MixBin. In section 4.7, we visualize the binning results of Sim-10G in assembly graph.

In section 4.8, we conduct ablation study of variant of MixBin, parameter analysis and CPU running time analysis of MixBin. The results of parameter analysis show MaxBin is not sensitive to the changes of parameter. The results of ablation study show incorporate constraints in bipartite graph matching improves the binning results. The results of the CPU running time show MixBin require relative short time to bin contigs.

Table 4.4: The experimental metrics of MixBin and baselines on five data sets for binning contigs (%).

Datasets		MetaWatt	CON COCT	MaxBin2	BusyBee Web	MetaBAT2	SolidBin	VAMB	RepBin	MixBin
Sim-5G	Precision	100.00	91.60	91.13	86.57	100.00	90.00	100.00±0.00	99.69±0.10	99.69±0.18
	Recall	24.59	40.50	46.69	49.79	6.61	46.49	33.92±0.90	99.69±0.10	99.69±0.18
	F1	39.47	56.16	61.75	63.22	12.40	61.31	50.66±1.02	99.69±0.10	99.69±0.18
	Pred. bins	12	7	5	4	34	5	6	5	5
Sim-10G	Precision	99.29	86.99	89.43	84.47	100.00	91.58	99.93±0.15	99.20±0.00	99.52±0.05
	Recall	26.13	39.72	40.30	45.53	6.39	41.70	33.80±0.20	99.55±0.08	99.57±0.05
	F1	41.38	54.54	55.56	59.17	12.01	57.30	50.51±0.23	99.37±0.04	99.55±0.00
	Pred. bins	20	12	10	6	56	10	11	10	10
Sim-20G	Precision	96.85	84.02	88.25	77.39	96.77	96.51	99.35±0.10	97.31±0.31	98.72±0.03
	Recall	32.01	42.27	41.69	44.51	7.73	85.04	36.88±0.60	96.98±0.69	96.86±0.13
	F1	48.12	56.24	56.63	56.52	14.32	90.41	53.79±0.64	97.15±0.61	97.78±0.05
	Pred. bins	33	22	21	12	88	20	22	20	20
Sim-50G	Precision	79.26	63.22	66.78	8.58	78.41	77.52	84.22±0.73	80.31±0.48	83.24±1.63
	Recall	17.65	38.76	40.89	4.21	5.67	38.67	39.32±0.45	90.59±2.01	92.49±2.42
	F1	41.42	47.65	51.23	5.65	11.32	51.60	55.37±1.56	84.55±1.80	87.62±1.91
	Pred. bins	75	56	53	12	98	45	48	50	47
Sim-100G	Precision	63.22	52.31	54.78	50.95	67.32	77.93	65.31±1.21	66.42±1.72	64.39± 0.72
	Recall	15.73	22.61	27.62	41.69	4.81	12.22	32.29±0.39	83.79±2.02	81.63± 2.43
	F1	32.34	34.59	36.73	45.86	9.63	21.12	45.21±0.79	74.08±0.74	71.97±0.73
	Pred. bins	157	132	127	28	256	84	87	100	92
	Stand-alone Binning Tools + GraphBin									
Sim-20G	Precision	98.02	92.96	96.77	90.27	96.77	96.51	98.15±0.04	97.31±0.31	98.72±0.03
	Recall	68.71	81.86	83.96	91.84	7.73	85.04	86.35±2.21	96.98±0.69	96.86±0.13
	F1	80.79	87.06	89.91	91.05	14.32	90.41	91.86±1.25	97.15±0.61	97.78±0.05
	Pred. bins	33	22	21	12	88	20	22	20	20

Chapter 5

Concluding Remarks

5.1 Conclusion

In this thesis, we have studied how to simultaneously use the assembly graph, composition and coverage information, and single-copy marker genes to improve the results of metagenomic contigs binning. Existing binning tools have three challenges, (1) discard short sequences when binning contigs; (2) stand-alone binning approaches neglect the homophilious relationships of contigs; (3) ignore the implicit information of the single-copy marker genes. To solve these challenges, we propose a contrastive graph representation learning over the assembly graph meanwhile inputting the composition information of the sequences. Then, we introduce a bipartite graph matching algorithm to mine the implicit information of the single-copy marker genes. Finally, a constraint-based graph label propagation model are designed to bin unannotated contigs. To evaluate the performance of the proposed MixBin model, we implement the MixBin on five simulated metagenomic datasets. Extensive experiments show that MixBin outperforms other unsupervised graph representation learning, graph clustering, and contigs binning algorithms in the task of binning metagenomic contigs.

5.2 Future Work

In this thesis, we integrate the assembly graph and the sequence information into the process of metagenomic binning simultaneously. We also propose a bipartite graph matching algorithm to mine the implicit relationships among different contigs. Extensive experiments show the superiority of combining the assembly graph and nucleotide sequence information, which can bin short contigs that are usually discarded by other stand-alone binning tools. The bipartite graph matching effectively improves the binning results of the constrained contigs and estimates the number of species in the mixed samples.

It has several directions of the metagenomics binning valuable to be extended according to the finds in this thesis. We summarized these possible directions in the following part.

- **Effective Assembly Graph Learning.** MixBin introduces a contrastive graph representation learning model to capture the topological information of the assembly graph meanwhile integrating the nucleotide sequence features. The input assembly graph, in which nodes are contigs and edges represent the homophilious relationships among different contigs, are different from the graph generated by the SPAdes algorithm. The nodes in the original assembly graph are reads and the edges denotes the overlapping relationships between different reads. Therefore, it is important and necessary to propose a graph representation learning over the original assembly graph, which can model the homophilious information of the read-overlapping graph and represent both reads and contigs as low-dimensional features. The initial investigation of learning the original read-overlapping graph has demonstrated its superior on binning metagenomic contigs.
- **Estimated Number of Species.** The number of species is estimated based on the constraints derived from the single-copy marker genes. Because each single-copy marker gene usually exists once at each species, the two contigs are prone to belong to distinct species if the same single-copy marker gene is contained by two different contigs. Thus, the number of contigs in each single-copy marker gene set should be equal to the number of species in the ideal case. However, in the real world, the number of contigs that contain the same marker gene is usually less than the number of true species in the sample, which makes the number of species estimation more difficult. One possible way is to explore the relations among different single-copy marker gene sets to obtain a more precise number of bins estimation.
- **Application to Viral Genomes.** Many bacterial, viral, and other DNA sequences can be attributed to the set of metagenomic data. The technique of metagenomics binning can be introduced into the area of mining viral genomes Kieft et al. (2021); Johansen et al. (2021); Song (2021). Most of existing binning algorithms used in analyzing viral genomes belong to the statistics category, which calculates the composition and coverage information of the viral sequences. One direction is to introduce the assembly graph and graph representation learning into the area of analyzing viral genomes.

Bibliography

Alneberg, J., Bjarnason, B. S., De Bruijn, I., Schirmer, M., Quick, J., Ijaz, U. Z., Lahti, L., Loman, N. J., Andersson, A. F., and Quince, C. (2014). Binning metagenomic contigs by coverage and composition. *Nature methods*, 11(11):1144–1146. [Cited on pages 2, 8, and 27.]

Alneberg, J., Bjarnason, B. S., de Bruijn, I., Schirmer, M., Quick, J., Ijaz, U. Z., Loman, N. J., Andersson, A. F., and Quince, C. (2013). Concoct: clustering contigs on coverage and composition. *arXiv preprint arXiv:1312.4038*. [Cited on pages 12, 13, 15, 18, and 30.]

Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., et al. (2012). Spades: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology*, 19(5):455–477. [Cited on pages 6 and 7.]

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. (2017). Minimum information about a single amplified genome (misag) and a metagenome-assembled genome (mimag) of bacteria and archaea. *Nature biotechnology*, 35(8):725–731. [Cited on page 2.]

Chen, P., Wu, Z., Zhao, Y., Wei, Y., Xu, R., Yan, L., and Li, H. (2016). Cultivation-independent comprehensive investigations on bacterial communities in serofluid dish, a traditional chinese fermented food. *Genomics data*, 7:127–128. [Cited on page 2.]

Davis, K. E., Sangwan, P., and Janssen, P. H. (2011). Acidobacteria, rubrobacteridae and chloroflexi are abundant among very slow-growing and mini-colony-forming soil bacteria. *Environmental microbiology*, 13(3):798–805. [Cited on page 1.]

Dick, G. J., Andersson, A. F., Baker, B. J., Simmons, S. L., Thomas, B. C., Yelton, A. P., and Banfield, J. F. (2009). Community-wide analysis of microbial genome sequence signatures. *Genome biology*, 10(8):1–16. [Cited on page 8.]

Eddy, S. R. (2011). Accelerated profile hmm searches. *PLoS Computational Biology*, 7. [Cited on pages 9, 18, and 27.]

Fan, S., Wang, X., Shi, C., Lu, E., Lin, K., and Wang, B. (2020). One2multi graph autoencoder for multi-view graph clustering. WWW '20, page 3070–3076. Association for Computing Machinery. [Cited on page 29.]

Frank, J. A., Pan, Y., Tooming-Klunderud, A., Eijsink, V. G., McHardy, A. C., Nederbragt, A. J., and Pope, P. B. (2016). Improved metagenome assemblies and taxonomic binning using long-read circular consensus sequence data. *Scientific reports*, 6(1):1–10. [Cited on page 8.]

Garza, D. R. and Dutilh, B. E. (2015). From cultured to uncultured genome sequences: metagenomics and modeling microbial ecosystems. *Cellular and Molecular Life Sciences*, 72(22):4287–4308. [Cited on page 1.]

Gleeson, T. J. and Staden, R. (1991). An x windows and unix implementation of our sequence analysis package. *Bioinformatics*, 7(3):398–398. [Cited on page 6.]

Gourlé, H., Karlsson-Lindsjö, O., Hayer, J., and Bongcam-Rudloff, E. (2019). Simulating illumina metagenomic data with insilicoseq. *Bioinformatics*, 35:521 – 522. [Cited on page 27.]

Hamilton, W. L., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *NeurIPS*. [Cited on page 29.]

Handelsman, J., Rondon, M. R., Brady, S. F., Clardy, J., and Goodman, R. M. (1998). Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chemistry & biology*, 5(10):R245–R249. [Cited on page 1.]

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. [Cited on page 21.]

Idury, R. M. and Waterman, M. S. (1995). A new algorithm for dna sequence assembly. *Journal of computational biology*, 2(2):291–306. [Cited on page 6.]

Johansen, J., Plichta, D. R., Nissen, J. N., Jespersen, M. L., Shah, S. A., Deng, L., Stokholm, J., Bisgaard, H., Nielsen, D. S., Sørensen, S. J., and Rasmussen, S. H. (2021). Genome binning of viral entities from bulk metagenomics data. *Nature Communications*, 13. [Cited on page 42.]

Kang, D. D. and et al. (2019). Metabat 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies. *PeerJ*, 7:e7359. [Cited on pages 12, 14, 15, and 30.]

Karlin, S. and Ladunga, I. (1994). Comparisons of eukaryotic genomic sequences. *Proceedings of the National Academy of Sciences*, 91(26):12832–12836. [Cited on page 8.]

Kieft, K., Adams, A. M., Salamzade, R., Kalan, L. R., and Anantharaman, K. (2021). vrhyme enables binning of viral genomes from metagenomes. *Nucleic Acids Research*, 50:e83 – e83. [Cited on page 42.]

Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning*. [Cited on page 29.]

Klicpera, J., Bojchevski, A., and Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*. [Cited on page 20.]

Klicpera, J., Weißenberger, S., and Günnemann, S. (2019). Diffusion improves graph learning. *arXiv preprint arXiv:1911.05485*. [Cited on page 20.]

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. (2020). metaflye: scalable long-read metagenome assembly using repeat graphs. *Nature Methods*, 17(11):1103–1110. [Cited on page 7.]

Kolmogorov, M., Yuan, J., Lin, Y., and Pevzner, P. A. (2019). Assembly of long, error-prone reads using repeat graphs. *Nature biotechnology*, 37(5):540–546. [Cited on page 6.]

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., and Phillippy, A. M. (2017). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736. [Cited on page 6.]

Köpke, B., Wilms, R., Engelen, B., Cypionka, H., and Sass, H. (2005). Microbial diversity in coastal subsurface sediments: a cultivation approach using various electron acceptors and substrate gradients. *Applied and environmental microbiology*, 71(12):7819–7830. [Cited on page 1.]

Laczny, C. C., Kiefer, C., Galata, V., Fehlmann, T., Backes, C., and Keller, A. (2017). Busybee web: metagenomic data analysis by bootstrapped supervised binning and annotation. *Nucleic Acids Research*, pages W171–W179. [Cited on pages 12, 13, 15, and 30.]

Land, M., Hauser, L., Jun, S.-R., Nookaew, I., Leuze, M. R., Ahn, T.-H., Karpinets, T., Lund, O., Kora, G., Wassenaar, T., et al. (2015). Insights from 20 years of bacterial genome sequencing. *Functional & integrative genomics*, 15(2):141–161. [Cited on page 8.]

Lander, E. S. and Waterman, M. S. (1988). Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, 2(3):231–239. [Cited on page 8.]

Li, D., Liu, C.-M., Luo, R., Sadakane, K., and Lam, T.-W. (2015). Megahit: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, 31(10):1674–1676. [Cited on page 6.]

Lin, H.-H. and Liao, Y.-C. (2016). Accurate binning of metagenomic contigs via automated clustering sequences using information of genomic signatures and marker genes. *Scientific reports*, 6(1):1–8. [Cited on page 9.]

Mallawaarachchi, V., Wickramarachchi, A., and Lin, Y. (2020a). Graphbin: refined binning of metagenomic contigs using assembly graphs. *Bioinformatics*, 36(11):3307–3313. [Cited on pages 2, 11, 15, 27, and 30.]

Mallawaarachchi, V. G., Wickramarachchi, A. S., and Lin, Y. (2020b). Graphbin2: refined and overlapped binning of metagenomic contigs using assembly graphs. In *20th International Workshop on Algorithms in Bioinformatics (WABI 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. [Cited on page 11.]

Meyer, F., Fritz, A., Deng, Z.-L., Koslicki, D., Lesker, T. R., Gurevich, A., Robertson, G., Alser, M., Antipov, D., Beghini, F., et al. (2022). Critical assessment of metagenome interpretation: the second round of challenges. *Nature methods*, 19(4):429–440. [Cited on page 7.]

Mou, X., Sun, S., Edwards, R. A., Hodson, R. E., and Moran, M. A. (2008). Bacterial carbon processing by generalist species in the coastal ocean. *Nature*, 451(7179):708–711. [Cited on page 2.]

Nissen, J. N. and et al. (2021). Improved metagenome binning and assembly using deep variational autoencoders. *Nature biotechnology*, 39(5):555–560. [Cited on pages 2, 15, and 30.]

Nurk, S., Meleshko, D., Korobeynikov, A., and Pevzner, P. A. (2017). metaspades: a new versatile metagenomic assembler. *Genome research*, 27(5):824–834. [Cited on pages 7, 12, 18, and 27.]

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab. [Cited on page 20.]

Pevzner, P. A., Tang, H., and Waterman, M. S. (2001). An eulerian path approach to dna fragment assembly. *Proceedings of the national academy of sciences*, 98(17):9748–9753. [Cited on page 6.]

Pulschen, A. A., Bendia, A. G., Fricker, A. D., Pellizari, V. H., Galante, D., and Rodrigues, F. (2017). Isolation of uncultured bacteria from antarctica using long incubation periods and low nutritional media. *Frontiers in microbiology*, 8:1346. [Cited on page 1.]

Qin, J., Li, R., Raes, J., Arumugam, M., Burgdorf, K. S., Manichanh, C., Nielsen, T., Pons, N., Levenez, F., Yamada, T., et al. (2010). A human gut microbial gene catalogue established by metagenomic sequencing. *nature*, 464(7285):59–65. [Cited on page 2.]

Rho, M., Tang, H., and Ye, Y. (2010). Fraggenescan: predicting genes in short and error-prone reads. *Nucleic acids research*, 38(20):e191–e191. [Cited on pages 9, 18, and 27.]

Saeed, I., Tang, S.-L., and Halgamuge, S. K. (2012). Unsupervised discovery of microbial population structure within metagenomes using nucleotide base composition. *Nucleic acids research*, 40(5):e34–e34. [Cited on page 8.]

Sanger, F., Nicklen, S., and Coulson, A. R. (1977). Dna sequencing with chain-terminating inhibitors. *Proceedings of the national academy of sciences*, 74(12):5463–5467. [Cited on page 5.]

Sedlar, K., Kupkova, K., and Provaznik, I. (2017). Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics. *Computational and Structural Biotechnology Journal*, 15:48–55. [Cited on page 8.]

Segata, N., Waldron, L., Ballarini, A., Narasimhan, V., Jousson, O., and Huttenhower, C. (2012). Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature methods*, 9(8):811–814. [Cited on page 10.]

Simon, C., Wiezer, A., Strittmatter, A. W., and Daniel, R. (2009). Phylogenetic diversity and metabolic potential revealed in a glacier ice metagenome. *Applied and environmental microbiology*, 75(23):7519–7526. [Cited on page 2.]

Simpson, J. T. and Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data structures. *Genome research*, 22(3):549–556. [Cited on pages 6 and 12.]

Song, K. (2021). Reads binning improves the assembly of viral genome sequences from metagenomic samples. *Frontiers in Microbiology*, 12. [Cited on page 42.]

Strous, M., Kraft, B., Bisdorf, R., and Tegetmeyer, H. (2012). The binning of metagenomic contigs for microbial physiology of mixed cultures. *Frontiers in Microbiology*, 3:410. [Cited on pages 12, 13, 15, and 30.]

Su, C.-H., Hsu, M.-T., Wang, T.-Y., Chiang, S., Cheng, J.-H., Weng, F. C., Kao, C.-Y., Wang, D., and Tsai, H.-K. (2011). Metaabc—an integrated metagenomics platform for data adjustment, binning and clustering. *Bioinformatics*, 27(16):2298–2299. [Cited on page 10.]

Sutton, G. G., White, O., Adams, M. D., and Kerlavage, A. R. (1995). Tigr assembler: A new tool for assembling large shotgun sequencing projects. *Genome Science and Technology*, 1(1):9–19. [Cited on page 6.]

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph Attention Networks. In *ICLR*. [Cited on page 29.]

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep Graph Infomax. In *ICLR*. [Cited on pages 20 and 29.]

Wang, X., Qian, B., and Davidson, I. (2014). On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, 28(1):1–30. [Cited on page 29.]

Wang, Z., Wang, Z., Lu, Y. Y., Sun, F., and Zhu, S. (2019). Solidbin: improving metagenome binning with semi-supervised normalized cut. *Bioinformatics*, 35(21):4229–4238. [Cited on pages 12, 13, 15, and 30.]

Wood, D. E., Lu, J., and Langmead, B. (2019). Improved metagenomic analysis with kraken 2. *Genome biology*, 20(1):1–13. [Cited on page 10.]

Wood, D. E. and Salzberg, S. L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, 15(3):1–12. [Cited on page 10.]

Wu, Y.-W. and et al. (2015). MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics*, pages 605–607. [Cited on pages 9, 12, 13, 15, 27, and 30.]

Wu, Y.-W., Tang, Y.-H., Tringe, S. G., Simmons, B. A., and Singer, S. W. (2014). Maxbin: an automated binning method to recover individual genomes from metagenomes using an expectation-maximization algorithm. *Microbiome*, 2(1):1–18. [Cited on pages 9, 13, and 27.]

Xue, H., Mallawaarachchi, V., Zhang, Y., Rajan, V., and Lin, Y. (2022). Repbin: Constraint-based graph representation learning for metagenomic binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4637–4645. [Cited on pages 2, 11, 15, 27, and 30.]

Zerbino, D. R. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829. [Cited on page 6.]

Zhang, H., Basu, S., and Davidson, I. (2019a). A framework for deep constrained clustering–algorithms and advances. [Cited on page 29.]

Zhang, X., Liu, H., Li, Q., and Wu, X.-M. (2019b). Attributed graph clustering via adaptive graph convolution. In *IJCAI-19*. [Cited on page 29.]