

Metagenomics Binning Using Assembly Graphs

Vijini Mallawaarachchi

A thesis submitted for the degree of Doctor of Philosophy

School of Computing
The Australian National University



August, 2022

Declaration

This dissertation is an account of research undertaken between August 2018 and February 2022 at the School of Computing, The Australian National University, Canberra, Australia.

The work presented in this thesis is that of the candidate alone, except where indicated by due literature reference and acknowledgements in the text. It has not been submitted in whole or in part for any other degree at this or any other university.

This thesis is based on five papers published in refereed scientific journals and conferences. The development of ideas and research was undertaken with guidance from my supervisor Dr Yu Lin and published papers presented in the thesis were written collaboratively with him and members of the ANU Computational Genomics Group. Below I describe my contributions to the research in each chapter and associated papers.

- Chapter 3 is the paper [Mallawaarachchi, Wickramarachchi et al. \(2020a\)](#) where my research introduces how to use assembly graphs to improve metagenomic binning results.
- Chapter 4 contains the papers [Mallawaarachchi, Wickramarachchi et al. \(2020b\)](#) and [Mallawaarachchi, Wickramarachchi et al. \(2021\)](#) where my research presents a new overlapped binning tool which can determine metagenomic sequences belonging to multiple species with the use of assembly graphs.
- Chapter 5 is the paper [Mallawaarachchi and Lin \(2022\)](#) where my research presents a stand-alone binning tool that makes use of composition, coverage, single-copy marker genes and assembly graphs to bin metagenomic contigs.
- Section 6.4.1 in Chapter 6 is based on the ideas from the co-authored paper [Xue et al. \(2022\)](#) where our research explores machine learning methods for metagenomics binning and shows how graph representation learning can be applied to bin metagenomic contigs with the use of assembly graphs.

Vijini Mallawaarachchi
9 August 2022

Acknowledgements

I acknowledge and celebrate the First Australians on whose traditional lands we meet, and pay my respect to the elders past, present and emerging.

I would like to extend my deepest gratitude to my primary supervisor Dr Yu Lin from the Australian National University (ANU) for the continuous support and guidance provided throughout the projects on problem solving, algorithm development, academic writing and presentation. Besides my primary supervisor, I am extremely grateful for my PhD supervisory panel members including Prof Gavin Huttley from the ANU Research School of Biology and Prof Weifa Liang from City University of Hong Kong for their insightful comments and suggestions. In addition, I thank Dr Benjamin Kaehler from the University of New South Wales, Canberra for the various suggestions and comments provided to improve my work during the early stages of my PhD.

I wish to thank to my collaborators Prof Liang Qiao from Fudan University, China, Dr Vaibhav Rajan from National University of Singapore, Dr Lianrong Pu from Tel Aviv University, Israel and everyone at the ANU Computational Genomics Group for all the support, guidance and insights provided during our collaborations.

I would like to acknowledge the Australian National University for funding my PhD and the National Computational Infrastructure (NCI) Australia for providing all the computational resources and assistance throughout the course of my PhD.

I am extremely grateful to my supervisors from University of Moratuwa (UoM), Sri Lanka where I obtained my Bachelors degree, Prof Dulani Meedeniya and Prof Indika Perera who have continuously encouraged, guided and supported me to pursue higher studies in the field of bioinformatics. Thank you to all the lovely members of the Department of Computer Science and Engineering at UoM for all the guidance and support you have offered.

I would like to pay my lifelong gratitude to my loving parents Asoka and Bennelette Mallawaarachchi, and grandparents Chandrarathna and Podihamine Mallawaarachchi for their continuous and unparalleled love, help and support. And last but not least, I thank my loving husband, Anuradha Wickramarachchi, who accompanied me and joined ANU to pursue his PhD with me at the ANU Computational Genomics Group. He has inspired me for my academic research and supported me during the ups and downs throughout my PhD. This accomplishment would not have been possible without them.

Thank you!

Vijini

Abstract

Metagenomics involves the study of various genetic material obtained directly from communities of microorganisms living in natural environments. The field of metagenomics has provided valuable insights into the structure, diversity and ecology within microbial communities. Recent developments in high-throughput sequencing technologies have enabled metagenomics to analyse samples from environments, without having to rely on culture-based methods. Once an environmental sample is sequenced, a process called metagenomics binning is used to cluster the sequences into bins that represent different taxonomic groups such as species, genera or higher levels. Various efforts have been made throughout the past to bin metagenomic sequences. One approach followed is to bin raw sequencing reads prior to assembly. However, reads are considered too short to produce accurate and reliable binning results for downstream analysis. Hence, the standard approach followed during metagenomics analysis is to assemble short reads into longer sequences called contigs and then bin these resulting contigs. Existing metagenomic contig-binning methods rely on the composition and abundance information of the contigs, and face challenges when binning short contigs and contigs with similar composition and abundance.

Contigs are derived from the underlying assembly graph which contains valuable connectivity information among contigs. However, existing metagenomic contig-binning methods do not consider the assembly graph in the binning process. Firstly, this thesis describes a bin refinement tool named GraphBin that improves existing metagenomic binning results using assembly graphs. GraphBin makes use of the assembly graph and a label propagation method to refine binning results of existing contig-binning tools by correcting mis-binned contigs and recovering short contigs that are discarded. Secondly, this thesis explains how to enable the detection of shared sequences among multiple species from assembly graphs and introduces a tool named GraphBin2 which can perform overlapped binning. GraphBin2 makes use of the assembly graph and the coverage information of contigs which enables the detection of contigs that may belong to multiple species. Thirdly, this thesis introduces a stand-alone approach named MetaCoAG to bridge metagenomics binning and assembly by incorporating composition, coverage and assembly graphs. MetaCoAG uses single-copy marker genes to estimate the number of initial bins, assigns contigs into bins iteratively and adjusts the number of bins dynamically throughout the binning process. In summary, this thesis discusses the challenges in binning metagenomic contigs, the shortcomings of existing metagenomic contig-binning tools and presents how the assembly graph can be incorporated to improve metagenomics binning.

Keywords: metagenomics, binning, bioinformatics algorithms, assembly graphs

Contents

Acknowledgements	ii
Abstract	iii
Figures	vi
Tables	viii
Abbreviations	ix
1 Introduction	1
1.1 Research Background	1
1.1.1 What is Metagenomics?	1
1.1.2 Why Metagenomics?	2
1.1.3 What is Assembly?	2
1.1.4 What is Metagenomics Binning?	2
1.2 Contributions	3
1.3 Thesis Outline	4
2 Literature Review	5
2.1 Genome Sequencing	5
2.2 Genome Assembly	6
2.3 Metagenome assembly	7
2.4 Metagenomics Binning	7
2.4.1 Reference-based Binning	7
2.4.2 Reference-free Binning	8
2.5 Challenges in Metagenomics Binning	11
3 Refined Binning of Metagenomic Contigs using Assembly Graphs	15
3.1 Motivation and Overview	15
3.2 Methods	15
3.2.1 Preprocessing	16
3.2.2 Refine Labels of Contigs based on the Assembly Graph	17
3.2.3 Run Label Propagation and Refinement	18
3.3 Experimental Setup	19
3.3.1 Datasets	19
3.3.2 Derive the Contigs, the Assembly Graph and Labels	20
3.3.3 Evaluation Criteria	21
3.4 Results and Discussion	22
3.4.1 Results on Simulated Datasets	23
3.4.2 Results on Sharon Datasets	25
3.4.3 Results on CAMI Datasets	25
3.4.4 Visualisation of the Assembly Graph	26
3.4.5 Propagation of Labels in the Assembly Graph	26
3.4.6 Implementation and Runtime	28
4 Overlapped Binning of Metagenomic Contigs using Assembly Graphs	31
4.1 Motivation and Overview	31
4.2 Methods	32
4.2.1 Preprocessing	32
4.2.2 Step 1: Remove Labels of Unsupported Vertices	32
4.2.3 Step 2: Correct Labels of Inconsistent Vertices	34

4.2.4	Step 3: Propagate Labels to Unlabelled Vertices	34
4.2.5	Step 4: Infer Multi-Labelled Vertices	35
4.3	Experimental Setup	37
4.3.1	Datasets	37
4.3.2	Tools Used	37
4.3.3	Evaluation Criteria	38
4.4	Results and Discussion	38
4.4.1	Binning Results	38
4.4.2	Multi-Labelled Inference Results	44
4.4.3	Visualisation of the Assembly Graph	48
4.4.4	Implementation, Running Time and Memory Usage	49
5	Binning Metagenomic Contigs using Composition, Coverage and Assembly Graphs	51
5.1	Motivation and Overview	51
5.2	Methods	51
5.2.1	Step 0: Assemble Reads into Contigs and Construct the Assembly Graph	51
5.2.2	Step 1: Identify Contigs with Single-Copy Marker Genes	52
5.2.3	Step 2: Order Single-copy Marker Genes and Estimate the Number of Initial Bins	53
5.2.4	Step 3: Bin Contigs with Single-copy Marker Genes	54
5.2.5	Step 4: Bin Remaining Contigs Using Label Propagation	56
5.3	Experimental Setup	57
5.3.1	Datasets	57
5.3.2	Tools Used	57
5.3.3	Evaluation Criteria	58
5.4	Results and Discussion	58
5.4.1	Benchmarks using simHC+ Dataset	58
5.4.2	Benchmarks using CAMI2 Toy Human Microbiome Project Datasets	60
5.4.3	Benchmarks using real metagenomic datasets	61
5.4.4	Implementation, Running Time and Memory Usage	63
6	Conclusion and Future Work	67
6.1	Refined Binning of Metagenomic Contigs using Assembly Graphs	67
6.2	Overlapped Binning of Metagenomic Contigs using Assembly Graphs	67
6.3	Binning Metagenomic Contigs using Composition, Coverage and Assembly Graphs	68
6.4	Future Work	68
6.4.1	Graph Representation Learning for Metagenomics Binning	68
6.4.2	Improving Taxonomic Binning Pipelines	69
6.4.3	Applications to Study Medical Data	69
	Bibliography	70
A	GraphBin Results	79
B	Datasets	84
B.1	Datasets used in experiments of GraphBin	84
B.2	Datasets used in experiments of GraphBin2	86
B.3	Datasets used in experiments of MetaCoAG	90
C	simHC+ Dataset	91
C.1	Binning results of simHC+ Dataset	91
D	Commands	94
D.1	Assembly Tools	94
D.2	Binning Tools	94
D.3	Bin Refinement Tools	96

Figures

1.1	Binning metagenomic reads and contigs to construct genomes	3
2.1	Assembly process	6
2.2	Metagenomics binning approaches	8
2.3	Reference-based binning	8
2.4	Reference-free binning	8
2.5	Constructing the feature vectors for composition-based binning	9
2.6	Composition and abundance-based binning of contigs	11
2.7	Coverage vs. composition plots of contigs	12
2.8	Binning shared contigs by existing contig-binning methods	13
2.9	Tetranucleotide composition profiles of three <i>Streptococcus</i> genomes	13
2.10	Tetranucleotide composition profiles of contigs from a species with high variance in composition	14
2.11	Coverage distribution of contigs from a species with high variance in coverage	14
3.1	Typical pipeline for binning metagenomic contigs	15
3.2	The workflow of GraphBin	16
3.3	Refined binning results of MetaWatt with GraphBin for the SGA assemblies	22
3.4	Refined binning results of MaxBin2 with GraphBin for the MEGAHIT assemblies	23
3.5	Refined binning results of MetaBAT2 with GraphBin for the metaSPAdes assemblies	23
3.6	Refined binning results of SolidBin with GraphBin for the metaSPAdes assemblies	24
3.7	Refined binning results of BusyBee Web with GraphBin for the MEGAHIT assemblies	24
3.8	Visualisation of the ES+metaSPAdes assembly graph in GraphBin	26
3.9	Labelling of the ES+metaSPAdes assembly graph in GraphBin	27
3.10	Labelling of the ESC+metaSPAdes assembly graph in GraphBin	28
3.11	Variation of running times for binning tools	30
4.1	Shared contigs in the assembly graph	31
4.2	The workflow of GraphBin2	33
4.3	Step-by-step illustration of how labels are propagated in GraphBin2	36
4.4	Binning results of CONCOCT, GraphBin and GraphBin2 for metaSPAdes assemblies	40
4.5	Binning results of MaxBin2, GraphBin and GraphBin2 for metaSPAdes assemblies	40
4.6	Binning results of SolidBin, GraphBin and GraphBin2 for metaSPAdes assemblies	41
4.7	Binning results of CONCOCT, GraphBin and GraphBin2 for SGA assemblies	41
4.8	Binning results of MaxBin2, GraphBin and GraphBin2 for SGA assemblies	42
4.9	Binning results of SolidBin, GraphBin and GraphBin2 for SGA assemblies	42
4.10	Binning results of CONCOCT, MaxBin2, SolidBin, GraphBin and GraphBin2 for complex datasets	43
4.11	Violin plots for the alignment ratio of the single and multi-labelled inference results of GraphBin2 for the metaSPAdes assemblies	45
4.12	Violin plots for the alignment ratio of the single and multi-labelled inference results of GraphBin2 for the complex datasets	46
4.13	Violin plots for the alignment ratio of the single and multi-labelled inference results of GraphBin2 for the SGA assemblies	47
4.14	Labelling of the assembly graph of Sim-5G dataset in GraphBin2	48
4.15	Labelling of the assembly graph of Sim-10G dataset in GraphBin2	49
5.1	The workflow of MetaCoAG	52
5.2	Single-copy marker genes in bacterial genomes	53

5.3	Estimating the number of bins using contigs containing single-copy marker genes . .	53
5.4	Cases of assigning contigs to bins in MetaCoAG	56
5.5	AMBER and CheckM results for the simHC+ dataset	59
5.6	Example visualization of the binning results from MaxBin2 and MetaCoAG in the simHC+ dataset	60
5.7	Visualization of the tetranucleotide composition and binning results of three <i>Streptococ-</i> <i>cus</i> genomes in the simHC+ dataset	61
5.8	F1-score of the CAMI datasets	62
5.9	F1-score of the real datasets	63
6.1	Traditional metagenomics binning and graph representation learning for metagenomics binning	69
A.1	Binning results of MetaWatt with GraphBin for the metaSPAdes assemblies	79
A.2	Binning results of BusyBee Web with GraphBin for the metaSPAdes assemblies	79
A.3	Binning results of MetaWatt with GraphBin for the MEGAHIT assemblies	80
A.4	Binning results of MaxBin2 with GraphBin for the metaSPAdes assemblies	80
A.5	Binning results of MaxBin2 with GraphBin for the SGA assemblies	81
A.6	Binning results of MetaBAT2 with GraphBin for the SGA assemblies	81
A.7	Binning results of MetaBAT2 with GraphBin for the MEGAHIT assemblies	82
A.8	Binning results of SolidBin with GraphBin for the SGA assemblies	82
A.9	Binning results of SolidBin with GraphBin for the MEGAHIT assemblies	83
A.10	Binning results of BusyBee Web with GraphBin for the SGA assemblies	83

Tables

2.1	Comparison of metagenomic contig-binning tools	10
3.1	Details of simulated datasets used in GraphBin	20
3.2	Number of bins identified by the binning tools in the experiments of GraphBin	25
3.3	Running times for assembly and binning with GraphBin	29
4.1	The number of bins identified by the binning tools in the experiments of GraphBin2	39
4.2	The number of multi-labelled contigs identified by GraphBin2	44
4.3	Running times and peak memory usage for the binning tools and GraphBin2	50
5.1	AMBER and CheckM evaluation results for the simHC+ dataset	59
5.2	High-quality species found from the GTDB-Tk annotations of the real datasets	64
5.3	Bin quality results for all the datasets used in the experiments of MetaCoAG	65
5.4	Running time and memory usage for the binning tools and MetaCoAG	66
B.1	Information on the datasets used for experiments of GraphBin	85
B.2	Information on the datasets used for experiments of GraphBin2	86
B.3	Species details of the simulated short-read datasets used in the experiments of GraphBin2	87
B.4	Species details of the 50G-SR dataset used in the experiments of GraphBin2	88
B.5	Samples used for the real datasets in experiments of MetaCoAG	90
B.6	Information of the datasets used in experiments of MetaCoAG	90
C.1	Information about the recovered species for the simHC+ dataset	91

Abbreviations

AMBER	Assessment of Metagenome BinnERs
ANU	Australian National University
ARI	Adjusted Rand Index
BAM	Binary Alignment Map
CAMI	Critical Assessment of Metagenome Interpretation
CIM	Culture Independent Method
CONCOCT	Clustering cONTigs on COverage and ComposiTion
DBG	De Bruijn Graph
DNA	Deoxyribo Nucleic Acid
EM	Expectation Maximisation
GC	Guanine-Cytosine
GSOM	Growing Self-Organising Map
HMP	Human Microbiome Project
LCA	Lowest Common Ancestor
LP	Lable Propagation
MAG	Metagenome-Assembled Genome
MCMC	Markov Chain Monte Carlo
NCBI	National Center for Biotechnology Information
NCI	National Computational Infrastructure
NGS	Next Generation Sequencing
OLC	Overlap Layout Consensus
ONT	Oxford Nanopore Technologies
PacBio	Pacific Biosciences
PCA	Principal Component Analysis
SGA	String Graph Assembler
SOM	Self-Organizing Map
TGS	Third Generation Sequencing
VAMB	Variational Autoencoders for Metagenomic Binning

Dedicated to my dear husband, parents and grandparents

Chapter 1

Introduction

The human body is home to a vast number of microorganisms. Also known as *microbes*, these microscopic organisms can be found on and in various parts of the human body such as skin, hair, mouth, nose and gut. It is believed that the number of microorganisms residing in the human body outnumbers the number of cells. Apart from the human body, microbes are naturally found as communities in various environments including soil, air and lake water, and occur under various conditions such as hot, cold, high pressure and even radioactive environments.

Microbes play an important role in biological activities of the human body such as digestion and development of immunity. Microbes also facilitate various natural processes such as decomposition and fermentation which have given rise to many industries including food production (*e.g.*, bread, cheese and alcoholic beverages), waste treatment, fertiliser and many industrial chemicals. Moreover, microbes have become essential tools in biotechnology and pharmaceutical related applications. However, microorganisms are causative agents of many infectious diseases that can cause harmful effects to the human body (*e.g.*, plague, tuberculosis and malaria). Given the benefits and drawbacks of microorganisms, it has become crucial to understand their behaviour and functions within their communities.

With the development of microscopes capable of viewing single-celled organisms by Antonj van Leeuwenhoek in the 17th century, scientists were able to directly view microorganisms (Garza and Dutilh, 2015). However, viewing and analysing microorganisms in their natural environments is a very complex and difficult process. As a solution, enrichment culture techniques were developed in the 19th century, and since then, isolation and cultivation became the most commonly used method to study and characterise microorganisms (Garza and Dutilh, 2015).

Even though enrichment culture techniques allowed scientists to study microorganisms in isolation, previous studies of environmental microbes have claimed that less than 2% of all the known microorganisms can be successfully cultured in laboratory conditions (Hofer, 2018; Overmann et al., 2017; Steen et al., 2019; Wade, 2002) and our understanding on the vast majority that has not been cultured is very limited. As a result, culture-independent methods (CIMS) (Garza and Dutilh, 2015; Su, Lei et al., 2012) have been developed, including *metagenomics*, to study microbial communities from various environments.

1.1 Research Background

1.1.1 What is Metagenomics?

Metagenomics is defined as the study of genetic content directly obtained from microbial communities found in various environments (Riesenfeld et al., 2004; Thomas et al., 2012) such as soil, sea water, space and niches of the human body including the gastrointestinal tract. Metagenomic samples directly obtained from natural environments are processed to separate the genetic material (referred as the *metagenomes*) and are sequenced to obtain the nucleotide information. This nucleotide information is analysed to characterise the different microorganisms present in the microbial community and understand their behaviours and functions. Metagenomics can provide valuable insights into novel molecules, enzymes and bio-catalysts as well as allows us to investigate microbial communities and their interactions.

The term *metagenomics* first appeared in publication in 1998 (Handelsman et al., 1998) with the application of genome sequencing to analyse complex and diverse (“meta”) populations of microbes (Wooley and Ye, 2010). In 2005, Kevin Chen and Lior Pachter defined metagenomics as “the application of modern genomics technique without the need for isolation and lab cultivation of individual species” (Chen and Pachter, 2018). With the advent of high throughput sequencing approaches, metagenomics has enabled us to access and study the genomes of entire microbial communities (Quince et al., 2017; Thomas et al., 2012). The field of metagenomics has paved the way for significant advances in microbiology and evolution over the past few decades and has been applied exclusively to study complex microbial communities.

In metagenomics analysis, three basic analytical strategies can be applied to quantify and characterise the different species present in a microbial community. They are (1) *marker gene analysis* (2) *assembly* and (3) *binning* (Sharpton, 2014). In marker gene analysis, taxonomically informative marker genes in the sequences are analysed. During assembly, short sequences called *reads* obtained from genome sequencing are combined together to form longer sequences called *contigs*. The binning process places sequences into imaginary bins which represent groups that belong to different taxonomic groups. This thesis will focus on metagenomic binning strategies for assembled contigs.

1.1.2 Why Metagenomics?

Traditional microbial studies have been based on pure culturing based techniques, where microorganisms are allowed to reproduce in a special culture medium under controlled laboratory conditions. However, most microorganisms cannot be grown in pure cultures under laboratory conditions. This is illustrated from the “Great Plate Count Anomaly” (Staley and Konopka, 1985) that showed only about 0.01–1% of the microorganisms from natural environments observed under the microscope could be isolated and grown in a culture medium (Garza and Dutilh, 2015). Microorganisms can fail to grow in culture media due to many reasons. Some are related to the culturing process such as not supplying the correct nutrients, use of the wrong pH of the medium and inadequate incubation periods (Davis et al., 2011; Köpke et al., 2005; Pulschen et al., 2017). Some reasons cannot be controlled within the culturing process such as dependence on the activities of other microorganisms and the need to develop as microcolonies (Davis et al., 2011; He et al., 2015; Pulschen et al., 2017). These factors have limited the study of the microbial diversity existing in the world.

To overcome the issues of culture-based techniques, culture-independent methods (CIMs) have been developed to study microbial communities without the need of culturing. Metagenomics is one such CIM that provides a relatively unbiased view of the structure and functions of microbial communities (Hugenholtz and Tyson, 2008). Metagenomics studies have facilitated the discovery of novel species (Andreani et al., 2018), genes (Boulund et al., 2017) and enzymes (Berini et al., 2017) which in turn has been applied to solve practical challenges in fields such as medicine, agriculture and ecology.

1.1.3 What is Assembly?

Assembly is the process of connecting reads of DNA together to reconstruct the original genome from which the DNA originated (Baker, 2012; Pop, 2009). Programs called *assemblers* have been designed to perform this computationally challenging task. The main goal of assemblers is to construct long contiguous pieces of DNA called *contigs* that together make the genome.

From a metagenomic perspective, assembly may fail to produce complete/near-complete genomes due to different complexities. Different organisms may have different levels of abundances, can be very closely related and there can be different repetitive structures within the genomes of organisms (Ghurye et al., 2016). Moreover, the genomes of some organisms may be sequenced to high depths of coverage than others resulting in a drastic variability (Ghurye et al., 2016). These factors complicate single genome assembly and hence cannot be applied directly to metagenomic data. Hence, specialised assemblers known as *metagenomic assemblers* are used to assemble metagenomic data.

1.1.4 What is Metagenomics Binning?

Sequences from different microbial genomes will be mixed together in a metagenomics sample and metagenomic assemblers may still find it challenging to produce complete/near-complete genomes

due to aforementioned complexities. To characterise the composition of a metagenomics sample, metagenomic sequences are placed into imaginary bins which represent groups that belong to different taxonomic groups such as species, genera or higher levels (Sedlar et al., 2017). This process is known as *metagenomics binning*. Once the sequences are separated into such groups, the data can be used to provide efficient taxonomic classification, construct metagenome-assembled genomes (MAGs) and enable accurate genome reconstruction (Frank et al., 2016).

Various efforts have been made to bin reads directly prior to assembly (Cleary et al., 2015; Girotto et al., 2016; Luo et al., 2018; Ounit et al., 2015; Schaeffer et al., 2017) and individually assemble reads in those bins as shown in Figure 1.1 (a). However, reads are considered as too short to produce accurate and reliable binning results for downstream analysis (Wang, Mawet et al., 2018). To overcome this issue and due to the failure of metagenomic assemblers to produce complete/near-complete genomes, the standard approach followed during metagenomics analysis is to assemble short reads into longer contigs and then cluster these resulting contigs into bins (Figure 1.1 (b)) that represent different taxonomic groups (Sedlar et al., 2017). Hence, this thesis will focus on methods to bin contigs.

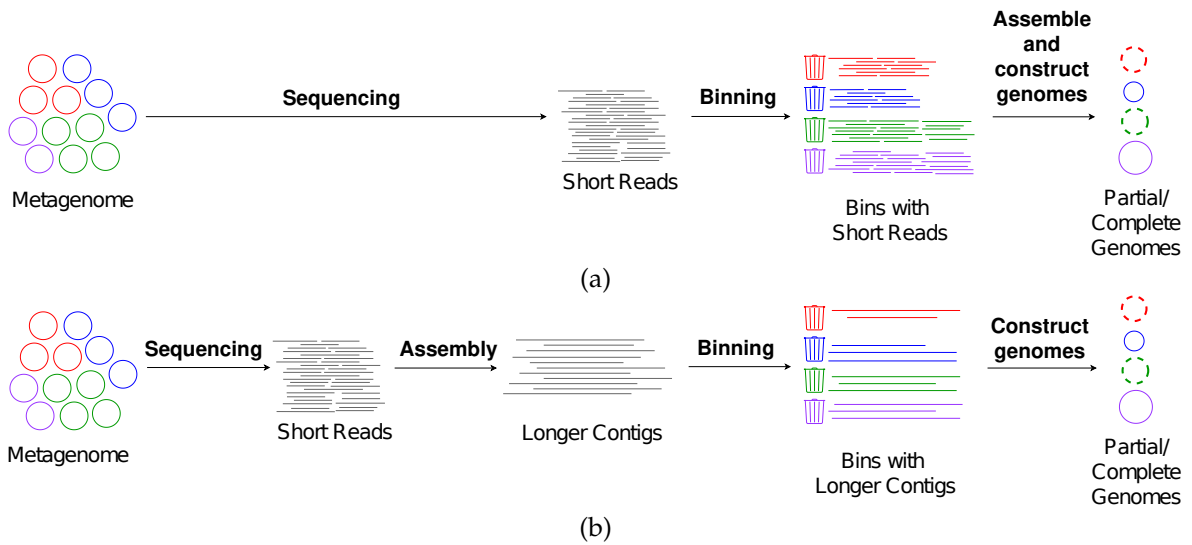


Figure 1.1: Binning metagenomic (a) reads and (b) contigs to construct genomes.

Existing metagenomic contig-binning methods can be divided into two categories as (1) *reference-based binning (taxonomic)* and (2) *reference-free binning (taxonomy independent)* (Sczyrba et al., 2017; Sedlar et al., 2017). Reference-based binning approaches rely on a database consisting of reference genomes and thus may not be applicable in many metagenomic samples when the reference genomes of novel species are not available. On the contrary, reference-free binning tools use unsupervised approaches to group contigs into unlabelled bins which correspond to different taxonomic groups, solely based on the information obtained from the contigs (Sedlar et al., 2017). These reference-free binning methods become very convenient when analysing environmental samples, especially when many species are not found in currently available reference databases (Laczny, Kiefer et al., 2017). Hence, this thesis will focus on reference-free binning methods.

Most of the available reference-free metagenomic contig-binning tools make use of the nucleotide composition and abundance information to perform binning. These tools achieve improved performance by combining both the composition and the coverage information. However, it still remains challenging for these binning tools to accurately reconstruct microbial genomes from complex datasets. Hence, it is worth investigating how to make use of other types of biological information and explore more features of contigs that can improve the binning result.

1.2 Contributions

This thesis explores how the assembly graph can be incorporated into metagenomics binning to improve the binning results. The major contributions of this thesis are as follows.

1. **Refined Binning of Metagenomic Contigs using Assembly Graphs.** Metagenomic contigs derived from the assembly process are obtained from the underlying assembly graph which contains valuable connectivity information between contigs that can be used for binning. However, existing contig-binning tools discard this connectivity information and use only the composition and coverage information of contigs for binning. Moreover, these tools discard many short contigs which can contain important genomic signatures. To address this issue, we have developed GraphBin, a metagenomics bin-refinement tool that makes use of the assembly graph. GraphBin can make use of assembly graphs (both the de Bruijn graph and the overlap-layout-consensus approach) to correct mis-binned contigs and recover short contigs discarded by existing binning tools. GraphBin is published in the Bioinformatics journal (Oxford University Press) ([Mallawaarachchi, Wickramarachchi et al., 2020a](#)).
2. **Overlapped Binning of Metagenomic Contigs using Assembly Graphs.** As different species in a metagenomic sample may share common sequences in their genomes, one assembled contig may belong to multiple species. However, existing tools for binning contigs only support non-overlapped binning, *i.e.*, each contig is assigned to at most one bin (species). To address this issue, we have developed GraphBin2, a metagenomics overlapped binning tool that can identify contigs shared between multiple species. GraphBin2 uses the connectivity and coverage information from assembly graphs to adjust existing binning results on contigs and to infer contigs shared by multiple species. GraphBin2 was presented at the 20th International Workshop on Algorithms in Bioinformatics (WABI 2020) ([Mallawaarachchi, Wickramarachchi et al., 2020b](#)) and is extended to a publication at BMC Algorithms for Molecular Biology journal ([Mallawaarachchi, Wickramarachchi et al., 2021](#)).
3. **Binning Metagenomic Contigs using Composition, Coverage and Assembly Graphs.** Our previous work GraphBin and GraphBin2 were dependent on the initial binning result from an existing metagenomic contig-binning tool. Errors made while identifying bins and during binning can be propagated and may affect the final result of these tools. Hence, we developed MetaCoAG, a stand-alone metagenomics contig-binning tool that makes use of composition, coverage and assembly graph information. MetaCoAG uses single-copy marker genes to estimate the number of initial bins, assigns contigs into bins iteratively and adjusts the number of bins dynamically throughout the binning process. MetaCoAG was presented at the 26th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2022) and is published in the Lecture Notes in Computer Science book series (Springer) ([Mallawaarachchi and Lin, 2022](#)).

1.3 Thesis Outline

This thesis is organised as follows. Chapter 1 provides an introduction to this thesis, including the research background and contributions of this thesis. Chapter 2 presents a comprehensive literature review on the background of genome sequencing, genome assembly, metagenomics binning and the challenges in metagenomics binning. In Chapter 3, I present GraphBin, a new binning method that makes use of the assembly graph and applies a label propagation algorithm to refine the binning result of existing tools. In Chapter 4, I present GraphBin2, a tool that refines the binning results obtained from existing tools and, more importantly, is able to detect contigs shared between multiple species and assign them to multiple bins. In Chapter 5, I present MetaCoAG, a stand-alone metagenomics contig-binning tool that makes use of assembly graphs and single-copy marker genes apart from the composition and coverage information of contigs. Finally, Chapter 6 concludes the thesis with a summary of the work presented, conclusions and future work.

Chapter 2

Literature Review

With the development of high-throughput sequencing technologies, the field of metagenomics has paved us the path to gain deep insights into various microbial communities. Traditional culturing-based analysis techniques can only obtain a very limited amount of genetic material from an environment under laboratory conditions. However, metagenomics analysis techniques allow us to obtain genetic material directly from the environment, without culturing under laboratory conditions. This chapter provides the background on genome sequencing, genome assembly, metagenomics binning, currently available binning approaches and challenges in metagenomics binning.

2.1 Genome Sequencing

The genome of an organism is made up of *deoxyribonucleic acid* (DNA) which carries the genetic instructions for all vital functions. DNA is made up of the nucleotides: *adenine* (A), *thymine* (T), *guanine* (G) and *cytosine* (C). *Genome sequencing* is the process of determining the DNA sequence of an organism. Special machines named *sequencers* have been developed to automate the sequencing process where they determine the order of the four nucleotides. These machines output strings of A, C, G, and T characters known as *reads*.

Many sequencing technologies have been developed during the past few decades to determine the nucleotide sequences in the DNA. Fredric Sanger introduced the *Sanger* sequencing method (also known as “chain-termination or dideoxy method”) in year 1977 which produced the first full DNA genome of the bacteriophage ϕ X174 (Sanger et al., 1977). This method produced reads of length 400-900 bp and had an extremely low error rate of about 0.001%. However, this method was expensive in terms of time and cost which resulted in a low throughput (Adams, 2008).

Second Generation or Next-Generation Sequencing (NGS) technologies such as *Illumina* (Canard and Sarfati, 1994) and *pyrosequencing* (454 sequencing) (Hyman, 1988) were able to almost replace Sanger sequencing due to their reduced cost and high throughput (hence the term *high-throughput sequencing* (Reuter et al., 2015)). NGS technologies have allowed scientists to sequence metagenomes at large scales and analyse multiple samples at once (e.g., Human Microbiome Project (HMP) (The Human Microbiome Project Consortium, 2012)). However, NGS technologies produce much shorter reads ranging from 100-300 bp with an error rate of 0.1%. The use of short NGS reads has led to many problems in determining genes of multiple species in multiple samples (Kang, Froula et al., 2015) and in resolving complex and repetitive regions during assembly (Pollard et al., 2018).

The more recent, Third Generation Sequencing (TGS) technologies (Schadt et al., 2010) such as Pacific Biosciences (PacBio) and Oxford Nanopore Technologies (ONT) allow sequencing from a single DNA molecule. They produce much larger reads ranging from 10 kbp to 2 Mbp with error percentages varying between 5-15%. Scientists have shown rising interest towards using data generated from TGS technologies as TGS data can overcome short-comings of short-read data (e.g., resolve repeat regions and produce more contiguous and complete assemblies). This is because PacBio and ONT reads have longer read-lengths (> 1,000 bp) allowing them to span across longer genomic regions and thus contain more genomic information than short NGS reads.

Please refer to the recent surveys and reviews on sequencing technologies by Reuter et al. (2015), Mardis (2017), Giani et al. (2020) and Hu et al. (2021).

2.2 Genome Assembly

Genome assembly is the process of connecting reads of DNA together to reconstruct the original genome from which the DNA originated (Baker, 2012; Pop, 2009). Programs called *assemblers* have been designed to perform this computationally challenging task. The main goal of assemblers is to construct long contiguous pieces of DNA called *contigs* that together make the genome.

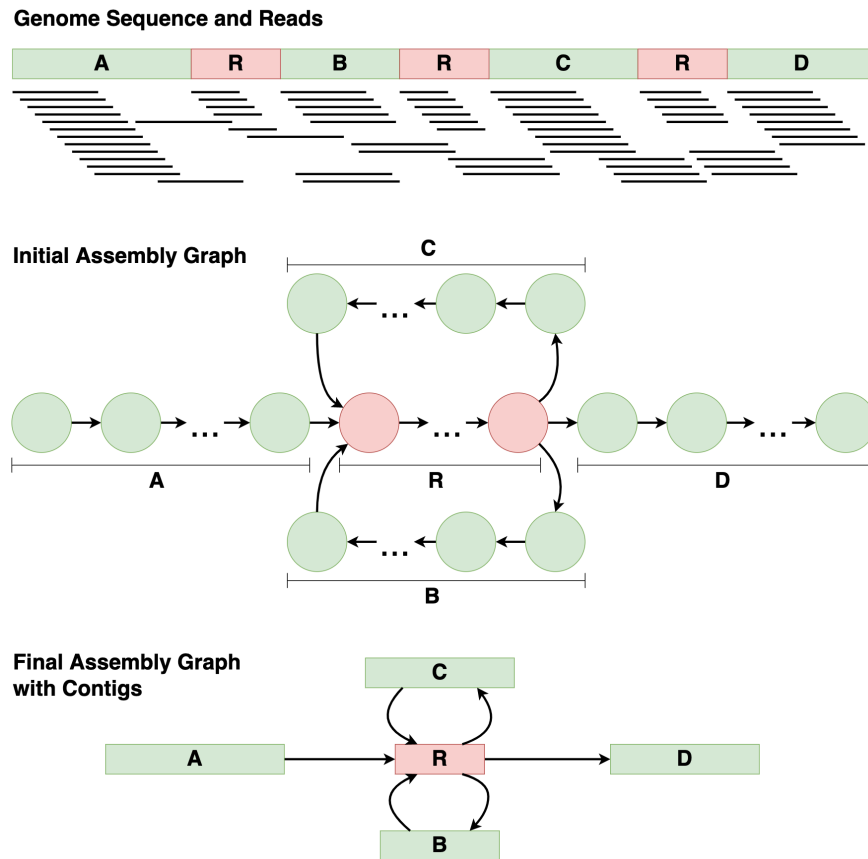


Figure 2.1: Assembly process

Genome assembly becomes extremely challenging due to the presence of large numbers of identical sequences, known as *repeats*. These repeats can range from a few hundreds to thousands of base pairs. Since short reads may not be long enough to span over these repeat regions, it becomes very difficult to resolve these regions during the assembly process.

Once reads are obtained by sequencing a genome with repeats (denoted by 'R' in Figure 2.1), the assembly process starts by identifying overlaps between reads. The initial assembly graph is constructed by connecting reads with overlaps (refer to initial assembly graph in Figure 2.1). Most existing assemblers use graphs as the underlying data structure to capture the overlaps between reads. Then the final assembly graph is constructed by collapsing non-branching paths (refer to final assembly graph with contigs in Figure 2.1) as non-branching paths correspond to longer contigs. The final assembly graph will contain the contigs and their overlaps.

Currently used genome assemblers fall under two broad categories. They are (1) overlap-layout-consensus (OLC) and (2) de-bruijn-graph (DBG) (Flicek and Birney, 2009; Li, Chen et al., 2012; Miller et al., 2010; Schatz et al., 2010). The OLC assemblers (Bonfield et al., 1995; Kececiloglu and Myers, 1995; Myers, 1995; Sutton et al., 1995) first identify overlaps among reads. Then they carry out a layout of all the reads and build a graph with reads as vertices and overlaps as edges. Finally, the consensus sequence is inferred from the graph. Identifying overlaps requires all-vs-all mapping of reads which is very time consuming. Early OLC assemblers include XBAP (Gleeson and Staden, 1991), GCG (Dolz, 1994), FALCON (Gryan, 1994), GAP (Bonfield et al., 1995), TIGR (Sutton et al., 1995), PHRAP (Green, 1999) and CAP3 (Huang and Madan, 1999), ARACHNE (Batzoglou et al., 2002), CELERA (Myers

et al., 2000) and Phusion (Mullikin and Ning, 2003). More recent assemblers that make use of the OLC method include SGA (Simpson and Durbin, 2012), Omega (Haider et al., 2014) and the long-read assembler Canu (Koren et al., 2017).

DBG assemblers first break reads into short k-mers, build a de Bruijn graph from all the k-mers and infer the genome sequence. The DBG algorithm was originally introduced in 1995 by Iddury and Waterman (Idury and Waterman, 1995) and the first assembler that used this approach, EULER was published in 2001 by Pevzner, Tang and Waterman (Pevzner et al., 2001). Popular assemblers that make use of the DBG method include Velvet (Zerbino and Birney, 2008), SPAdes (Bankevich et al., 2012), MEGAHIT (Li, Liu et al., 2015) and Flye (Kolmogorov, Yuan et al., 2019). Currently DBG assemblers are more popular as they are faster than OLC assemblers. Please refer to the recent surveys and reviews on genome assembly and different assemblers by Sohn and Nam (2016), Breitwieser et al. (2017) Giani et al. (2020) and Yang et al. (2021).

2.3 Metagenome assembly

Methods developed for single genome assembly cannot be directly applied on metagenomic data due to many reasons (Ghurye et al., 2016). As stated previously, assembly of a single genome becomes extremely challenging due to the presence of repeats. By assuming a uniform sequencing process for a single organism, such repeats can be identified as anomalies in the depth of coverage. However, due to the presence of many different organisms with different levels of abundances and genomes of some organisms being sequenced to high depths of coverage than others, simple coverage statistics cannot be used. Moreover, there can be multiple closely-related genomes (strain variants) that have very little genetic difference and it is hard to distinguish such differences from sequencing errors. Hence, it is challenging from a computational as well as biological perspective, to decide whether to consider or ignore these differences when constructing genomes from metagenomic data.

More recently, assemblers have been developed specifically to assemble metagenomic datasets, known as *metagenomic assemblers* (Kolmogorov, Bickhart et al., 2020; Nurk et al., 2017). These assemblers are able to overcome inherent challenges of assembling metagenomic data up to some extent and produce sequences belonging to different organisms found in different abundances within the metagenomic sample. However, they do not always produce complete/near-complete genomes and hence, methods to recover metagenome-assembled genomes (MAGs) that correspond to draft genomes are employed after metagenomic assembly.

Popular metagenomic assemblers such as MEGAHIT (Li, Liu et al., 2015), metaSPAdes (Nurk et al., 2017) for NGS data and metaFlye (Kolmogorov, Bickhart et al., 2020) for TGS data are widely used in large-scale metagenomic studies. MEGAHIT, metaSPAdes (comes as a mode in SPAdes), metaFlye (comes as a mode in Flye) were found to perform well in the recent second round of Critical Assessment of Metagenome Interpretation (CAMI) challenge (Meyer, Fritz et al., 2022). These assemblers are preferred to generate the contigs and assembly graphs which are required as input for the binning tools presented in this thesis.

2.4 Metagenomics Binning

Existing metagenomics binning methods can be broadly divided into two categories as (1) *reference-based binning (taxonomic)* and (2) *reference-free binning (taxonomy independent)* (Sczyrba et al., 2017; Sedlar et al., 2017) as shown in Figure 2.2.

2.4.1 Reference-based Binning

Reference-based binning tools (Figure 2.3) make use of a reference database with known microbial genomes, map the sequences to each reference genome, determine the genome having the highest similarity and assign the corresponding taxonomic label to sequences (e.g., MetaABC (Su, Hsu et al., 2011), MetaPhlAn (Segata et al., 2012), Kraken (Wood and Salzberg, 2014), Kraken2 (Wood, Lu et al., 2019), etc.). These tools make use of attributes that indicate taxonomic origin such as sequence similarity and compare them with sequences of known reference genomes (Dröge and McHardy, 2012). Methods such as sequence alignment, lowest common ancestor (LCA), use of specific marker

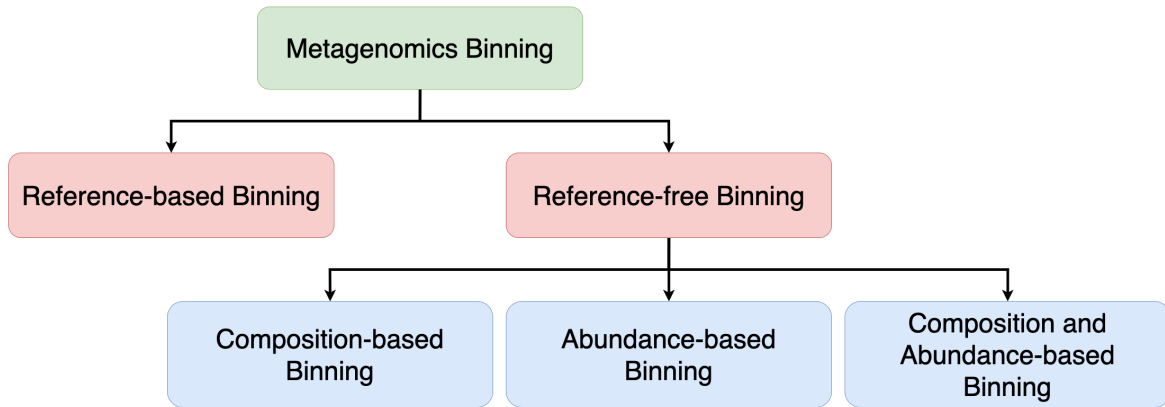


Figure 2.2: Metagenomics binning approaches

genes and phylogenetic tree construction are used in recently introduced reference-based binning tools. Moreover, several machine learning techniques are used to develop taxonomic classifiers that can handle large amounts of high dimensional data.

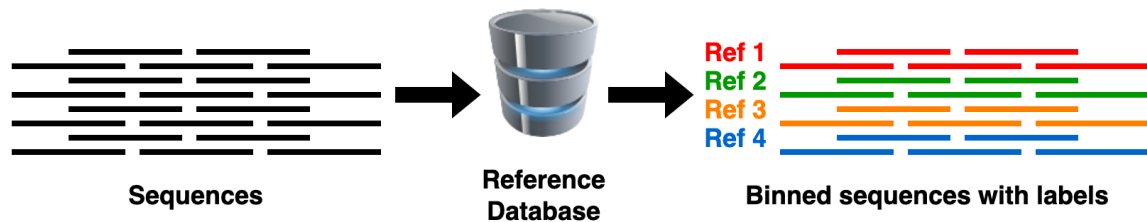


Figure 2.3: Reference-based binning

Although reference-based binning tools achieve a high accuracy when classifying sequences from known microbial genomes, they suffer from a serious limitation when classifying at lower levels (*i.e.*, species or genus levels) for some microbial genomes that are unknown or not included in the currently available reference databases.

2.4.2 Reference-free Binning

Reference-free binning tools do not make use of any reference database and they simply group sequences into unlabelled bins based on the genomic information of the sequences (Figure 2.4). Reference-free binning methods can be divided into three categories: (1) *composition-based*, (2) *abundance-based* and (3) *composition and abundance-based* (Sangwan et al., 2016).



Figure 2.4: Reference-free binning

Composition-based Binning

Previous studies have shown that frequencies of oligonucleotides in genomic sequences (referred to as *k-mers*, denoted in Figure 2.5) carry species-specific signals (Karlin and Burge, 1995; Karlin and Ladunga, 1994). Following these observations, composition-based methods have been developed based on the fact that each taxonomic group (such as species, genus, etc.) has a unique nucleotide

composition and performs binning by comparing the nucleotide content (Sedlar et al., 2017), such as the normalised frequencies of oligonucleotides (Dick et al., 2009) and the guanine-cytosine (GC) content (Saeed et al., 2011). Most of the tools employ conventional machine learning based approaches such as Markov chain Monte Carlo (MCMC) methods (used in LikelyBin (Kislyuk et al., 2009)) and neural network-based approaches such as self-organizing maps (SOMs) (used in Databionic ESOM Tools (Ultsch and Mörchen, 2005)).

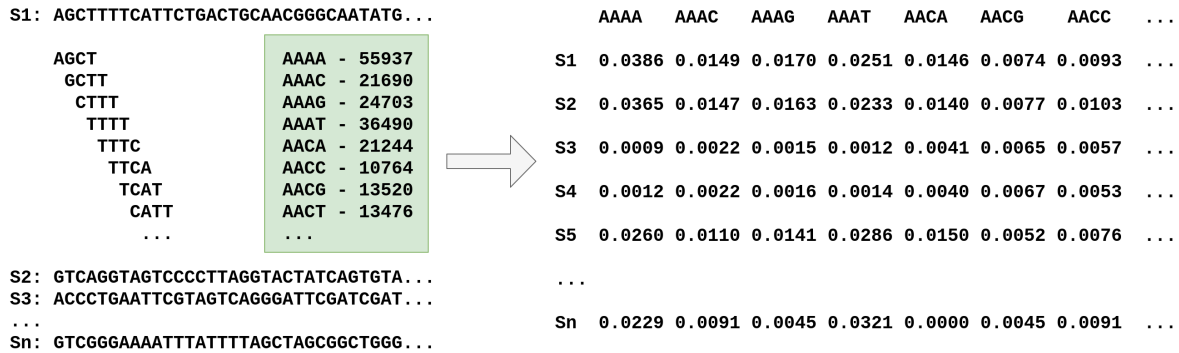


Figure 2.5: Constructing the feature vectors for composition-based binning

Abundance-based Binning

The abundance information of contigs is generally represented by the *coverage* which is the average number of reads that map to each base of the contig. Abundance-based binning methods are based on the assumption that abundance profiles of metagenomic sequences either follow the Lander-Waterman model (Lander and Waterman, 1988) in a single sample, or are highly correlated across multiple metagenomic samples (Sedlar et al., 2017). For example, AbundanceBin (Wu and Ye, 2011) models the sequenced reads as a mixture of Poisson distributions. Canopy (Nielsen et al., 2014) computes and compares the abundance profiles to cluster sequences from multiple samples. These methods have shown improved results for sequences of closely related strains (Sangwan et al., 2016), but pairwise calculations involved can be computationally expensive.

Composition and abundance-based Binning

Composition and abundance-based methods make use of both the variation of oligonucleotide frequencies and coverage information (e.g., Figure 2.6 illustrates an example workflow followed for binning contigs based on composition and abundance-based methods). These methods have often outperformed composition-based methods and abundance-based methods. Techniques such as principal component analysis (PCA), probabilistic models, expectation maximisation (EM) algorithms and more recently, machine learning models have been used to develop these binning tools.

Apart from the three main binning categories, researchers have proposed other techniques to perform metagenomic binning and improve the binning result of existing tools. BMC3C (Wang, Jiang et al., 2018) utilises codon usage in addition to the composition and coverage information. COCACOLA (Lu et al., 2016) considers linkage information from paired-end reads to improve the binning process. d_2^S Bin (Wang, Wang et al., 2017) makes use of the result of existing binning tools and adjusts the contigs among bins by measuring their dissimilarity. *Metabinner*s such as DAS tools (Sieber et al., 2018) and MetaWRAP (Uritskiy et al., 2018) have been introduced to integrate and optimize the results of multiple binning approaches.

Single-copy marker genes are special genes which are found in majority of the bacterial genomes and they appear only once in each genome (Albertsen et al., 2013; Dupont et al., 2012; Wu, Tang et al., 2014). Hence, single-copy marker genes have been utilised by some binning tools to estimate the number of bins during binning and for refinement of the binning results. MaxBin (Wu, Tang et al., 2014) and MaxBin 2.0 (Wu, Simmons et al., 2015) use single-copy marker genes to estimate the number of bins for initialisation of the binning process. MyCC (Lin and Liao, 2016) makes use of single-copy marker genes to refine the final clustering result.

Table 2.1: Comparison of metagenomic contig-binning tools

Binning tool	Composition	Coverage	Single-copy marker genes	Default/ minimum contig length cut-off	Main techniques/models/ algorithms used
MetaWatt (Strous et al., 2012)	✓	✓	✗	N/D	Interpolated markov models
CONCOCT (Alneberg et al., 2014)	✓	✓	✗	1,000 bp	Gaussian mixture model fit with a variational Bayesian approximation
GroopM (Imelfort et al., 2014)	✓	✓	✗	500 bp	Two-way clustering and Hough partitioning
MaxBin 2.0 (Wu, Simmons et al., 2015)	✓	✓	✓	1,000 bp	Expectation maximisation algorithm
MyCC (Lin and Liao, 2016)	✓	✓	✓	1,000 bp	Affinity propagation
VizBin (Laczny, Sternal et al., 2015)	✓	✗	✗	1,000 bp	Barnes-Hut stochastic neighbor embedding and manual clustering
COCACOLA (Lu et al., 2016)	✓	✓	✗	1,000 bp	K-means clustering with L_1 distance
BusyBee Web (Laczny, Kiefer et al., 2017)	✓	✗	✗	500 bp	Bootstrapped supervised binning
BMC3C (Wang, Jiang et al., 2018)	✓	✓	✗	1,000 bp	Ensemble k-means and graph partitioning using normalised cut
SolidBin (Wang, Wang et al., 2019)	✓	✓	✓	1,000 bp	Constraint-based spectral clustering with normalised cut
MetaBAT 2 (Kang, Li et al., 2019)	✓	✓	✗	1,500 bp	Graph partitioning using a modified label propagation algorithm
VAMB (Nissen et al., 2021)	✓	✓	✗	100 bp	Deep variational autoencoders and an iterative medoid clustering algorithm

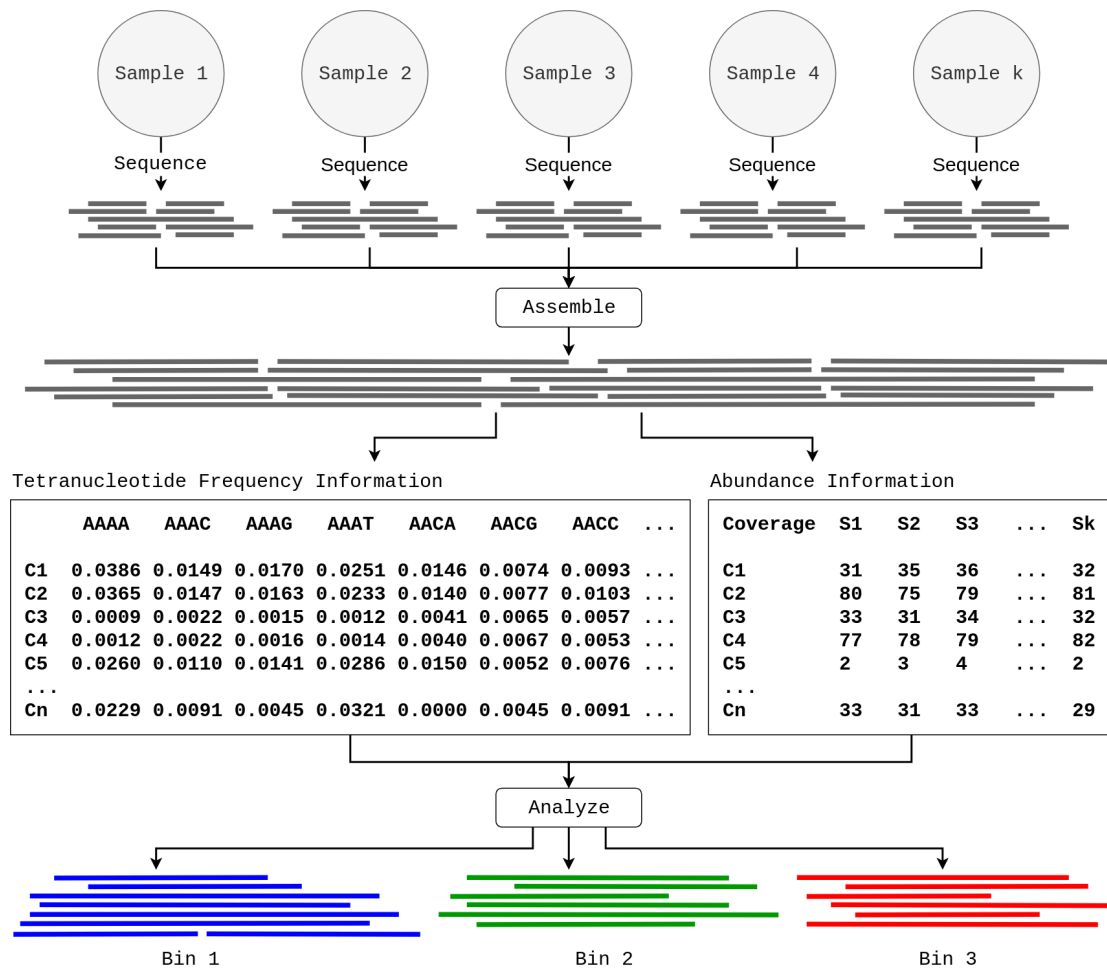


Figure 2.6: Composition and abundance-based binning of contigs

Table 2.1 summarises some of the popular metagenomic contig-binning tools. The inputs, whether the composition, coverage and marker gene information are used, default/minimum contig cut-off length and the main techniques/models/algorithms used for binning are compared for the listed tools. It can be seen that all the tools use composition information and majority of the tools use coverage information for binning. MaxBin 2.0, MyCC and SolidBin make use of single-copy marker genes. Furthermore, most of the tools have a contig cut-off length of 1,000 bp. Please refer to the recent surveys and reviews on binning metagenomic data by [Breitwieser et al. \(2017\)](#) and [Yang et al. \(2021\)](#) and benchmark studies on metagenomic binning tools by [Sczyrba et al. \(2017\)](#), [Yue et al. \(2020\)](#) and [Meyer, Fritz et al. \(2022\)](#).

2.5 Challenges in Metagenomics Binning

Existing metagenomic contig-binning tools mainly face problems when binning;

1. short sequences (generally shorter than 1,000 bp)
2. shared sequences that belong to multiple species
3. sequences of species belonging to the same genus
4. sequences of species with high intra-variance in composition
5. sequences of species with high intra-variance in genome coverage

Short sequences may not provide accurate genome-specific signals due to the low resolution in a single sample. The composition profiles of such short sequences can be highly deviated from that

of their constituent genomes. Binning such short contigs solely based on composition and coverage information can be challenging (as shown in Figure 2.7). Hence, majority of the available metagenomic contig-binning tools discard these short contigs during the binning process (as denoted by the contig length cut-off values in Table 2.1). For example, CONCOCT (Alneberg et al., 2014), MaxBin2 (Wu, Simmons et al., 2015) COCACOLA (Lu et al., 2016) and SolidBin (Wang, Wang et al., 2019) has a minimum contig length cut-off of 1,000 bp and MetaBAT2 Kang, Li et al. (2019) has a minimum contig length cut-off of 1,500 bp. It is worth exploring methods to recover these short contigs as they can contain useful biological information (e.g., can contain genes).

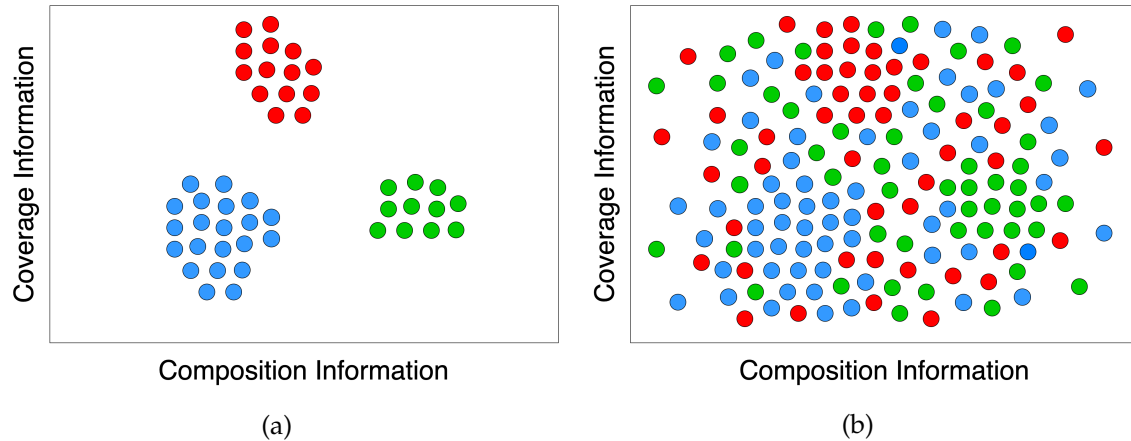


Figure 2.7: Coverage vs. composition plots of (a) long contigs and (b) both short and long contigs. Existing metagenomic contig-binning tools show a reasonable separation only for fairly long contigs, generally longer than 1,000 bp. However, if we try to bin all the contigs including both long and short contigs, we won't be able to see a clear separation between the species. This is because short contigs are too short to capture the species-specific composition and coverage patterns.

Different bacterial genomes in a metagenomic sample may share similar genes and genomic regions (Riesenfeld et al., 2004), which is a major challenge in assembling metagenomic reads into contigs (Nurk et al., 2017). Therefore, some assembled contigs from metagenomic reads may be shared by multiple species in the sample. However, most of the binning tools will assign these contigs to a single bin (refer to Figure 2.8). Failure to recover shared contigs and replicate them in their corresponding bins will affect the completeness of the MAGs constructed afterwards. Very few contig-binning tools support overlapped binning (i.e., assigning shared contigs to multiple species). S-GSOM (Chan et al., 2008) abstracts the flanking sequences of highly conserved 16S rRNA and incorporates them into Growing Self-Organising Maps (GSOM) to bin contigs into overlapping bins. MetaPhase (Burton et al., 2014) uses Hi-C reads to scaffold assembled contigs into assemblies of individual species and allows certain contigs to belong to multiple species. However, the applications of S-GSOM and MetaPhase are limited due to their required additional sequencing effort (e.g., 16S RNA or Hi-C sequencing). As shared contigs correspond to shared vertices between different genomic paths on the assembly graph (Nurk et al., 2017), it is worth exploring methods to infer such shared contigs from the assembly graph without additional sequencing requirements.

As microbial communities can be composed of different taxonomic groups (or taxa) ranging from species, genera, to higher orders, the sequencing data obtained from these samples can be very complex and heterogeneous. Even for the same genus, there can be several different species with very similar genomic signatures and they may co-exist in similar abundance (e.g., refer to Figure 2.9 where the nucleotide composition of three *Streptococcus* genomes are visualised). Sequences of such closely related species may be binned into a single bin as they show high similarity in composition (Sczyrba et al., 2017) or they may be mis-binned among the *Streptococcus* bins. Hence, it can be very challenging to distinguish sequences of such species during metagenomics binning.

Previous studies have shown that for a given bacterial genome, the nucleotide content can vary locally along the chromosome, especially in terms of the GC content (Bohlin, Eldholm et al., 2017; Bohlin, Snipen et al., 2010). It has been observed that protein coding regions tend to have higher GC content than non-coding regions (Bohlin, Skjerve et al., 2008). Moreover, the nucleotide composition of microbial genomes has shown to vary with different factors such as genome size, oxygen requirement

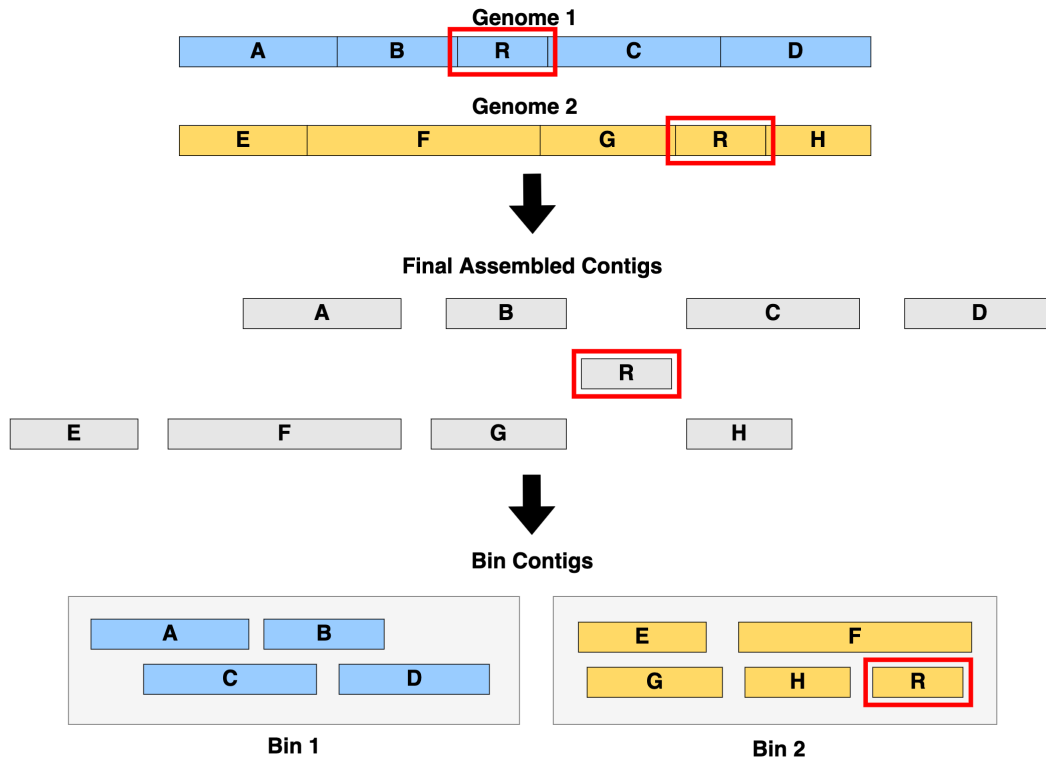


Figure 2.8: Binning shared contigs by existing contig-binning methods. The shared region *R* found in both genome 1 and genome 2 is binned only in to one bin.

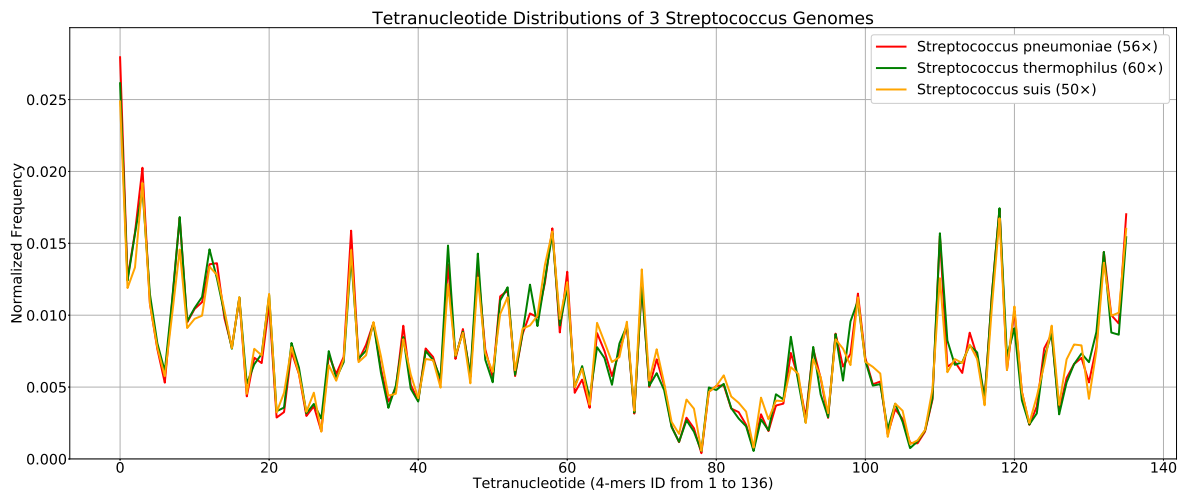


Figure 2.9: Tetranucleotide composition profiles of three *Streptococcus* genomes: *Streptococcus pneumoniae*, *Streptococcus thermophilus* and *Streptococcus suis*. The nucleotide composition of these three species is very similar and existing binning tools can mis-bin sequences belonging to them.

and nitrogen abundance (McEwan et al., 1998; Mitchell, 2007; Naya et al., 2002). Due to such reasons, even among the sequences that belong to the same species, there can be a high variance in nucleotide composition (as shown in Figure 2.10). This can confuse many existing metagenomic binning tools as their nucleotide profiles can be affected and sequences can be mis-binned.

Many bacterial genomes contain repetitive DNA regions called *repeats* (Kuśmirek and Nowak, 2018). The assembly process reconstructs these repeat regions using the information from reads. Contigs corresponding to these repeat regions usually have elevated coverage values. Hence, even among the contigs that belong to the same species, there can be a high variance in coverage values (as shown in

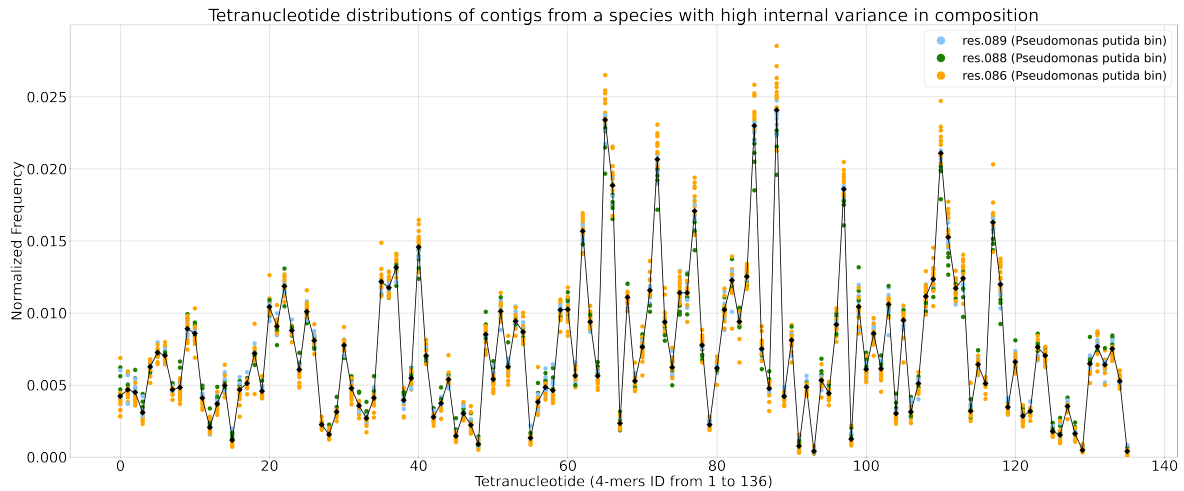


Figure 2.10: Tetranucleotide composition profiles of contigs from a species with high variance in composition (*Pseudomonas putida*). Even though the contigs belong to the same species, they have been split into three bins (denoted by the three colours) during the binning process of MaxBin2 (Wu, Simmons et al., 2015).

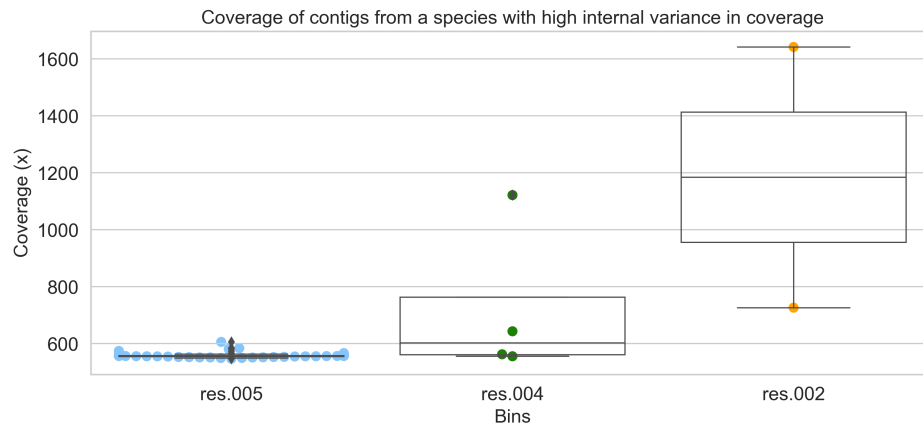


Figure 2.11: Coverage distribution of contigs from a species with high variance in coverage (*Aeromonas veroni*). Even though the contigs belong to the same species, they have been split into three bins during the binning process of MaxBin2 (Wu, Simmons et al., 2015).

Figure 2.11). This can lead to mis-binning of contigs.

Metagenomics binning results can be significantly affected by the aforementioned challenges and in turn affect downstream analyses as well. This thesis explores additional information of contigs provided from the assembly process, biological information and improved techniques that can be employed to tackle these challenges in metagenomics binning. The binning tools developed in this thesis incorporate a novel feature, the assembly graph information, which has not been used in previous binning tools. The first tool GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) is a bin-refinement tool that makes use of the connectivity information from the assembly graph to refine binning results produced by existing contig-binning tools. The next tool GraphBin2 (Mallawaarachchi, Wickramarachchi et al., 2020b, 2021) is the improved version of GraphBin that makes use of the coverage information of contigs apart from the connectivity information from the assembly graph to refine existing binning results and predict contigs shared between multiple species. The final tool MetaCoAG (Mallawaarachchi and Lin, 2022) removes the need to have an initial binning result from an existing contig-binning tool and bins the contigs from the beginning using composition, coverage and assembly graph information.

Chapter 3

Refined Binning of Metagenomic Contigs using Assembly Graphs

This chapter was published as

V. Mallawaarachchi, A. Wickramarachchi et al. (March 2020a). ‘GraphBin: refined binning of metagenomic contigs using assembly graphs’. *Bioinformatics*, 36(11), pp. 3307–3313. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btaa180](https://doi.org/10.1093/bioinformatics/btaa180)

3.1 Motivation and Overview

Binning of contigs plays an important role in metagenomics and most available binning algorithms bin contigs using genomic features such as oligonucleotide/k-mer composition and contig coverage. As metagenomic contigs are derived from the assembly process, they are output from the underlying assembly graph which contains valuable connectivity information between contigs that can be used for binning. However, existing metagenomic binning tools discard this connectivity information found in the assembly graph and treat contigs as independent sequences without any mutual connections (as shown in Figure 3.1).

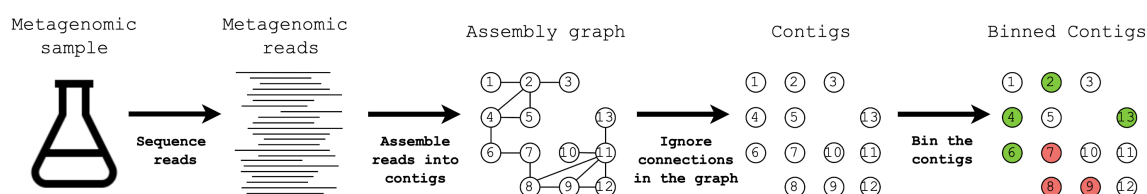


Figure 3.1: Typical pipeline for binning metagenomic contigs

In this chapter, we propose GraphBin, a new binning method that makes use of the assembly graph. We exploit the potential of using the assembly graph to refine the binning results from existing tools. We utilise contig connectivity information from the assembly graphs (produced from two types of assemblers; based on the de Bruijn graph and the string graph), and correct the contigs which were mis-binned by the existing tools. Moreover, a label propagation approach was used to predict the bins of the contigs discarded by existing tools. To the best of our knowledge, this is the first time to use the assembly graph along with a label propagation approach in metagenomics binning. Experimental results show that GraphBin was able to gain a significant improvement on top of existing binning results.

3.2 Methods

Figure 3.2 shows a flow diagram of GraphBin, to refine the binning results from existing metagenomic contig-binning tools using assembly graphs. During the preprocessing step, we obtain the assembly graph by assembling reads into contigs. Then we bin the contigs using an existing binning tool and

label the assembly graph. Next, GraphBin refines the labels of the contigs in the assembly graph (Step 1 in Figure 3.2). Finally, GraphBin performs label propagation and refinement (Step 2 in Figure 3.2) and outputs the refined binning result.

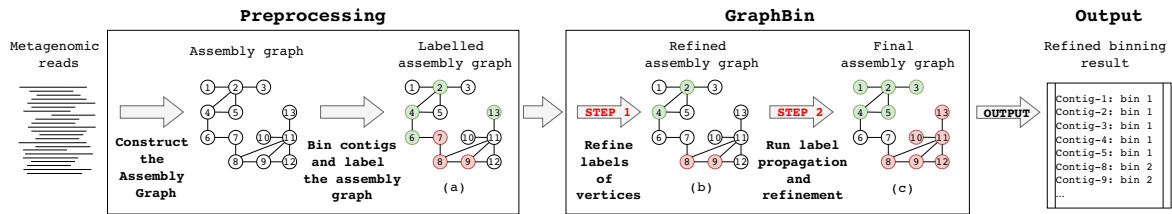


Figure 3.2: The workflow of GraphBin. During the preprocessing step, we obtain the assembly graph by assembling reads into contigs. Then we bin the contigs using an existing binning tool and label the assembly graph. Next, GraphBin refines the labels of the assembly graph. At last, GraphBin performs label propagation and refinement to obtain the final graph. GraphBin outputs the refined binning result.

3.2.1 Preprocessing

Assemble Reads into Contigs and Construct the Assembly Graph

In general, a metagenomic assembly tool first constructs the assembly graph using the connection (overlapping) information between reads, and then traverses through these connections in such a way, that each read is visited in the correct order, thereby linking them together to form a contiguous sequence known as a *contig* (Nurk et al., 2017; Vollmers et al., 2017). The assembly graph represents contigs as vertices and connection information between the contigs as edges. If there is a significant overlap between two contigs, there will be an edge connecting two corresponding vertices in the assembly graph. In this paper, we consider that all the edges have the same weight. Figure 3.2 contains an example assembly graph.

As each genome typically corresponds to a long path in the assembly graph, contigs connected to each other are more likely to belong to the same species. Connectivity information between contigs can thus provide valuable insights to bin contigs and previous studies have shown improved binning results through the manual refinement of contigs after visualising the assembly graph (Barnum et al., 2018). In the following, we use metaSPAdes (Nurk et al., 2017), SGA (Simpson and Durbin, 2012) and MEGAHIT (Li, Liu et al., 2015) to build the assembly graph from reads. metaSPAdes first constructs the de Bruijn graph from raw reads, transforms it into the assembly graph through various simplification heuristics and reconstructs paths in the assembly graph that correspond to contigs within metagenomes (Nurk et al., 2017). SGA employs the overlap-layout-consensus idea, first corrects errors in raw reads and then constructs a string graph based on the overlaps between the error-corrected reads (Simpson and Durbin, 2012). MEGAHIT is based on the *succinct de Bruijn graph* which is a compressed representation of the de Bruijn graph (Li, Liu et al., 2015).

Bin Contigs using an Existing Binning Tool and Label the Assembly Graph

In this paper, we selected MetaWatt (Strous et al., 2012) and MaxBin2 (Wu, Simmons et al., 2015) to obtain the initial binning result as these two tools have shown the best performance in the recent CAMI challenge (Sczyrba et al., 2017). Moreover, we also selected three recent reference-free binning tools MetaBAT2 (Kang, Li et al., 2019), SolidBin (Wang, Wang et al., 2019) and BusyBee Web (Laczny, Kiefer et al., 2017).

MetaWatt makes use of the multivariate statistics of tetranucleotide frequencies, builds interpolated Markov models of genomes and assigns contigs to bins based on maximum likelihood. MaxBin2 considers tetranucleotide frequencies and coverage of contigs to perform binning. It makes use of probabilistic models and an EM algorithm to iteratively bin the contigs. However, MaxBin2 only bins contigs which are longer than 1,000 bp. MetaBAT2 employs a graph clustering approach, where the graph construction is done using tetra nucleotide frequency scores. However, MetaBAT2 only bins contigs of length 1,500 bp or longer. SolidBin makes use of a spectral clustering approach with additional biological information. However, SolidBin only bins contigs which are longer than 1,000 bp. BusyBee Web makes use of a bootstrapped supervised binning approach to bin contigs. By

default, it bins contigs which are 500 bp or longer. As BusyBee Web is a web-based application, certain restrictions have been enforced on the size of the input data due to limited computational resources.

In the assembly graph, contigs are represented as vertices and connections between the contigs are denoted by edges. Once the contigs are binned, the corresponding vertices in the initial assembly graph are labelled accordingly. As shown in Figure 3.2 (a), vertices are illustrated in different colours (labels) to represent contigs belonging to different bins.

3.2.2 Refine Labels of Contigs based on the Assembly Graph

Considering the fact that the connected contigs are more likely to belong to the same species, we refine the labels of the vertices in the assembly graph. It is highly probable that vertices connected together or vertices located close-by belong to the same cluster as they are less distant (*i.e.*, have more similarity) to each other. Hence, such vertices should have the same label. The refinement process is done based on the labels of such vertices.

Let $d(i, j)$ be the distance between vertex i and vertex j in the assembly graph. The label status of vertex i is represented by the label function L : if i is labelled, then $L(i) = 1$; if i is not labelled, then $L(i) = 0$. Let $d_L(i)$ be the shortest distance between any labelled vertex and vertex i , *i.e.*, $d_L(i) = \min_{\forall j, L(j)=1} d(i, j)$.

If none of the labelled vertices is reachable from vertex i , $d_L(i) = \infty$. Let $CLV(i)$ be the set of *closest labelled vertices* for a vertex i where $CLV(i)$ consists of the closest labelled vertices of vertex i , *i.e.*, $CLV(i) = \{j | d(i, j) = d_L(i) \text{ and } L(j) = 1\}$. Note that $CLV(i) = \emptyset$ when $d_L(i) = \infty$. For example, in Figure 3.2 (a), $d_L(4) = 1$ and $CLV(4) = \{2, 6\}$, $d_L(13) = 2$ and $CLV(13) = \{8, 9\}$. For a given vertex i , the set of closest labelled vertices $CLV(i)$ is found by performing a breadth-first search starting from vertex i . The breadth-first search terminates at a certain depth level (*i.e.*, $d_L(i)$) when some labelled vertices are found, or it has visited all the reachable vertices. Once at least one labelled vertex is found, all the labelled vertices at that particular depth level (*i.e.*, $d_L(i)$) are considered as the set of closest labelled vertices and the breadth-first search stops. If no labelled vertices are encountered after visiting all the reachable vertices (*i.e.*, $d_L(i) = \infty$), then the breadth-first search stops and the set of closest labelled vertices is empty (*i.e.*, $CLV(i) = \emptyset$). Algorithm 1 denotes the pseudo-code of the *getCLV* function. It takes in the assembly graph (G), vertex (i) and the label function (L) and outputs the set of closest labelled vertices $CLV(i)$. Please note that $G.neighbours(i)$ in Algorithm 1 is the set of neighbours of vertex i in the assembly graph G .

Algorithm 1: Compute $CLV(i)$ in the assembly graph

Input: Assembly graph (G), vertex (i), Label (L)

Output: $CLV(i)$, the closest labelled vertices of vertex i

```

1 function getCLV( $G, i, L$ )
2    $SET_{current} \leftarrow G.neighbours(i)$ 
3   mark  $i$  as visited
4    $CLV_{temp} \leftarrow \emptyset$  and  $SET_{next} \leftarrow \emptyset$ 
5   while  $SET_{current} \neq \emptyset$  and  $CLV_{temp} = \emptyset$  do
6     forall  $v \in SET_{current}$  do
7       mark  $v$  as visited
8       if  $L(v) = 1$  then
9          $CLV_{temp} \leftarrow CLV_{temp} \cup \{v\}$ 
10      else
11        forall  $r \in G.neighbours(v)$  do
12          if  $r$  is not visited then
13             $SET_{next} \leftarrow SET_{next} \cup \{r\}$ 
14     $SET_{current} \leftarrow SET_{next}$ 
15   $CLV(i) \leftarrow CLV_{temp}$ 
16  return  $CLV(i)$ 

```

In the assembly graph, a vertex i is denoted as an *ambiguous* vertex if at least one of its closest labelled vertices $CLV(i)$ has a label that is different than the label of vertex i . For example, in Figure 3.2 (a),

vertex 13 is ambiguous because one of its closest labelled vertices (vertex 8 or 9 in $CLV(13)$) has a red label while vertex 13 itself has a green label. Similarly, vertex 6 and 7 are also ambiguous as one of its closest labelled vertices has a different label. The labels of ambiguous vertices are removed as the corresponding contigs may belong to multiple genomes or the binning tool may have predicted the wrong bin, and these contigs might misguide the followup label propagation process. Please refer to Step 3 in Figure 3.2 for an example. Labels of vertices 6, 7 and 13 are removed. Figure 3.2 (b) denotes the graph after refining the labels.

3.2.3 Run Label Propagation and Refinement

Label propagation (LP) approaches have been applied to cluster metagenomic sequences (Kang, Li et al., 2019; Li, Wang et al., 2019) without using the assembly graph. Label propagation is a semi-supervised machine learning technique that can propagate labels to neighbouring unlabelled data with the use of labelled data (Zhu and Ghahramani, 2002). The label propagation algorithm initialises by assigning labels to the vertices of corresponding labelled data in a graph. The vertices then propagate their labels to their neighbouring vertices in further iterations. Finally, the vertices with similar labels are clustered. The label propagation algorithm proposed by Zhu and Ghahramani (Zhu and Ghahramani, 2002) was applied to the assembly graph in order to predict the labels of the unlabelled vertices. We follow the same notations and definitions from Zhu and Ghahramani (2002) and the details about the algorithm are as follows.

Label Propagation Algorithm

Consider a graph G to have vertices where some are labelled and others are not. Let $X_L = \{x_1 \dots x_l\}$ be the set of labelled vertices where $Y_L = \{y_1 \dots y_l\}$ are the corresponding vertex labels. The number of classes C is known as we can obtain it from the initial binning result. Assume that all the classes are present in the labelled data. Let $X_U = \{x_{l+1} \dots x_{l+u}\}$ be the unlabelled vertices where $Y_U = \{y_{l+1} \dots y_{l+u}\}$ are unknown. Let $X = \{x_1 \dots x_{l+u}\}$. We have to estimate Y_U from X and Y_L .

We define weights of edges between vertices x_i and x_j as w_{ij} . Since we consider edge weights to be uniform, all edge weights will have a weight of 1; i.e., if vertices x_i and x_j are connected by an edge, then $w_{ij} = 1$, else $w_{ij} = 0$.

We define a $(l + u) \times (l + u)$ probabilistic transition matrix T where T_{ij} is the probability of travelling from vertex x_j to vertex x_i .

$$T_{ij} = P(j \rightarrow i) = \frac{w_{ij}}{\sum_{k=1}^{l+u} w_{kj}} \quad (3.1)$$

Next, we define a $(l + u) \times C$ label matrix Y where the i^{th} row of Y denotes the label probability distribution of vertex x_i . Label probabilities of labelled vertices will be defined as 1 and those of unknown vertices are defined as 0. Now we will present the label propagation process as denoted in Algorithm 2. The function takes in the probabilistic transition matrix (T), the matrix with label probability distributions of the vertices (Y), the threshold (eps) and the maximum number of iterations (max_iter) as inputs and outputs the set of predicted labels of all the vertices in G .

According to Algorithm 2, all vertices will propagate their labels (line 8). Clamping labels is very important as we want the label sources (labelled data) to be persistent (line 9). These two steps are repeated until Y converges or the number of iterations exceeds the maximum number.

Convergence of Y is determined by summing up the differences in all the corresponding label probabilities between $Y^{(t)}$ and $Y^{(t+1)}$ (lines 10 to 13). If the sum of differences in label probabilities is less than eps , then Y is considered to be converged. Once Y has converged or the number of iterations has exceeded, for each vertex v_i , the label with the maximum probability is determined from $Y^{(t)}$ and assigned to $labels(i)$ (lines 18 to 19).

The parameters eps and max_iter are set by default to 0.1 and 100 respectively in the GraphBin source code. Moreover, GraphBin provides the facility for the user to specify eps and max_iter parameters as required when executing.

Algorithm 2: Label Propagation

Input: Probabilistic transition matrix (T), matrix with label probability distributions of the vertices (Y), threshold (eps), maximum number of iterations (max_iter)

Output: $labels$, set of predicted labels of all the vertices

```

1 function propagateLabels( $T, Y, eps, max\_iter$ )
2    $labels \leftarrow \emptyset$ 
3    $t \leftarrow 0$ 
4    $n \leftarrow Y.rows$ 
5    $m \leftarrow Y.columns$ 
6    $Y^{(0)} \leftarrow Y$ 
7   while  $t < max\_iter$  do
8      $Y^{(t+1)} \leftarrow T \times Y^{(t)}$ 
9     clamp label probabilities of labelled data in  $Y^{(t+1)}$ 
10     $diff \leftarrow 0$ 
11    for  $i \leftarrow 1$  to  $n$  do
12      for  $j \leftarrow 1$  to  $m$  do
13         $diff \leftarrow diff + |Y^{(t+1)}[i][j] - Y^{(t)}[i][j]|$ 
14    if  $diff < eps$  then
15      break
16    else
17       $t \leftarrow t + 1$ 
18  for  $i \leftarrow 1$  to  $n$  do
19     $labels(i) \leftarrow \underset{j}{\operatorname{argmax}} Y^{(t)}[i][j]$ 
20  return  $labels$ 

```

Please note that the labels of vertices in isolated components cannot be inferred through label propagation if none of the vertices in such components have predicted labels by an existing binning tool.

After the label propagation, we further refine the labelling by removing the labels of ambiguous vertices. The resulting graph consists of labelled vertices as shown in Figure 3.2 (c). Please refer to Step 4 in Figure 3.2 as an example. Labels of vertices 6 and 7 will be removed after the label propagation process as they are ambiguous (as illustrated in Figure 3.2 (c)).

3.3 Experimental Setup

3.3.1 Datasets

Simulated Datasets

We simulated several metagenomic datasets according to a preborn infant gut metagenome, commonly known as the **Sharon** dataset (Sharon et al., 2013). We obtained the three most abundant species from the run **SRR492184** of the **Sharon** dataset (NCBI accession number **SRA052203**) and their corresponding proportions. Those species and their corresponding proportions are as follows,

1. *Enterococcus faecalis* - 70.8%
2. *Staphylococcus aureus* - 18.9%
3. *Cutibacterium avidum* - 3.4%

We also downloaded their complete reference genomes from the NCBI nucleotide database. Two datasets were simulated each containing 2 species (*Enterococcus faecalis* and *Staphylococcus aureus*, referred as **ES**) and 3 species (*Enterococcus faecalis*, *Staphylococcus aureus* and *Cutibacterium avidum*, referred as **ESC**) respectively. Paired-end reads were simulated using the tool InSilicoSeq (Gourlé

et al., 2018) modelling a MiSeq instrument with 300 bp mean read length and the predefined MiSeq error model. Further details about the simulated datasets can be found in Table 3.1.

Table 3.1: Information of the datasets simulated according to the Sharon dataset

Dataset	Species present	Number of reads used for assembly	Genome size	Coverage	Number of contigs classified as ground truth	Total number of contigs classified as ground truth
ES+metaSPAdes	<i>Enterococcus faecalis</i>	1,578,595	2.6 Mb	181x	67	156
	<i>Staphylococcus aureus</i>	421,405	2.9 Mb	44x	89	
ES+SGA	<i>Enterococcus faecalis</i>	1,578,595	2.6 Mb	181x	1,007	1,482
	<i>Staphylococcus aureus</i>	421,405	2.9 Mb	44x	475	
ESC+metaSPAdes	<i>Enterococcus faecalis</i>	3,802,363	2.6 Mb	436x	60	181
	<i>Staphylococcus aureus</i>	1,015,038	2.9 Mb	105x	98	
	<i>Cutibacterium avidum</i>	182,599	2.5 Mb	21x	23	
ESC+SGA	<i>Enterococcus faecalis</i>	3,802,363	2.6 Mb	436x	4,216	4,434
	<i>Staphylococcus aureus</i>	1,015,038	2.9 Mb	105x	193	
	<i>Cutibacterium avidum</i>	182,599	2.5 Mb	21x	25	

Note: The number of reads from each species were simulated to correspond to their proportions in the run *SRR492184* of the **Sharon** dataset.

Sharon Dataset

The **Sharon** dataset (Sharon et al., 2013) (available on the NCBI Sequence Read Archive under accession number *SRA052203*) is composed of a time-series of 11 fecal samples from a pre-born infant and consists of reads from 18 Illumina (Illumina HiSeq 2000) runs. Reads from all the 18 runs were combined to form the **Sharon** dataset.

CAMI Datasets

We selected 3 publicly available datasets from the first Critical Assessment of Metagenome Interpretation (CAMI) challenge (Sczyrba et al., 2017) representing microbiomes of three complexities low, medium and high;

- **CAMI_l**: Sample from low complexity
- **CAMI_m**: Sample 1 with 5 kbp insert size from medium complexity
- **CAMI_h**: Sample 1 from high complexity

Please refer to Appendix B for further details about the CAMI datasets.

3.3.2 Derive the Contigs, the Assembly Graph and Labels

The reads in each of the simulated datasets and the publicly available datasets were assembled into contigs and the assembly graph was obtained by running metaSPAdes (Nurk et al., 2017) (from SPAdes 3.13.0 (Bankevich et al., 2012)), SGA 0.10.15 (Simpson and Durbin, 2012) and MEGAHIT 1.2.9 (Li, Liu et al., 2015) as discussed in the preprocessing step. The commands used for assembly can be found in Appendix D. The resulting datasets are referred as follows.

- **ES+metaSPAdes** - metaSPAdes assembly of the ES dataset
- **ES+SGA** - SGA assembly of the ES dataset
- **ES+MEGAHIT** - MEGAHIT assembly of the ES dataset
- **ESC+metaSPAdes** - metaSPAdes assembly of the ESC dataset
- **ESC+SGA** - SGA assembly of the ESC dataset
- **ESC+MEGAHIT** - MEGAHIT assembly of the ESC dataset
- **Sharon+metaSPAdes** - metaSPAdes assembly of the Sharon dataset
- **Sharon+SGA** - SGA assembly of the Sharon dataset

- **Sharon+MEGAHIT** - MEGAHIT assembly of the Sharon dataset
- **CAMI_l+metaSPAdes** - metaSPAdes assembly of the CAMI_l dataset
- **CAMI_l+SGA** - SGA assembly of the CAMI_l dataset
- **CAMI_l+MEGAHIT** - MEGAHIT assembly of the CAMI_l dataset
- **CAMI_m+metaSPAdes** - metaSPAdes assembly of the CAMI_m dataset
- **CAMI_m+SGA** - SGA assembly of the CAMI_m dataset
- **CAMI_m+MEGAHIT** - MEGAHIT assembly of the CAMI_m dataset
- **CAMI_h+metaSPAdes** - metaSPAdes assembly of the CAMI_h dataset
- **CAMI_h+SGA** - SGA assembly of the CAMI_h dataset
- **CAMI_h+MEGAHIT** - MEGAHIT assembly of the CAMI_h dataset

Further details of the resulting datasets can be found in Appendix B.

MetaWatt 3.5.3 (Strous et al., 2012), MaxBin 2.2.5 (Wu, Simmons et al., 2015), BusyBee Web (Laczny, Kiefer et al., 2017) with their default parameters, SolidBin 1.3 (Wang, Wang et al., 2019) in SolidBin-SFS mode, and MetaBAT2 (Kang, Li et al., 2019) with minimum contig length set to 1,500 bp were selected to obtain the initial binning results as discussed in the preprocessing step. Please note that the recently published tool Vamb (Nissen et al., 2021) was not used in the evaluations as it was not published at the time GraphBin was published. The commands used for binning can be found in Appendix D.

3.3.3 Evaluation Criteria

In order to determine the ground-truth species of the contigs in all the datasets, we used TAXAassign v0.4 (can be found at <https://github.com/umerijaz/TAXAassign>) to label the contigs. TAXAassign v0.4 uses BLAST to search matches in the NCBI nucleotide database with a given percentage of identity. Isolated contigs (corresponding vertices with zero degree) were not considered for the ground truth set of the datasets.

To evaluate GraphBin, we applied the four common criteria (1) *Precision*, (2) *Recall*, (3) *F1-score* and (4) *Adjusted Rand Index (ARI)* that have been used in previous binning studies Alneberg et al. (2014); Herath et al. (2017); Wang, Wang et al. (2017) at the species level. The binning result is denoted as a $K \times S$ matrix with K number of bins and S number of species. In this matrix, the element a_{ks} denotes the number of contigs binned to the k^{th} bin and belongs to the s^{th} species. U denotes the number of unclassified contigs. Following are the definitions and equations that were used to calculate the *precision*, *recall*, *F1-score* and *ARI* values in our experiments.

Precision

For each of the bins, we obtain the species with the maximum number of contigs assigned. Then, we sum the maximum numbers of each bin and divide it by the total number of binned contigs. The resulting value is the *precision* and it is calculated as follows.

$$Precision = \frac{\sum_k \max_s \{a_{ks}\}}{\sum_k \sum_s a_{ks}} \quad (3.2)$$

Recall

For each of the species, we obtain the bin with the maximum number of contigs assigned. Then, we sum the maximum numbers of each species and divide it by the total number of binned contigs and unclassified contigs. The resulting value is the *recall* and it is calculated as follows.

$$Recall = \frac{\sum_s \max_k \{a_{ks}\}}{(\sum_k \sum_s a_{ks} + U)} \quad (3.3)$$

F1-score

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

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.4)$$

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.

$$ARI = \frac{\sum_{k,s} \binom{a_{ks}}{2} - t_3}{\frac{1}{2}(t_1 + t_2) - t_3} \quad (3.5)$$

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

3.4 Results and Discussion

The performance of GraphBin was compared with the original performance of the five binning tools; MetaWatt (Strous et al., 2012), MaxBin2 (Wu, Simmons et al., 2015), MetaBAT2 (Kang, Li et al., 2019), SolidBin (Wang, Wang et al., 2019) and BusyBee Web (Laczny, Kiefer et al., 2017). Note that three different assemblers metaSPAdes (Nurk et al., 2017), SGA (Simpson and Durbin, 2012) and MEGAHIT (Li, Liu et al., 2015) were used to build the assembly graphs. Figure 3.3 denotes a representative result of GraphBin on improving the binning result of MetaWatt for the SGA assemblies, Figure 3.4 denotes a representative result of GraphBin on improving the binning result of MaxBin2 for the MEGAHIT assemblies, Figure 3.5 denotes a representative result of GraphBin on improving the binning result of MetaBAT2 for the metaSPAdes assemblies, Figure 3.6 denotes a representative result of GraphBin on improving the binning result of SolidBin for the metaSPAdes assemblies and Figure 3.7 denotes a representative result of GraphBin on improving the binning result of BusyBee Web for the MEGAHIT assemblies over all the datasets with respect to the four evaluation criteria *precision*, *recall*, *F1-score* and *ARI*. The rest of the improved binning results can be found in Appendix A.

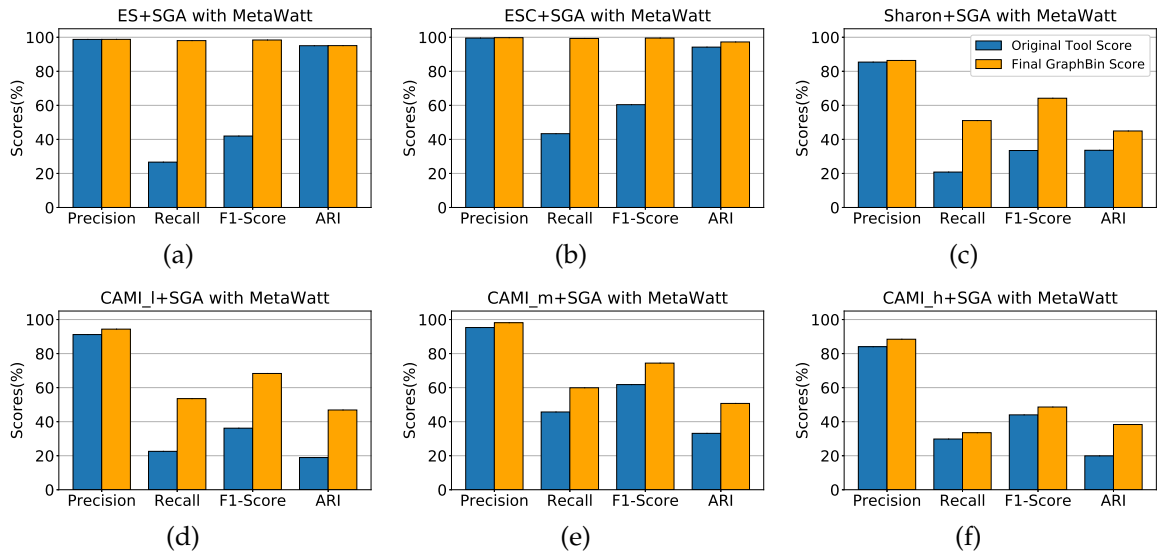


Figure 3.3: Refined binning results of MetaWatt with GraphBin for the datasets ES+SGA, ESC+SGA, Sharon+SGA, CAMI_l+SGA, CAMI_m+SGA and CAMI_h+SGA. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool (MetaWatt) compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

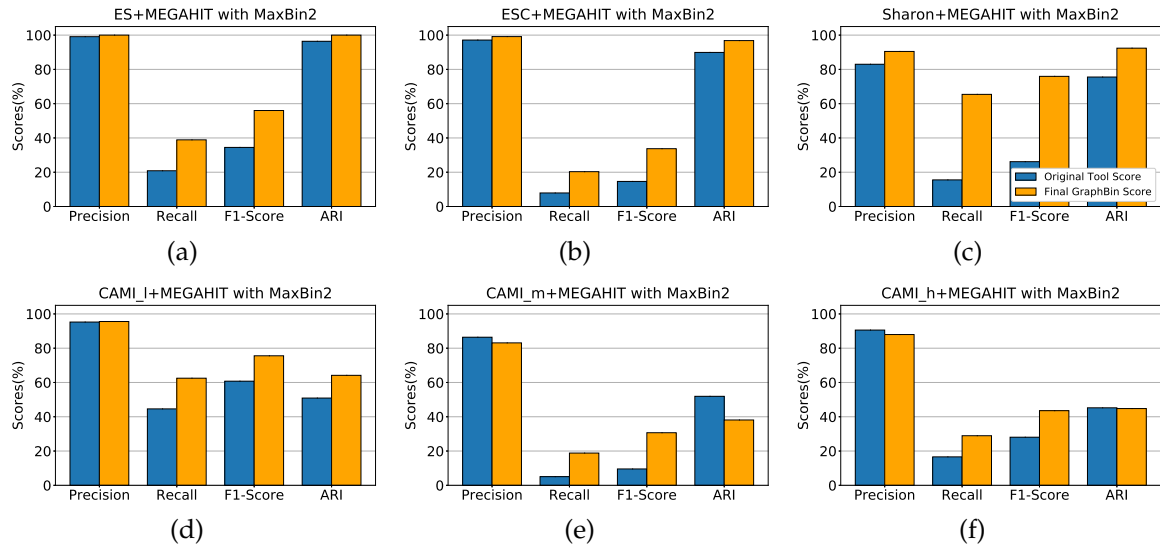


Figure 3.4: Refined binning results of MaxBin2 with GraphBin for the datasets ES+MEGAHIT, ESC+MEGAHIT, Sharon+MEGAHIT, CAMI_l+MEGAHIT, CAMI_m+MEGAHIT and CAMI_h+MEGAHIT. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool (MaxBin2) compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

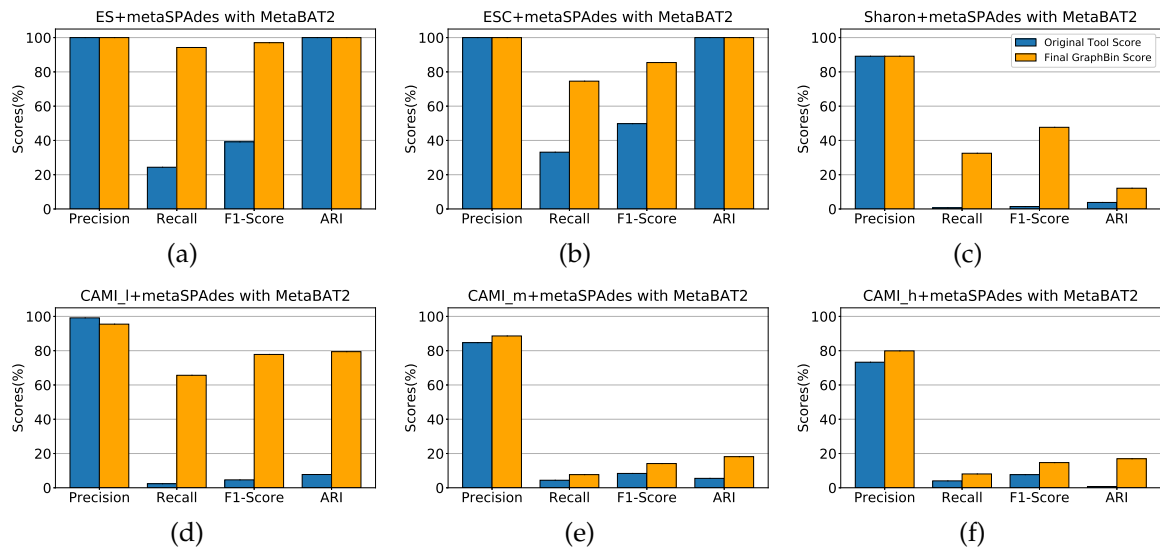


Figure 3.5: Refined binning results of MetaBAT2 with GraphBin for the datasets ES+metaSPAdes, ESC+metaSPAdes, Sharon+metaSPAdes, CAMI_l+metaSPAdes, CAMI_m+metaSPAdes and CAMI_h+metaSPAdes. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool (MetaBAT2) compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

Overall, GraphBin has been shown to improve the binning results of different combinations of the above assemblers and binning tools including MetaWatt, MaxBin2, MetaBAT2, SolidBin and BusyBee Web. One possible factor that has contributed to the improved performance of GraphBin is that the assembly graph provides valuable information regarding connections among contigs which is not used by other binning tools.

3.4.1 Results on Simulated Datasets

MetaWatt, MaxBin2, MetaBAT2 and SolidBin have shown very high precision values on the simulated datasets. However, they have shown poor *recall* values below 50%. The reason for such poor *recall*

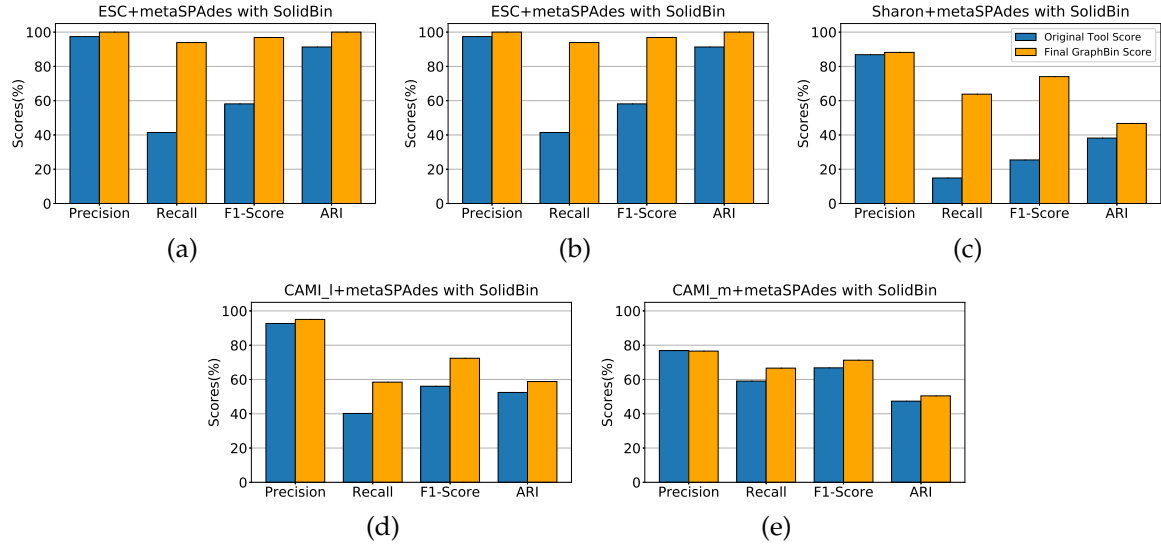


Figure 3.6: Refined binning results of SolidBin with GraphBin for the datasets ES+metaSPAdes, ESC+metaSPAdes, Sharon+metaSPAdes, CAMI_l+metaSPAdes and CAMI_m+metaSPAdes. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool (SolidBin) compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin. CAMI_h+metaSPAdes dataset could not be binned using SolidBin due to insufficient memory (576GB).

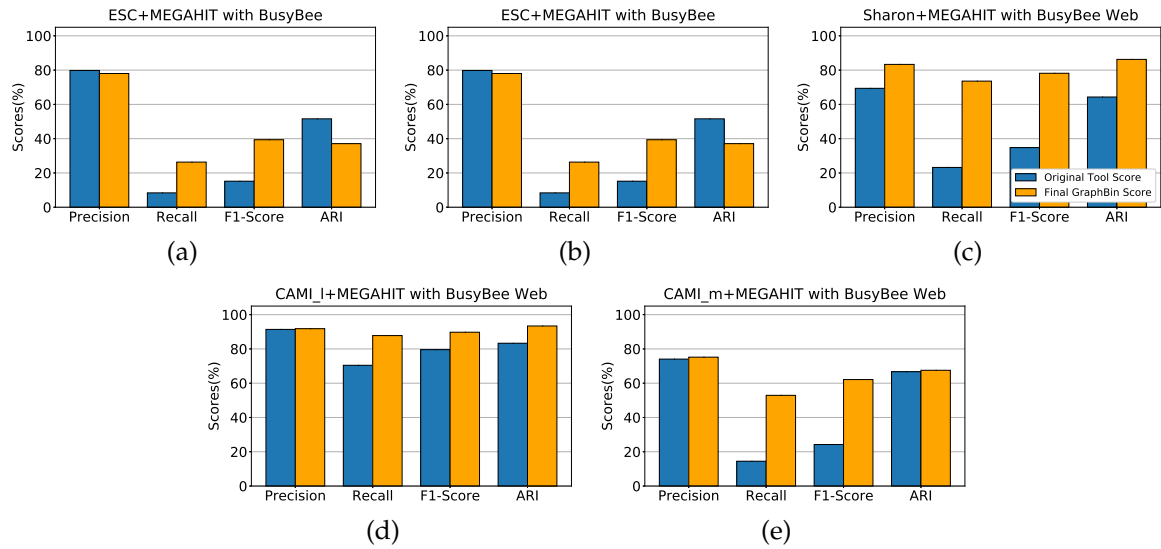


Figure 3.7: Refined binning results of BusyBee Web with GraphBin for the datasets ES+MEGAHIT, ESC+MEGAHIT, Sharon+MEGAHIT, CAMI_l+MEGAHIT and CAMI_m+MEGAHIT. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool (BusyBee Web) compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin. CAMI_h+MEGAHIT dataset could not be binned using BusyBee Web due to the restriction on input file size.

values is that these tools have discarded more than 50% of the contigs due to short lengths in each dataset. MaxBin2 and SolidBin do not bin contigs which are shorter than 1,000 bp and BusyBee Web does not bin contigs which are shorter than 500 bp due to insufficient k -mer frequency information. MetaBAT2 has a fixed value for the minimum contig length which is set to 1,500 bp while MetaWatt discards short contigs as the estimation of their relative coverage and tetranucleotide frequency may not be accurate. For example, Figure 3.3 (a) demonstrates that GraphBin has significantly increased the *recall* and *F1-score* from 26.65% and 41.98% to 98.04% and 98.40% respectively in the **ES+SGA** dataset. Similarly, GraphBin has improved the *recall* and *F1-score* from 43.37% and 60.41% to 99.77% and 99.54% respectively for the **ESC+SGA** dataset (refer to Figure 3.3 (b)). Further results can be

found in Appendix A.

3.4.2 Results on Sharon Datasets

We also tested GraphBin using the **Sharon** dataset (Sharon et al., 2013). As shown in Figure 3.3 (c), GraphBin has improved all the evaluation criteria in the **Sharon+SGA** dataset using the MetaWatt result. Specifically, GraphBin has been able to improve the *recall* and *F1-score* from 20.81% and 33.47% to 51.06% and 64.19% respectively, using the MetaWatt results of the **Sharon+SGA** dataset (refer to Figure 3.3 (c)). Moreover, similar improvements on *recall* and *F1-score* can be seen from GraphBin with other binning tools. Note that the increase of *recall* and *F1-score* of GraphBin on **Sharon** datasets is not as significant as that on the simulated datasets. This is mainly due to the increased complexity in the real dataset which results in imperfect and fragmented assembly graphs due to noisy or chimeric reads, uneven coverage and shared genomic segments between species.

3.4.3 Results on CAMI Datasets

We further tested GraphBin using the CAMI datasets of different complexities (Sczyrba et al., 2017). As shown in Figures 3.3 (d), (e) and (f), GraphBin has improved all the evaluation criteria for binning results from MetaWatt for the SGA assemblies of three CAMI datasets. Similarly, GraphBin has

Table 3.2: Number of bins identified by TAXAassign (ground truth) and the initial binning tools for each dataset

Dataset	Number of species identified by TAXAassign	MetaWatt	MaxBin2	MetaBAT2	SolidBin	BusyBee Web
ES+metaSPAdes	2	2	2	2	2	N/A*
ES+SGA	2	2	2	2	2	2
ES+MEGAHIT	2	2	2	2	2	2
ESC+metaSPAdes	3	3	3	3	3	N/A*
ESC+SGA	3	3	3	3	3	3
ESC+MEGAHIT	3	3	3	3	3	3
Sharon+metaSPAdes	26	39	11	34	9	5
Sharon+SGA	19	36	8	36	5	9
Sharon+MEGAHIT	24	38	10	65	9	8
CAMI_l+metaSPAdes	25	65	22	101	20	16
CAMI_l+SGA	27	74	15	77	15	15
CAMI_l+MEGAHIT	13	59	24	112	21	20
CAMI_m+metaSPAdes	25	48	32	152	26	37
CAMI_m+SGA	25	46	20	100	16	17
CAMI_m+MEGAHIT	128	69	23	68	19	18
CAMI_h+metaSPAdes	143	233	134	892	N/A [†]	N/A [†]
CAMI_h+SGA	130	118	49	311	N/A [‡]	58
CAMI_h+MEGAHIT	162	261	167	225	N/A [†]	N/A [†]

* BusyBee Web was unable to bin the datasets **ES+metaSPAdes** and **ESC+metaSPAdes** as there were not enough sequences that had the minimum contig length of 500 bp.

[†] **CAMI_h+metaSPAdes** and **CAMI_h+MEGAHIT** datasets could not be binned using BusyBee Web due to the restriction on input file size and could not be binned using SolidBin due to insufficient memory (576GB).

[‡] SolidBin did not complete binning the **CAMI_h+SGA** dataset after 72 hours.

improved most of the evaluation criteria for binning results obtained from MetaWatt, MaxBin2, MetaBAT2, SolidBin and BusyBee based on different assembly graphs built by metaSPAdes, SGA and MEGAHIT. Note that the improvement of binning results tends to diminish as the complexity of the datasets increases. Assembly graphs built from high-complexity datasets are more likely to contain misassembled contigs or shared contigs among species which can lead to errors in the initial binning results and get further propagated by GraphBin. Such assembly graphs are also prone to contain more false edges which can affect the label propagation process, causing the *precision* and *ARI* to drop. Moreover, as shown in Table 3.2, the number of bins estimated by different binning tools becomes inaccurate, especially for high-complexity datasets. Therefore, initial individual bins may contain contigs from more than one species while contigs from the same species may split into multiple bins, which may hinder the ability of GraphBin to improve the binning result through label propagation.

3.4.4 Visualisation of the Assembly Graph

Figure 3.8 denotes the labelling of the assembly graph of the **ES+metaSPAdes** dataset with the four evaluation criteria *precision*, *recall*, *F1-score* and *ARI* for the results of MaxBin2 and GraphBin. It can be seen that GraphBin has been able to improve the binning result significantly.

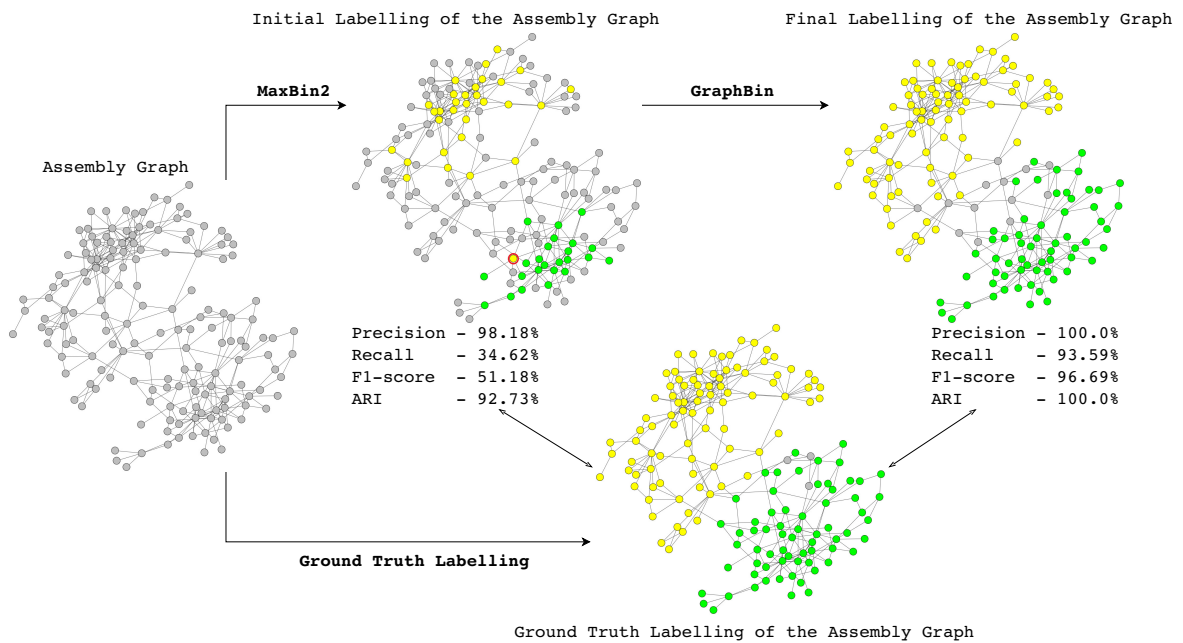


Figure 3.8: The labelling of the assembly graph of ES+metaSPAdes dataset based on the initial MaxBin2 result and GraphBin result. Contigs belonging to *E. faecalis* and *S. aureus* are coloured in green and yellow respectively. Contigs which were not considered in the classification and discarded during the binning process are coloured in grey. The vertices circled in red correspond to the mis-binned contigs. Finally, the four evaluation criteria are denoted by comparing with the ground truth labelling of the assembly graph.

3.4.5 Propagation of Labels in the Assembly Graph

The refinement of labels in the assembly graph for two simulated datasets which were used for the experiments are presented in this section. Figures 3.9 and 3.10 illustrate the refinement of labels in the assembly graph of **ES+metaSPAdes** and **ESC+metaSPAdes** datasets respectively.

ES+metaSPAdes Dataset

Figure 3.9 (a) denotes the ground truth labelling of the contigs obtained from TAXAassign. It resulted in a total of 156 labelled contigs where 67 belonged to *E. faecalis* (vertices coloured in green) and 89 belonged to *S. aureus* (vertices coloured in yellow). MaxBin2 had labelled 55 contigs. According to Figure 3.9 (b), MaxBin2 has mis-binned one contig (vertex circled in red colour). As denoted in

Figure 3.9 (d), it can be seen that the final graph after applying GraphBin has more correctly binned contigs and less discarded contigs.

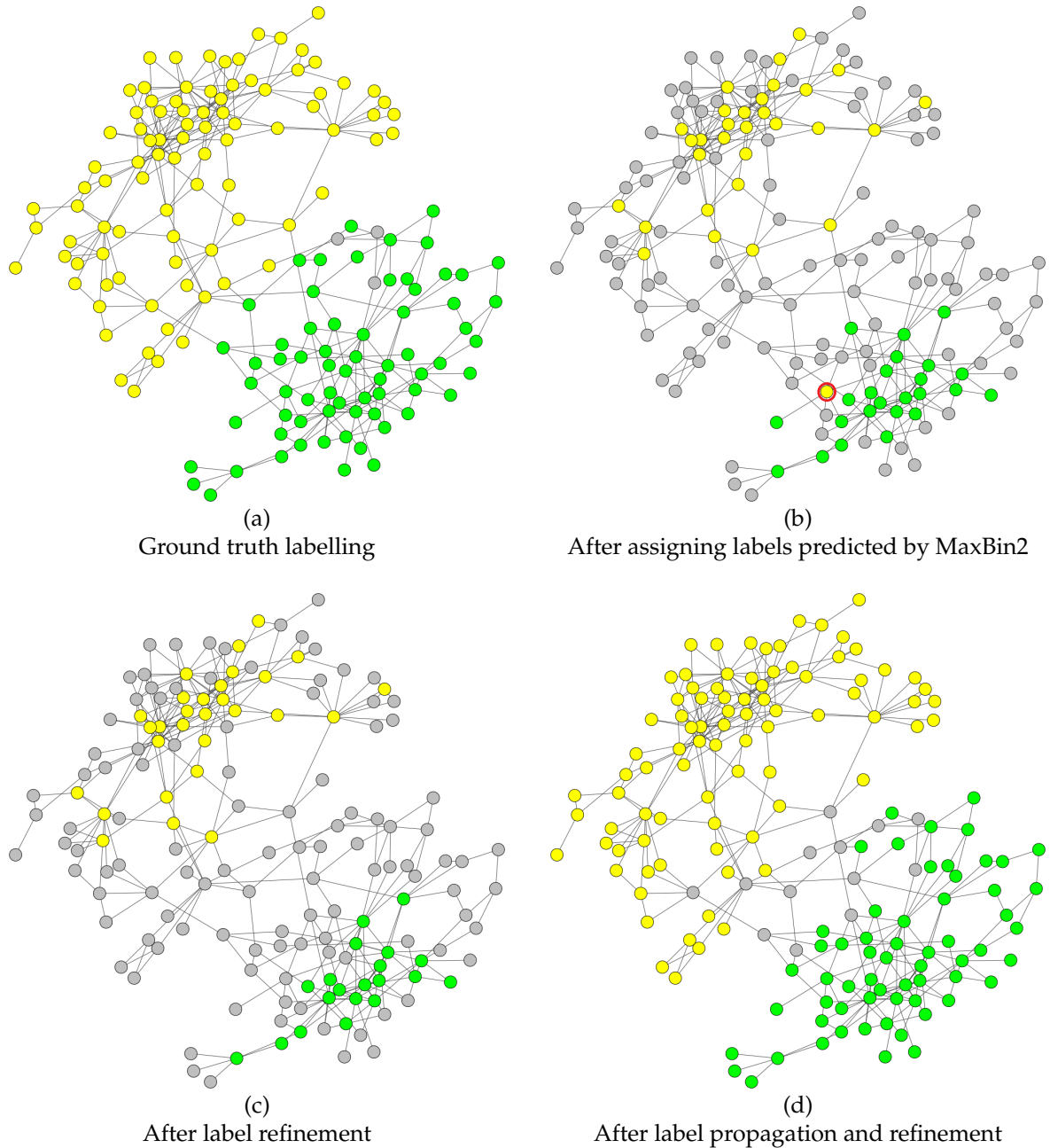


Figure 3.9: The assembly graph of the ES+metaSPAdes dataset as it undergoes the different steps in the GraphBin algorithm. It contains contigs of *E. faecalis* (green) and *S. aureus* (yellow). Contigs which were not considered in the classification and discarded during the binning process are denoted in grey. The vertices circled in red correspond to the mis-binned contigs.

ESC+metaSPAdes Dataset

Figure 3.10 (a) denotes the ground truth labelling of the contigs obtained from TAXAassign. It resulted in a total of 181 labelled contigs where 60 belonged to *E. faecalis* (vertices coloured in green), 98 belonged to *S. aureus* (vertices coloured in yellow) and 23 belonged to *C. avidum* (vertices coloured in orange). MaxBin2 had labelled 74 contigs. According to Figure 3.10 (b), MaxBin2 has mis-binned two contigs (vertices circled in red colour). As denoted in Figure 3.10 (d), it can be seen that the final graph after applying GraphBin has more correctly binned contigs and less discarded contigs.

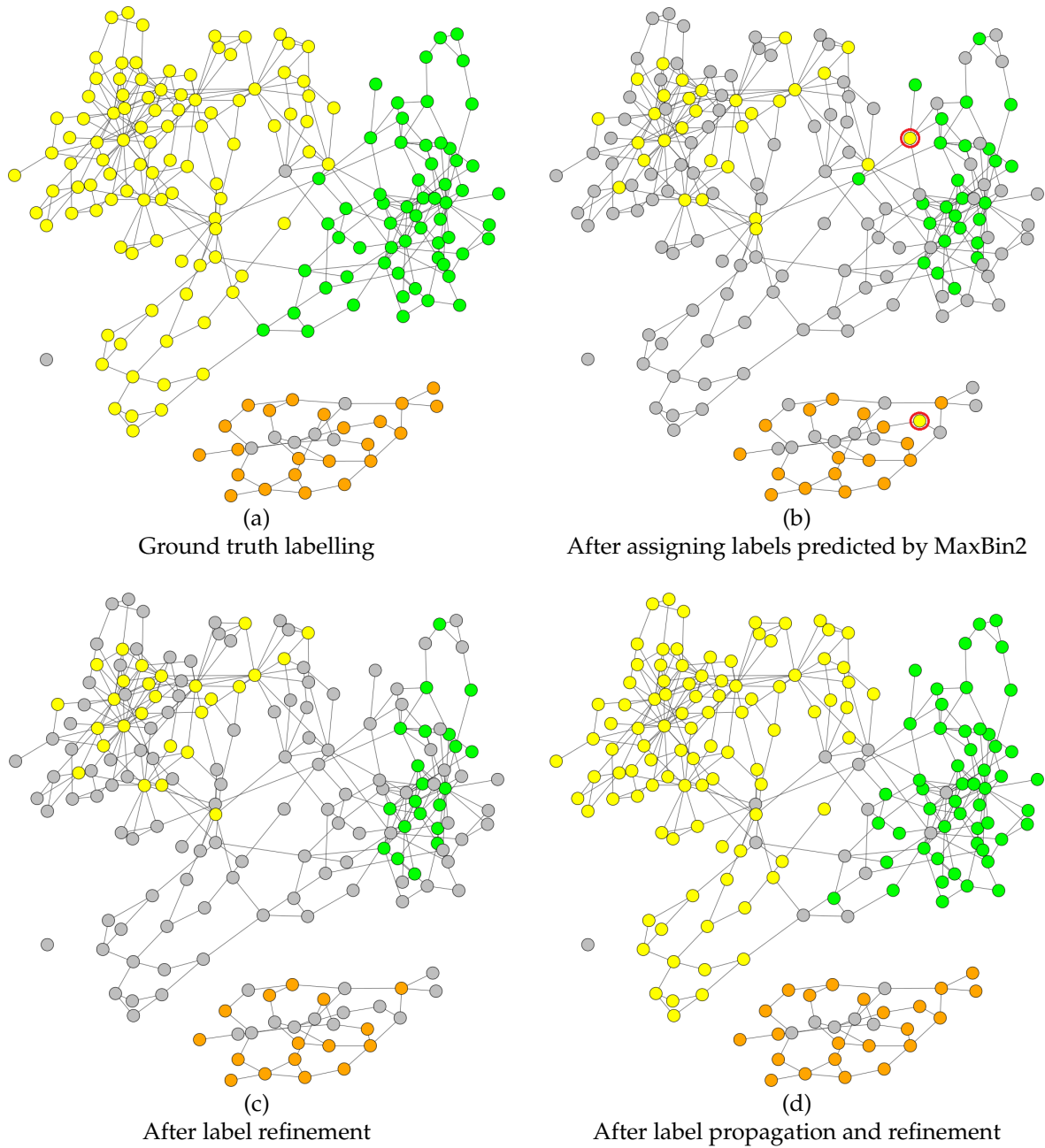


Figure 3.10: The assembly graph of the ESC+metaSPAdes dataset as it undergoes the different steps in the GraphBin algorithm. It contains contigs of *E. faecalis* (green), *S. aureus* (yellow) and *C. avidum* (orange). Contigs which were not considered in the classification and discarded during the binning process are denoted in grey. The vertices circled in red correspond to the mis-binned contigs.

3.4.6 Implementation and Runtime

The source code for the experiments was implemented using Python 3.6.5 and run on a Linux system with Ubuntu 18.04.1 LTS, 16G memory and Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz with 4 CPU cores.

The running times for assembly and binning using MaxBin2, MetaWatt, MetaBAT2, SolidBin and GraphBin were recorded for each dataset. These values are denoted in Table 3.3. Assembly of the datasets was carried out using the resources from the National Computational Infrastructure (NCI) Australia. All the binning tools (except for SolidBin on the **CAMI_m** and **CAMI_h** datasets) and GraphBin were run on a Linux system with Ubuntu 18.04.1 LTS, 16G memory and Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz with 4 CPU cores. For the **CAMI_m** and **CAMI_h** datasets, SolidBin was run

Table 3.3: Running times for assembly and binning using each tool

Dataset	Assembly (CPU time)	MetaWatt	GraphBin on MetaWatt result	MaxBin2	GraphBin on MaxBin2 result	MetaBAT2	GraphBin on MetaBAT2 result	SolidBin	GraphBin on SolidBin result
ES+metaSPAdes	1h 2m 24s	3m 45s	1s	8s	1s	5s	1s	2m 10s	1s
ES+SGA	2h 24m 43s	4m 7s	1s	8s	1s	6s	1s	1m 42s	1s
ES+MEGAHIT	1h 47m 47s	3m 27s	3s	15s	1s	6s	1s	1m 49s	1s
ESC+metaSPAdes	3h 6m 2s	5m 56s	1s	11s	1s	12s	1s	3m 11s	1s
ESC+SGA	7h 26m 10s	10m 15s	8s	11s	7s	15s	7s	4m 8s	4s
ESC+MEGAHIT	4h 45m 43s	6m 27s	2s	12s	3s	15s	2s	3m 55s	1s
Sharon+metaSPAdes	78h 57m 24s	44m 36s	54s	59s	42s	4m 27s	24s	1h 6m 57s	51s
Sharon+SGA	90h 35m 55s	1h 37m 39s	46s	45s	17s	4m 2s	10s	1h 8m 59s	17s
Sharon+MEGAHIT	29h 58m 23s	13m 54s	20s	59s	7s	4m 55s	20s	1h 10m 40s	10s
CAMIL_l+metaSPAdes	94h 27m 48s	1h 6m 31s	1m 10s	3m 46s	18s	2m 30s	39s	41m 24s	24s
CAMIL_l+SGA	73h 24m 25s	49m 42s	45m 14s	1m 41s	20m 34s	3m 13s	10m 58s	55m 25s	54m 21s
CAMIL_l+MEGAHIT	59h 54m 43s	35m 15s	38s	3m 7s	24s	2m 49s	4s	1h 3m 51s	29s
CAMIL_m+metaSPAdes	40h 59m 12s	4h 9m 30s	4m 16s	12m 33s	1m 37s	2m 39s	26s	26h 2m 50s	1m 45s
CAMIL_m+SGA	26h 50m 59s	1h 46m 34s	33s	2m 47s	25s	1m 59s	27s	23m 41s	28s
CAMIL_m+MEGAHIT	62h 55m 31s	4h 52m 33s	3h 34m 12s	5m 32s	14m 49s	4m 25s	1m 40s	23 h 27m 50s	17m 11s
CAMIL_h+metaSPAdes	371h 59m 8s	1h 28m 8s	34m 29s	2h 37m 16s	30m 17s	26m 6s	26m 17s	N/A [†]	N/A [†]
CAMIL_h+SGA	100h 42m 10s	1h 6m 50s	3h 11m 19s	19m 59s	18m 38s	7m 24s	14m 5s	N/A [†]	N/A [†]
CAMIL_h+MEGAHIT	178h 25m 27s	1h 57m 1s	4h 34m 23s	3h 6m 2s	2h 40m 49s	33m 7s	19m 3s	N/A [†]	N/A [†]

s denotes seconds, m denotes minutes and h denotes hours. The times denotes as **GraphBin on <Tool> result** represent the times taken to run GraphBin on a single CPU based on the results from MetaWatt, MaxBin2, MetaBAT2 and SolidBin.

[†]CAMIL_h+metaSPAdes and CAMIL_h+MEGAHIT datasets could not be binned using BusyBee Web due to the restriction on input file size and could not be binned using SolidBin due to insufficient memory.

[‡]SolidBin did not complete binning the CAMIL_h+SGA dataset after 72 hours.

on the NCI using 1 CPU (as SolidBin is single threaded) and 576GB (maximum memory per node in NCI) of memory. The times elapsed for binning are denoted as wall time.

Figure 3.11 denotes the graphs showing the variation of running times with the number of contigs and variation of running times with the dataset complexity (number of species present) for the metaSPAdes, SGA and MEGAHIT assemblies for the **Sharon**, **CAMI_1**, **CAMI_m** and **CAMI_h** datasets, using the three tools MetaWatt, MaxBin2 and MetaBAT2. GraphBin was run ten times with each dataset, and the average running times were used to plot the graphs. Overall, it can be seen that the running time of GraphBin increases with the increasing number of contigs and increasing number of species. Moreover, it can be seen that when MetaWatt is used, GraphBin has resulted in the highest times. This is because, MetaWatt produced the most number of labelled contigs initially out of the three tools, resulting in more number of contigs for the label refinement process of GraphBin. On the contrary, when MetaBAT2 is used, GraphBin has resulted in the lowest running times as MetaBAT2 produced the lowest number of labelled contigs initially.

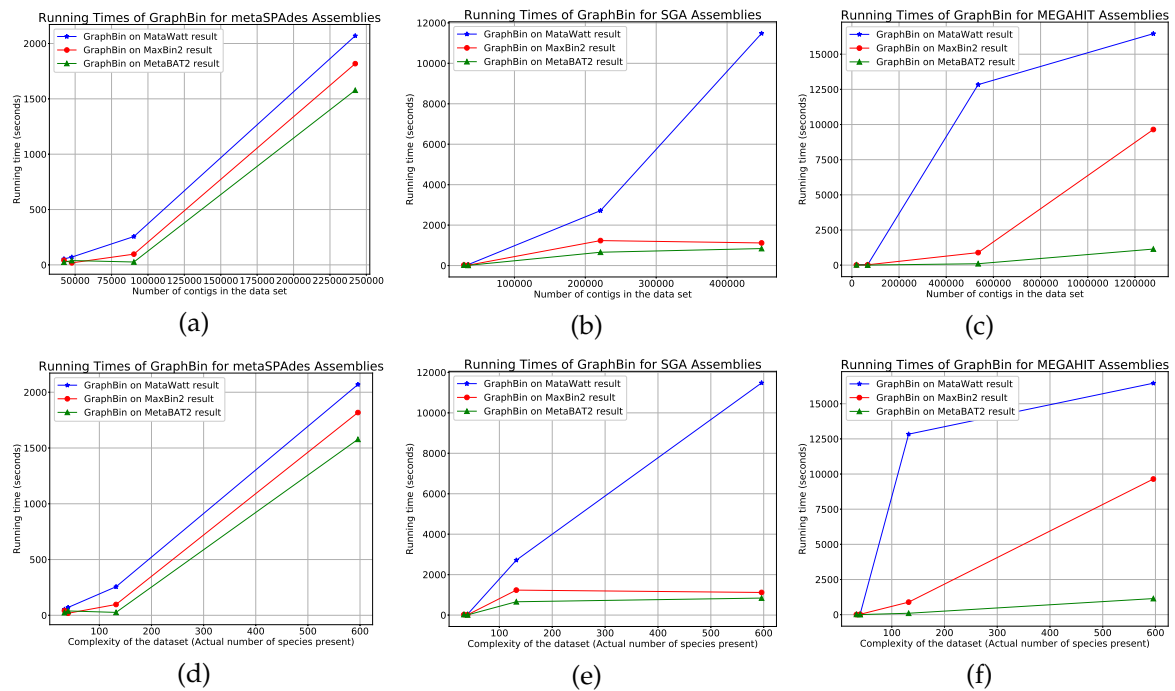


Figure 3.11: Variation of running times with the (a) - (c) number of contigs and (d) - (f) complexity of the dataset (the actual number of species present) for each of the metaSPAdes, SGA and MEGAHIT assemblies of the Sharon, CAMI_1, CAMI_m and CAMI_h datasets using the binning tools MetaWatt, MaxBin2 and MetaBAT2.

Discussion

GraphBin uses the binary connectivity information (*i.e.*, unweighted edges) between contigs, and the label propagation algorithm suffers from the problem of inconsistency when predicting labels of contigs on species boundaries. This can lead to mis-binned contigs and reduce the precision of the final GraphBin result. Moreover, the final label refinement step of GraphBin can remove a large number of labels of shared contigs along species boundaries in very complex datasets and in turn reduce the recall of binning results. Hence, it is worth exploring methods to recover these shared contigs, determine which species (or bins) they belong to and, improve the label propagation algorithm by introducing more features such as graph distance and coverage information of contigs.

With the advancement of third-generation sequencing, assembly graphs built from long reads will have more reliable connectivity information. Hence, it is worth exploring how to make use of assembly graphs from long reads in improving metagenomics binning.

Chapter 4

Overlapped Binning of Metagenomic Contigs using Assembly Graphs

This chapter was published as

V. G. Mallawaarachchi, A. S. Wickramarachchi et al. (2020b). ‘GraphBin2: Refined and Overlapped Binning of Metagenomic Contigs Using Assembly Graphs’. In: *20th International Workshop on Algorithms in Bioinformatics (WABI 2020)*. Ed. by C. Kingsford and N. Pisanti. Vol. 172. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 8:1–8:21. ISBN: 978-3-95977-161-0. DOI: [10.4230/LIPIcs.WABI.2020.8](https://doi.org/10.4230/LIPIcs.WABI.2020.8)

V. G. Mallawaarachchi, A. S. Wickramarachchi et al. (May 2021). ‘Improving metagenomic binning results with overlapped bins using assembly graphs’. *Algorithms for Molecular Biology*, 16(1), p. 3. ISSN: 1748-7188. DOI: [10.1186/s13015-021-00185-6](https://doi.org/10.1186/s13015-021-00185-6) has been extended.

4.1 Motivation and Overview

Different species may share common sequences in their genomes and an assembled contig containing such a common region may belong to multiple species. However, existing tools for binning contigs only support non-overlapped binning, *i.e.*, each contig is assigned to at most one bin. If we observe the assembly graph of two species as shown in Figure 4.1, we can see that the shared contig (contig ‘R’ in purple) is connected to contigs of both the species.

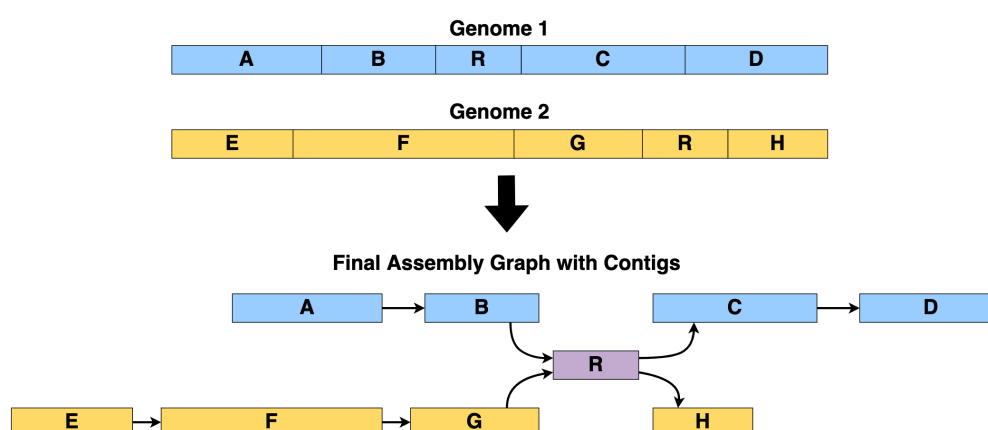


Figure 4.1: Shared contigs in the assembly graph.

In this chapter, we present GraphBin2, the new generation of GraphBin, to improve binning results using the assembly graph. While GraphBin only uses the topology information of the assembly graph, GraphBin2 improves the algorithms to adjust existing binning results and to support overlapped binning based on both the connectivity and coverage information of assembly graphs. Experimental results show that GraphBin2 not only improves existing binning results, but also infers contigs that

may belong to multiple species. Furthermore, we have experimentally shown that GraphBin2 could be applied to long-read assemblies as well.

4.2 Methods

Figure 4.2 denotes the workflow of GraphBin2. The preprocessing steps of GraphBin2 assemble reads into contigs using the assembly graph and then bin the contigs (*i.e.*, assign coloured labels to contigs) using existing contig-binning tools. GraphBin2 takes this labelled assembly graph as input, removes unsupported labels, corrects the labels of inconsistent vertices, propagates labels to unlabelled vertices and finally infers vertices with multiple labels (colours).

4.2.1 Preprocessing

In this step, we assemble the next generation reads (*e.g.*, Illumina reads with length ranging from 75 bp to 300 bp) into contigs using the assembly graph. There are two dominant paradigms for genome assembly: overlap-layout-consensus (or string graphs) (Myers, 2005) and de Bruijn graphs (Pevzner et al., 2001). We select one representative assembler from each paradigm, SGA (Simpson and Durbin, 2012) and metaSPAdes (Nurk et al., 2017) respectively, to demonstrate the adaptability of GraphBin2. In order to show that GraphBin2 could be in principle applied to long-read assemblies, we also considered a simulated dataset which was assembled using metaFlye (Kolmogorov, Bickhart et al., 2020), a popular metagenomics long-read assembler.

In the assembly graph, each vertex represents a contig with *coverage* denoting the average number of reads that map to each base of the contig and each edge indicates a significant overlap between a pair of contigs. In an ideal case, a genome corresponds to a path in the assembly graph and its genomic sequence corresponds to the concatenation of contigs along this path. Hence, if two contigs are connected by an edge in the assembly graph, they are more likely to belong to the same genome. Previous studies (Barnum et al., 2018; Mallawaarachchi, Wickramarachchi et al., 2020a) have shown that the connectivity information between contigs can be used to refine and improve binning results. In the assembly graph of metagenomic datasets, different genomes usually correspond to different paths in the assembly graph. If two genomes share a common contig (*e.g.*, unresolved “interspecies repeat” (Nurk et al., 2017)), the corresponding vertex would be shared by two genomic paths in the assembly graph.

After assembling reads into contigs using assembly graphs, GraphBin2 uses an existing contig-binning tool to derive an initial binning result. Note that most of the existing tools for binning contigs require a minimum length for the contigs (*e.g.*, 1,000 bp for MaxBin2 (Wu, Simmons et al., 2015) and SolidBin (Wang, Wang et al., 2019), 500 bp for BusyBee Web (Laczny, Kiefer et al., 2017) and 1,500 bp for MetaBAT2 (Kang, Li et al., 2019)). Therefore, many short contigs in the assembly graph will be discarded, resulting in low recall values as a common limitation of existing binning tools. For example, 65% of the contigs in the metaSPAdes assembly of the **Sharon-All** dataset were discarded by MaxBin2 due to their short length.

4.2.2 Step 1: Remove Labels of Unsupported Vertices

A linear (or circular) chromosome usually corresponds to a path (or a cycle) that traverses multiple vertices in the assembly graph. If two contigs belong to the same chromosome, they are likely to be connected by a path which consists of other contigs from the same chromosome. Therefore, a labelled vertex is defined as *supported* if and only if one of the following conditions hold:

- It is an isolated vertex
- It directly connects to a vertex of the same label
- It connects to a vertex of the same label through a path that consists of only unlabelled vertices

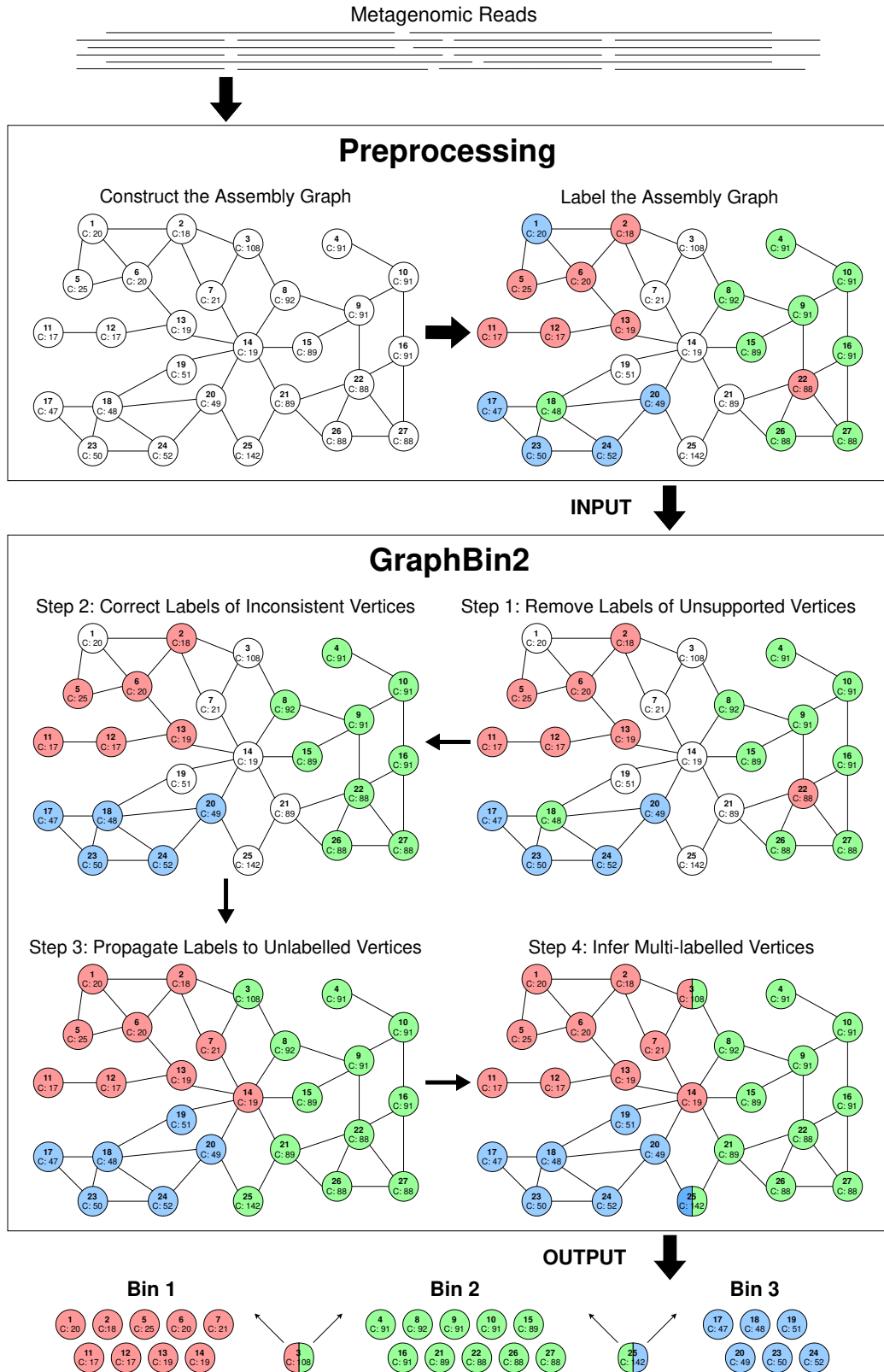


Figure 4.2: The workflow of GraphBin2.

Otherwise, a labelled vertex is defined as *unsupported*. Note that the definition of *unsupported* vertices

in GraphBin2 is more strict than *ambiguous* vertices in GraphBin.¹ For example, in the initial labelled assembly graph of Figure 4.2, vertex 2 in red is supported by vertex 6 in red as they are directly connected. Note that vertex 18 in green is also supported by vertex 15 in green as there exists a path (*i.e.*, $18 \rightarrow 19 \rightarrow 14 \rightarrow 15$) between them that traverses only unlabelled vertices (*i.e.*, 19 and 14). However, vertex 1 in blue is unsupported as it cannot reach another blue vertex through a path consisting of only unlabelled (white coloured) vertices.

To check whether a labelled vertex is *supported* or *unsupported*, a naive approach is to perform a breadth-first-search from each labelled vertex. A refined algorithm first initialises all labelled vertices as unsupported and scans the graph to identify all labelled vertices that are either isolated or directly connected to a vertex of the same label and classifies them as supported vertices. This refined algorithm then uses breadth-first-search to find all connected components that consist of only unlabelled vertices and for each component *Component* stores a set of labelled vertices $N(\text{Component})$ that are connected to vertices in *Component*. If multiple labelled vertices in $N(\text{Component})$ have the same label, these vertices are supported because they connect to each other through a path that consists of only unlabelled vertices in *Component*. GraphBin2 removes the labels for all unsupported vertices because these labels may not be reliable. For example, the label of the unsupported vertex 1 is removed by GraphBin2 in Step 1 of Figure 4.2.

4.2.3 Step 2: Correct Labels of Inconsistent Vertices

After Step 1, each non-isolated labelled vertex v is supported by at least one vertex with the same label. The closer two vertices are in the assembly graph, the more likely they have the same label. For each vertex v , we introduce a *labelled score*, $S(v, x)$, for each label x by considering all vertices of label x that are directly connected to v or connected to v through a path that consists of only unlabelled vertices. A vertex t of label x contributes to $S(v, x)$ by $2^{-D(v, t)}$ where $D(v, t)$ is the shortest distance between v and t using only unlabelled vertices. This distance is measured by the number of edges in a path and $D(v, t) = 1$ if v and t are directly connected. Therefore, the *labelled score* $S(v, x)$ is the sum of contributions from all vertices of label x that are directly connected to v or connected to v through a path that consists of only unlabelled vertices. In Step 1 of Figure 4.2, vertex 17 contributes $1/2$ to $S(18, \text{blue})$ because $D(17, 18) = 1$ and vertex 8 contributes $1/8$ to $S(18, \text{green})$ because $D(8, 17) = 3$. The labelled score of $S(18, \text{blue})$ is 2 to which all four blue vertices 17, 20, 23 and 24 contribute $1/2$ respectively while $S(18, \text{green}) = 5/16$ to which vertex 8 contributes $1/8$, vertex 15 contributes $1/8$ and vertex 26 contributes $1/16$.

A labelled vertex v of label x is defined as *inconsistent* if and only if the *labelled score* of its current label x times α is less than or equal to the *labelled score* of another label y where α is a parameter, *i.e.*, $\alpha \times S(v, x) \leq S(v, y)$. We have set $\alpha = 1.5$ in the default settings of GraphBin2. In Step 1 of Figure 4.2, vertex 18 in green is an inconsistent vertex because $1.5 \times S(18, \text{green}) = 1.5 \times 5/16 = 0.47$ is less than $S(18, \text{blue}) = 2$.

Again, GraphBin2 uses the breadth-first-search to check if a labelled vertex is *inconsistent*. GraphBin2 corrects the label of an inconsistent vertex v to another label that maximises the labelled score. For example, GraphBin2 corrects the label of vertex 18 from green to blue and corrects the label of vertex 22 from red to green (refer from Step 1 to Step 2 in Figure 4.2).

4.2.4 Step 3: Propagate Labels to Unlabelled Vertices

As existing contig-binning tools discard contigs due to their short lengths in the initial binning, many vertices are still unlabelled in the current assembly graph. In this step, we will propagate existing labels to the remaining unlabelled vertices using the assembly graph. There are two intuitions behind this label propagation process. Firstly, vertices that are closer to each other in the assembly graph are more likely to have the same label. Secondly, vertices with similar coverages are more likely to have the same label because contigs from the same genome usually have similar coverages (Herath et al.,

¹In GraphBin, a vertex i is denoted as an *ambiguous* vertex if at least one of its closest labelled vertices has a label that is different than the label of the vertex i .

An *ambiguous* vertex in GraphBin may be *supported* (in GraphBin2) by another vertex of the same label if they are directly connected or connected through a path consisting of only unlabelled vertices. An *unsupported* vertex in GraphBin2 is always *ambiguous* in GraphBin.

2017; Wu, Tang et al., 2014). GraphBin2 uses both the connectivity and coverage information of the assembly graph to propagate the labels.

For each unlabelled vertex v with coverage $c(v)$ (i.e., coverage of the contig that corresponds to the vertex), a candidate propagation action $(D(v, t), |c(v) - c(t)|, t, v)$ is recorded as a tuple where t is the nearest labelled vertex to v , $c(t)$ is the coverage of t and $D(v, t)$ is the shortest distance between v and t (as defined in Step 2). Given two candidate propagation actions, (d_1, c_1, t_1, v_1) and (d_2, c_2, t_2, v_2) , GraphBin2 will execute (d_1, c_1, t_1, v_1) before (d_2, c_2, t_2, v_2) , i.e., propagating the label of t_1 to v_1 before propagating the label of t_2 to v_2 , if $(d_1 < d_2)$ or $(c_1 < c_2 \text{ and } d_1 = d_2)$. In other words, GraphBin2 puts more emphasis on the connectivity information than the coverage information because the edges in the assembly graph are expected to be more reliable than the coverage information on vertices, especially for vertices corresponding to short contigs (which are discarded by initial binning tools).

GraphBin2 first uses the breadth-first-search to compute all candidate propagation actions for unlabelled vertices and sort them into a ranked list according to the order defined above. At each iteration, GraphBin2 executes the first candidate propagation action and then updates the ranked list of candidate propagation actions. Note that one unlabelled vertex receives its label at each iteration and updating the ranked list of candidate propagation actions can be done efficiently by breadth-first-search from this unlabelled vertex.

Figure 4.3 shows how GraphBin2 propagates labels from Step 2 to Step 3 in Figure 4.2. Figure 4.3 (a) denotes the assembly graph after correcting labels of inconsistent vertices (after Step 2). The following candidate propagation actions will be executed in the given order as shown in Figure 4.3.

- (1) The candidate propagation action $(1, 0, 6, 1)$ is executed. Vertex 1 receives the red label from vertex 6 as shown in Figure 4.3 (b).
- (2) The candidate propagation action $(1, 0, 13, 14)$ is executed. Vertex 14 receives the red label from vertex 13 as shown in Figure 4.3 (c).
- (3) The candidate propagation action $(1, 1, 22, 21)$ is executed. Vertex 21 receives the green label from vertex 22 as shown in Figure 4.3 (d).
- (4) The candidate propagation action $(1, 2, 14, 7)$ is executed. Vertex 7 receives the red label from vertex 14 as shown in Figure 4.3 (e).
- (5) The candidate propagation action $(1, 3, 18, 19)$ is executed. Vertex 19 receives the blue label from vertex 18 as shown in Figure 4.3 (f).
- (6) The candidate propagation action $(1, 16, 8, 3)$ is executed. Vertex 3 receives the green label from vertex 8 as shown in Figure 4.3 (g).
- (7) The candidate propagation action $(1, 53, 21, 25)$ is executed. Vertex 25 receives the green label from vertex 21 as shown in Figure 4.3 (h).

Note that this label propagation process in GraphBin2 improves on the label propagation algorithm in GraphBin by incorporating both the connectivity and coverage information in the assembly graph. So far, GraphBin2 does not generate multi-labelled vertices. In the next step, we will show how GraphBin2 uses the labelling, connectivity and coverage information together on the assembly graph to infer multi-labelled vertices.

4.2.5 Step 4: Infer Multi-Labelled Vertices

Contigs belonging to multiple genomes correspond to multi-labelled vertices in the assembly graph. What are the characteristics of shared contigs between multiple species? Firstly, a contig shared by multiple genomes may connect other contigs in these genomes. Secondly, the coverage of a contig shared by multiple genomes should be equal to the sum of coverages of these genomes in the ideal case. After label propagation, vertices of the same label are likely to form connected components in the assembly graph and multi-labelled vertices are likely to be located along the borders between multiple connected components where distinct labels meet and have a coverage similar to the sum of the average coverages of multiple components that they belong to.

GraphBin2 checks labelled vertices that are connected to vertices of multiple different labels. The average coverage of a connected component P is calculated by $\frac{\sum c(i) \times L(i)}{\sum L(i)}$ for each vertex i in the

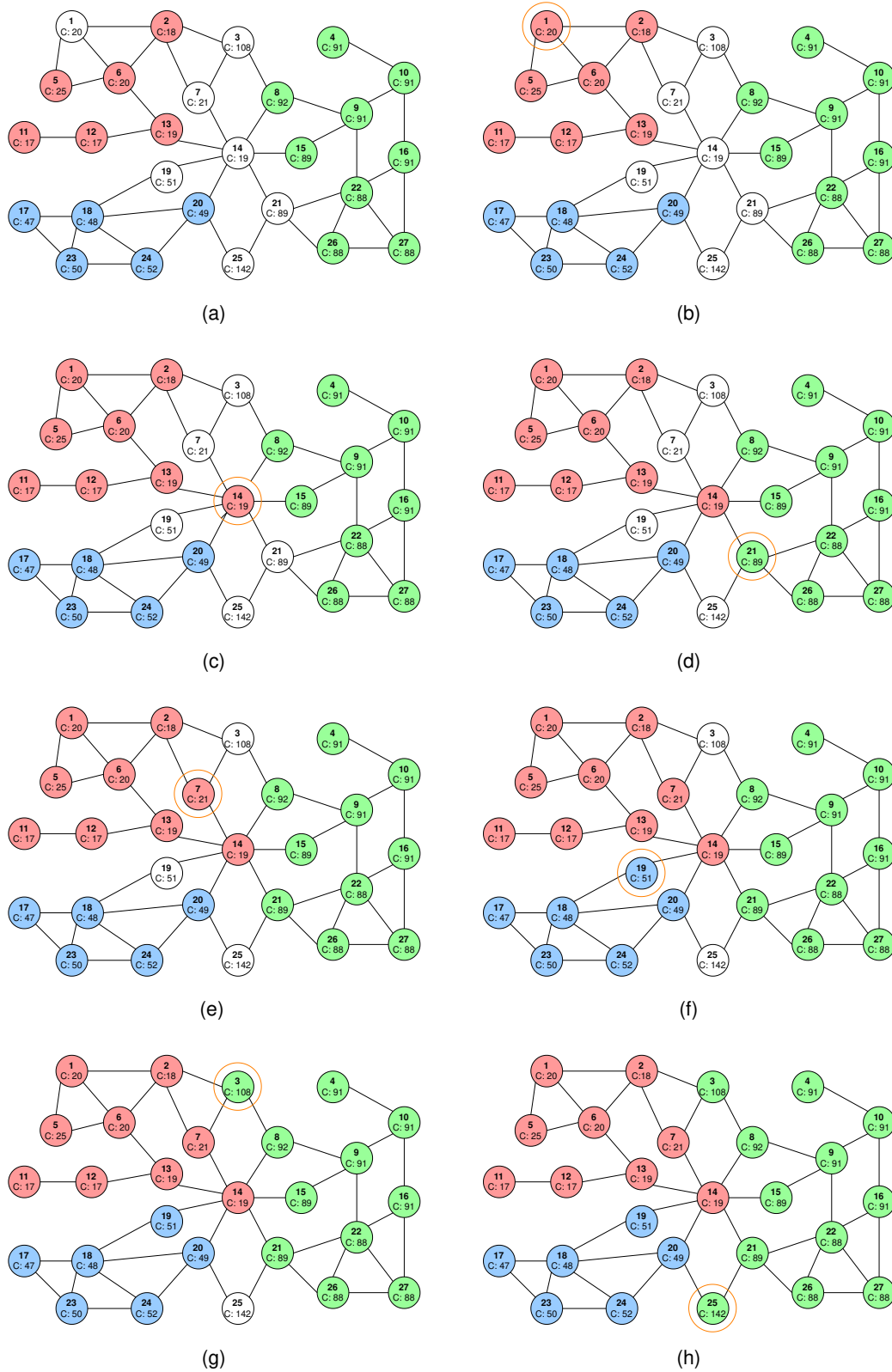


Figure 4.3: Step-by-step illustration of how labels are propagated in Step 3 of the GraphBin2 Workflow on the example assembly graph.

connected component P , where $c(i)$ is the coverage of the vertex i and $L(i)$ is the length of the contig corresponding to vertex i . Assume v is a labelled vertex v from a component P , the coverage of v is $c(v)$ and the average coverage of P is $c(P)$. When $c(v)$ is larger than $c(P)$ and v is connected

to other components P_1, P_2, \dots, P_k with different labels, it is possible that v also belongs to one or more components (in addition to P). For example, if v belongs to P, P_i and P_j in the ground-truth, the coverage of v , $c(v)$, is expected to be close to the sum of average coverages of the above three components, $c(P) + c(P_i) + c(P_j)$. In fact, finding which components in $\{P_1, P_2, \dots, P_k\}$ that v also belongs to (in addition to P) can be modelled as the following subset sum problem (Garey and Johnson, 1979). Given a set of positive numbers $\{c(P_1), c(P_2), \dots, c(P_k)\}$, find a subset whose sum is or is closest to $c(v) - c(P)$. Then v will be assigned to the corresponding components in this subset as well as to P . Note that it is possible that the selected subset is empty and thus v only belongs to P .

In all of our experiments, the maximum number of different components that a vertex connects to in the assembly graph is less than 5. We use a brute-force way to enumerate all possible combinations of components and find out the combinations that best explain the observed coverages. For example, after Step 3 in Figure 4.2, vertex 3 in green connects to another red component. The coverage of vertex 3 is 108 while the average coverage of the green component is 95 and the average coverage of the red components is 19. Because the coverage of vertex 3 (108) is closer to the sum of average coverages of green and red components ($95+19=114$) compared to the average coverage of the green component (95), vertex 3 is assigned both green and red labels. Similarly, the coverage of vertex 25 (142) is closer to the sum of average coverages of green and blue components ($95+49=144$) compared to the average coverage of the green component (95). Hence, vertex 25 is assigned both green and blue labels. In the same assembly graph after Step 3 in Figure 4.2, vertex 14 in red does not gain any other labels because its own coverage is closest to the average coverage of the red component (19) compared to other possible combinations (*i.e.*, red+blue, red+green, green+blue and red+green+blue).

4.3 Experimental Setup

4.3.1 Datasets

Simulated Datasets

We simulated three metagenomic datasets according to the species found in the *simMC+* dataset (Wu, Simmons et al., 2015). These datasets were simulated each containing 5 species (referred as **Sim-5G**), 10 species (referred as **Sim-10G**) and 20 species (referred as **Sim-20G**) respectively. Paired-end reads were simulated using the tool InSilicoSeq (Gourlé et al., 2018) modelling a MiSeq instrument with 300 bp mean read length.

To benchmark the performance of GraphBin2 on complex metagenomic datasets, we simulated a dataset with the 50 most abundant species found in the *simMC+* dataset (Wu, Simmons et al., 2015). This dataset consisting of MiSeq reads is referred as **50G-SR**. Moreover, we used the *100-genomes* long-read dataset (Wickramarachchi et al., 2020) which consisted of simulated PacBio reads of 100 species to evaluate the performance of GraphBin2 on long-read assemblies. This dataset has been simulated by the long-read simulator SimLoRD (Stöcker et al., 2016) using default parameters for PacBio reads. We refer to this dataset as **100G-LR**. Further details about the simulated datasets can be found in Appendix B.

Real Datasets

We used the preborn infant gut metagenome, commonly known as the **Sharon** dataset (Sharon et al., 2013) (NCBI accession number *SRA052203*). There are 18 Illumina (Illumina HiSeq 2000) runs available for this dataset. One run *SRR492184* is included as a representative dataset (referred as **Sharon-1**) and all the 18 Illumina runs are combined to form the **Sharon-All** dataset in our experiments.

We also used the Lake Biwa bacterioplankton metagenome dataset ((Mehrshad et al., 2018)) which consists of bacterioplankton obtained from the Lake Biwa, Japan (NCBI BioProject number *PRJDB6644*, run *DRR125127*, referred as **Lake Water**) and consists of Illumina MiSeq paired-end reads. Further details on the **Sharon** and **Lake Water** datasets can be found in Appendix B.

4.3.2 Tools Used

To derive the assembly graph from short reads, there are two dominant assembly paradigms, de Bruijn graphs (Pevzner et al., 2001) and overlap-overlap-layout-consensus (or string graphs) (Myers, 2005).

We selected one representative tool from each paradigm to show the effectiveness of GraphBin2. To represent the de Bruijn graph paradigm, we used metaSPAdes (Nurk et al., 2017) (from SPAdes version 3.13.0 (Bankevich et al., 2012)) with its default parameters to generate the assembly graph. As for the overlap-layout-consensus paradigm, we selected SGA (version 0.10.15) (Simpson and Durbin, 2012) to derive the assembly graph. We used the long-read metagenomic assembler metaFlye (Kolmogorov, Bickhart et al., 2020) (available in Flye version 2.4.2 (Kolmogorov, Yuan et al., 2019)) with its default parameters to assemble the 100G-LR dataset.

We used CONCOCT (version 1.1.0) (Alneberg et al., 2014) and MaxBin2 (version 2.2.5) (Wu, Simmons et al., 2015) with default parameters, and SolidBin (version 1.3) (Wang, Wang et al., 2019) in SolidBin-SFS mode to obtain the initial binning results for our experiments. CONCOCT, MaxBin2 and SolidBin are considered as hybrid contig-binning tools as they use both the composition and coverage information. They make use of tetranucleotide frequencies and coverages of reads with different machine learning approaches to bin contigs. Note that CONCOCT, MaxBin2 and SolidBin only bin contigs which are longer than 1,000 bp by default. We also compared GraphBin2 with its predecessor GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a). The commands used to run all the assembly and binning tools can be found in Appendix D.

4.3.3 Evaluation Criteria

Since the reference genomes of the simulated datasets were known, we used BWA-MEM (Li, 2013) to align the contigs to their reference genomes to determine the ground truth species to which the contigs actually belonged to. For each contig, the alignment lengths for each species were recorded. A contig is considered to belong to one species if the longest alignment to this species covers at least 50% of the contig length. Furthermore, isolated contigs (corresponding vertices with zero degree in the assembly graph) were not considered for the ground-truth set of the datasets.

For the Sharon dataset, we considered the annotated contigs from 12 species which are available at <https://ggkbase.berkeley.edu/carrol/organisms> as references. For the Lake Water dataset, we considered the assembled genomes provided by the authors as ground truth species. A process similar to the simulated datasets was followed for the Sharon and Lake Water datasets to determine the origin species of contigs and alignment lengths to species.

To evaluate the binning results of CONCOCT (Alneberg et al., 2014), MaxBin2 (Wu, Simmons et al., 2015), SolidBin (Wang, Wang et al., 2019), GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2, we used the metrics (1) precision, (2) recall and (3) F1-score as defined in the evaluation metrics of GraphBin in Chapter 3.

To evaluate whether a vertex in the assembly graph corresponds to a contig that may belong to multiple species, we align this contig to genomes of ground-truth species and record the best alignment against each species, respectively. Then we introduce a parameter $Ratio_{(2^{nd}/1^{st})}$ as the ratio between the alignment lengths of the second longest alignment and the longest alignment. If a contig is aligned to only one species (*i.e.*, there is no alignment to another species), then $Ratio_{(2^{nd}/1^{st})} = 0$. If a contig is aligned to multiple species, the higher the $Ratio_{(2^{nd}/1^{st})}$ is, the more likely that this contig belongs to multiple species. The violin plots of $Ratio_{(2^{nd}/1^{st})}$ are computed for both inferred multi-labelled and single-labelled contigs respectively in the next section to demonstrate how $Ratio_{(2^{nd}/1^{st})}$ varies for each type of contigs.

4.4 Results and Discussion

4.4.1 Binning Results

Figures 4.4, 4.5 and 4.6 demonstrate the results of CONCOCT (Alneberg et al., 2014), MaxBin2 (Wu, Simmons et al., 2015) and SolidBin (Wang, Wang et al., 2019), respectively with GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 on top of the initial binning results for the metaSPAdes assemblies. Binning results of the SGA assemblies can be found in Figures 4.7 to 4.9. Figure 4.10 denotes the binning results of all the tools for the complex datasets 50G-SR, Lake Water and 100G-LR. The number of bins identified by the binning tools for each dataset can be found in Table 4.1.

Table 4.1: The number of bins identified by the binning tools for each dataset.

Dataset	Ground truth bins	Assembly type	Binning tool	Number of bins identified
Sim-5G	5	metaSPAdes	CONCOCT	7
			MaxBin2	5
			SolidBin	5
		SGA	CONCOCT	11
			MaxBin2	5
			SolidBin	5
Sim-10G	10	metaSPAdes	CONCOCT	12
			MaxBin2	10
			SolidBin	10
		SGA	CONCOCT	14
			MaxBin2	9
			SolidBin	9
Sim-20G	20	metaSPAdes	CONCOCT	22
			MaxBin2	21
			SolidBin	20
		SGA	CONCOCT	28
			MaxBin2	20
			SolidBin	19
Sharon-1 (Sharon et al., 2013)	12	metaSPAdes	CONCOCT	27
			MaxBin2	5
			SolidBin	5
		SGA	CONCOCT	25
			MaxBin2	5
			SolidBin	4
Sharon-All (Sharon et al., 2013)	12	metaSPAdes	CONCOCT	48
			MaxBin2	11
			SolidBin	9
		SGA	CONCOCT	27
			MaxBin2	8
			SolidBin	5
50G-SR	50	metaSPAdes	CONCOCT	44
			MaxBin2	44
			SolidBin	45
Lake Water (Mehrshad et al., 2018)	57	metaSPAdes	CONCOCT	149
			MaxBin2	57
			SolidBin	N/A*
100G-LR (Wickramarachchi et al., 2020)	100	metaFlye	CONCOCT	76
			MaxBin2	76
			SolidBin	86

* SolidBin could not be run on the Lake Water dataset due to insufficient memory.

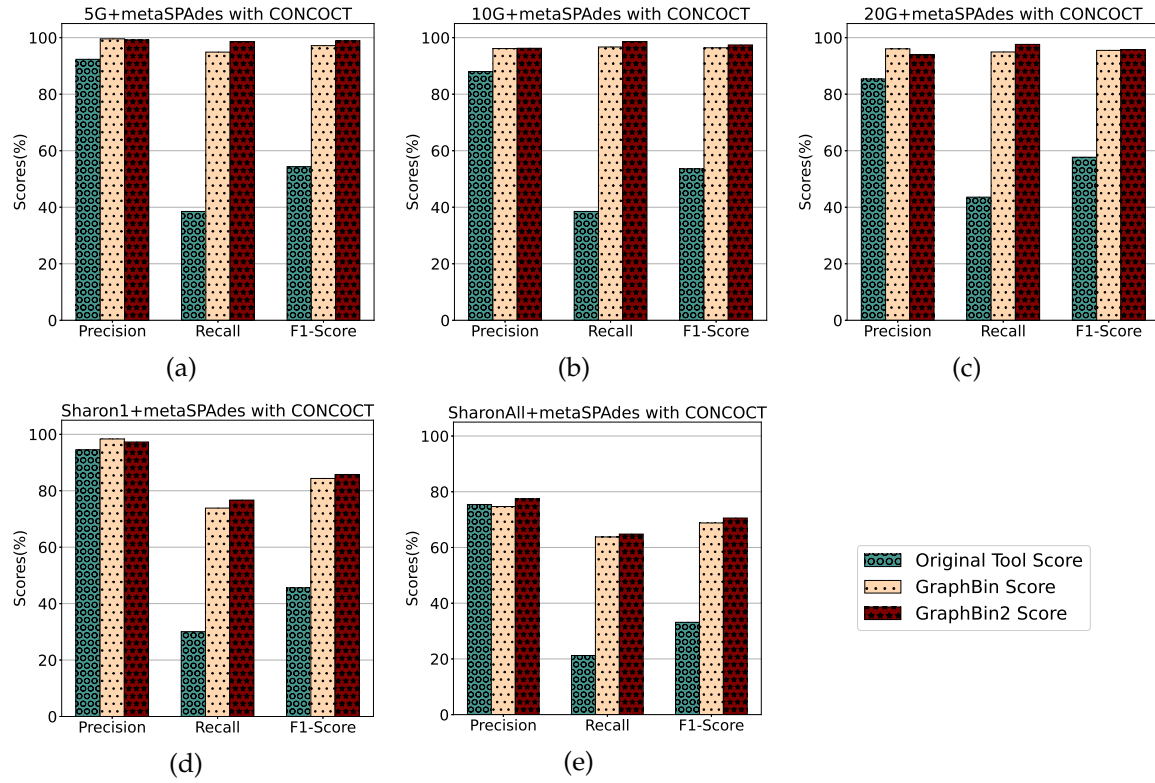


Figure 4.4: Comparison of binning results of CONCOCT (Alneberg et al., 2014), GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 (on top of CONCOCT results) using assembly graphs built by metaSPAdes (Nurk et al., 2017).

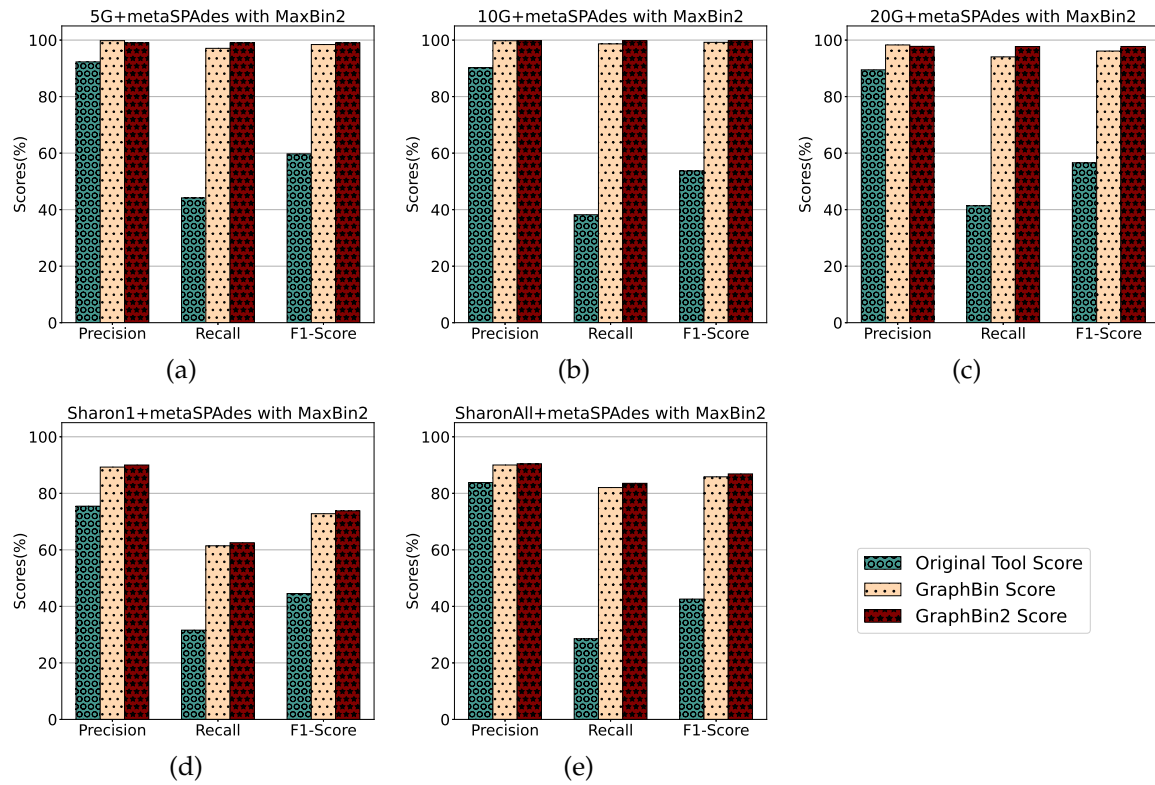


Figure 4.5: Comparison of binning results of MaxBin2 (Wu, Simmons et al., 2015), GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 (on top of MaxBin2 results) using assembly graphs built by metaSPAdes (Nurk et al., 2017).

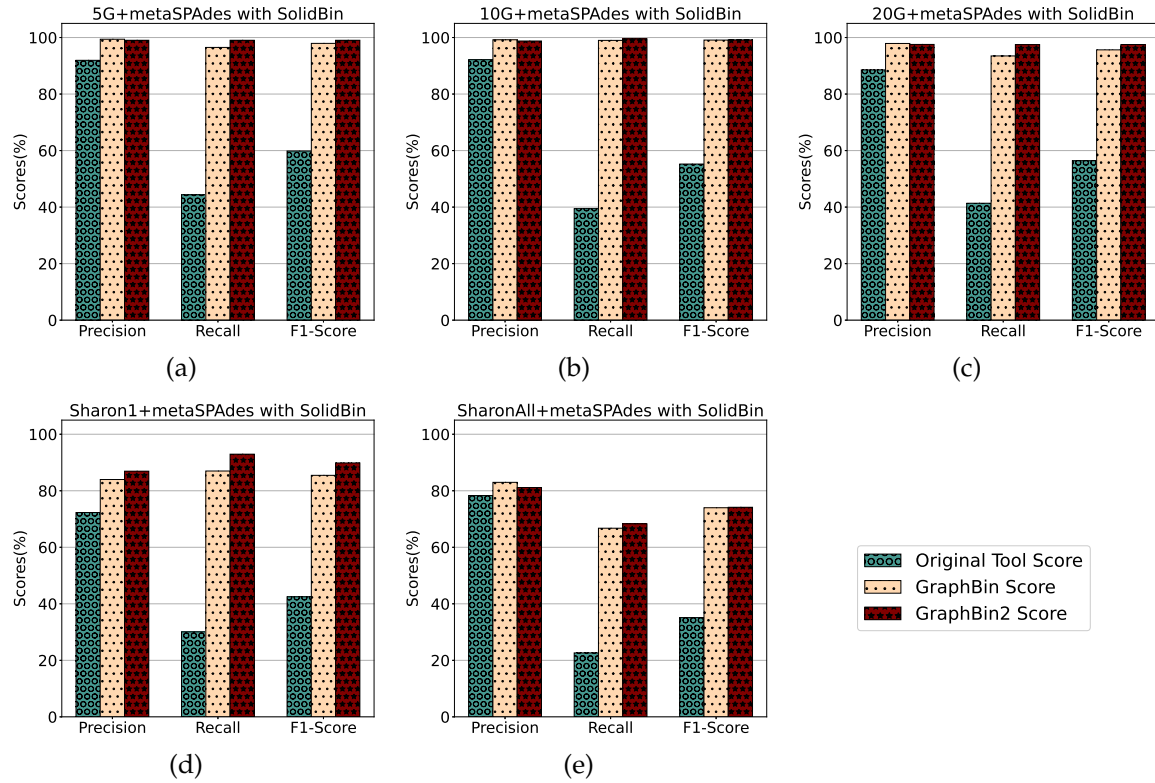


Figure 4.6: Comparison of binning results of SolidBin (Wang, Wang et al., 2019), GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 (on top of SolidBin results) using assembly graphs built by metaSPAdes (Nurk et al., 2017).

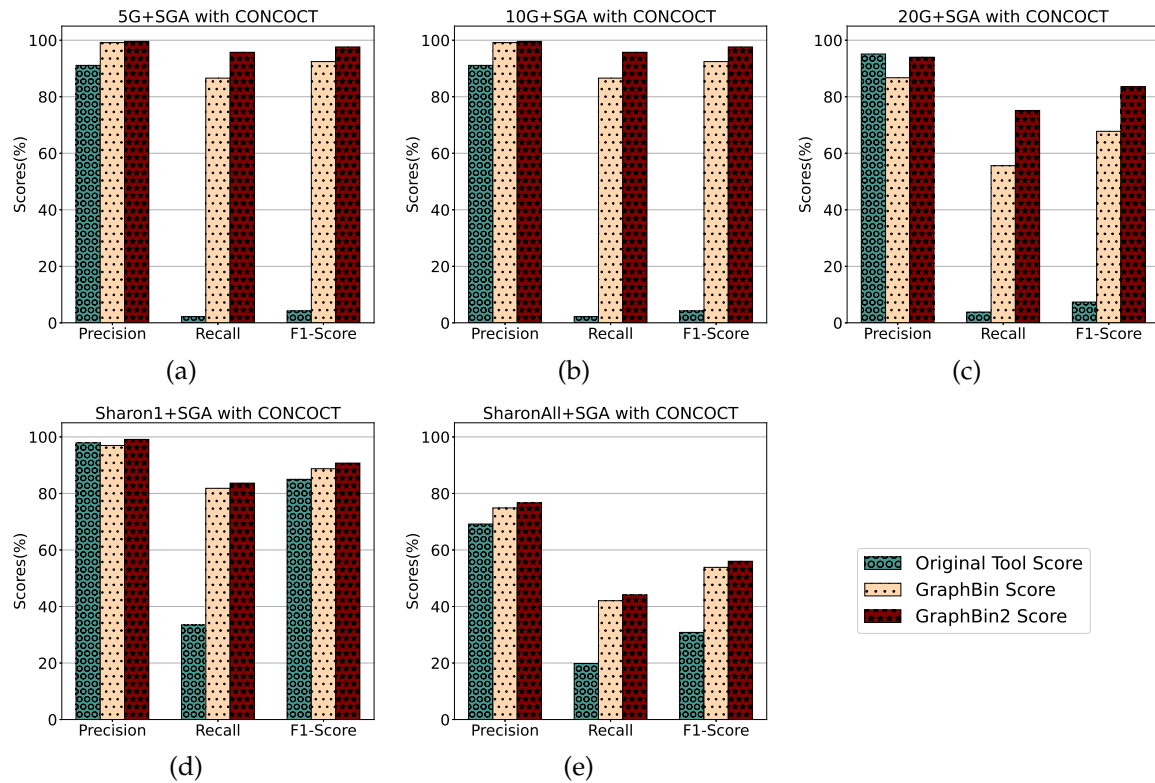


Figure 4.7: Comparison of binning results of CONCOCT (Alneberg et al., 2014), GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 (on top of CONCOCT results) using assembly graphs built by SGA (Simpson and Durbin, 2012).

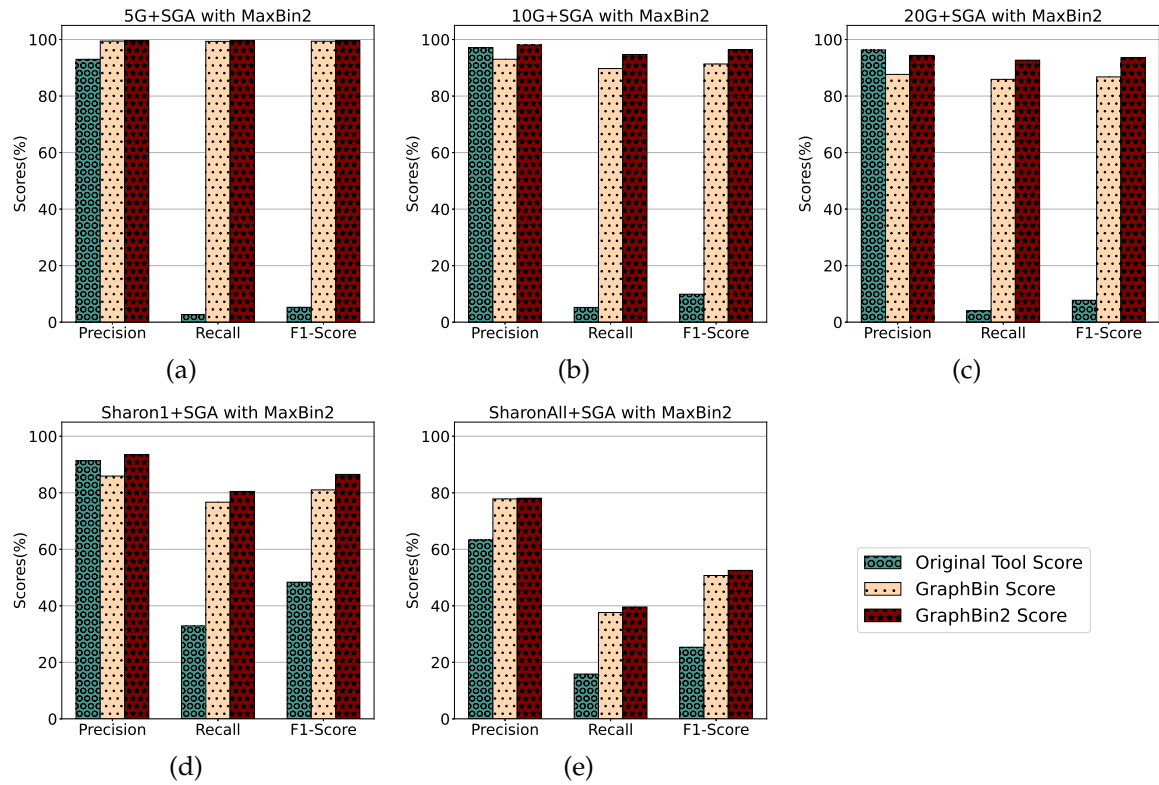


Figure 4.8: Comparison of binning results of MaxBin2 (Wu, Simmons et al., 2015), GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 (on top of MaxBin2 results) using assembly graphs built by SGA (Simpson and Durbin, 2012).

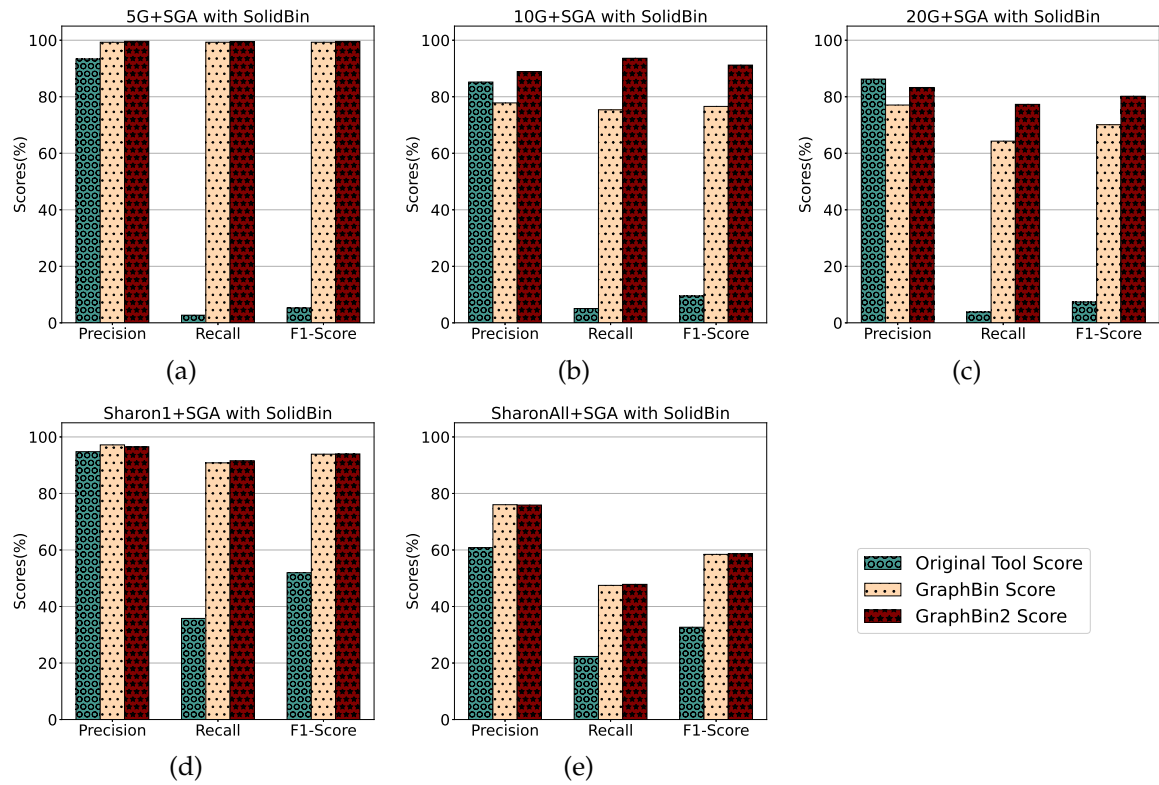


Figure 4.9: Comparison of binning results of SolidBin (Wang, Wang et al., 2019), GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 (on top of SolidBin results) using assembly graphs built by SGA (Simpson and Durbin, 2012).

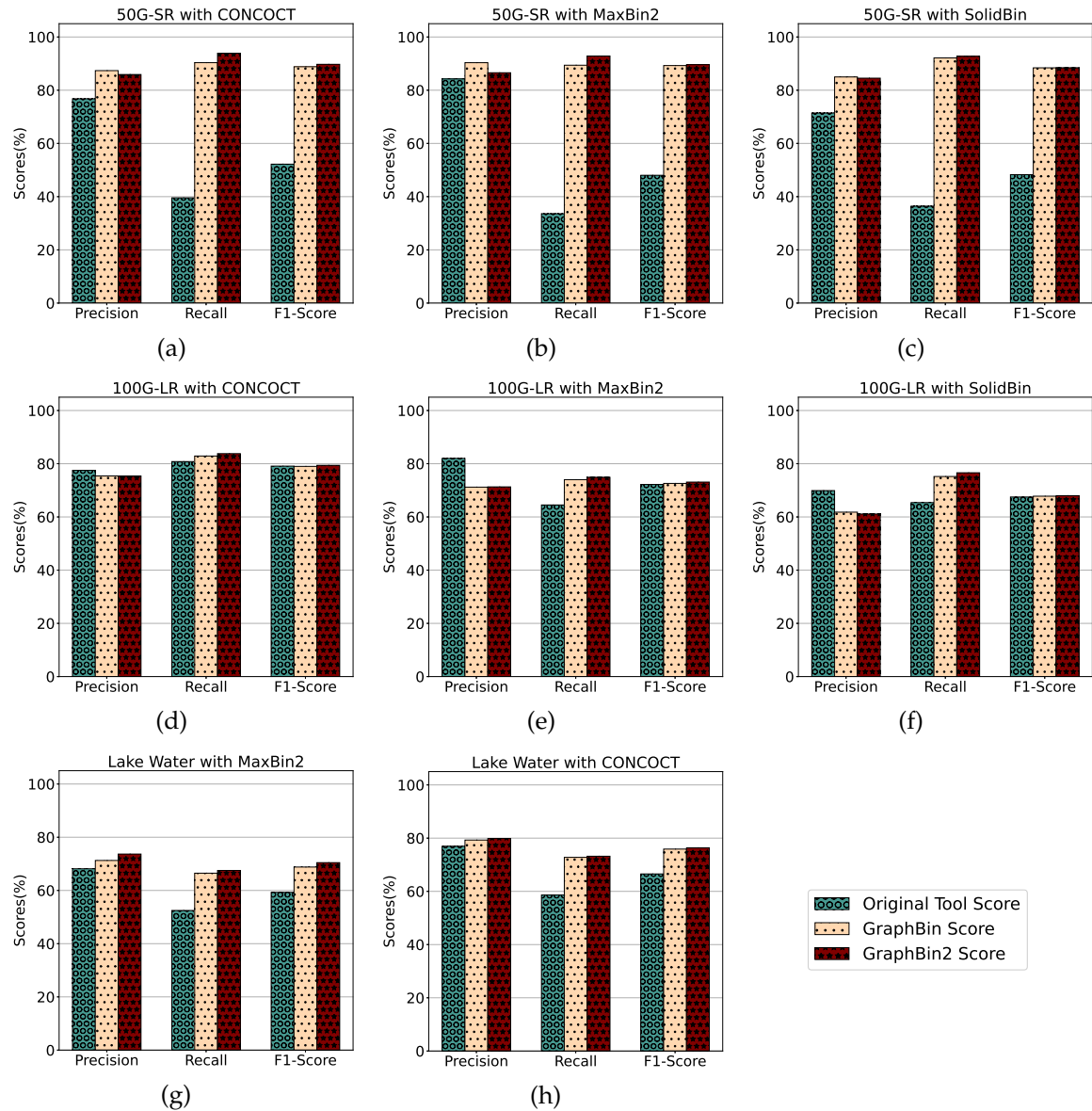


Figure 4.10: Comparison of binning results of CONCOCT (Alneberg et al., 2014), MaxBin2 (Wu, Simmons et al., 2015), SolidBin (Wang, Wang et al., 2019), GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 for the complex datasets 50G-SR, 100G-LR and Lake Water.

The binning results show that GraphBin2 achieves the best performance in most of the scenarios. The improvement over GraphBin is because GraphBin2 makes use of coverage information additionally, rather than relying only on the graph topology as GraphBin does. Both GraphBin and GraphBin2 have shown significant improvements on recall compared to CONCOCT, MaxBin2 and SolidBin. While CONCOCT, MaxBin2 and SolidBin filter contigs with length shorter than 1,000 bp, GraphBin and GraphBin2 are able to bin short contigs using assembly graphs. In a few scenarios, GraphBin2 improved on the recall with a bit of a compromise on the precision compared to GraphBin because GraphBin removes ambiguous labels in the final step. Furthermore, the existence of weak edges (*i.e.*, edges that are not well supported from the data) can form false connections between contigs and can mislead the label propagation process.

4.4.2 Multi-Labelled Inference Results

One key novelty of GraphBin2 is the introduction of the multi-labelled inference for contigs where GraphBin2 detects possible contigs that may belong to multiple species. Table 4.2 denotes the number of multi-labelled contigs identified by GraphBin2 for the metaSPAdes assemblies, SGA assemblies and assemblies of the complex datasets using the initial binning result of the binning tools CONCOCT (Alneberg et al., 2014), MaxBin2 (Wu, Simmons et al., 2015) and SolidBin (Wang, Wang et al., 2019). Moreover, for each combination of dataset and initial binning tool, we calculated the ratio $Ratio_{(2^{nd}/1^{st})}$ (please refer to Section 3.3) of single and multi-labelled contigs produced by GraphBin2. Then we plotted the violin plots of $Ratio_{(2^{nd}/1^{st})}$ in Figures 4.11 and 4.12 to demonstrate how $Ratio_{(2^{nd}/1^{st})}$ varies for different datasets. Multi-labelled inference results of the SGA assemblies can be found in Figure 4.13.

According to Figures 4.11 and 4.12, the multi-labelled contigs identified by GraphBin2 for most of the datasets have a high mean value (much greater than zero) for $Ratio_{(2^{nd}/1^{st})}$, suggesting that these identified contigs have significant alignments to multiple species. Moreover, the mean value of $Ratio_{(2^{nd}/1^{st})}$ for the single-labelled contigs identified by GraphBin2 is close to zero, suggesting that the majority of the contigs only belong to one species. The clear distinction between the $Ratio_{(2^{nd}/1^{st})}$ of inferred single and multi-labelled contigs in these datasets demonstrates the effective detection of contigs that may belong to multiple species by GraphBin2. Note that the relatively low mean value of $Ratio_{(2^{nd}/1^{st})}$ for the **Sharon-All** dataset can be due to repeats and weak edges in complex assembly graphs, *i.e.*, contigs that represent repeats within one species tend to have higher coverage and may be misinterpreted as multi-labelled contigs if there exist weak edges connecting them to contigs in other species. The possible multi-labelled contigs in the **50G-SR** and **100G-LR** datasets which are not identified by GraphBin2 may be due to the underestimation of the number of bins, misassemblies and fragmentation of the assembly graphs, especially for datasets with a large number of species.

Table 4.2: The number of multi-labelled contigs identified by GraphBin2 for the metaSPAdes assemblies, SGA assemblies and assemblies of the complex datasets using the initial binning result of each binning tool.

Dataset	Assembler	With CONCOCT result	With MaxBin2 result	With SolidBin result
Sim-5G	metaSPAdes	3	4	5
	SGA	31	6	8
Sim-10G	metaSPAdes	6	7	7
	SGA	81	9	2
Sim-20G	metaSPAdes	5	11	10
	SGA	156	15	11
Sharon1	metaSPAdes	3	3	2
	SGA	6	2	2
SharonAll	metaSPAdes	69	38	30
	SGA	40	37	17
50G-SR	metaSPAdes	89	74	74
Lake Water	metaSPAdes	178	329	N/A*
100G-LR	metaSPAdes	17	10	10

* SolidBin could not be run on the Lake Water dataset due to insufficient memory.

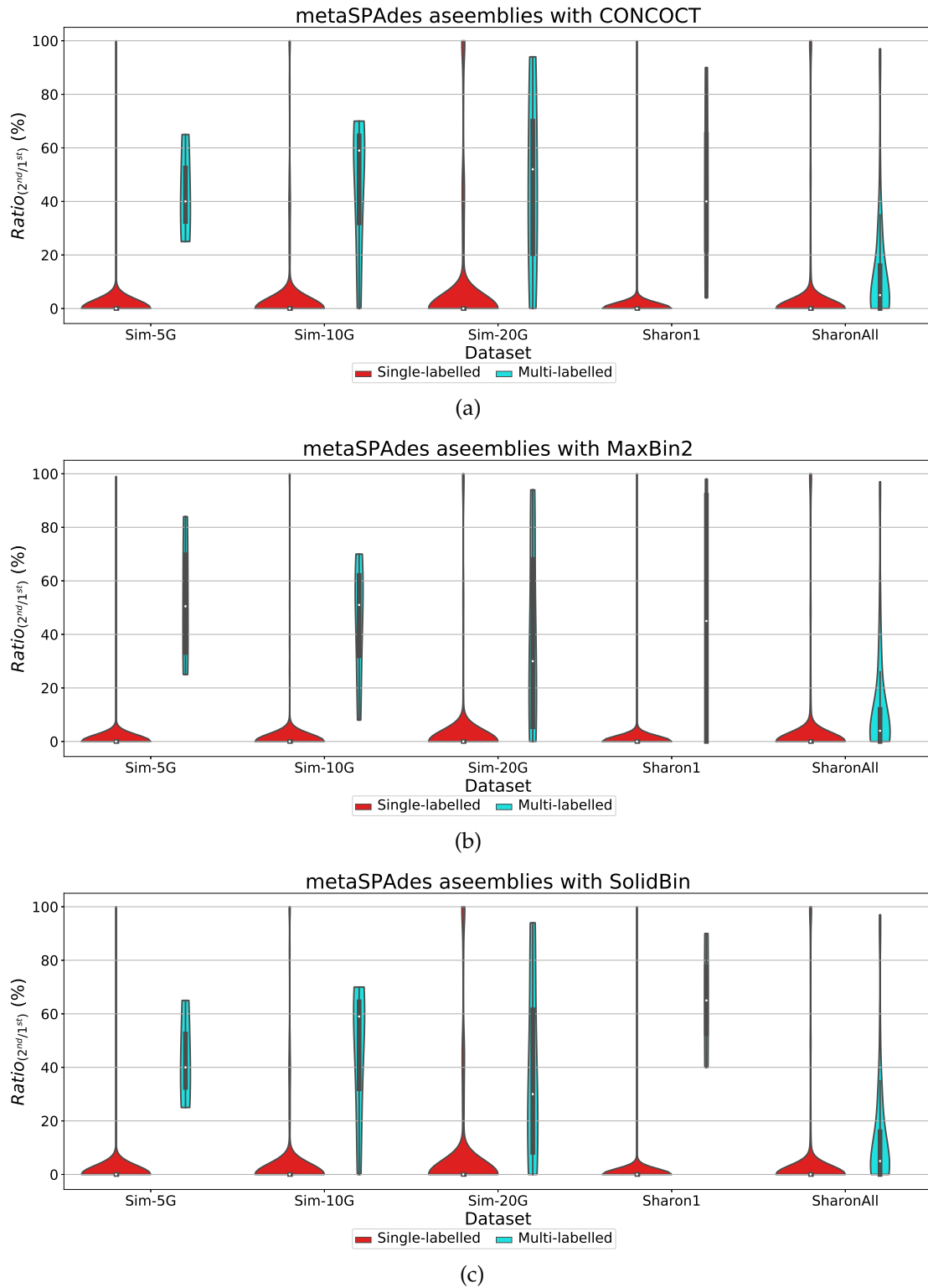


Figure 4.11: Violin plots for the ratio between the alignment lengths of the second longest alignment and the longest alignment of the single and multi-labelled inference results using GraphBin2 on top of (a) CONCOCT, (b) MaxBin2 and (c) SolidBin results for the metaSPAdes assemblies.

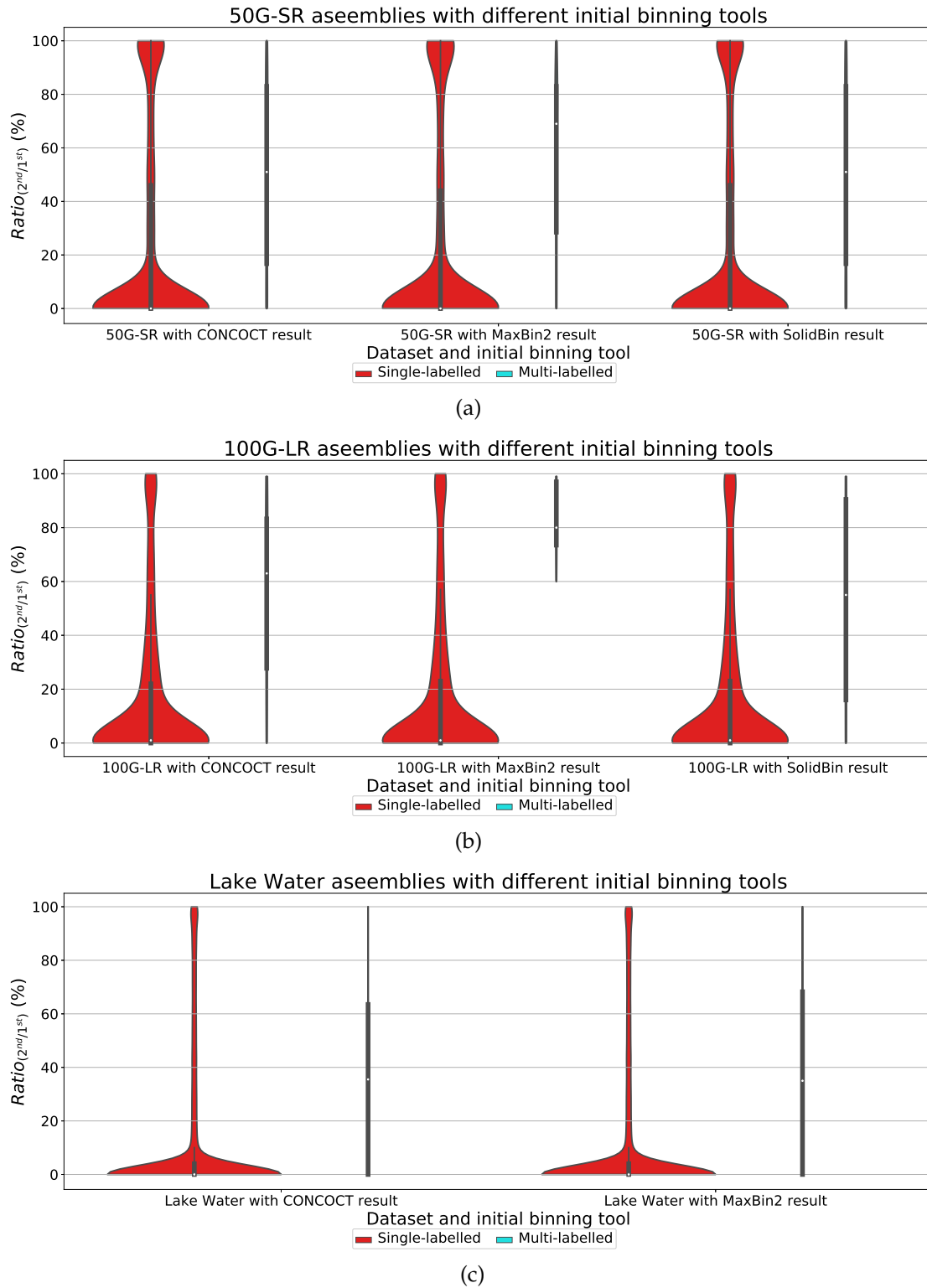
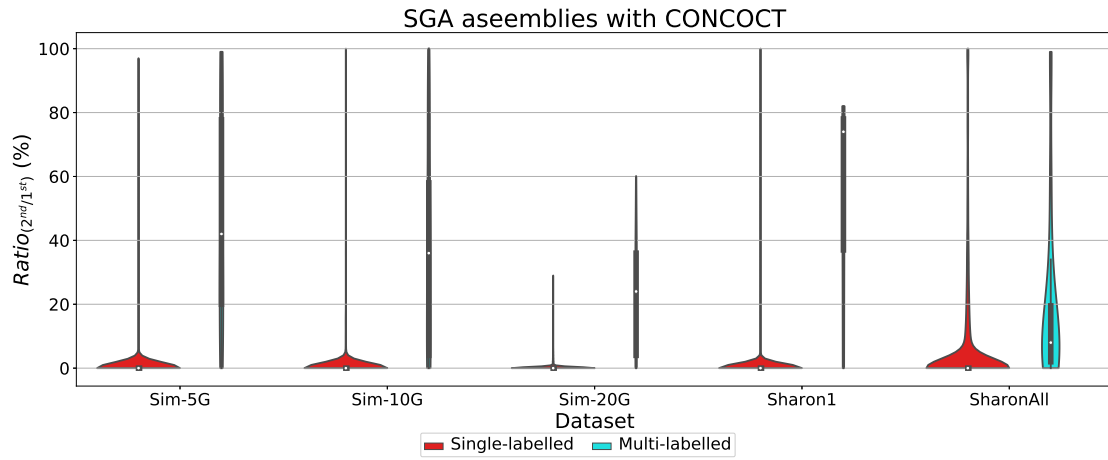
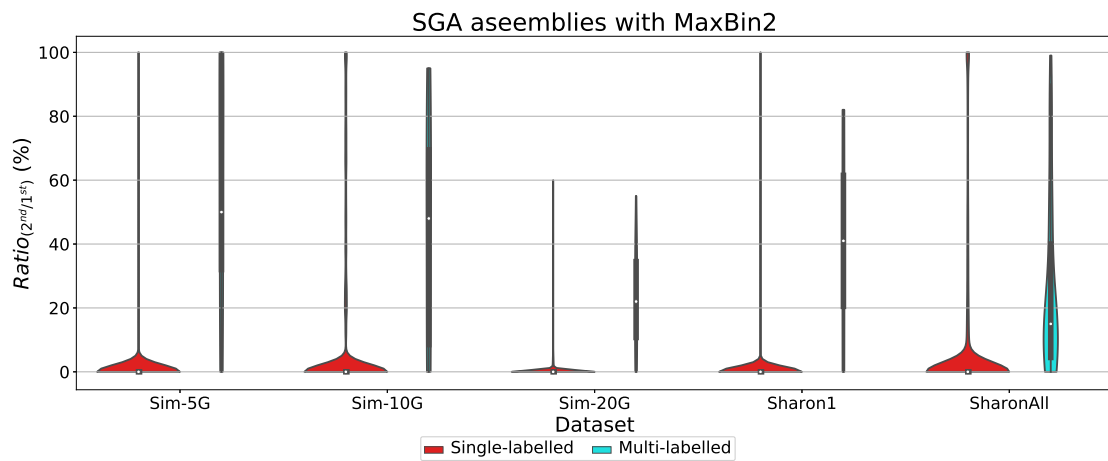


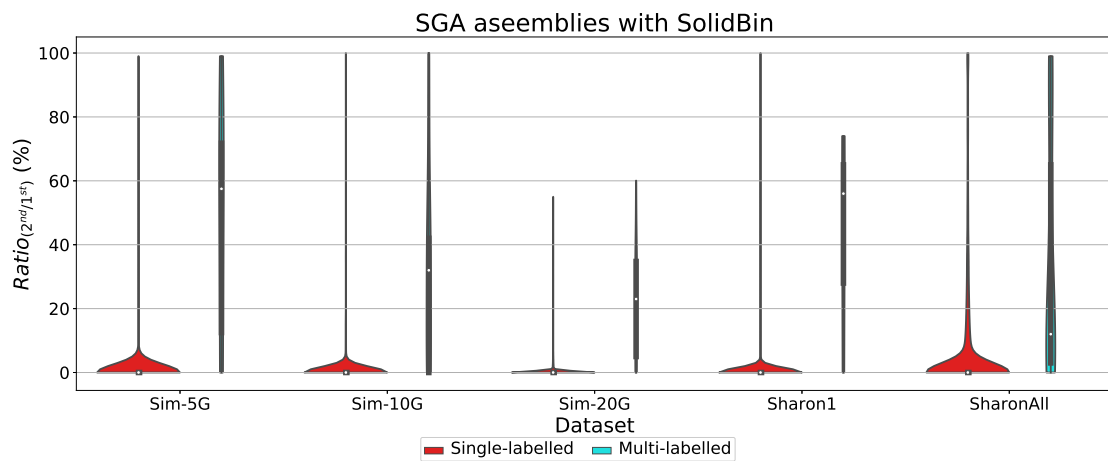
Figure 4.12: Violin plots for the ratio between the alignment lengths of the second longest alignment and the longest alignment of the single and multi-labelled inference results using GraphBin2 on top of CONCOCT, MaxBin2 and SolidBin results for the complex datasets.



(a)



(b)



(c)

Figure 4.13: Violin plots for the ratio between the alignment lengths of the second longest alignment and the longest alignment of the single and multi-labelled inference results using GraphBin2 on top of (a) CONCOCT, (b) MaxBin2 and (c) SolidBin results for the SGA assemblies.

4.4.3 Visualisation of the Assembly Graph

Figures 4.14 and 4.15 denote the labelling of the contigs in the metaSPAdes assembly graphs of the **Sim-5G** and **Sim-10G** datasets at different stages as it undergoes the processing of GraphBin2. White coloured vertices denote un-binned contigs and the rest of the coloured vertices denote the labelled contigs.

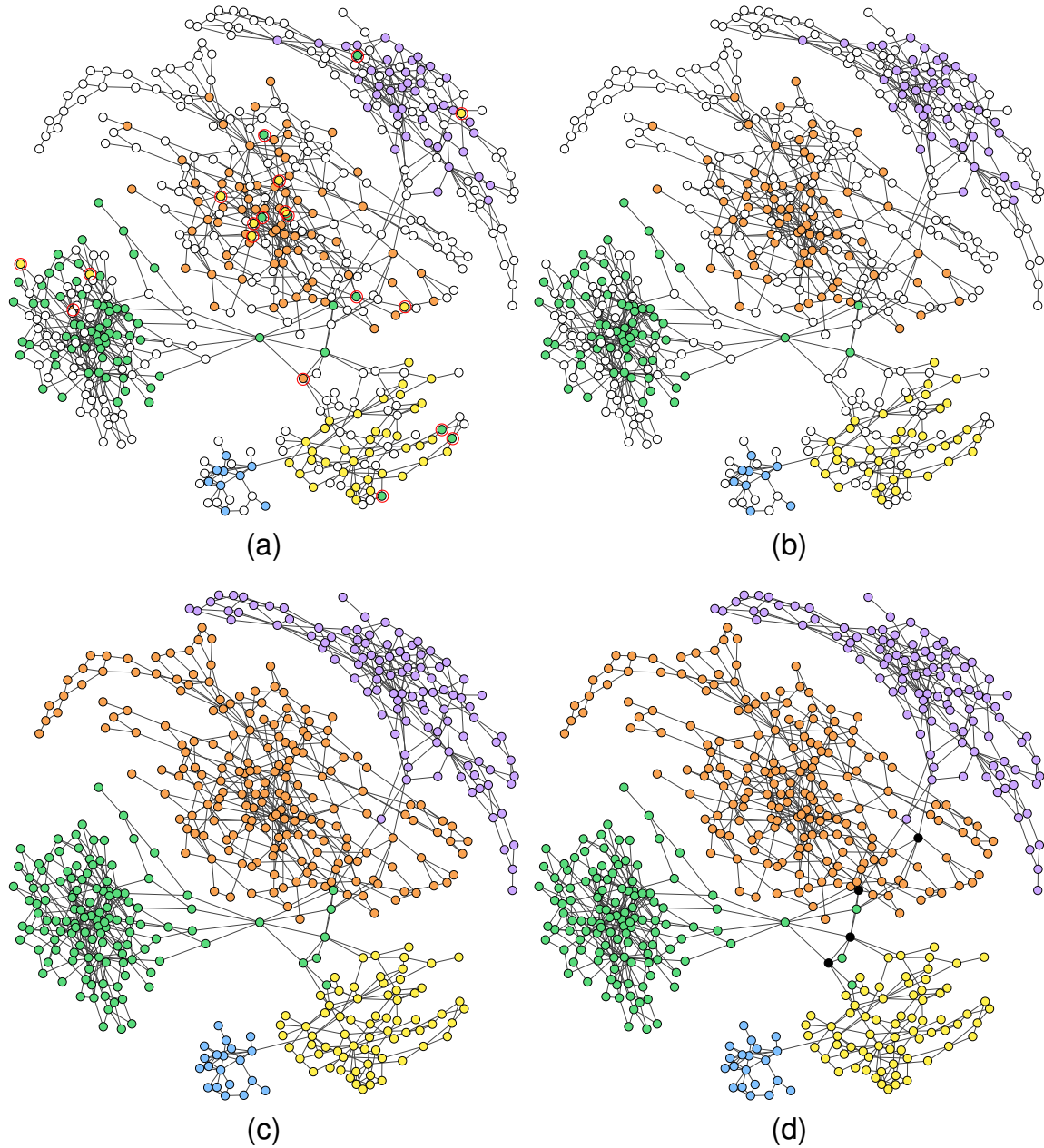


Figure 4.14: The labelling of the assembly graph of Sim-5G dataset based on (a) the initial MaxBin2 result (un-binned contigs are denoted by white coloured vertices and mis-binned contigs are circled in red), (b) after removing labels of unsupported vertices and correcting labels of inconsistent vertices, (c) after propagating labels of unlabelled vertices (d) after determining multi-labelled vertices (black coloured vertices) by GraphBin2.

In Figures 4.14 (a) and 4.15 (a), we can see that some mis-binned contigs are identified (circled in red) as differently coloured contigs within components of a single colour. Figures 4.14 (b) and 4.15 (b) show the refined assembly graph where GraphBin2 has removed labels of unsupported vertices and corrected labels of inconsistent vertices. After GraphBin2 propagates labels to the remaining unlabelled vertices, the assembly graph will look as denoted in Figures 4.14 (c) and 4.15 (c). Finally,

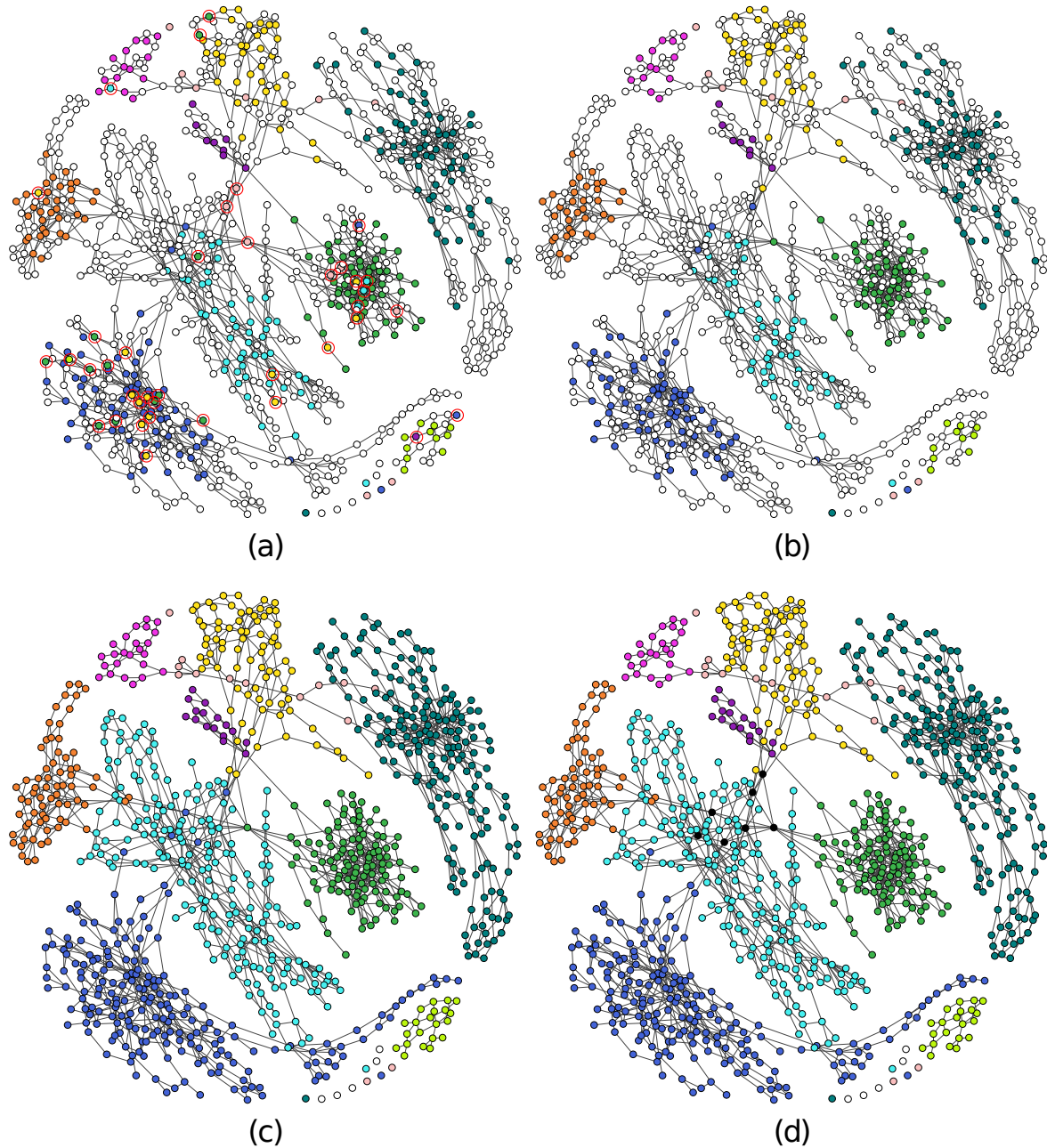


Figure 4.15: The labelling of the assembly graph of Sim-10G dataset based on (a) the initial MaxBin2 result (un-binned contigs are denoted by white coloured vertices and mis-binned contigs are circled in red), (b) after removing labels of unsupported vertices and correcting labels of inconsistent vertices, (c) after propagating labels of unlabelled vertices (d) after determining multi-labelled vertices (black coloured vertices) by GraphBin2.

GraphBin2 will detect multi-labelled vertices that correspond to contigs that may belong to multiple species as shown by the black coloured vertices in Figures 4.14 (d) and 4.15 (d).

4.4.4 Implementation, Running Time and Memory Usage

The source code for the experiments was implemented using Python 3.7.3 and run on a Darwin system with macOS Mojave 10.14.6, 16 GB memory and Intel Core i7 CPU @ 2.8 GHz with 4 CPU cores. In our experiments, we restrict the depth of the breadth-first-search in Steps 2-3 to be 5 to speed up GraphBin2. Moreover, we have set the parameter $\alpha = 1.5$ by default for GraphBin2. Furthermore, the process of inferring multi-labelled vertices was performed in parallel using multithreading (set to 8 threads by default in GraphBin2).

The running times (wall time) and the peak memory used by the initial binning tools (CONCOCT, MaxBin2 and SolidBin) and GraphBin2 can be found in Tables 4.3.

Table 4.3: Running times (wall time) and peak memory usage for binning using each tool for all the datasets. *s* denotes seconds, *m* denotes minutes and *MB* denotes megabytes.

Dataset	Assembly	Criteria	CONCOCT	GraphBin2 with CONCOCT	MaxBin2	GraphBin2 with MaxBin2	SolidBin	GraphBin2 with SolidBin
Sim-5G	metaSPAdes	Running time	29s	1s	12s	1s	3s	1s
		Memory usage (MB)	172	35	2,389	36	155	36
	SGA	Running time (MB)	20s	3m 58s	15s	3m 12s	3m 1s	3m 54s
		Memory usage	169	127	394	124	794	124
Sim-10G	metaSPAdes	Running time	25s	2s	20s	2s	3s	2s
		Memory usage (MB)	175	40	2,859	41	164	41
	SGA	Running time (MB)	14s	4m 33s	28s	5m	8m 25s	5m 2s
		Memory usage	204	101	285	101	1,423	101
Sim-20G	metaSPAdes	Running time	41s	3s	32s	3s	4s	5s
		Memory usage (MB)	193	44	2,854	44	193	45
	SGA	Running time (MB)	25s	28m 45s	49s	29m 40s	18m 47s	29m 54s
		Memory usage	211	194	364	192	2,064	193
Sharon1	metaSPAdes	Running time (MB)	12s	4s	9s	5s	6s	5s
		Memory usage	166	45	1,389	45	290	45
	SGA	Running time (MB)	20s	3s	12s	3s	15s	3s
		Memory usage	172	33	203	33	654	33
SharonAll	metaSPAdes	Running time (MB)	1m 8s	9m 54s	30s	10m 50s	2m 7s	11m 12s
		Memory usage	189	137	1,378	163	1,416	163
	SGA	Running time (MB)	1m 46s	1m 13s	28s	1m 21s	2m 51s	1m 15s
		Memory usage	201	50	241	50	2,612	50
50G-SR	metaSPAdes	Running time (MB)	1m 35s	19s	1m 33s	33s	13s	21s
		Memory usage	237	75	3,978	77	500	75
Lake Water	metaSPAdes	Running time (MB)	22m 2s	58m 42s	23m 27s	55m 17s	N/A*	N/A*
		Memory usage	807	855	1,004	862	N/A*	N/A*
100G-LR	metaSPAdes	Running time (MB)	3m 7s	9s	4m 8s	4s	14m 59s	4s
		Memory usage	399	54	3,976	57	4,840	57

* SolidBin could not be run on the Lake Water dataset due to insufficient memory.

Discussion

Although bin-refinement tools such as GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a) and GraphBin2 (Mallawaarachchi, Wickramarachchi et al., 2020b) achieve improved binning performance, they still require initial binning results obtained from other existing binning tools. The number of bins in the existing result cannot be dynamically adjusted during the binning process and erroneous binning results can be propagated during the label propagation process. Hence, it is worth exploring methods to develop a stand-alone contig-binning tool that makes use of the assembly graph information.

Chapter 5

Binning Metagenomic Contigs using Composition, Coverage and Assembly Graphs

This chapter is based on the work

V. Mallawaarachchi and Y. Lin (2022). ‘MetaCoAG: Binning Metagenomic Contigs via Composition, Coverage and Assembly Graphs’. In: *Research in Computational Molecular Biology (RECOMB 2022)*. Ed. by I. Pe’er. Vol. 13278. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 70–85. ISBN: 978-3-031-04749-7. DOI: [10.1007/978-3-031-04749-7_5](https://doi.org/10.1007/978-3-031-04749-7_5)

5.1 Motivation and Overview

In this chapter, we introduce MetaCoAG, a reference-free stand-alone approach for binning metagenomic contigs. In addition to composition and abundance information, MetaCoAG also makes use of the connectivity information from assembly graphs to bin contigs. More specifically, MetaCoAG estimates the number of initial bins using single-copy marker genes, assigns contigs into bins iteratively and adjusts the number of bins dynamically through graph-matching algorithms, and bins the remaining contigs using a label propagation method based on the assembly graph. To the best of our knowledge, MetaCoAG is the first stand-alone contig-binning tool to make direct use of the assembly graph information. We benchmark MetaCoAG against state-of-the-art contig-binning tools using simulated and real datasets. The experimental results show that MetaCoAG significantly outperforms other contig-binning tools, *e.g.*, improving the completeness of bins while maintaining high purity levels and producing more high-quality bins.

5.2 Methods

Figure 5.1 shows the overall workflow of MetaCoAG. Note that metagenomic assemblers assemble reads into contigs using assembly graphs which are considered as the input for MetaCoAG. MetaCoAG first identifies a list of contigs that contain single-copy marker genes. Next, MetaCoAG counts the number of contigs containing each single-copy marker gene and estimates the initial number of bins. Then, MetaCoAG applies a graph-matching algorithm to assign contigs that contain single-copy marker genes into bins iteratively and adjust the number of bins dynamically. Finally, MetaCoAG bins the remaining contigs using label propagation algorithms based on the assembly graph, performs a postprocessing step, and outputs the bins along with their corresponding contigs. Each step of MetaCoAG is explained in detail in the following sections.

5.2.1 Step 0: Assemble Reads into Contigs and Construct the Assembly Graph

This preprocessing step is carried out to assemble the reads into contigs and obtain the assembly graph. Metagenomic assemblers first use graph models to connect overlapping reads or *k*-mers and to infer contigs as non-branching paths. After graph simplification, the vertices represent contigs

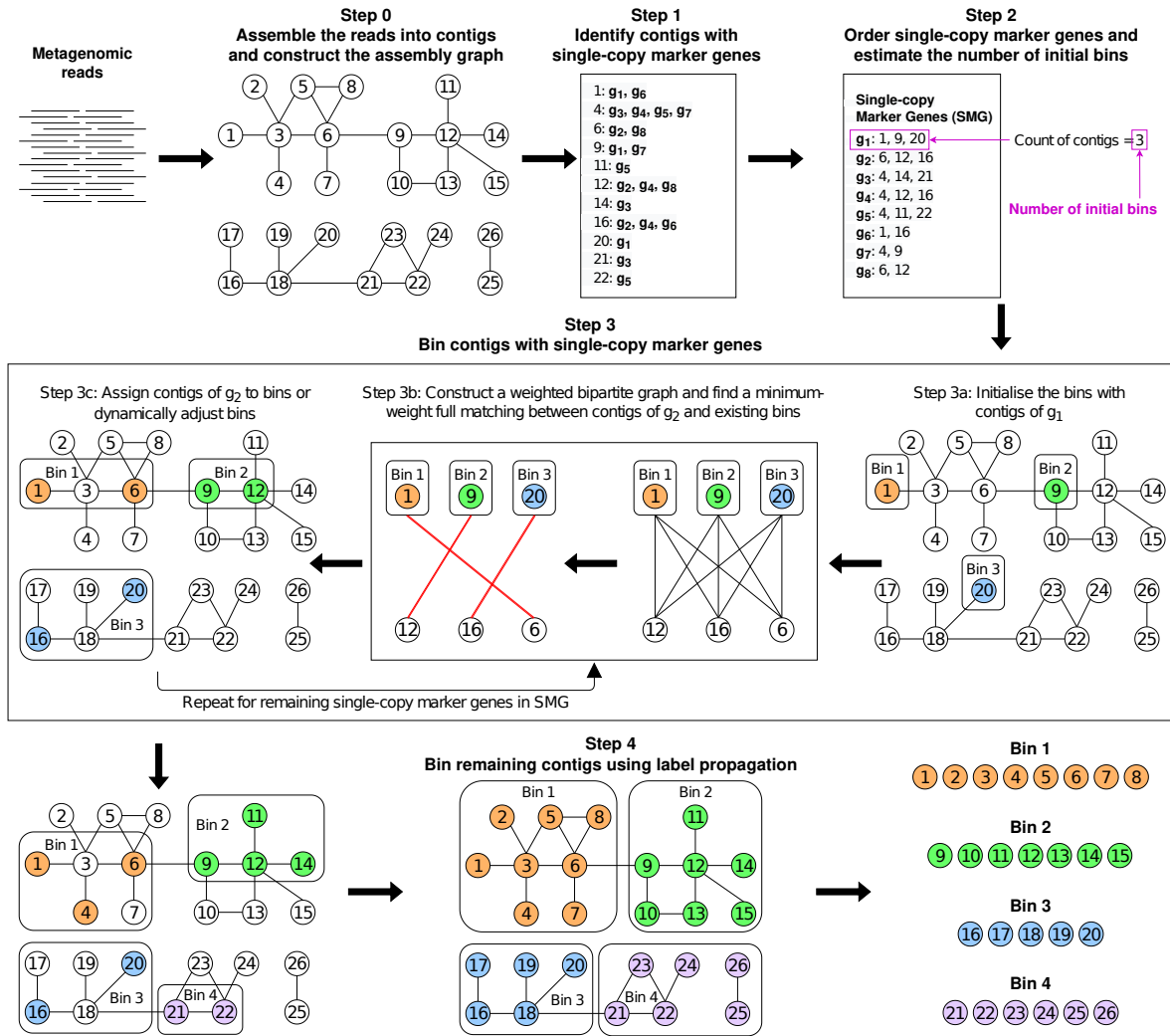


Figure 5.1: The workflow of MetaCoAG.

and edges represent connections between contigs in the assembly graph. Here we use the popular metagenomic assembler metaSPAdes (Nurk et al., 2017) to derive input contigs and assembly graphs. Note that the assembly graphs can also be obtained similarly using other metagenomic assemblers such as MEGAHIT (Li, Liu et al., 2015) and metaFlye (Kolmogorov, Bickhart et al., 2020).

5.2.2 Step 1: Identify Contigs with Single-Copy Marker Genes

Single-copy marker genes appear only once in a bacterial genome and are conserved in the majority of bacterial genomes (Albertsen et al., 2013; Dupont et al., 2012; Wu, Tang et al., 2014) (please refer to Figure 5.2 for a visualisation of single-copy marker genes). These single-copy marker genes have been used in previous work to determine the completeness and contamination of binning results (Parks et al., 2015). Since these genes appear only once in a bacterial genome, we can use these to estimate the number of species present in a sample. For example, if we consider the blue marker gene in Figure 5.2, it appears only once in each of the genomes and there are three such genes in this mixed sample. So, our mixed sample is most likely to contain three species.

For each single-copy marker gene, we use FragGeneScan (Rho et al., 2010) and HMMER (Eddy, 2011) to identify the contigs which contain this marker gene (refer to Figure 5.3 (a)). A single-copy marker gene is considered to be contained in a contig if more than 50% of the gene length is aligned to this contig. Similar to approaches such as MaxBin (Wu, Tang et al., 2014) and MaxBin2 (Wu, Simmons et al., 2015), MetaCoAG uses single-copy marker genes to distinguish contigs belonging to different species (*i.e.*, if these contigs contain the same single-copy marker gene).

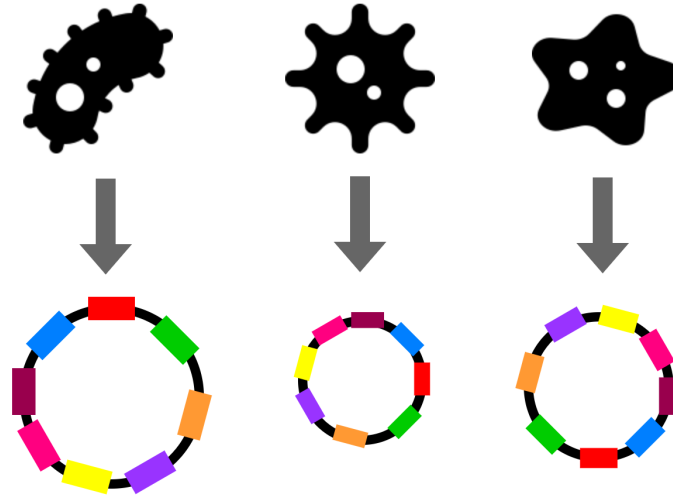


Figure 5.2: Single-copy marker genes in bacteria. Coloured rectangles show the different single-copy marker genes found on three different bacterial genomes and they appear only once in each genome.

5.2.3 Step 2: Order Single-copy Marker Genes and Estimate the Number of Initial Bins

For a given single-copy marker gene, the contigs containing this marker gene should come from different species (*e.g.*, if two contigs contain the same marker gene, then the two contigs should belong to two different species). In the ideal case, if we have a near-perfect assembly, the number of contigs that contain the same single-copy marker gene should be equal to the number of species present in the sample. However, in reality, assemblies can be fragmented and erroneous, which may make it challenging to recover all single-copy marker genes and hence, lowering the counts of contigs containing each single-copy marker gene.

To get a better estimation of the number of species, we obtain the counts of contigs containing each single-copy marker gene. We also record the single-copy marker genes found in each contig. For a single-copy marker gene, the number of contigs that it can distinguish is the number of contigs containing this gene. Therefore, we order all the single-copy marker genes according to the descending order of the number of contigs containing them. We refer to this list of ordered marker genes as *SMG* where a single-copy marker gene g_i has a set of contigs $C(g_i)$ containing g_i .

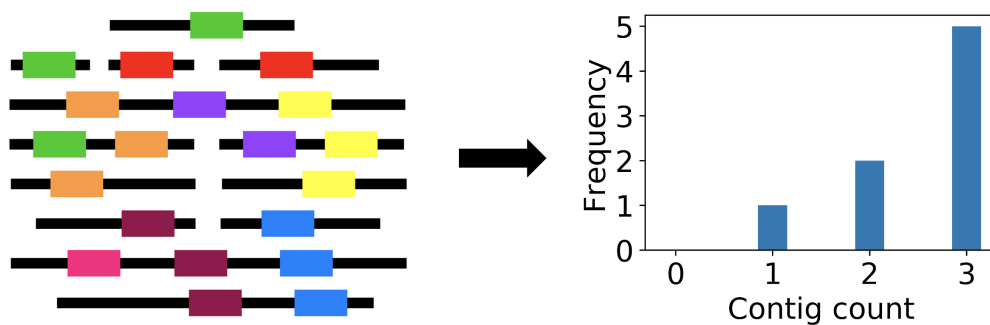


Figure 5.3: MetaCoAG identifies contigs containing each of the single-copy marker genes and counts the contigs containing each marker gene. For example, the green marker gene is contained in three contigs, the red marker gene is contained in two contigs and the pink marker gene is contained in one contig. Then we plot a histogram of the contig counts. There are five marker genes where each is found in three contigs, two marker genes where each is found in two contigs and one marker gene found in one contig. Since these genes appear only once in a bacterial genome, we can estimate that this sample contains three species, as most of the marker genes appear in three contigs. The number of initial bins is set to be the largest count of contigs a marker gene is present which is three in this example.

The number of initial bins is empirically set to be the number of contigs that contain the first gene in SMG, in order to recover the maximum number of species possible from the marker gene information (refer to Figure 5.3 for an example of estimating the number of bins in a sample).

5.2.4 Step 3: Bin Contigs with Single-copy Marker Genes

Step 3a: Initialise Bins

We initialise the bins using the contigs of the first single-copy marker gene g_1 in SMG; *i.e.*, we initialise a new bin B for each contig in $C(g_1)$ (as shown in Step 3a of Figure 5.1). We define the initialised set of bins as $BINS$. Please note that the number of bins $|BINS|$ may change during the binning process.

Calculating Composition and Coverage similarities

Previous studies on metagenomics binning have used genomic signatures as they follow species-specific patterns (Deschavanne et al., 1999; Wu, Tang et al., 2014). The most commonly used genomic signatures to characterise composition information are *tetranucleotide frequencies* (136 canonical 4-mers, also known as *tetramers*) (Alneberg et al., 2014; Kang, Li et al., 2019; Nissen et al., 2021; Wang, Wang et al., 2019; Wu, Simmons et al., 2015; Wu, Tang et al., 2014). For each contig c , we normalise the tetranucleotide frequencies using its total number of tetranucleotides to obtain the normalised tetranucleotide frequency vector, $tetra(c)$. We obtain the tetranucleotide composition distance between contigs c and c' as $d_{tetra}(c, c') = dist_E(tetra(c), tetra(c'))$ where $dist_E$ is the Euclidean distance function.

$$d_{tetra}(c, c') = dist_E(tetra(c), tetra(c')) \quad (5.1)$$

We use the same formula proposed by Wu et al. (Wu, Tang et al., 2014) to estimate how similar c and c' are (*i.e.*, belonging to the same species) based on their composition, $S_{comp}(c, c')$ as shown in equation 5.2.

$$S_{comp}(c, c') = \frac{N_{intra}(d_{tetra}(c, c') | \mu_{intra}, \sigma_{intra}^2)}{N_{intra}(d_{tetra}(c, c') | \mu_{intra}, \sigma_{intra}^2) + N_{inter}(d_{tetra}(c, c') | \mu_{inter}, \sigma_{inter}^2)} \quad (5.2)$$

N_{intra} and N_{inter} are Gaussian distributions with μ_{intra} , σ_{intra} , μ_{inter} and σ_{inter} set according to the latest values of MaxBin 2.2.7 (Wu, Simmons et al., 2015) which have been calculated by analysing the Euclidean distance between the tetranucleotide frequencies of pairs of sequences sampled from the same genome (*intra*) and different genomes (*inter*). If the distance is lower between two sequences, they are more similar, and are more likely to belong to the same genome.

We use the coverage information of the contigs as coverage carries important information about the abundance of species and has been used in previous metagenomics binning studies (Albertsen et al., 2013; Kang, Li et al., 2019; Nissen et al., 2021; Wang, Wang et al., 2019; Wu, Tang et al., 2014). Shotgun sequencing has shown to follow the Lander-Waterman model (Lander and Waterman, 1988) and the Poisson distribution has been used to obtain the sequencing coverage of nucleotides and applied in metagenomics binning (Wu, Tang et al., 2014; Wu and Ye, 2011). Modifying the definition found in Wu et al. (Wu, Tang et al., 2014), we estimate how similar c and c' are in terms of their coverage values in each sample, $S_{cov}(c, c')$ as shown in equation 5.3.

$$S_{cov}(c, c') = \min \left(\prod_{n=1}^M Poisson(cov_n(c) | cov_n(c')), \prod_{n=1}^M Poisson(cov_n(c') | cov_n(c)) \right) \quad (5.3)$$

Here $cov_n(c)$ and $cov_n(c')$ refer to the coverage values of the contigs c and c' respectively in the sample n where M is the number of samples. *Poisson* is the Poisson probability mass function.

Step 3b: Construct a Weighted Bipartite Graph and Find a Minimum-Weight Full Matching

In the previous steps, we have used single-copy marker genes to identify pairs of contigs that belong to different species. Remind that contigs in different bins in $BINS$ are expected to belong to different species and contigs in $C(g_i)$ are also expected to belong to different species. However, there is no measurement to measure how likely a contig c in $C(g_i)$ belongs to an existing bin B in $BINS$. Therefore, we introduce a bipartite graph between $C(g_i)$ and $BINS$ and propose a weight $w_{c2B}(c, B)$ between a contig c in $C(g_i)$ and an existing bin B in $BINS$ as shown in equation 5.4 (averaging over all the contigs in bin B).

$$w_{c2B}(c, B) = \frac{\sum_{c' \in B} w_{c2c}(c, c')}{|B|} \quad (5.4)$$

In equation 5.4, $w_{c2c}(c, c')$ is the weight that measures how likely a pair of contigs c and c' belong to the same species and is computed using equation 5.5.

$$w_{c2c}(c, c') = -(\log(S_{comp}(c, c')) + \log(S_{cov}(c, c'))) \quad (5.5)$$

In equation 5.5, $S_{comp}(c, c')$ and $S_{cov}(c, c')$ are calculated according to equations 5.2 and 5.3 respectively.

Now we find a minimum-weight full matching (minimum-cost assignment) (Karp, 1980) for the above bipartite graph between $C(g_i)$ and $BINS$ where every contig c in $C(g_i)$ will get paired with exactly one bin B in $BINS$. For this purpose, we use the minimum-weight full matching algorithm implemented in the *NetworkX* python library which is based on the algorithm proposed by Karp (Karp, 1980) and the time complexity is $O(|C(g_i)| \times |BINS| \times \log(|BINS|))$.

In the next step, we will see how we can assign the contigs to existing bins based on the minimum-weight full matching we have obtained.

Step 3c: Assign Contigs to Existing Bins or Dynamically Adjust Bins

Previous studies have observed that contigs connected to each other in the assembly graph are more likely to belong to the same taxonomic group (Barnum et al., 2018; Mallawaarachchi, Wickramarachchi et al., 2020a). While $w_{c2B}(c, B)$ considers both composition and coverage information, the assembly graph has not yet been incorporated into the binning process. Therefore, we introduce $d_{graph}(c, B)$ to measure how well contig c is connected to contigs in bin B within the assembly graph. Specifically, $d_{graph}(c, B)$ is defined as the average length of the shortest-path distances between contig c and all the contigs in bin B in the assembly graph. Note that both $w_{c2B}(c, B)$ and $d_{graph}(c, B)$ will be used to assign contigs to existing bins or dynamically adjust the bins.

We define the thresholds w_{intra} and w_{inter} as follows where M is the number of samples in the dataset.

$$w_{intra} = -(\log(p_{intra})) \times M \quad (5.6)$$

$$w_{inter} = -(\log(p_{inter})) \times M \quad (5.7)$$

Each candidate pair (c, B) obtained from the minimum-weight full matching falls under one of the following three cases as shown in Figure 5.4.

- **Case 1:** If the weight of the candidate pair $w_{c2B}(c, B)$ is less than or equal to w_{intra} and the average distance $d_{graph}(c, B)$ is less than or equal to d_{limit} , then contig c will be assigned to bin B , i.e., $B \leftarrow B \cup \{c\}$ (e.g., contig 4 and Bin 1 in Figure 5.4).
- **Case 2:** If the weight of the candidate pair $w_{c2B}(c, B)$ is greater than w_{inter} and the average distance $d_{graph}(c, B)$ is greater than d_{limit} , then a new bin B' is created and contig c is assigned to that new bin, i.e., $B' = \{c\}$ and $BINS \leftarrow BINS \cup \{B'\}$. (e.g., contig 21 in Figure 5.4).
- **Case 3:** If $w_{c2B}(c, B)$ and $d_{graph}(c, B)$ satisfy neither Case 1 nor Case 2, then contig c will not be assigned to any bin (e.g., contig 14 in Figure 5.4).

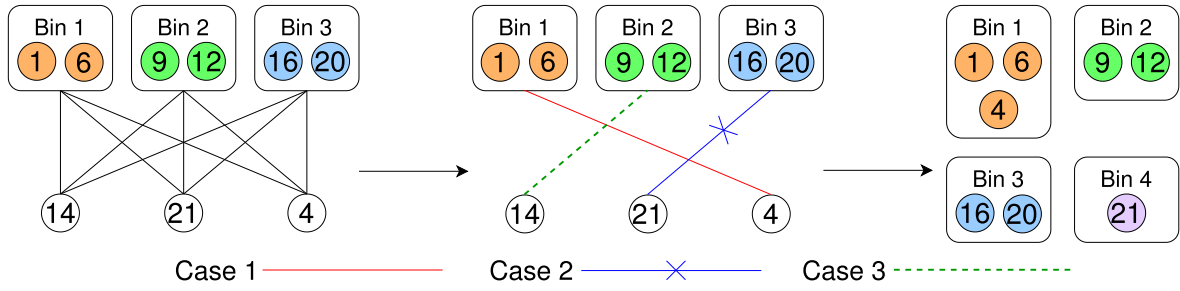


Figure 5.4: Cases 1, 2 and 3 in assigning contigs to existing bins or adjusting bins

The default values for parameters p_{intra} , p_{inter} , d_{limit} were chosen empirically and set to 0.1, 0.01 and 20 respectively. Now we iteratively perform Steps 3b and 3c to process all the contigs containing single-copy marker genes. The remaining challenge is to bin the contigs which do not contain single-copy marker genes which will be addressed in Step 4.

5.2.5 Step 4: Bin Remaining Contigs Using Label Propagation

After we bin the contigs with single-copy marker genes, each such contig receives a label corresponding to its bin. Now we will propagate labels from these contigs to other unlabeled contigs within the same connected component.

Step 4a: Propagate Labels Within Connected Components

MetaCoAG uses composition, coverage and distance information from the assembly graph to propagate labels from labeled contigs to the unlabeled contigs located within the same connected components. More specifically, for each unlabeled long contig c (at least 1,000 bp long because short contigs result in unreliable composition and coverage information) directly connected or connected via short contigs to a labeled contig c' , MetaCoAG computes a candidate propagation action $(c', c, d(c, c'), w_{c2B}(c, B'))$ where $d(c, c')$ is the shortest distance between c and c' using only unlabeled vertices and $w_{c2B}(c, B')$ is computed according to equation 5.4 where B' is the bin to which contig c' is assigned. Given two candidate propagation actions (a, b, d, w) and (a', b', d', w') , (a, b, d, w) has a higher priority than (a', b', d', w') if $d < d'$ or $(w < w' \text{ and } d = d')$. MetaCoAG iteratively selects the candidate propagation action with the highest priority and executes the corresponding label propagation. If a contig to be labeled contains single-copy marker genes, the relevant candidate propagation action is executed if the single-copy marker genes of the contig are not present in the intended bin. We restrict the depth of the search for labeled contigs in this step to 10 in order to speed up MetaCoAG.

Step 4b: Propagate Labels Across Different Components

Note that some components in the assembly graph may not have any labeled contigs and we need to propagate labels from labeled bins to unlabeled contigs across components. Calculating pair-wise weights $w_{c2c}(c, c')$ for all the remaining contigs becomes time consuming. Hence, for each bin B we create a representative contig $c(B)$ which has a composition profile and a coverage profile calculated by averaging the normalised tetranucleotide frequency vectors and coverage vectors of all the contigs in bin B , respectively. These profiles will provide a better representation of the composition and coverage of the bins. Then, for each unlabeled contig c , MetaCoAG identifies a bin B that minimises $w_{c2c}(c, c(B))$ which is calculated according to equation 5.5, and assigns contig c into that bin B . This propagation is limited to long contigs (at least 1,000 bp long by default). If an unlabeled contig contains single-copy marker genes, it is assigned to bin B that minimises $w_{c2c}(c, c(B))$ if the single-copy marker genes of the contig are not present in bin B . Then, Step 4a is performed again to further propagate labels.

Step 4e: Postprocessing

In this step, we will make final adjustment on the current bins. Two bins B and B' are *mergeable* if they have no common marker genes and $w_{c2c}(c(B), c(B'))$ (calculated by equation 5.5) is upper bounded

by w_{intra} (defined in Step 3c). Then, MetaCoAG creates a graph where vertices denote current bins and edges between two vertices denote that the corresponding two bins are mergeable. Now we use the implementation of python-igraph library to find maximal cliques (https://igraph.org/c/doc/igraph-Cliques.html#igraph_maximal_cliques) in this graph and merge the bins found in each maximal clique. After merging bins, we also remove the bins which contain less than one third (set by default) of the single-copy marker genes. Finally, MetaCoAG outputs the bins along with their corresponding contigs.

5.3 Experimental Setup

5.3.1 Datasets

Simulated Datasets

We evaluated the binning performance on the simulated **simHC+** dataset (Wu, Tang et al., 2014) which consists of 100 bacterial species. Paired-end MiSeq reads were simulated using InSilicoSeq (Gourlé et al., 2018) with 300 bp mean read length and the predefined MiSeq error model.

CAMI2 Toy Human Microbiome Project Datasets

We used the simulated metagenome data from the toy Human Microbiome project of the second CAMI challenge (Meyer, Fritz et al., 2022). Metagenomes were simulated from five different body sites of the human host as follows.

1. Urogenital tract - referred as **CAMI UG**
2. Skin - referred as **CAMI Skin**
3. Oral cavity - referred as **CAMI Oral**
4. Gastrointestinal tract - referred as **CAMI GI**
5. Airways - referred as **CAMI Airways**

Real Datasets

We used the following real datasets to evaluate the binning performance on real-world metagenomic data.

1. Pre-born infant gut metagenome, (Sharon et al., 2013) - referred as **Sharon**
2. Metagenomics of the Chronic Obstructive Pulmonary Disease (COPD) Lung Microbiome (Cameron et al., 2016) - referred as **COPD**
3. Human metagenome sample from tongue dorsum of a participant from the Deep WGSHP clinical samples (Lloyd-Price et al., 2017) - referred as **Deep HMP TD**

Please refer to Appendix B for further details of all the datasets.

5.3.2 Tools Used

We used the popular metagenomic assembler metaSPAdes (Nurk et al., 2017) (from SPAdes version 3.15.2 (Bankevich et al., 2012)) to assemble reads into contigs and obtain the assembly graphs. The mean coverage of each contig in each sample was calculate using CoverM (available at <https://github.com/wwood/CoverM>).

MetaCoAG was benchmarked against the binning tools MaxBin2 (version 2.2.7) (Wu, Simmons et al., 2015) in its default settings, MetaBAT2 (version 2.12.1) (Kang, Li et al., 2019) with `-m 1500` and Vamb (version 3.0.1) (Nissen et al., 2021) in co-assembly mode (for a fair comparison with other tools) with the parameter `--minfasta 200000` as suggested by the authors. The commands used to run these tools can be found in Appendix D.

The binning results were evaluated using the tools AMBER (Meyer, Hofmann et al., 2018) (version 2.0.2), CheckM (Parks et al., 2015) (version 1.1.3) and GTDB-Tk (Chaumeil et al., 2019) (version 1.5.0).

5.3.3 Evaluation Criteria

Since the ground truth species for the simHC+ dataset were available, we used Minimap2 (Li, 2018) to map the contigs to the reference genomes and determine the ground truth. With this ground truth annotation of contigs, we used AMBER (Meyer, Hofmann et al., 2018) to assess the binning results of the simHC+ dataset. We set the recall as AMBER completeness and precision as AMBER purity and calculated the F1-score as $2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$ for each bin/species.

For all the datasets, we determined the completeness and contamination of the bins produced by each tool using CheckM (Parks et al., 2015). We set the completeness as CheckM completeness and purity as $1 / (1 + \text{CheckM contamination})$. To check the trade-off between completeness and purity, we set the recall as completeness and precision as purity, and calculated the F1-score as $2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$ for each bin. Furthermore, we counted the number of high-quality bins (bins which have >80% recall and >90% precision), medium-quality bins (bins which have >50% recall and >80% precision) and low-quality bins (bins which are not considered as high-quality or medium-quality).

To determine the species identified by the binning tools, we annotated all the high-quality bins of the real metagenomic datasets produced from the three best-performing tools; MetaCoAG, MaxBin2 and Vamb using GTDB-Tk (Chaumeil et al., 2019) up to the species level. The species were determined by the classification string produced by GTDB-Tk.

5.4 Results and Discussion

5.4.1 Benchmarks using simHC+ Dataset

We first benchmarked MetaCoAG against two popular contig-binning tools, MaxBin2 (Wu, Simmons et al., 2015) and MetaBAT2 (Kang, Li et al., 2019) on the simulated dataset **simHC+** (Wu, Tang et al., 2014) which consists of 100 bacterial genomes (please refer to Appendix C for further details of the simHC+ dataset)¹. We evaluated the binning results of the simHC+ dataset produced by all the tools using the two popular evaluation tools AMBER (Meyer, Hofmann et al., 2018) and CheckM (Parks et al., 2015). AMBER assesses the quality of bins based on the ground truth annotations provided and CheckM assesses the quality of bins based on sets of single-copy marker genes. We analysed the purity, completeness and F1-score of the binning results calculated by AMBER (at the nucleotide level) and CheckM. MetaCoAG has recovered bins with a better trade-off between purity and completeness when compared to other binning tools (Figure 5.5 (a)) with an average purity of 91.07% and an average completeness of 82.73% from AMBER and an average purity of 97.55% and an average completeness of 87.17% from CheckM. This better trade-off is demonstrated from the best F1-score results produced by MetaCoAG with a median F1-score of 95.69% from AMBER (Figure 5.5 (b)) and a median F1-score of 98.48% from CheckM (Figure 5.5 (c)) when compared with other binning tools. Even though MetaBAT2 has recorded the highest average purity (98.30% from AMBER and 100.0% from CheckM), it has a very low average completeness (13.02% from AMBER and 29.59% from CheckM) because all contigs shorter than 1,500bp (*i.e.*, 60.49% of the contigs in the entire dataset) were discarded. Please refer to Table 5.1 for the exact values of the AMBER and CheckM results of the simHC+ dataset. We also used CheckM to count the number of high-, medium- and low-quality bins produced by all the binning tools for the simHC+ dataset (Please refer to Table 5.3). MetaCoAG has recovered the highest number of high-quality bins (69 bins) and the lowest number of low-quality bins (13 bins) for the simHC+ dataset.

We further used AMBER to analyse the species recovered by each binning tool for the simHC+ dataset. Out of the 100 species, MetaCoAG was able to recover more species than other tools (Appendix C), thanks to its adaptable bin-breaking mechanism that allows to separate more species rather than combining them together. We also analysed the F1-score of these recovered species, and observed that MetaCoAG has recovered more species with high F1-score than the other binning tools (Please refer to Appendix C for comparison of the F1-score of the species recovered by MaxBin2, MetaBAT2 and MetaCoAG). Many existing binning tools assume that the oligonucleotide composition and coverage

¹Please note that the recently published tool Vamb (Nissen et al., 2021) was not used to evaluate the simHC+ dataset as the number of contigs was less than the number recommended by the authors (<https://github.com/RasmussenLab/vamb#recommended-workflow>).

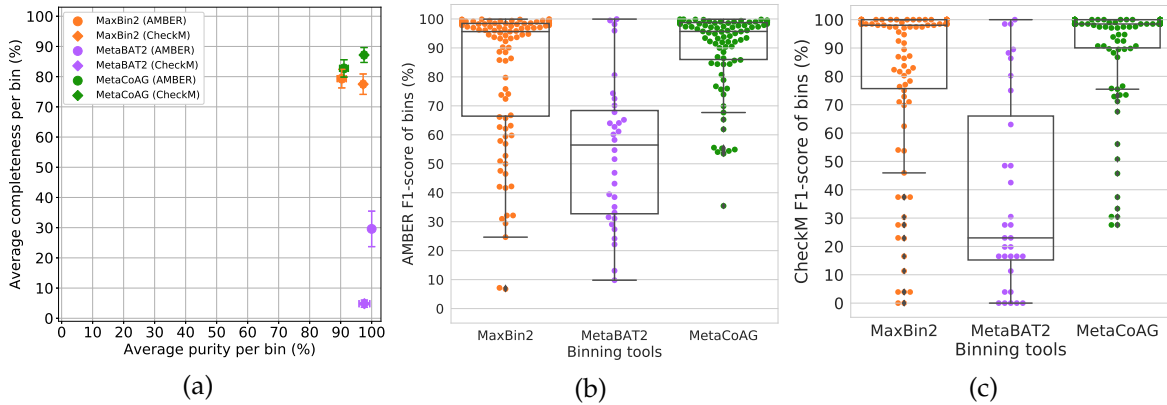


Figure 5.5: AMBER and CheckM results of the bins of the simHC+ dataset. (a) Quality of bins in terms of average completeness per bin vs. average purity per bin obtained from AMBER and CheckM. (b) F1-score of the bins obtained from AMBER. (c) F1-score of the bins obtained from CheckM.

Table 5.1: AMBER and CheckM evaluation results for the simHC+ dataset

Evaluation criteria	MaxBin score (%)	MetaBAT2 score (%)	MetaCoAG score (%)
Average purity per bin (AMBER)	90.36	98.30	91.07
Average purity per bin (CheckM)	97.25	100.0	97.55
Average completeness per bin (AMBER)	79.34	13.02	82.73
Average completeness per bin (CheckM)	77.51	29.59	87.17
F1-score per bin (AMBER)	84.50	23.00	86.70
F1-score per bin (CheckM)	80.64	37.25	89.44
Accuracy (AMBER)	77.07	14.38	84.46
Binned fraction (AMBER)	84.90	14.79	92.04

are conserved across the genome. Hence it is challenging for such tools to bin species with high variance in oligonucleotide composition and/or coverage. Moreover, these tools face difficulties when recovering species with low abundance due to the rare occurrence of species-specific signals. In Figure 5.6, we visualise and compare the binning results of MaxBin2 and MetaCoAG² against the ground truth for the following species, *Pseudomonas putida* and *Arthrobacter arilaitensis*. The species *Pseudomonas putida* has a high variance in oligonucleotide composition (standard deviation > 0.015 for the tetranucleotide composition of its contigs) and thus MaxBin2 has split this species into multiple bins incorrectly (refer to Figure 5.6 (a)). The species *Arthrobacter arilaitensis* has a high variance in genome coverage (standard deviation $> 50\times$ for the coverages of its contigs) and thus MaxBin2 has mis-binned some high-coverage contigs into other species with high coverage (refer to Figure 5.6 (b)). However, MetaCoAG has been able to recover these species with high F1-score values, *e.g.*, improving the F1 score for *Pseudomonas putida* from 59.78% to 99.56% and improving the F1-score for *Arthrobacter arilaitensis* from 97.65% to 98.99%. Despite the high variance in oligonucleotide composition and coverage, MetaCoAG has been able to recover these species accurately, thanks to the additional connectivity information from the assembly graph.

Another challenge faced by the majority of the existing binning tools is the inability to accurately separate contigs of species belonging to the same genus, where such species tend to have similar oligonucleotide composition and appear in similar abundances. For example, the three species, *S. pneumoniae*, *S. thermophilus* and *S. suis* from simHC+ belong to the *Streptococcus* genus, and they

²MetaBAT2 was not included in this comparison as it had not recovered the species *Pseudomonas putida* and *Arthrobacter arilaitensis*.

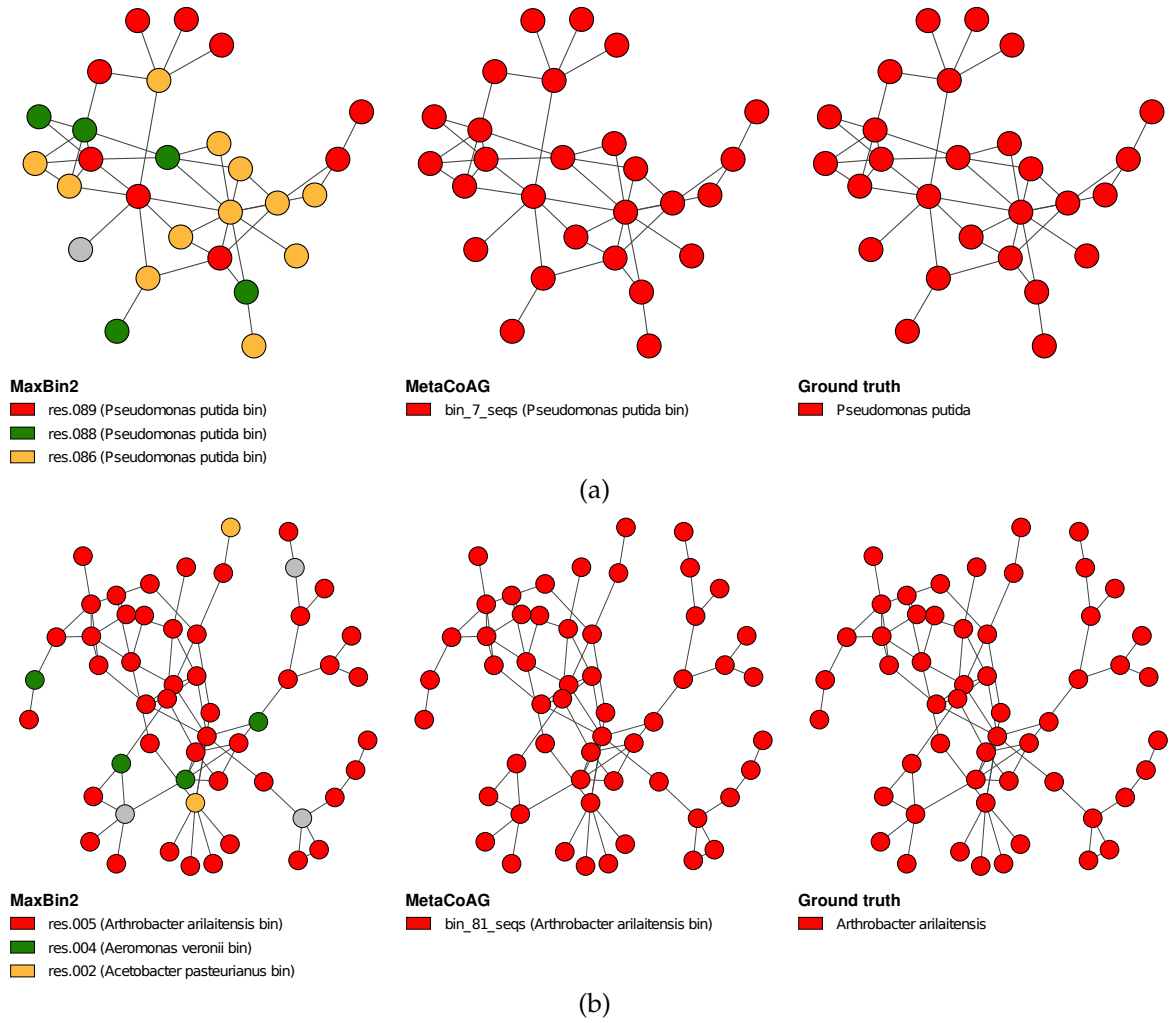


Figure 5.6: Visualization of the binning results from MaxBin2 and MetaCoAG for a species with (a) high variance in oligonucleotide composition (standard deviation > 0.015) and (b) high variance in coverage (standard deviation $> 50\times$). Gray colour nodes denote contigs which were binned to bins other than the ones specified in the figure.

have very similar oligonucleotide composition and coverage values (Refer to Figure 5.7 (a)) and similar coverages (*Streptococcus pneumoniae*: $56\times$, *Streptococcus thermophilus*: $60\times$ and *Streptococcus suis*: $50\times$, please refer to Appendix C for further information). Not surprisingly, contigs from these three species were incorrectly binned by MaxBin2 and even ignored by MetaBAT2 because they share similar composition and coverage profiles (Refer to Figure 5.7 (b)). On the contrary, MetaCoAG was able to accurately bin most of the contigs from these three species because they naturally form three subgraphs in the assembly graph (Refer to Figure 5.7 (b)), thus improving the F1-scores of *Streptococcus pneumoniae* from 46.51% to 93.40%, *Streptococcus thermophilus* from 49.97% to 95.67% and *Streptococcus suis* from 72.39% to 95.95%. Figure 5.7 (b) demonstrates that the use of assembly graph in MetaCoAG can assist in the separation of species, despite the high similarity in oligonucleotide composition and coverage of certain species.

5.4.2 Benchmarks using CAMI2 Toy Human Microbiome Project Datasets

We benchmarked MetaCoAG against MaxBin2 (Wu, Simmons et al., 2015), MetaBAT2 (Kang, Li et al., 2019), and Vamb (Nissen et al., 2021) on five publicly available datasets from the toy Human Microbiome Project dataset of the second Critical Assessment of Metagenomic Interpretation (CAMI) (Meyer, Fritz et al., 2022) challenge (Please refer to Appendix B for further details of the CAMI datasets). Multiple samples from each dataset were co-assembled together to obtain the final contigs for binning.

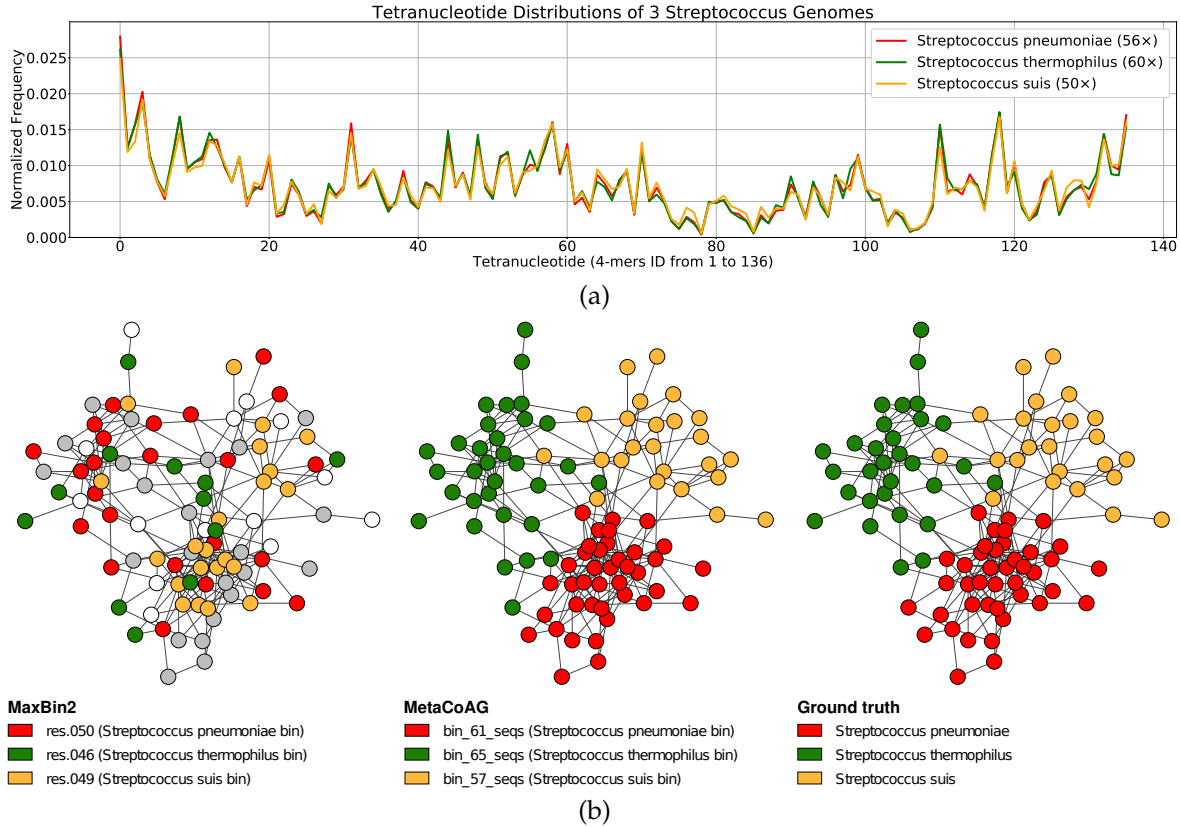


Figure 5.7: Visualization of the tetranucleotide composition and binning results of three *Streptococcus* genomes in the simHC+ dataset. (a) Tetranucleotide distributions of the three *Streptococcus* genomes; *Streptococcus pneumoniae* (red) with 56× coverage, *Streptococcus suis* (yellow) with 60× coverage and *Streptococcus thermophilus* (green) with 50× coverage. (b) Visualization of the binning results from MaxBin2 and MetaCoAG for three *Streptococcus* genomes. White colour nodes denote discarded contigs and gray colour nodes denote contigs which were binned to bins other than the three *Streptococcus* genomes. MetaBAT2 was not included as it had not recovered these three species.

We evaluated the binning results of the CAMI datasets using CheckM (Parks et al., 2015) and reported the F1-score of the bins produced by all the binning tools. Figure 5.8 (a)-(e) shows that overall MetaCoAG has achieved the best binning results among all the binning tools. The overall median F1-scores averaging from all 5 CAMI datasets for MetaCoAG, MaxBin2, MetaBAT2 and Vamb are 86.77%, 75.41%, 1.57% and 33.30%, respectively. More specifically, MetaCoAG has recovered more complete bins with higher purity when compared to other tools. MetaCoAG produced the highest numbers of high-quality and medium-quality bins combined together for all the CAMI datasets (Please refer to Table 5.3). Note that only MaxBin2 outperforms MetaCoAG in terms of the number of high-quality bins just for the GI dataset. This dataset had a low density in its assembly graph (Please refer to Appendix B for density of the assembly graph) which prevented MetaCoAG from making full use of the assembly graphs.

5.4.3 Benchmarks using real metagenomic datasets

We benchmarked MetaCoAG against MaxBin2 Wu, Simmons et al. (2015), MetaBAT2 Kang, Li et al. (2019) and Vamb Nissen et al. (2021) on three real metagenomic datasets; Sharon Sharon et al. (2013), COPD Cameron et al. (2016) and Deep HMP TD Lloyd-Price et al. (2017). Similar to the simHC+ dataset, we again use CheckM Parks et al. (2015) to evaluate the bins produced by all the binning tools and identify high-quality bins. Fig. 5.9 shows that MetaCoAG has also achieved the best binning result in terms of the median F1-score for the real datasets. For the Sharon dataset, MetaCoAG records a median F1-score of 99.24% while the second-best tool (Vamb) has a median F1-score of 83.88%. For the COPD dataset, MetaCoAG records a median F1-score of 75.68% while the second-best tool (MaxBin2) has a median F1-score of 25.13%. For the Deep HMP TD dataset, MetaCoAG records a

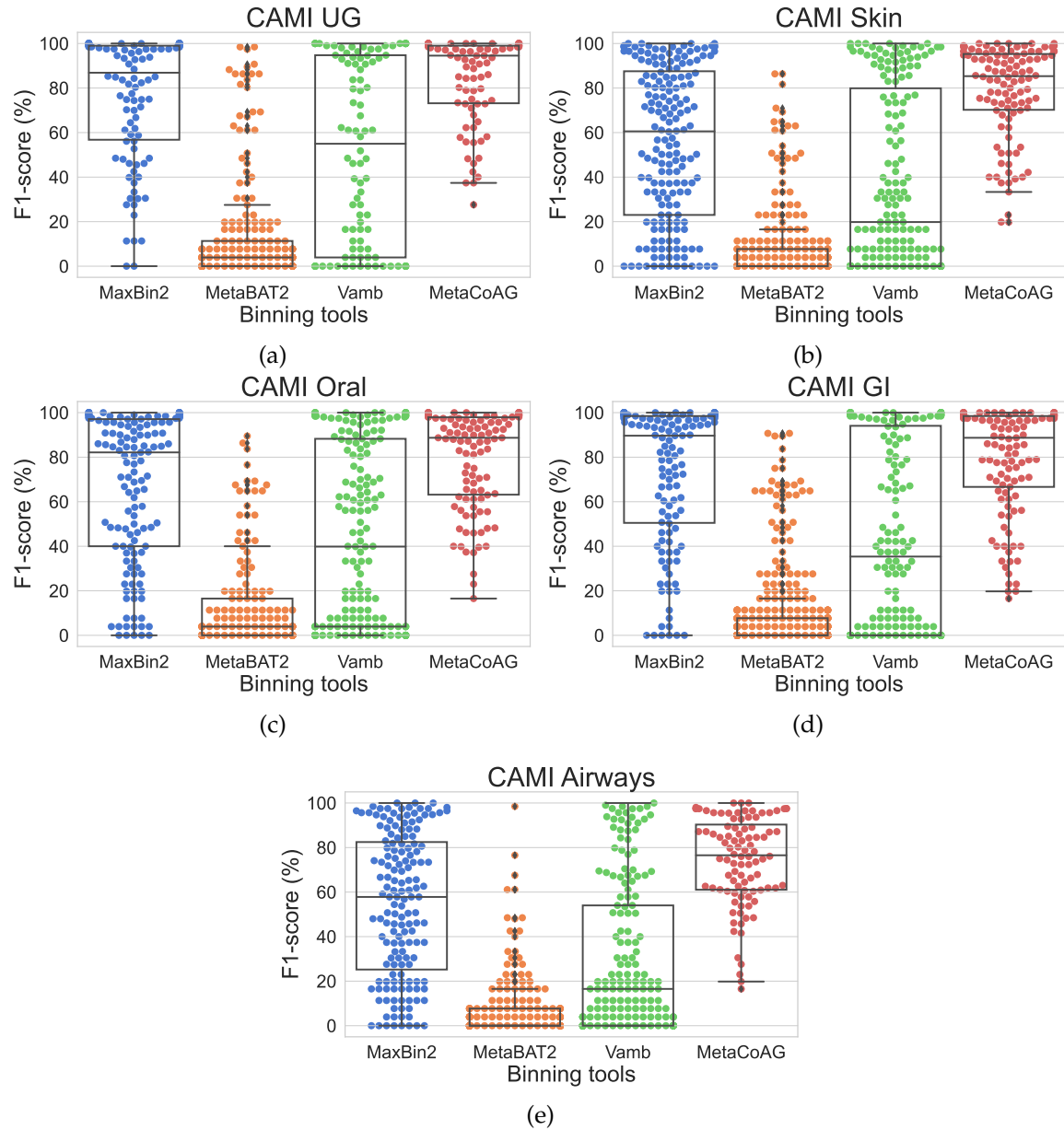


Figure 5.8: The F1-scores of the bins found in the CAMI datasets by all the binning tools.

median F1-score of 76.34% while the second-best tool (MaxBin2) has a median F1-score of 37.40%. Furthermore, MetaCoAG has produced the highest number of high-quality bins for all the real datasets (Please refer to Table 5.3 for the exact counts).

We used GTDB-Tk (Chaumeil et al., 2019) to annotate all the high-quality bins produced by MetaCoAG, MaxBin2 and Vamb³ for both datasets. Then we compared the taxonomic annotations (up to the species level) with the analysis results reported by the authors of these datasets (Refer to Table 5.2). Table 5.2 shows that MetaCoAG achieves the best consistency with the original analysis reported by the authors. In the Sharon dataset, the five most abundant species reported according to the authors (Sharon et al., 2013); *Staphylococcus epidermidis*, *Enterococcus faecalis*, *Cutibacterium avidum*, *Peptoniphilus lacydonensis* and *Staphylococcus aureus* have been successfully identified by all the three binning tools. However, Vamb missed *Staphylococcus hominis*, which is reported as a rare species in the Sharon dataset (Sharon et al., 2013). Moreover, MetaCoAG is the only tool that is able to recover *Leuconostoc citreum*, which is also identified as a rare species in the Sharon dataset (Sharon et al., 2013).

³MetaBAT2 results were not considered for GTDB-Tk annotations as the results had very low number of high-quality bins compared to the other binning tools.

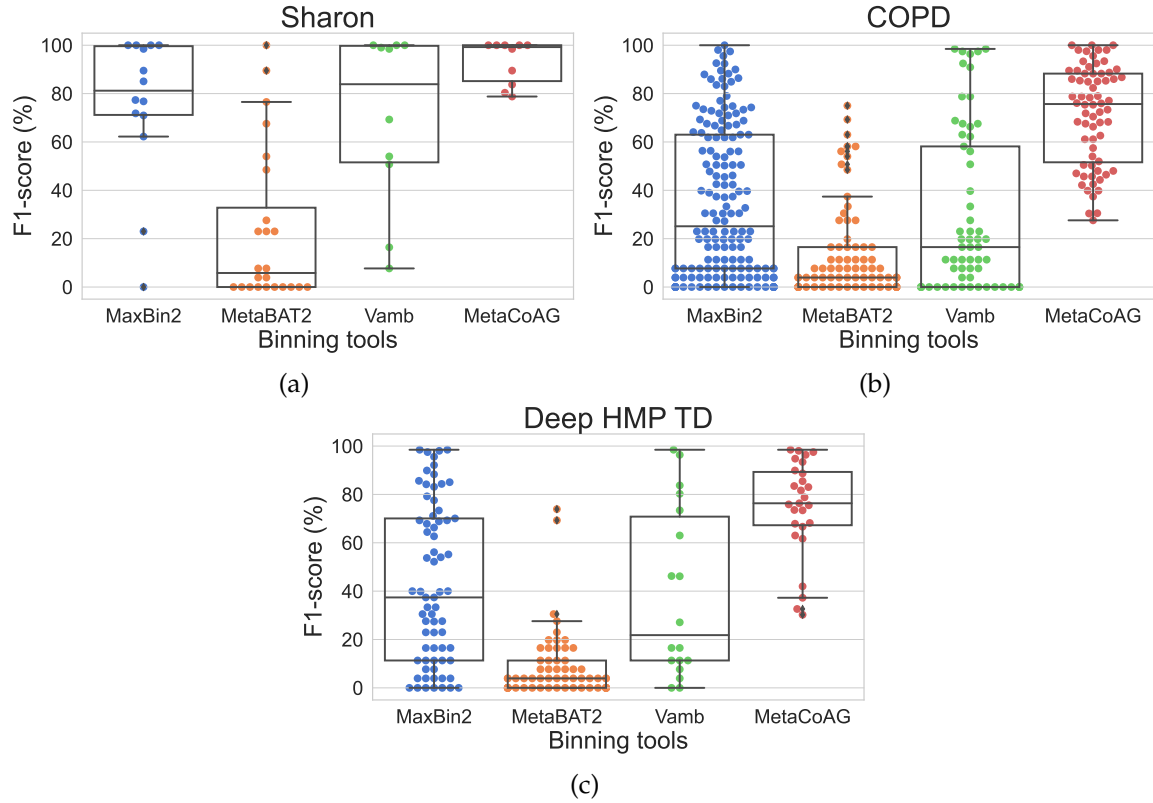


Figure 5.9: The F1-scores of the bins found in the real datasets by all the binning tools.

These results denote the ability of MetaCoAG to recover rare species in real metagenomics samples that are ignored by other binning tools.

In the COPD dataset, there is a larger discrepancy among MaxBin2, Vamb and MetaCoAG. Only two species, *Peptostreptococcus* sp. and *SR1 bacterium human oral taxon HOT-345*, have been identified by all the three binning tools. *SR1 bacterium human oral taxon HOT-345* and *Lachnospiraceae bacterium oral taxon 096* have been added to NCBI taxonomy recently (Schoch et al., 2020) and hence are not found in the original analysis (Cameron et al., 2016). Compared to MetaCoAG, MaxBin2 failed to identify three species *Prevotella pallens*, *Prevotella shahii* and *Prevotella histicola* while Vamb only identified *Prevotella pallens* under the genus *Prevotella*. Similarly, Vamb failed to identify two species, *Capnocytophaga gingivalis* and *Capnocytophaga leadbetteri*, both of which are identified by MaxBin2 and MetaCoAG. Moreover, the species *Anaeroglobus micronuciformis* only identified by MaxBin2 was not present in the top 50 genera ranked by abundance in the original analysis (Cameron et al., 2016), which is likely to be a false-positive. In the Deep HMP TD dataset, the species identified by MetaCoAG show best consistency with the original analysis, while being the only tool to identify the species from the genus *Eubacterium*. These results demonstrate that MetaCoAG has been able to recover more species correctly with respect to the original analysis of these real datasets.

5.4.4 Implementation, Running Time and Memory Usage

Table 5.4 denotes the running times (wall time) and the peak memory used by all the binning tools. All the binning tools were run on a Linux system with Ubuntu 18.04.1 LTS, 16 GB memory and Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz with 4 CPU cores. 8 threads were used for all the binning tools for parallel execution.

Table 5.2: High-quality species found from the GTDB-Tk annotations of MetaCoAG, MaxBin2 and Vamb for the real metagenomic datasets.

Dataset	Species	MaxBin2	Vamb	MetaCoAG	Present in original analysis
Sharon	Cutibacterium avidum	✓	✓	✓	✓
	Enterococcus faecalis	✓	✓	✓	✓
	Peptoniphilus lacydonensis	✓	✓	✓	✓
	Staphylococcus aureus	✓	✓	✓	✓
	Staphylococcus epidermidis	✓	✓	✓	✓
	Staphylococcus hominis	✓	✗	✓	✓
	Leuconostoc citreum	✗	✗	✓	✓
COPD*	Peptostreptococcus sp.	✓	✓	✓	✓
	SR1 bacterium human oral taxon HOT-345	✓	✓	✓	✗ [†]
	Prevotella pallens	✗	✓	✓	✓
	Haemophilus sputorum	✗	✓	✓	✓
	Herbaspirillum huttiense	✗	✓	✓	✓
	Capnocytophaga gingivalis	✓	✗	✓	✓
	Capnocytophaga leadbetteri	✓	✗	✓	✓
	Lancefieldella sp000564995	✓	✗	✓	✓
	Actinomyces graevenitzi	✓	✗	✗	✓
	Actinomyces oris	✓	✗	✗	✓
	Anaeroglobus micronuciformis	✓	✗	✗	✗
	Eubacterium sulci	✗	✗	✓	✓
	Prevotella shahii	✗	✗	✓	✓
	Prevotella histicola	✗	✗	✓	✓
	Lachnospiraceae bacterium oral taxon 096	✗	✗	✓	✗ [†]
Deep HMP TD*	Actinomyces graevenitzi	✓	✓	✓	✓
	Saccharimonadaceae TM7x sp900557595	✓	✓	✓	✗ [†]
	Neisseria subflava_C	✓	✗	✓	✓
	Prevotella pallens	✓	✗	✓	✓
	Anaeroglobus micronuciformis	✓	✗	✓	✗
	Actinomyces sp. ICM47	✓	✗	✓	✓
	Lancefieldella sp000564995	✓	✗	✗	✗
	Eubacterium_B sulci	✗	✗	✓	✓

We annotated all the high-quality bins of the real metagenomic datasets produced from MetaCoAG, MaxBin2 and Vamb using GTDB-Tk up to the species level.

Then we determined whether these taxonomic groups are actually present in the original analysis.

The species were determined by the classification string produced by GTDB-Tk up to species level.

✓ denotes that the species is present and ✗ denotes that the species is absent in the result/analysis.

Green coloured items match the original analysis whereas the red coloured items do not match the original analysis.

*For the COPD and Deep HMP TD datasets, the species were determined as present in the original analysis based on the 50 most abundant genera presented.

[†] These species were added to NCBI taxonomy in year 2020 (Schoch et al., 2020) which is after the relevant analyses.

Table 5.3: The number of high-quality, medium-quality and low-quality bins produced by each binning tool for all the datasets. The best values are highlighted in bold.

Dataset	Tool	Total number of bins detected	Number of high-quality bins	Number of medium-quality bins	Number of low-quality bins
simHC+	MaxBin2	95	59	11	25
	MetaBAT2	32	4	4	24
	MetaCoAG	90	69	8	13
CAMI UG	MaxBin2	98	49	17	32
	MetaBAT2	202	5	12	185
	Vamb	100	34	10	56
	MetaCoAG	83	50	17	16
CAMI Skin	MaxBin2	176	42	30	104
	MetaBAT2	240	0	5	235
	Vamb	167	36	15	116
	MetaCoAG	106	49	33	24
CAMI Oral	MaxBin2	137	54	24	59
	MetaBAT2	152	1	7	144
	Vamb	176	45	19	112
	MetaCoAG	106	58	19	29
CAMI GI	MaxBin2	127	59	22	46
	MetaBAT2	389	4	9	376
	Vamb	156	44	14	98
	MetaCoAG	113	57	28	28
CAMI Airways	MaxBin2	155	32	26	97
	MetaBAT2	205	1	2	202
	Vamb	173	20	14	139
	MetaCoAG	96	33	29	34
Sharon	MaxBin2	14	6	2	6
	MetaBAT2	24	2	2	20
	Vamb	10	5	1	4
	MetaCoAG	10	7	3	0
COPD	MaxBin2	156	9	24	123
	MetaBAT2	76	0	2	74
	Vamb	61	6	7	48
	MetaCoAG	68	17	25	26
Deep HMP TD	MaxBin2	69	8	15	46
	MetaBAT2	61	0	1	60
	Vamb	29	2	3	13
	MetaCoAG	27	8	9	10

Table 5.4: Running time and memory usage of the different binning tools for all the datasets.

Dataset	Tool	Running time (wall time)	Memory usage (kbytes)
simHC+	MaxBin2	5m 37s	4,063,345
	MetaBAT2	12s	619,345
	MetaCoAG	26m 34s	621,584
CAMI UG	MaxBin2	26m 04s	2,604,084
	MetaBAT2	1m 51s	985,860
	Vamb	2h 37m 32s	701,764
	MetaCoAG	19m 24s	1,944,632
CAMI Skin	MaxBin2	1h 58m 56s	1,184,476
	MetaBAT2	7m 21s	2,185,080
	Vamb	8h 18m 42s	976,644
	MetaCoAG	1h 01m 44s	5,291,572
CAMI Oral	MaxBin2	1h 11m 49s	1,130,420
	MetaBAT2	3m 42s	1,585,200
	Vamb	6h 31m 47s	891,608
	MetaCoAG	43m 24s8	4,411,072
CAMI GI	MaxBin2	59m 53s	1,874,764
	MetaBAT2	2m 53s	1,531,248
	Vamb	3h 26m 43s	741,960
	MetaCoAG	32m 16s	2,569,092
CAMI Airways	MaxBin2	1h 51m 35s	1,338,220
	MetaBAT2	6m 28s	2,145,720
	Vamb	10h 04m 16s	1,035,716
	MetaCoAG	1h 03m 13s	6,476,888
Sharon	MaxBin2	1m 27s	384,264
	MetaBAT2	9s	212,252
	Vamb	34m 20s	607,708
	MetaCoAG	7s	417,344
COPD	MaxBin2	1h 09m 17s	352,888
	MetaBAT2	1m 31s	950,584
	Vamb	6h 06m 54s	894,956
	MetaCoAG	13m 50s	3,967,256
Deep HMP TD	MaxBin2	8m 33s	698,652
	MetaBAT2	7s	473,924
	Vamb	2h 56m 54s	743,728
	MetaCoAG	10m 40s	1,282,096

Chapter 6

Conclusion and Future Work

In this thesis, we have studied how the assembly graph and its connectivity information can be utilised to improve metagenomic contig-binning results. While existing binning tools face challenges when binning short contigs, contigs shared between multiple species, contigs of species with high intra-variance in composition and coverage and contigs of species belonging to the same genus, this thesis has showed that the connectivity information among contigs in the assembly graph can be used to accurately bin contigs in above scenarios and improve binning results. The following sections summarise the conclusions for each of the problems presented in this thesis and discuss some future directions which the methods can be extended to.

6.1 Refined Binning of Metagenomic Contigs using Assembly Graphs

Existing binning tools face problems when binning short sequences, sequences of low abundance species and sequences of closely related species. Composition and abundance based binning tools have been able to overcome these shortcomings up to a certain extent. In most cases, the assembly process can produce a large number of short contigs having limited information about their nucleotide composition and abundance. Most existing binning tools will discard such short sequences (*e.g.*, shorter than 1,000 bp) due to their limited *k-mer* frequency information.

In Chapter 3, we proposed GraphBin, a new method to refine the binning results of existing binning tools by using the information found in the assembly graph followed by label propagation. We studied the assembly graphs constructed from both the *de Bruijn graph* and the *overlap-layout-consensus* (more recent *string graph*) approaches, and how contigs are connected in them. As shown in previous studies (Barnum et al., 2018), we observed that contigs connected to each other in the assembly graph are most likely to belong to the same taxonomic group. We used the assembly graph, the binning results of existing tools and a label propagation technique to correct mis-binned contigs and predict the labels of discarded contigs. Finally, experimental results confirmed that our approach refined the binning results by correcting mis-binned contigs and binning contigs which were discarded by existing tools.

6.2 Overlapped Binning of Metagenomic Contigs using Assembly Graphs

In Chapter 4, we presented a novel algorithm, GraphBin2, that incorporates the coverage information into the assembly graph as an improvement of GraphBin (Mallawaarachchi, Wickramarachchi et al., 2020a). While GraphBin uses only the topology of the assembly graph to refine and propagate labels, GraphBin2 makes use of the coverage information on vertices to perform label propagation. Moreover, in comparison to the label propagation algorithm used in GraphBin, GraphBin2 uses an improved label propagation algorithm that takes into consideration the distance and coverage of neighbouring contigs. Furthermore, GraphBin2 enables the detection of contigs that may belong to multiple species.

The performance of GraphBin2 was evaluated against its predecessor and three other contig-binning tools on top of contigs obtained from short-reads assembled using metaSPAdes (Nurk et al., 2017) and SGA (Simpson and Durbin, 2012) which represent the two assembly paradigms; *de Bruijn graphs*

and overlap-layout-consensus (string graphs). The results showed that GraphBin2 achieves the best binning performance in both simulated and real datasets. Furthermore, GraphBin2 shows the potential to infer contigs shared by multiple species and we have experimentally shown that GraphBin2 could be in principle applied to long-read assemblies.

6.3 Binning Metagenomic Contigs using Composition, Coverage and Assembly Graphs

Metagenomic sequencing and *de novo* assembly, coupled with binning methods have facilitated the characterization of different microbial communities. The majority of existing metagenomic contig-binning tools do not make use of the valuable connectivity information found in assembly graphs from which the contigs are derived. Moreover, existing tools do not make use of multiple single-copy marker genes throughout the entire binning process. Furthermore, existing bin-refinement tools and metabinner rely upon the bins produced by an existing binning tool and cannot dynamically adjust the number of bins.

In Chapter 5, we have presented MetaCoAG, a tool for binning metagenomic contigs that makes use of composition, coverage and assembly graphs simultaneously and does not rely on an initial binning result. The use of connectivity information from the assembly graphs makes the binning process of MetaCoAG robust against similar inter-species oligonucleotide composition and coverage (within the same genus) as well as high variance of intra-species oligonucleotide composition and coverage. Experimental results on both simulated and real datasets show that MetaCoAG achieves the best binning results compared to state-of-the-art tools, especially in terms of bin quality.

6.4 Future Work


This thesis explores how the assembly graph can be incorporated in the process of binning metagenomic contigs and presents methods to obtain improved binning results. We showed that the assembly graph contains valuable connectivity information among contigs that can be used in the binning process and contigs connected to each other in the assembly graph are more likely to belong to the same taxonomic group. We showed that the assembly graph can be used to recover short contigs that are discarded by existing binning tools and determine contigs shared by multiple species. Finally, we propose a stand-alone binning tool to bin metagenomic contigs using assembly graphs, composition, coverage and single-copy marker genes. We showed that the usage of assembly graphs has contributed to accurately bin challenging contigs such as those belonging to species of the same genus and species with high intra-variance in composition and coverage. The findings in this thesis suggest several future directions of research where the proposed methods can be extended.

6.4.1 Graph Representation Learning for Metagenomics Binning

The methods proposed in this thesis are based on combinatorial optimisation techniques. As new machine learning techniques are developed and their applications are getting popular, it is worth exploring if such techniques can be used to perform metagenomics binning based on assembly graphs.

Various machine learning techniques focusing on graphs have been used in many applications of node clustering and classification as it is convenient to represent real-world data by graphs, especially in biological networks (Chen, Wang et al., 2020). *Graph representation learning* methods try to learn representations that encode structural information of the graph while preserving the intrinsic graph properties (Chen, Wang et al., 2020; Hamilton, 2020). Once the learned graph representation is obtained, various machine learning techniques can be applied to perform downstream tasks.

Reference-free metagenomics binning can be considered as a clustering task. Hence, we can make use of graph learning techniques such as graph representation learning to obtain node embeddings of the assembly graph and perform node clustering. We can model contigs connected together in the assembly graph as homophily relations (that they should be in the same bin) and contigs containing the same single-copy marker gene as heterophily constraints (that they should not be in the same bin), learn the assembly graph with these constraints and perform constraint based clustering (Figure 6.1).

This co-authored work was carried out as a collaboration with National University of Singapore and is based on the paper H. Xue et al. (2022). *RepBin: Constraint-based Graph Representation Learning for Metagenomic Binning*.  that has been accepted at the 36th AAAI Conference on Artificial Intelligence (AAAI 22).

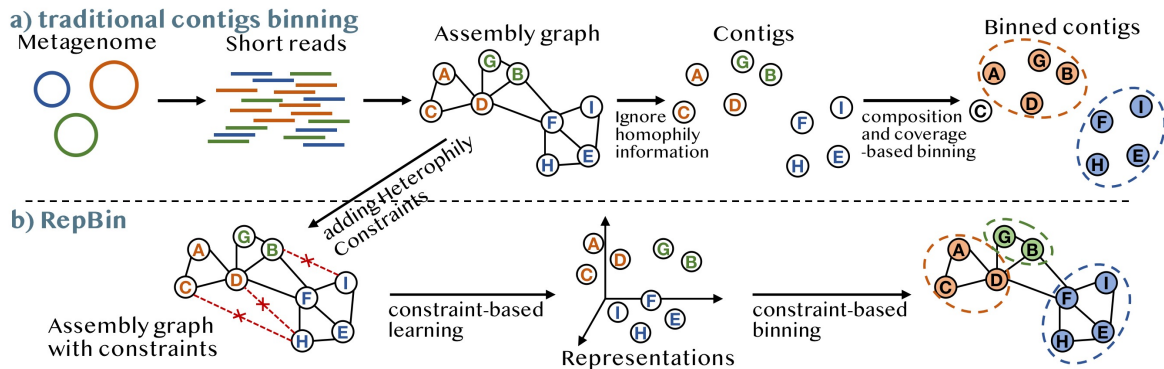


Figure 6.1: Traditional metagenomics binning and graph representation learning for metagenomics binning (Xue et al., 2022).

6.4.2 Improving Taxonomic Binning Pipelines

Recent studies on benchmarking taxonomic classification tools have identified that existing tools can produce a large number of low-abundance false positive classifications for metagenomic datasets, which can lower the precision of these results (Ye et al., 2019). Some tools provide certain confidence parameters as a means of filtering results, but they do not provide any recommendations for using them. Parameter settings can vary depending on the experimental design and even between different classifiers, which can make it very difficult to use. These parameter changes can be very sensitive when dealing with various metagenomic datasets containing different compositions of microorganisms and affect downstream analysis results.

A recent study by Sun et al. (2021) suggests that different tools report different relative abundance types; (1) *relative sequence abundance* which is the number of sequences assigned to a taxon relative to the total number of sequences classified and (2) *relative taxonomic abundance* which is the number of genomes of a given taxon relative to the total number of genomes identified in the sample. The comparison of these different abundance types together can be misleading when analysing results of metagenomic profiling studies.

We are currently working to improve taxonomic classification pipelines where we refine the search space for a given metagenomics sample and try to annotate sequences as much as possible. This work is carried out in collaboration with Prof. Liang Qiao's group at Fudan University, China. A manuscript is being prepared for submission at the moment.

6.4.3 Applications to Study Medical Data


Investigating the microbiome's role in diseases such as Inflammatory Bowel Disease (IBD) can improve the understanding of how microorganisms affect their hosts and develop efficient treatment methods. Previous studies have shown that the gut microbes play an important role in IBD (Ni et al., 2017). Studying the intestinal bacteria in healthy people and patients with IBD can identify what changes have occurred in terms of microbial community composition and relative abundance of taxa (Ma et al., 2021). In order to identify the intestinal bacteria present in healthy and patient samples, the binning tools presented in this thesis can be incorporated into metagenomics analysis pipelines.

With the development of high-throughput sequencing technologies to sample at the time-scale, longitudinal studies (Faust et al., 2015) have gained popularity in metagenomics (Coenen et al., 2020; Faust et al., 2015). These studies analyse microbiome time-series data and record the temporal variation of microbial communities in different environments (Coenen et al., 2020; Faust et al., 2015). The binning tools proposed in this thesis can incorporate the different coverage values of sequences in different samples and in turn can capture the changes in relative abundances of taxa.

Bibliography


- Adams, J. (2008). 'DNA sequencing technologies'. *Nature Education*, 1(1), p. 193.  (cited on p. 5).
- Albertsen, M., Hugenholtz, P., Skarshewski, A., Nielsen, K. L., Tyson, G. W. and Nielsen, P. H. (2013). 'Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes'. *Nature Biotechnology*, 31(6), pp. 533–538. DOI: [10.1038/nbt.2579](https://doi.org/10.1038/nbt.2579) (cited on pp. 9, 52, 54).
- Alneberg, J., Bjarnason, B. S., Bruijn, I. de, 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, pp. 1144–1146.  (cited on pp. 10, 12, 21, 38, 40, 41, 43, 44, 54).
- Andreani, J., Verneau, J., Raoult, D., Levasseur, A. and La Scola, B. (2018). 'Deciphering viral presences: two novel partial giant viruses detected in marine metagenome and in a mine drainage metagenome'. *Virology Journal*, 15(1), p. 66. DOI: [10.1186/s12985-018-0976-9](https://doi.org/10.1186/s12985-018-0976-9) (cited on p. 2).
- Baker, M. (2012). 'De novo genome assembly: what every biologist should know'. *Nature Methods*, 9(4), pp. 333–337. DOI: [10.1038/nmeth.1935](https://doi.org/10.1038/nmeth.1935) (cited on pp. 2, 6).
- Bankevich, A. et al. (2012). 'SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing'. *Journal of Computational Biology*, 19(5). PMID: 22506599, pp. 455–477. DOI: [10.1089/cmb.2012.0021](https://doi.org/10.1089/cmb.2012.0021) (cited on pp. 7, 20, 38, 57).
- Barnum, T. P., Figueroa, I. A., Carlström, C. I., Lucas, L. N., Engelbrektson, A. L. and Coates, J. D. (2018). 'Genome-resolved metagenomics identifies genetic mobility, metabolic interactions, and unexpected diversity in perchlorate-reducing communities'. *The ISME Journal*, 12(6), pp. 1568–1581. DOI: [10.1038/s41396-018-0081-5](https://doi.org/10.1038/s41396-018-0081-5) (cited on pp. 16, 32, 55, 67).
- Batzoglou, S., Jaffe, D. B., Stanley, K., Butler, J., Gnerre, S., Mauceli, E., Berger, B., Mesirov, J. P. and Lander, E. S. (2002). 'ARACHNE: a whole-genome shotgun assembler'. *Genome research*, 12(1), pp. 177–189 (cited on p. 6).
- Berini, F., Casciello, C., Marcone, G. L. and Marinelli, F. (2017). 'Metagenomics: novel enzymes from non-culturable microbes'. *FEMS Microbiology Letters*, 364(21). DOI: [10.1093/femsle/fnx211](https://doi.org/10.1093/femsle/fnx211) (cited on p. 2).
- Bohlin, J., Eldholm, V., Pettersson, J. H. O., Brynildsrud, O. and Snipen, L. (2017). 'The nucleotide composition of microbial genomes indicates differential patterns of selection on core and accessory genomes'. *BMC Genomics*, 18(1), p. 151. DOI: [10.1186/s12864-017-3543-7](https://doi.org/10.1186/s12864-017-3543-7) (cited on p. 12).
- Bohlin, J., Skjerve, E. and Ussery, D. W. (2008). 'Investigations of Oligonucleotide Usage Variance Within and Between Prokaryotes'. *PLOS Computational Biology*, 4(4), pp. 1–9. DOI: [10.1371/journal.pcbi.1000057](https://doi.org/10.1371/journal.pcbi.1000057) (cited on p. 12).
- Bohlin, J., Snipen, L., Hardy, S. P., Kristoffersen, A. B., Lagesen, K., Dønsvik, T., Skjerve, E. and Ussery, D. W. (2010). 'Analysis of intra-genomic GC content homogeneity within prokaryotes'. *BMC Genomics*, 11(1), p. 464. DOI: [10.1186/1471-2164-11-464](https://doi.org/10.1186/1471-2164-11-464) (cited on p. 12).
- Bonfield, J. K., Smith, K. F. and Staden, R. (1995). 'A new DNA sequence assembly program'. *Nucleic acids research*, 23(24), pp. 4992–4999 (cited on p. 6).
- Boulund, F., Berglund, F., Flach, C.-F., Bengtsson-Palme, J., Marathe, N. P., Larsson, D. J. and Kristiansson, E. (2017). 'Computational discovery and functional validation of novel fluoroquinolone resistance genes in public metagenomic data sets'. *BMC Genomics*, 18(1), p. 682. DOI: [10.1186/s12864-017-4064-0](https://doi.org/10.1186/s12864-017-4064-0) (cited on p. 2).


- Breitwieser, F. P., Lu, J. and Salzberg, S. L. (2017). 'A review of methods and databases for metagenomic classification and assembly'. *Briefings in Bioinformatics*, 20(4), pp. 1125–1136. DOI: [10.1093/bib/bbx120](https://doi.org/10.1093/bib/bbx120) (cited on pp. 7, 11).
- Burton, J. N., Liachko, I., Dunham, M. J. and Shendure, J. (2014). 'Species-level deconvolution of metagenome assemblies with Hi-C-based contact probability maps'. *G3: Genes, Genomes, Genetics*, 4(7), pp. 1339–1346 (cited on p. 12).
- Cameron, S. J. S., Lewis, K. E., Huws, S. A., Lin, W., Hegarty, M. J., Lewis, P. D., Mur, L. A. J. and Pachebat, J. A. (2016). 'Metagenomic Sequencing of the Chronic Obstructive Pulmonary Disease Upper Bronchial Tract Microbiome Reveals Functional Changes Associated with Disease Severity'. *PLOS ONE*, 11(2), pp. 1–16. DOI: [10.1371/journal.pone.0149095](https://doi.org/10.1371/journal.pone.0149095) (cited on pp. 57, 61, 63).
- Canard, B. and Sarfati, R. S. (1994). 'DNA polymerase fluorescent substrates with reversible 3'-tags'. *Gene*, 148(1), pp. 1–6 (cited on p. 5).
- Chan, C.-K. K., Hsu, A. L., Halgamuge, S. K. and Tang, S.-L. (2008). 'Binning sequences using very sparse labels within a metagenome'. *BMC bioinformatics*, 9(1), pp. 1–17 (cited on p. 12).
- Chaumeil, P.-A., Mussig, A. J., Hugenholtz, P. and Parks, D. H. (2019). 'GTDB-Tk: a toolkit to classify genomes with the Genome Taxonomy Database'. *Bioinformatics*, 36(6), pp. 1925–1927. DOI: [10.1093/bioinformatics/btz848](https://doi.org/10.1093/bioinformatics/btz848) (cited on pp. 57, 58, 62).
- Chen, F., Wang, Y.-C., Wang, B. and Kuo, C.-C. J. (2020). 'Graph representation learning: a survey'. *APSIPA Transactions on Signal and Information Processing*, 9, e15. DOI: [10.1017/ATSIP.2020.13](https://doi.org/10.1017/ATSIP.2020.13) (cited on p. 68).
- Chen, K. and Pachter, L. (2018). 'Bioinformatics for Whole-Genome Shotgun Sequencing of Microbial Communities'. *PLOS Computational Biology*, 1(2). DOI: [10.1371/journal.pcbi.0010024](https://doi.org/10.1371/journal.pcbi.0010024) (cited on p. 2).
- Cleary, B., Brito, I. L., Huang, K., Gevers, D., Shea, T., Young, S. and Alm, E. J. (2015). 'Detection of low-abundance bacterial strains in metagenomic datasets by eigengene partitioning'. *Nature Biotechnology*, 33, p. 1053. [DOI: 10.1038/nbt.3333](https://doi.org/10.1038/nbt.3333) (cited on p. 3).
- Coenen, A. R., Hu, S. K., Luo, E., Muratore, D. and Weitz, J. S. (2020). 'A Primer for Microbiome Time-Series Analysis'. *Frontiers in Genetics*, 11. DOI: [10.3389/fgene.2020.00310](https://doi.org/10.3389/fgene.2020.00310) (cited on p. 69).
- 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), pp. 798–805 (cited on p. 2).
- Deschavanne, P. J., Giron, A., Vilain, J., Fagot, G. and Fertil, B. (1999). 'Genomic signature: characterization and classification of species assessed by chaos game representation of sequences'. *Molecular Biology and Evolution*, 16(10), pp. 1391–1399. DOI: [10.1093/oxfordjournals.molbev.a026048](https://doi.org/10.1093/oxfordjournals.molbev.a026048) (cited on p. 54).
- 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), R85. DOI: [10.1186/gb-2009-10-8-r85](https://doi.org/10.1186/gb-2009-10-8-r85) (cited on p. 9).
- Dolz, R. (1994). 'GCG: fragment assembly programs'. *Methods Mol. Biol*, 24, pp. 9–23 (cited on p. 6).
- Dröge, J. and McHardy, A. C. (2012). 'Taxonomic binning of metagenome samples generated by next-generation sequencing technologies'. *Briefings in Bioinformatics*, 13(6), pp. 646–655. DOI: [10.1093/bib/bbs031](https://doi.org/10.1093/bib/bbs031) (cited on p. 7).
- Dupont, C. L., Rusch, D. B., Yooseph, S., Lombardo, M.-J., Richter, R. A., Valas, R., Novotny, M., Yee-Greenbaum, J., Selengut, J. D., Haft, D. H. et al. (2012). 'Genomic insights to SAR86, an abundant and uncultivated marine bacterial lineage'. *The ISME journal*, 6(6), pp. 1186–1199 (cited on pp. 9, 52).
- Eddy, S. R. (2011). 'Accelerated Profile HMM Searches'. *PLOS Computational Biology*, 7(10), pp. 1–16. DOI: [10.1371/journal.pcbi.1002195](https://doi.org/10.1371/journal.pcbi.1002195) (cited on p. 52).
- Faust, K., Lahti, L., Gonze, D., de Vos, W. M. and Raes, J. (2015). 'Metagenomics meets time series analysis: unraveling microbial community dynamics'. *Current Opinion in Microbiology*, 25. *Environmental microbiology • Extremophiles*, pp. 56–66. DOI: <https://doi.org/10.1016/j.mib.2015.04.004> (cited on p. 69).


- Flicek, P. and Birney, E. (2009). 'Sense from sequence reads: methods for alignment and assembly'. *Nature methods*, 6(11), S6–S12 (cited on p. 6).
- Frank, J. A., Pan, Y., Tooming-Klunderud, A. et al. (2016). 'Improved metagenome assemblies and taxonomic binning using long-read circular consensus sequence data'. *Scientific Reports*, 6. Article, p. 25373.  (cited on p. 3).
- Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co. ISBN: 0716710447 (cited on p. 37).
- 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), pp. 4287–4308. DOI: [10.1007/s00018-015-2004-1](https://doi.org/10.1007/s00018-015-2004-1) (cited on pp. 1, 2).
- Ghurye, J. S., Cepeda-Espinoza, V. and Pop, M. (2016). 'Focus: microbiome: metagenomic assembly: overview, challenges and applications'. *The Yale journal of biology and medicine*, 89(3), p. 353 (cited on pp. 2, 7).
- Giani, A. M., Gallo, G. R., Gianfranceschi, L. and Formenti, G. (2020). 'Long walk to genomics: History and current approaches to genome sequencing and assembly'. *Computational and Structural Biotechnology Journal*, 18, pp. 9–19. DOI: <https://doi.org/10.1016/j.csbj.2019.11.002> (cited on pp. 5, 7).
- Girotto, S., Pizzi, C. and Comin, M. (2016). 'MetaProb: accurate metagenomic reads binning based on probabilistic sequence signatures'. *Bioinformatics*, 32(17), pp. i567–i575. DOI: [10.1093/bioinformatics/btw466](https://doi.org/10.1093/bioinformatics/btw466) (cited on p. 3).
- Gleeson, T. J. and Staden, R. (1991). 'An X windows and UNIX implementation of our sequence analysis package'. *Bioinformatics*, 7(3), pp. 398–398. DOI: [10.1093/bioinformatics/7.3.398](https://doi.org/10.1093/bioinformatics/7.3.398) (cited on p. 6).
- Gourlé, H., Karlsson-Lindsjö, O., Hayer, J. and Bongcam-Rudloff, E. (2018). 'Simulating Illumina metagenomic data with InSilicoSeq'. *Bioinformatics*, 35(3), pp. 521–522. DOI: [10.1093/bioinformatics/bty630](https://doi.org/10.1093/bioinformatics/bty630) (cited on pp. 19, 37, 57).
- Green, P. (1999). 'Documentation for phrap and cross_match'. <http://bozeman.mbt.washington.edu/phrap.docs/phrap.html> (cited on p. 6).
- Gryan, G. (1994). 'Faster sequence assembly software for megabase shotgun assemblies'. In: *Genome Sequencing and Analysis Conference VI* (cited on p. 6).
- Haider, B., Ahn, T.-H., Bushnell, B., Chai, J., Copeland, A. and Pan, C. (2014). 'Omega: an Overlap-graph de novo Assembler for Metagenomics'. *Bioinformatics*, 30(19), pp. 2717–2722. DOI: [10.1093/bioinformatics/btu395](https://doi.org/10.1093/bioinformatics/btu395) (cited on p. 7).
- Hamilton, W. L. (2020). 'Graph Representation Learning'. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3), pp. 1–159. DOI: [10.2200/S01045ED1V01Y202009AIM046](https://doi.org/10.2200/S01045ED1V01Y202009AIM046) (cited on p. 68).
- 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 p. 2).
- He, X., McLean, J. S., Edlund, A., Yooseph, S., Hall, A. P., Liu, S.-Y., Dorrestein, P. C., Esquenazi, E., Hunter, R. C., Cheng, G. et al. (2015). 'Cultivation of a human-associated TM7 phylotype reveals a reduced genome and epibiotic parasitic lifestyle'. *Proceedings of the National Academy of Sciences*, 112(1), pp. 244–249 (cited on p. 2).
- Herath, D., Tang, S.-L., Tandon, K., Ackland, D. and Halgamuge, S. K. (2017). 'CoMet: a workflow using contig coverage and composition for binning a metagenomic sample with high precision'. *BMC Bioinformatics*, 18(Suppl 16), p. 571. DOI: [10.1186/s12859-017-1967-3](https://doi.org/10.1186/s12859-017-1967-3) (cited on pp. 21, 34).
- Hofer, U. (2018). 'The majority is uncultured'. *Nature Reviews Microbiology*, 16(12), pp. 716–717. DOI: [10.1038/s41579-018-0097-x](https://doi.org/10.1038/s41579-018-0097-x) (cited on p. 1).
- Hu, T., Chitnis, N., Monos, D. and Dinh, A. (2021). 'Next-generation sequencing technologies: An overview'. *Human Immunology*, 82(11). Next Generation Sequencing and its Application to Medical Laboratory Immunology, pp. 801–811. DOI: <https://doi.org/10.1016/j.humimm.2021.02.012> (cited on p. 5).


- Huang, X. and Madan, A. (1999). 'CAP3: A DNA sequence assembly program'. *Genome research*, 9(9), pp. 868–877 (cited on p. 6).
- Hugenholtz, P. and Tyson, G. W. (2008). 'Metagenomics'. *Nature*, 455(7212), pp. 481–483. DOI: [10.1038/455481a](https://doi.org/10.1038/455481a) (cited on p. 2).
- Hyman, E. D. (1988). 'A new method of sequencing DNA'. *Analytical Biochemistry*, 174(2), pp. 423–436. DOI: [https://doi.org/10.1016/0003-2697\(88\)90041-3](https://doi.org/10.1016/0003-2697(88)90041-3) (cited on p. 5).
- Idury, R. M. and Waterman, M. S. (1995). 'A new algorithm for DNA sequence assembly'. *Journal of computational biology*, 2(2), pp. 291–306 (cited on p. 7).
- Imelfort, M., Parks, D., Woodcroft, B. J., Dennis, P., Hugenholtz, P. and Tyson, G. W. (2014). 'GroopM: an automated tool for the recovery of population genomes from related metagenomes'. *PeerJ*, 2, e603 (cited on p. 10).
- Kang, D., Li, F., Kirton, E. S., Thomas, A., Egan, R. S., An, H. and Wang, Z. (2019). 'MetaBAT 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies'. *PeerJ Preprints*, 7, e27522v1. DOI: [10.7287/peerj.preprints.27522v1](https://doi.org/10.7287/peerj.preprints.27522v1) (cited on pp. 10, 12, 16, 18, 21, 22, 32, 54, 57, 58, 60, 61).
- Kang, D. D., Froula, J., Egan, R. and Wang, Z. (2015). 'MetaBAT, an efficient tool for accurately reconstructing single genomes from complex microbial communities'. *PeerJ*, 3, e1165. DOI: [10.7717/peerj.1165](https://doi.org/10.7717/peerj.1165) (cited on p. 5).
- Karilin, S. and Burge, C. (1995). 'Dinucleotide relative abundance extremes: a genomic signature'. *Trends in genetics*, 11(7), pp. 283–290 (cited on p. 8).
- Karlin, S. and Ladunga, I. (1994). 'Comparisons of eukaryotic genomic sequences'. *Proceedings of the National Academy of Sciences*, 91(26), pp. 12832–12836 (cited on p. 8).
- Karp, R. M. (1980). 'An algorithm to solve the $m \times n$ assignment problem in expected time $O(mn \log n)$ '. *Networks*, 10(2), pp. 143–152. DOI: [10.1002/net.3230100205](https://doi.org/10.1002/net.3230100205) (cited on p. 55).
- Kececioglu, J. D. and Myers, E. W. (1995). 'Combinatorial algorithms for DNA sequence assembly'. *Algorithmica*, 13(1), p. 7. DOI: [10.1007/BF01188580](https://doi.org/10.1007/BF01188580) (cited on p. 6).
- Kislyuk, A., Bhatnagar, S., Dushoff, J. and Weitz, J. S. (2009). 'Unsupervised statistical clustering of environmental shotgun sequences'. *BMC Bioinformatics*, 10(1), p. 316. DOI: [10.1186/1471-2105-10-316](https://doi.org/10.1186/1471-2105-10-316) (cited on p. 9).
- Kolmogorov, M., Bickhart, D. M. et al. (2020). 'metaFlye: scalable long-read metagenome assembly using repeat graphs'. *Nature Methods*, 17(11), pp. 1103–1110. DOI: [10.1038/s41592-020-00971-x](https://doi.org/10.1038/s41592-020-00971-x) (cited on pp. 7, 32, 38, 52).
- Kolmogorov, M., Yuan, J., Lin, Y. and Pevzner, P. A. (2019). 'Assembly of long, error-prone reads using repeat graphs'. *Nature Biotechnology*, 37(5), pp. 540–546 (cited on pp. 7, 38).
- Köpke, B., Wilms, R., Engelen, B., Cypionka, H. and Sass, H. (2005). 'Microbial diversity in coastal sub-surface sediments: a cultivation approach using various electron acceptors and substrate gradients'. *Applied and environmental microbiology*, 71(12), pp. 7819–7830 (cited on p. 2).
- 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), pp. 722–736 (cited on p. 7).
- Kuśmirek, W. and Nowak, R. (2018). 'De novo assembly of bacterial genomes with repetitive DNA regions by dnaasm application'. *BMC Bioinformatics*, 19(1), p. 273. DOI: [10.1186/s12859-018-2281-4](https://doi.org/10.1186/s12859-018-2281-4) (cited on p. 13).
- Laczny, C. C., Sternal, T., Plugaru, V., Gawron, P., Atashpendar, A., Margossian, H. H., Coronado, S., Van der Maaten, L., Vlassis, N. and Wilmes, P. (2015). 'VizBin-an application for reference-independent visualization and human-augmented binning of metagenomic data'. *Microbiome*, 3(1), pp. 1–7 (cited on p. 10).
- 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*, 45(W1), W171–W179. DOI: [10.1093/nar/gkx348](https://doi.org/10.1093/nar/gkx348) (cited on pp. 3, 10, 16, 21, 22, 32).

- Lander, E. S. and Waterman, M. S. (1988). 'Genomic mapping by fingerprinting random clones: A mathematical analysis'. *Genomics*, 2(3), pp. 231–239. DOI: [https://doi.org/10.1016/0888-7543\(88\)90007-9](https://doi.org/10.1016/0888-7543(88)90007-9) (cited on pp. 9, 54).
- 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), pp. 1674–1676. DOI: [10.1093/bioinformatics/btv033](https://doi.org/10.1093/bioinformatics/btv033) (cited on pp. 7, 16, 20, 22, 52).
- Li, H. (2013). *Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM* (cited on p. 38).
- Li, H. (2018). 'Minimap2: pairwise alignment for nucleotide sequences'. *Bioinformatics*, 34(18), pp. 3094–3100. DOI: [10.1093/bioinformatics/bty191](https://doi.org/10.1093/bioinformatics/bty191) (cited on p. 58).
- Li, K., Wang, L., Shi, L., Deng, L. and Wang, Z. (2019). 'Deconvolute individual genomes from metagenome sequences through read clustering'. *bioRxiv*. DOI: [10.1101/620666](https://doi.org/10.1101/620666) (cited on p. 18).
- Li, Z., Chen, Y., Mu, D., Yuan, J., Shi, Y., Zhang, H., Gan, J., Li, N., Hu, X., Liu, B. et al. (2012). 'Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph'. *Briefings in functional genomics*, 11(1), pp. 25–37 (cited on p. 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), p. 24175. DOI: [10.1038/srep24175](https://doi.org/10.1038/srep24175) (cited on pp. 9, 10).
- Lloyd-Price, J. et al. (2017). 'Strains, functions and dynamics in the expanded Human Microbiome Project'. *Nature*, 550(7674), pp. 61–66. DOI: [10.1038/nature23889](https://doi.org/10.1038/nature23889) (cited on pp. 57, 61).
- Lu, Y. Y., Chen, T., Sun, F. and Fuhrman, J. A. (2016). 'COCACOLA: binning metagenomic contigs using sequence COMposition, read CoverAge, CO-alignment and paired-end read LinkAge'. *Bioinformatics*, 33(6), pp. 791–798. DOI: [10.1093/bioinformatics/btw290](https://doi.org/10.1093/bioinformatics/btw290) (cited on pp. 9, 10, 12).
- Luo, Y., Yu, Y. W., Zeng, J., Berger, B. and Peng, J. (2018). 'Metagenomic binning through low-density hashing'. *Bioinformatics*, 35(2), pp. 219–226. DOI: [10.1093/bioinformatics/bty611](https://doi.org/10.1093/bioinformatics/bty611) (cited on p. 3).
- Ma, Y. et al. (2021). 'Metagenome Analysis of Intestinal Bacteria in Healthy People, Patients With Inflammatory Bowel Disease and Colorectal Cancer'. *Frontiers in Cellular and Infection Microbiology*, 11. DOI: [10.3389/fcimb.2021.599734](https://doi.org/10.3389/fcimb.2021.599734) (cited on p. 69).
- Mallawaarachchi, V. and Lin, Y. (2022). 'MetaCoAG: Binning Metagenomic Contigs via Composition, Coverage and Assembly Graphs'. In: *Research in Computational Molecular Biology (RECOMB 2022)*. Ed. by I. Pe'er. Vol. 13278. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 70–85. ISBN: 978-3-031-04749-7. DOI: [10.1007/978-3-031-04749-7_5](https://doi.org/10.1007/978-3-031-04749-7_5) (cited on pp. i, 4, 14, 51).
- Mallawaarachchi, V., Wickramarachchi, A. and Lin, Y. (2020a). 'GraphBin: refined binning of metagenomic contigs using assembly graphs'. *Bioinformatics*, 36(11), pp. 3307–3313. DOI: [10.1093/bioinformatics/btaa180](https://doi.org/10.1093/bioinformatics/btaa180) (cited on pp. i, 4, 14, 15, 32, 38, 40–43, 50, 55, 67).
- 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)*. Ed. by C. Kingsford and N. Pisanti. Vol. 172. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 8:1–8:21. ISBN: 978-3-95977-161-0. DOI: [10.4230/LIPIcs.WABI.2020.8](https://doi.org/10.4230/LIPIcs.WABI.2020.8) (cited on pp. i, 4, 14, 31, 50).
- Mallawaarachchi, V. G., Wickramarachchi, A. S. and Lin, Y. (2021). 'Improving metagenomic binning results with overlapped bins using assembly graphs'. *Algorithms for Molecular Biology*, 16(1), p. 3. DOI: [10.1186/s13015-021-00185-6](https://doi.org/10.1186/s13015-021-00185-6) (cited on pp. i, 4, 14, 31).
- Mardis, E. R. (2017). 'DNA sequencing technologies: 2006–2016'. *Nature Protocols*, 12(2), pp. 213–218. DOI: [10.1038/nprot.2016.182](https://doi.org/10.1038/nprot.2016.182) (cited on p. 5).
- Mcewan, C. E., Gatherer, D. and Mcewan, N. R. (1998). 'Nitrogen-fixing aerobic bacteria have higher genomic GC content than non-fixing species within the same genus'. *Hereditas*, 128(2), pp. 173–178 (cited on p. 13).

- Mehrshad, M., Salcher, M. M., Okazaki, Y., Nakano, S.-i., Šimek, K., Andrei, A.-S. and Ghai, R. (2018). 'Hidden in plain sight—highly abundant and diverse planktonic freshwater Chloroflexi'. *Microbiome*, 6(1), p. 176 (cited on pp. 37, 39).
- Meyer, F., Fritz, A. et al. (2022). 'Critical Assessment of Metagenome Interpretation: the second round of challenges'. *Nature Methods*, 19(4), pp. 429–440. DOI: [10.1038/s41592-022-01431-4](https://doi.org/10.1038/s41592-022-01431-4) (cited on pp. 7, 11, 57, 60).
- Meyer, F., Hofmann, P., Belmann, P., Garrido-Oter, R., Fritz, A., Sczyrba, A. and McHardy, A. C. (2018). 'AMBER: Assessment of Metagenome BinnERs'. *GigaScience*, 7(6). giy069. DOI: [10.1093/gigascience/giy069](https://doi.org/10.1093/gigascience/giy069) (cited on pp. 57, 58).
- Miller, J. R., Koren, S. and Sutton, G. (2010). 'Assembly algorithms for next-generation sequencing data'. *Genomics*, 95(6), pp. 315–327 (cited on p. 6).
- Mitchell, D. (2007). 'GC content and genome length in Chargaff compliant genomes'. *Biochemical and biophysical research communications*, 353(1), pp. 207–210 (cited on p. 13).
- Mullikin, J. C. and Ning, Z. (2003). 'The phusion assembler'. *Genome research*, 13(1), pp. 81–90 (cited on p. 7).
- Myers, E. W. (1995). 'Toward simplifying and accurately formulating fragment assembly'. *Journal of Computational Biology*, 2(2), pp. 275–290 (cited on p. 6).
- Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J., Kravitz, S. A., Mobarry, C. M., Reinert, K. H., Remington, K. A. et al. (2000). 'A whole-genome assembly of *Drosophila*'. *Science*, 287(5461), pp. 2196–2204 (cited on p. 6).
- Myers, E. W. (2005). 'The fragment assembly string graph'. *Bioinformatics*, 21(suppl_2), pp. ii79–ii85. DOI: [10.1093/bioinformatics/bti1114](https://doi.org/10.1093/bioinformatics/bti1114) (cited on pp. 32, 37).
- Naya, H., Romero, H., Zavala, A., Alvarez, B. and Musto, H. (2002). 'Aerobiosis increases the genomic guanine plus cytosine content (GC%) in prokaryotes'. *Journal of molecular evolution*, 55(3), pp. 260–264 (cited on p. 13).
- Ni, J., Wu, G. D., Albenberg, L. and Tomov, V. T. (2017). 'Gut microbiota and IBD: causation or correlation?' *Nature Reviews Gastroenterology & Hepatology*, 14(10), pp. 573–584. DOI: [10.1038/nrgastro.2017.88](https://doi.org/10.1038/nrgastro.2017.88) (cited on p. 69).
- Nielsen, H. B., Almeida, M., Juncker, A. S., Rasmussen, S., Li, J. et al. (2014). 'Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes'. *Nature Biotechnology*, 32, pp. 822–828.  (cited on p. 9).
- Nissen, J. N. et al. (2021). 'Improved metagenome binning and assembly using deep variational autoencoders'. *Nature Biotechnology*. DOI: [10.1038/s41587-020-00777-4](https://doi.org/10.1038/s41587-020-00777-4) (cited on pp. 10, 21, 54, 57, 58, 60, 61).
- Nurk, S., Meleshko, D., Korobeynikov, A. and Pevzner, P. A. (2017). 'metaSPAdes: a new versatile metagenomic assembler'. *Genome Research*, 27(5), pp. 824–834. DOI: [10.1101/gr.213959.116](https://doi.org/10.1101/gr.213959.116) (cited on pp. 7, 12, 16, 20, 22, 32, 38, 40, 41, 52, 57, 67).
- Ounit, R., Wanamaker, S., Close, T. J. and Lonardi, S. (2015). 'CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers'. *BMC Genomics*, 16(1), p. 236. DOI: [10.1186/s12864-015-1419-2](https://doi.org/10.1186/s12864-015-1419-2) (cited on p. 3).
- Overmann, J., Abt, B. and Sikorski, J. (2017). 'Present and Future of Culturing Bacteria'. *Annual Review of Microbiology*, 71(1). PMID: 28731846, pp. 711–730. DOI: [10.1146/annurev-micro-090816-093449](https://doi.org/10.1146/annurev-micro-090816-093449) (cited on p. 1).
- Parks, D. H., Imelfort, M., Skennerton, C. T., Hugenholtz, P. and Tyson, G. W. (2015). 'CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes'. *Genome Research*, 25(7), pp. 1043–1055. DOI: [10.1101/gr.186072.114](https://doi.org/10.1101/gr.186072.114) (cited on pp. 52, 57, 58, 61).
- 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), pp. 9748–9753. DOI: [10.1073/pnas.171285098](https://doi.org/10.1073/pnas.171285098) (cited on pp. 7, 32, 37).

- Pollard, M. O., Gurdasani, D., Mentzer, A. J. et al. (2018). 'Long reads: their purpose and place'. *Human Molecular Genetics*, 27(R2), R234–R241. DOI: [10.1093/hmg/ddy177](https://doi.org/10.1093/hmg/ddy177) (cited on p. 5).
- Pop, M. (2009). 'Genome assembly reborn: recent computational challenges'. *Briefings in Bioinformatics*, 10(4), pp. 354–366. DOI: [10.1093/bib/bbp026](https://doi.org/10.1093/bib/bbp026) (cited on pp. 2, 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, p. 1346. DOI: [10.3389/fmicb.2017.01346](https://doi.org/10.3389/fmicb.2017.01346) (cited on p. 2).
- Quince, C., Walker, A. W., Simpson, J. T., Loman, N. J. and Segata, N. (2017). 'Shotgun metagenomics, from sampling to analysis'. *Nature Biotechnology*, 35(9), pp. 833–844. DOI: [10.1038/nbt.3935](https://doi.org/10.1038/nbt.3935) (cited on p. 2).
- Reuter, J. A., Spacek, D. V. and Snyder, M. P. (2015). 'High-Throughput Sequencing Technologies'. *Molecular Cell*, 58(4), pp. 586–597. DOI: [10.1016/j.molcell.2015.05.004](https://doi.org/10.1016/j.molcell.2015.05.004) (cited on p. 5).
- Rho, M., Tang, H. and Ye, Y. (2010). 'FragGeneScan: predicting genes in short and error-prone reads'. *Nucleic Acids Research*, 38(20), e191–e191. DOI: [10.1093/nar/gkq747](https://doi.org/10.1093/nar/gkq747) (cited on p. 52).
- Riesenfeld, C. S., Schloss, P. D. and Handelsman, J. (2004). 'Metagenomics: Genomic Analysis of Microbial Communities'. *Annual Review of Genetics*, 38(1). PMID: 15568985, pp. 525–552. DOI: [10.1146/annurev.genet.38.072902.091216](https://doi.org/10.1146/annurev.genet.38.072902.091216) (cited on pp. 1, 12).
- Saeed, I., Tang, S.-L. and Halgamuge, S. K. (2011). 'Unsupervised discovery of microbial population structure within metagenomes using nucleotide base composition'. *Nucleic Acids Research*, 40(5), e34–e34. DOI: [10.1093/nar/gkr1204](https://doi.org/10.1093/nar/gkr1204) (cited on p. 9).
- Sanger, F., Nicklen, S. and Coulson, A. R. (1977). 'DNA sequencing with chain-terminating inhibitors'. *Proceedings of the National Academy of Sciences*, 74(12), pp. 5463–5467. DOI: [10.1073/pnas.74.12.5463](https://doi.org/10.1073/pnas.74.12.5463) (cited on p. 5).
- Sangwan, N., Xia, F. and Gilbert, J. A. (2016). 'Recovering complete and draft population genomes from metagenome datasets'. *Microbiome*, 4(1), p. 8. DOI: [10.1186/s40168-016-0154-5](https://doi.org/10.1186/s40168-016-0154-5) (cited on pp. 8, 9).
- Schadt, E. E., Turner, S. and Kasarskis, A. (2010). 'A window into third-generation sequencing'. *Human molecular genetics*, 19(R2), R227–R240 (cited on p. 5).
- Schaeffer, L., Pimentel, H., Bray, N., Melsted, P. and Pachter, L. (2017). 'Pseudoalignment for metagenomic read assignment'. *Bioinformatics*, 33(14), pp. 2082–2088. DOI: [10.1093/bioinformatics/btx106](https://doi.org/10.1093/bioinformatics/btx106) (cited on p. 3).
- Schatz, M. C., Delcher, A. L. and Salzberg, S. L. (2010). 'Assembly of large genomes using second-generation sequencing'. *Genome research*, 20(9), pp. 1165–1173 (cited on p. 6).
- Schoch, C. L., Ciufo, S., Domrachev, M., Hotton, C. L., Kannan, S., Khovanskaya, R., Leipe, D., Mcveigh, R., O'Neill, K., Robbertse, B. et al. (2020). 'NCBI Taxonomy: a comprehensive update on curation, resources and tools'. *Database*, 2020 (cited on pp. 63, 64).
- Sczyrba, A. et al. (2017). 'Critical Assessment of Metagenome Interpretation—a benchmark of metagenomics software'. *Nature Methods*, 14, pp. 1063–1071.  (cited on pp. 3, 7, 11, 12, 16, 20, 25).
- 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, pp. 48–55. DOI: <https://doi.org/10.1016/j.csbj.2016.11.005> (cited on pp. 3, 7, 9).
- 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), pp. 811–814. DOI: [10.1038/nmeth.2066](https://doi.org/10.1038/nmeth.2066) (cited on p. 7).
- Sharon, I., Morowitz, M. J., Thomas, B. C., Costello, E. K., Relman, D. A. and Banfield, J. F. (2013). 'Time series community genomics analysis reveals rapid shifts in bacterial species, strains, and phage during infant gut colonization'. *Genome Research*, 23(1), pp. 111–120. DOI: [10.1101/gr.142315.112](https://doi.org/10.1101/gr.142315.112) (cited on pp. 19, 20, 25, 37, 39, 57, 61, 62).

- Sharpton, T. J. (2014). 'An introduction to the analysis of shotgun metagenomic data'. *Frontiers in Plant Science*, 5, p. 209. DOI: [10.3389/fpls.2014.00209](https://doi.org/10.3389/fpls.2014.00209) (cited on p. 2).
- Sieber, C. M., Probst, A. J., Sharrar, A. et al. (2018). 'Recovery of genomes from metagenomes via a dereplication, aggregation and scoring strategy'. *Nature microbiology*, 3(7), pp. 836–843 (cited on p. 9).
- Simpson, J. T. and Durbin, R. (2012). 'Efficient de novo assembly of large genomes using compressed data structures'. *Genome Research*, 22(3), pp. 549–556. DOI: [10.1101/gr.126953.111](https://doi.org/10.1101/gr.126953.111) (cited on pp. 7, 16, 20, 22, 32, 38, 41, 42, 67).
- Sohn, J.-i. and Nam, J.-W. (2016). 'The present and future of de novo whole-genome assembly'. *Briefings in Bioinformatics*, 19(1), pp. 23–40. DOI: [10.1093/bib/bbw096](https://doi.org/10.1093/bib/bbw096) (cited on p. 7).
- Staley, J. T. and Konopka, A. (1985). 'Measurement of in situ activities of nonphotosynthetic microorganisms in aquatic and terrestrial habitats'. *Annual review of microbiology*, 39(1), pp. 321–346 (cited on p. 2).
- Steen, A. D., Crits-Christoph, A., Carini, P., DeAngelis, K. M., Fierer, N., Lloyd, K. G. and Cameron Thrash, J. (2019). 'High proportions of bacteria and archaea across most biomes remain uncultured'. *The ISME Journal*, 13(12), pp. 3126–3130. DOI: [10.1038/s41396-019-0484-y](https://doi.org/10.1038/s41396-019-0484-y) (cited on p. 1).
- Stöcker, B. K., Köster, J. and Rahmann, S. (2016). 'SimLoRD: Simulation of Long Read Data'. *Bioinformatics*, 32(17), pp. 2704–2706 (cited on p. 37).
- 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, p. 410. DOI: [10.3389/fmicb.2012.00410](https://doi.org/10.3389/fmicb.2012.00410) (cited on pp. 10, 16, 21, 22).
- Su, C., Lei, L., Duan, Y., Zhang, K.-Q. and Yang, J. (2012). 'Culture-independent methods for studying environmental microorganisms: methods, application, and perspective'. *Applied Microbiology and Biotechnology*, 93(3), pp. 993–1003. DOI: [10.1007/s00253-011-3800-7](https://doi.org/10.1007/s00253-011-3800-7) (cited on p. 1).
- 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), pp. 2298–2299. DOI: [10.1093/bioinformatics/btr376](https://doi.org/10.1093/bioinformatics/btr376) (cited on p. 7).
- Sun, Z. et al. (2021). 'Challenges in benchmarking metagenomic profilers'. *Nature Methods*, 18(6), pp. 618–626. DOI: [10.1038/s41592-021-01141-3](https://doi.org/10.1038/s41592-021-01141-3) (cited on p. 69).
- 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), pp. 9–19 (cited on p. 6).
- The Human Microbiome Project Consortium (2012). 'Structure, function and diversity of the healthy human microbiome'. *Nature*, 486, pp. 207–214.  (cited on p. 5).
- Thomas, T., Gilbert, J. and Meyer, F. (2012). 'Metagenomics - a guide from sampling to data analysis'. *Microbial Informatics and Experimentation*, 2(1), p. 3. DOI: [10.1186/2042-5783-2-3](https://doi.org/10.1186/2042-5783-2-3) (cited on pp. 1, 2).
- Utsch, A. and Mörfen, F. (2005). *ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM*. Tech. rep. Databionics Research Group, University of Marburg, Germany (cited on p. 9).
- Uritskiy, G. V., DiRuggiero, J. and Taylor, J. (2018). 'MetaWRAP—a flexible pipeline for genome-resolved metagenomic data analysis'. *Microbiome*, 6(1), pp. 1–13 (cited on p. 9).
- Vollmers, J., Wiegand, S. and Kaster, A.-K. (2017). 'Comparing and Evaluating Metagenome Assembly Tools from a Microbiologist's Perspective - Not Only Size Matters!' *PLOS ONE*, 12(1), pp. 1–31. DOI: [10.1371/journal.pone.0169662](https://doi.org/10.1371/journal.pone.0169662) (cited on p. 16).
- Wade, W. (2002). 'Unculturable bacteria—the uncharacterized organisms that cause oral infections'. *Journal of the Royal Society of Medicine*, 95(2), pp. 81–83 (cited on p. 1).
- Wang, J., Mawet, D., Ruane, G., Delorme, J.-r. and Klimovich, N. (2018). *Baseline Requirements For Detecting Biosignatures with the HabEx and LUVOIR Mission Concepts*. Tech. rep., pp. 1–25. DOI: [10.1117/12.2275222](https://doi.org/10.1117/12.2275222) (cited on p. 3).

- Wang, J., Jiang, Y., Yu, G., Zhang, H. and Luo, H. (2018). 'BMC3C: binning metagenomic contigs using codon usage, sequence composition and read coverage'. *Bioinformatics*, 34(24), pp. 4172–4179. DOI: [10.1093/bioinformatics/bty519](https://doi.org/10.1093/bioinformatics/bty519) (cited on pp. 9, 10).
- Wang, Y., Wang, K., Lu, Y. Y. and Sun, F. (2017). 'Improving contig binning of metagenomic data using d2S oligonucleotide frequency dissimilarity'. *BMC Bioinformatics*, 18(1), p. 425. DOI: [10.1186/s12859-017-1835-1](https://doi.org/10.1186/s12859-017-1835-1) (cited on pp. 9, 21).
- Wang, Z., Wang, Z. et al. (2019). 'SolidBin: improving metagenome binning with semi-supervised normalized cut'. *Bioinformatics*, 35(21), pp. 4229–4238. DOI: [10.1093/bioinformatics/btz253](https://doi.org/10.1093/bioinformatics/btz253) (cited on pp. 10, 12, 16, 21, 22, 32, 38, 41–44, 54).
- Wickramarachchi, A., Mallawaarachchi, V., Rajan, V. and Lin, Y. (2020). 'MetaBCC-LR: metagenomics binning by coverage and composition for long reads'. *Bioinformatics*, 36(Supplement_1), pp. i3–i11. DOI: [10.1093/bioinformatics/btaa441](https://doi.org/10.1093/bioinformatics/btaa441) (cited on pp. 37, 39).
- Wood, D. E., Lu, J. and Langmead, B. (2019). 'Improved metagenomic analysis with Kraken 2'. *Genome Biology*, 20(1), p. 257. DOI: [10.1186/s13059-019-1891-0](https://doi.org/10.1186/s13059-019-1891-0) (cited on p. 7).
- Wood, D. E. and Salzberg, S. L. (2014). 'Kraken: ultrafast metagenomic sequence classification using exact alignments'. *Genome Biology*, 15(3), R46. DOI: [10.1186/gb-2014-15-3-r46](https://doi.org/10.1186/gb-2014-15-3-r46) (cited on p. 7).
- Wooley, J. C. and Ye, Y. (2010). 'Metagenomics: facts and artifacts, and computational challenges'. *Journal of computer science and technology*, 25(1), pp. 71–81 (cited on p. 2).
- Wu, Y.-W., Simmons, B. A. and Singer, S. W. (2015). 'MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets'. *Bioinformatics*, 32(4), pp. 605–607. DOI: [10.1093/bioinformatics/btv638](https://doi.org/10.1093/bioinformatics/btv638) (cited on pp. 9, 10, 12, 14, 16, 21, 22, 32, 37, 38, 40, 42–44, 52, 54, 57, 58, 60, 61).
- 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), p. 26. DOI: [10.1186/2049-2618-2-26](https://doi.org/10.1186/2049-2618-2-26) (cited on pp. 9, 35, 52, 54, 57, 58).
- Wu, Y.-W. and Ye, Y. (2011). 'A Novel Abundance-Based Algorithm for Binning Metagenomic Sequences Using l-tuples'. *Journal of Computational Biology*, 18(3). PMID: 21385052, pp. 523–534. DOI: [10.1089/cmb.2010.0245](https://doi.org/10.1089/cmb.2010.0245) (cited on pp. 9, 54).
- Xue, H., Mallawaarachchi, V., Zhang, Y., Rajan, V. and Lin, Y. (2022). *RepBin: Constraint-based Graph Representation Learning for Metagenomic Binning*.  (cited on pp. i, 69).
- Yang, C., Chowdhury, D., Zhang, Z., Cheung, W. K., Lu, A., Bian, Z. and Zhang, L. (2021). 'A review of computational tools for generating metagenome-assembled genomes from metagenomic sequencing data'. *Computational and Structural Biotechnology Journal*, 19, pp. 6301–6314. DOI: <https://doi.org/10.1016/j.csbj.2021.11.028> (cited on pp. 7, 11).
- Ye, S. H., Siddle, K. J., Park, D. J. and Sabeti, P. C. (2019). 'Benchmarking Metagenomics Tools for Taxonomic Classification'. *Cell*, 178(4), pp. 779–794. DOI: [10.1016/j.cell.2019.07.010](https://doi.org/10.1016/j.cell.2019.07.010) (cited on p. 69).
- Yue, Y., Huang, H., Qi, Z., Dou, H.-M., Liu, X.-Y., Han, T.-F., Chen, Y., Song, X.-J., Zhang, Y.-H. and Tu, J. (2020). 'Evaluating metagenomics tools for genome binning with real metagenomic datasets and CAMI datasets'. *BMC Bioinformatics*, 21(1), p. 334. DOI: [10.1186/s12859-020-03667-3](https://doi.org/10.1186/s12859-020-03667-3) (cited on p. 11).
- Zerbino, D. R. and Birney, E. (2008). 'Velvet: algorithms for de novo short read assembly using de Bruijn graphs'. *Genome research*, 18(5), pp. 821–829 (cited on p. 7).
- Zhu, X. and Ghahramani, Z. (2002). *Learning from Labeled and Unlabeled Data with Label Propagation*. Tech. rep. School of Computer Science, Carnegie Mellon University (cited on p. 18).

Appendix A

GraphBin Results

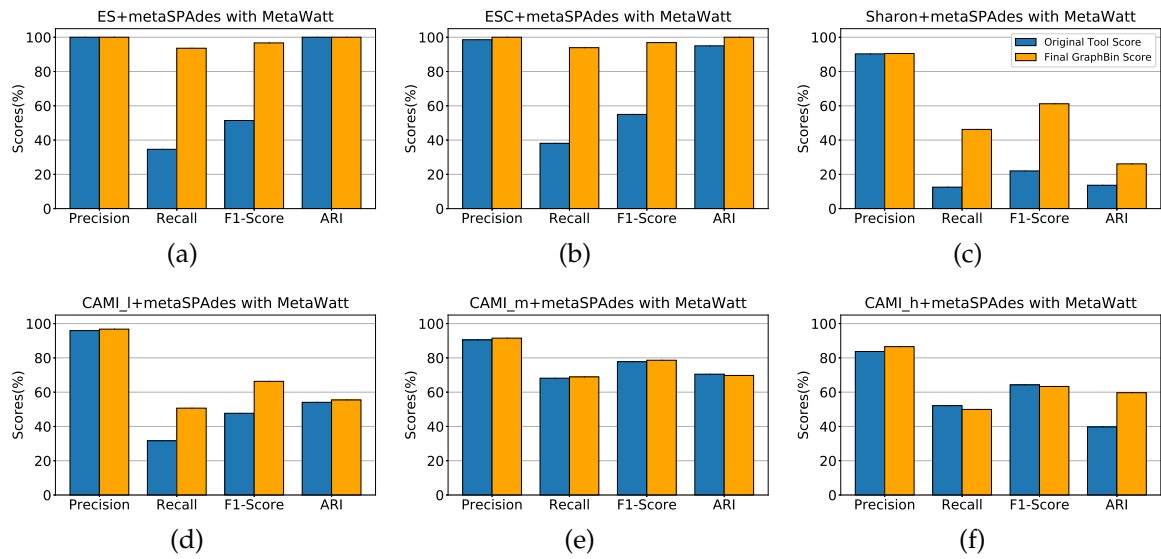


Figure A.1: Binning results of MetaWatt with GraphBin for the datasets ES+metaSPAdes, ESC+metaSPAdes, Sharon+metaSPAdes, CAMI_I+metaSPAdes, CAMI_m+metaSPAdes and CAMI_h+metaSPAdes. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

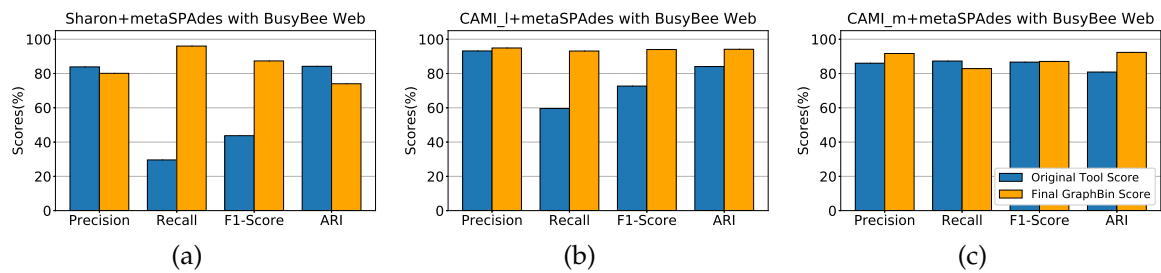


Figure A.2: Binning results of BusyBee Web with GraphBin for the datasets Sharon+metaSPAdes, CAMI_I+metaSPAdes and CAMI_m+metaSPAdes. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin. BusyBee Web was unable to bin the datasets ES+metaSPAdes and ESC+metaSPAdes as there were not enough sequences that had the minimum contig length of 500 bp. CAMI_h+metaSPAdes dataset could not be binned using BusyBee Web due to the restriction on input file size.

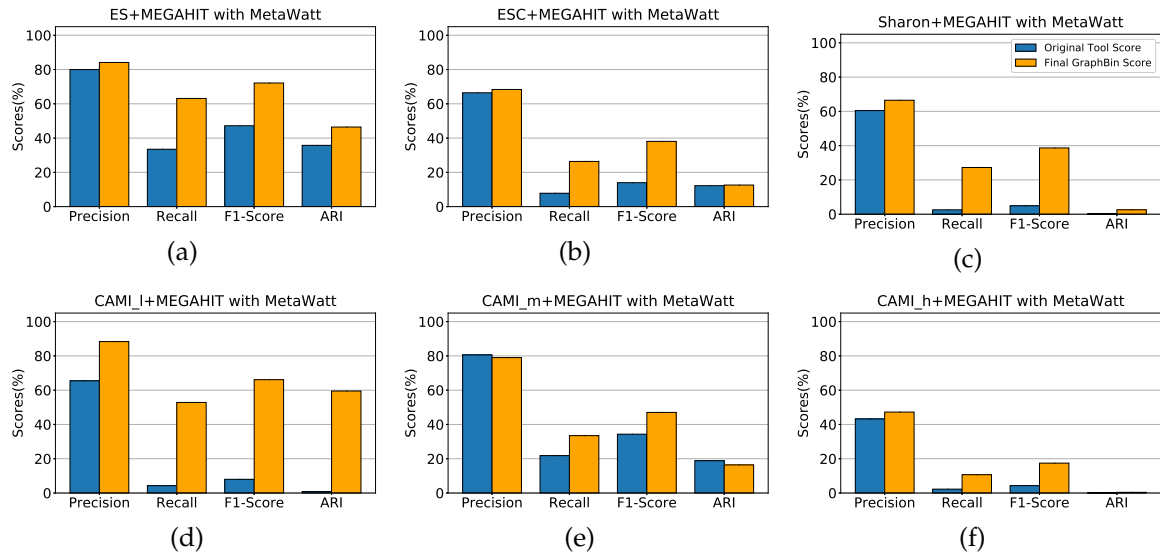


Figure A.3: Binning results of MetaWatt with GraphBin for the datasets ES+MEGAHIT, ESC+MEGAHIT, Sharon+MEGAHIT, CAMI_l+MEGAHIT, CAMI_m+MEGAHIT and CAMI_h+MEGAHIT. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

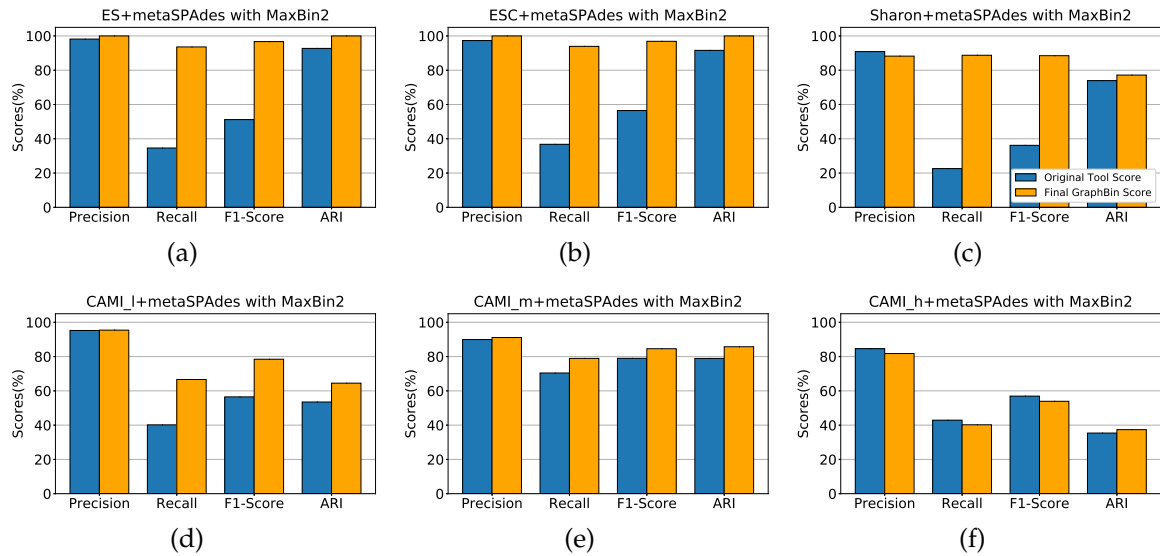


Figure A.4: Binning results of MaxBin2 with GraphBin for the datasets ES+metaSPAdes, ESC+metaSPAdes, Sharon+metaSPAdes, CAMI_l+metaSPAdes, CAMI_m+metaSPAdes and CAMI_h+metaSPAdes. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

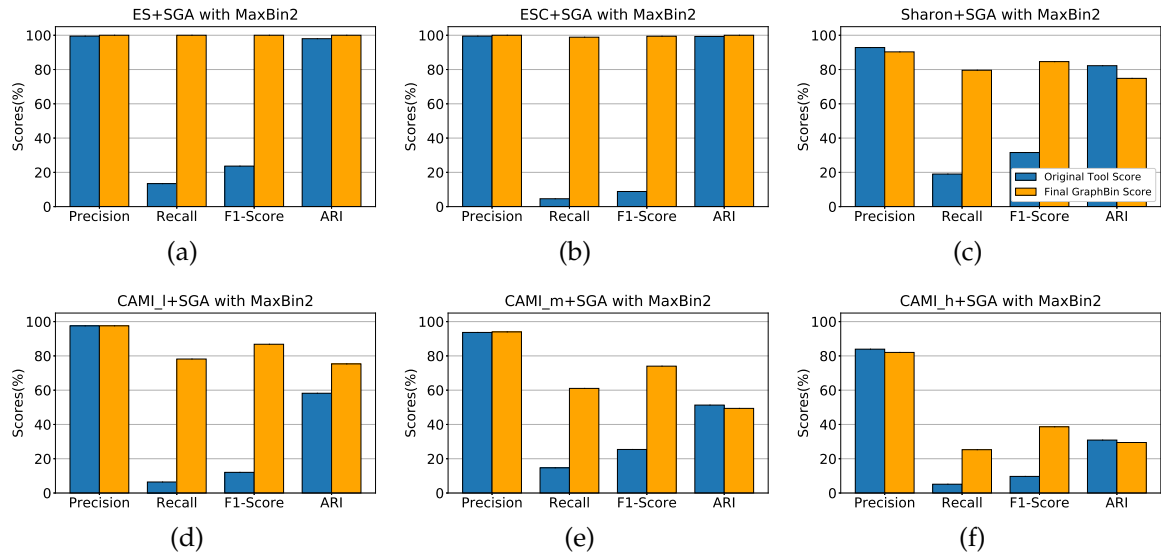


Figure A.5: Binning results of MaxBin2 with GraphBin for the datasets ES+SGA, ESC+SGA, Sharon+SGA, CAMI_l+SGA, CAMI_m+SGA and CAMI_h+SGA. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

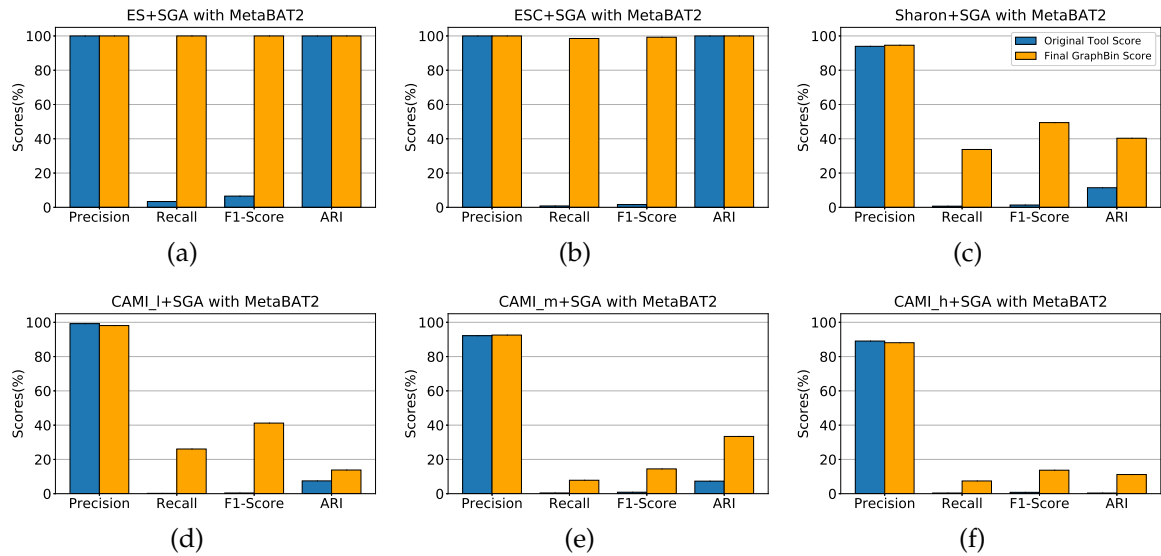


Figure A.6: Binning results of MetaBAT2 with GraphBin for the datasets ES+SGA, ESC+SGA, Sharon+SGA, CAMI_l+SGA, CAMI_m+SGA and CAMI_h+SGA. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

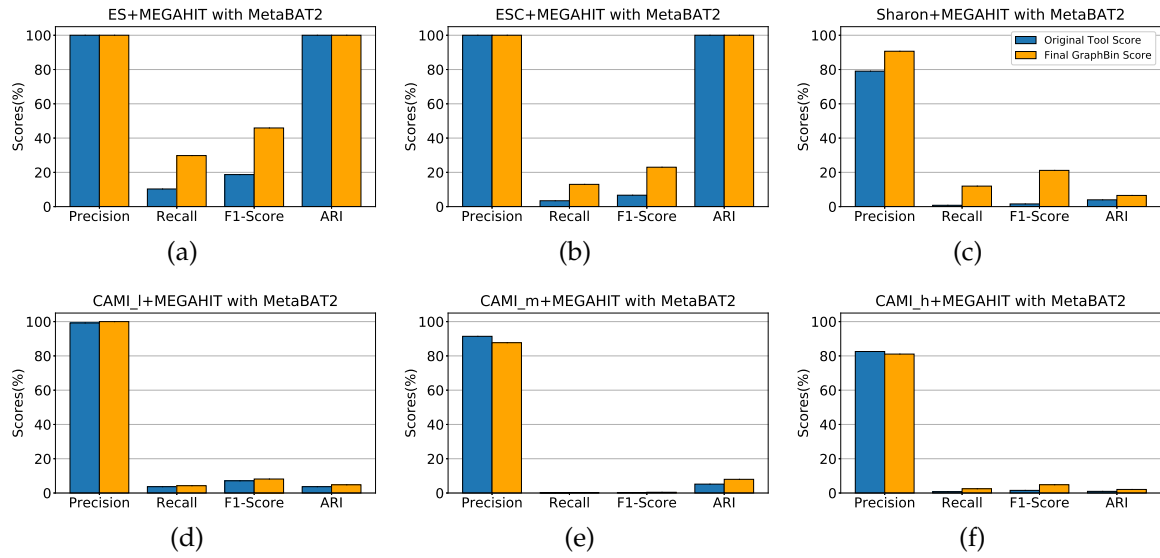


Figure A.7: Binning results of MetaBAT2 with GraphBin for the datasets ES+MEGAHIT, ESC+MEGAHIT, Sharon+MEGAHIT, CAMI_l+MEGAHIT, CAMI_m+MEGAHIT and CAMI_h+MEGAHIT. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

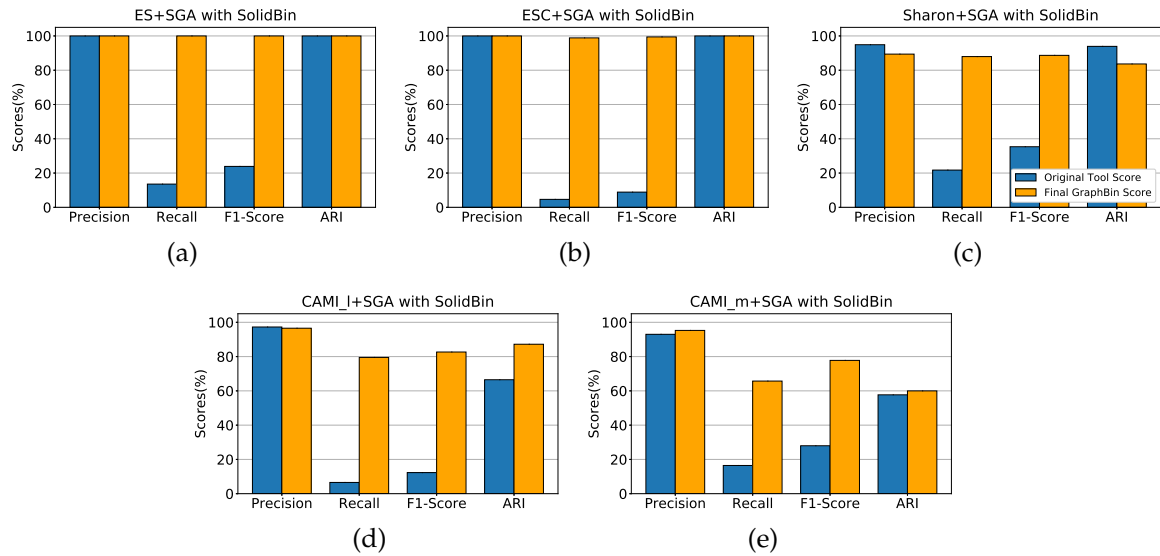


Figure A.8: Binning results of SolidBin with GraphBin for the datasets ES+SGA, ESC+SGA, Sharon+SGA, CAMI_l+SGA and CAMI_m+SGA. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin. SolidBin did not complete binning the CAMI_h+SGA dataset after 72 hours.

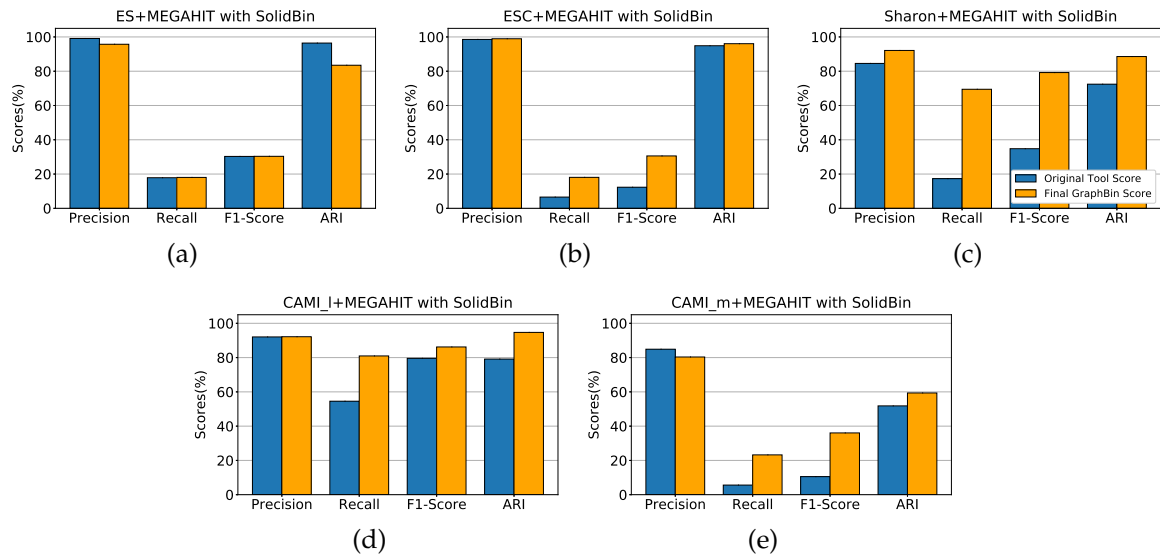


Figure A.9: Binning results of SolidBin with GraphBin for the datasets ES+MEGAHIT, ESC+MEGAHIT, Sharon+MEGAHIT, CAMI_l+MEGAHIT and CAMI_m+MEGAHIT. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin. CAMI_h+MEGAHIT dataset could not be binned using SolidBin due to insufficient memory (576GB).

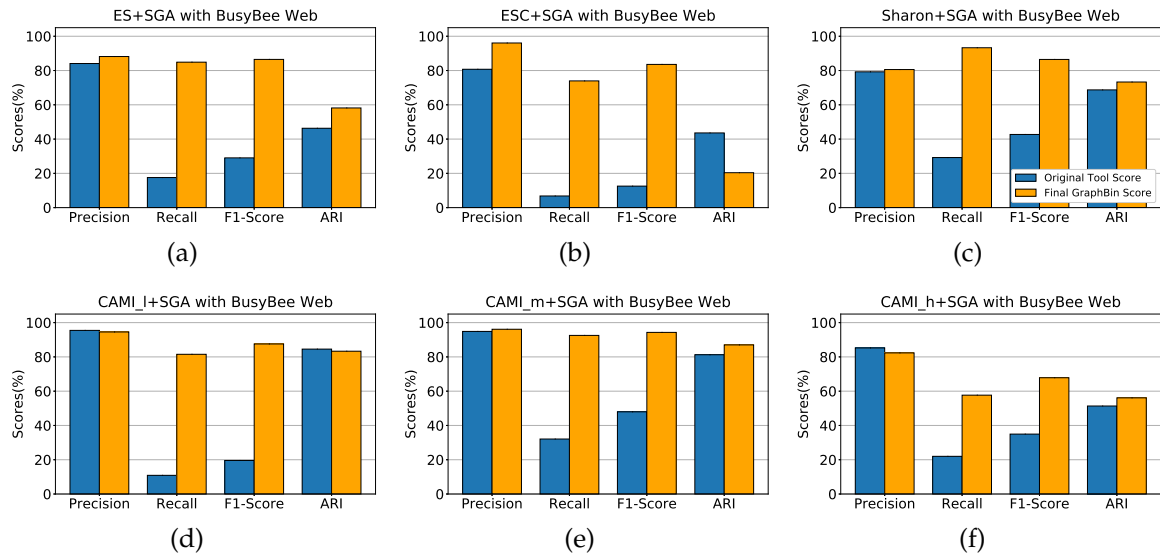


Figure A.10: Binning results of BusyBee Web with GraphBin for the datasets ES+SGA, ESC+SGA, Sharon+SGA, CAMI_l+SGA, CAMI_m+SGA and CAMI_h+SGA. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool compared with the scores obtained after applying GraphBin. The dark blue bars denote the original tool scores and the light orange bars denote the final scores of GraphBin.

Appendix B

Datasets

B.1 Datasets used in experiments of GraphBin

The data sets used for the experiments in GraphBin are as follows.

- **ES+metaSPAdes** - metaSPAdes assembly of the ES dataset
- **ES+SGA** - SGA assembly of the ES dataset
- **ES+MEGAHIT** - MEGAHIT assembly of the ES dataset
- **ESC+metaSPAdes** - metaSPAdes assembly of the ESC dataset
- **ESC+SGA** - SGA assembly of the ESC dataset
- **ESC+MEGAHIT** - MEGAHIT assembly of the ESC dataset
- **Sharon+metaSPAdes** - metaSPAdes assembly of the Sharon dataset
- **Sharon+SGA** - SGA assembly of the Sharon dataset
- **Sharon+MEGAHIT** - MEGAHIT assembly of the Sharon dataset
- **CAMI_l+metaSPAdes** - metaSPAdes assembly of the CAMI_l dataset
- **CAMI_l+SGA** - SGA assembly of the CAMI_l dataset
- **CAMI_l+MEGAHIT** - MEGAHIT assembly of the CAMI_l dataset
- **CAMI_m+metaSPAdes** - metaSPAdes assembly of the CAMI_m dataset
- **CAMI_m+SGA** - SGA assembly of the CAMI_m dataset
- **CAMI_m+MEGAHIT** - MEGAHIT assembly of the CAMI_m dataset
- **CAMI_h+metaSPAdes** - metaSPAdes assembly of the CAMI_h dataset
- **CAMI_h+SGA** - SGA assembly of the CAMI_h dataset
- **CAMI_h+MEGAHIT** - MEGAHIT assembly of the CAMI_h dataset

The details of the resulting data sets such as read length, assembly size, number of contigs, mean contig length, median contig length, number of edges in the assembly graph and the number of species identified can be found in Table [B.1](#).

To keep the experiments simple and less time consuming, we removed all the contigs which were shorter than 500 bp from the **CAMI_m+metaSPAdes** assembly and removed all the contigs which were shorter than 1,000 bp from the **CAMI_h+metaSPAdes** assembly.

Table B.1: Information on the datasets used for experiments of GraphBin

Data Set	Read length (bp)	Assembly size (Mb)	Total number of assembled contigs	Mean contig length (bp)	Median contig length (bp)	Number of edges in the assembly graph	Number of species found as ground truth by TAXAassign
ES+metaSPAdes	300	5.94	159	37,328	223	741	2
ES+SGA	300	6.37	1,753	3,633	347	2,330	2
ES+MEGAHIT	300	7.55	5,921	1,342	302	508	2
ESC+metaSPAdes	300	8.48	189	44,870	289	879	3
ESC+SGA	300	12.06	12,177	990	302	6,247	3
ESC+MEGAHIT	300	12.86	14,617	879	302	3,828	3
Sharon+metaSPAdes	100	46.08	42,313	1,089	230	102,918	26
Sharon+SGA	100	30.04	34,621	867	428	9,792	19
Sharon+MEGAHIT	100	42.32	19,401	2,181	409	17,232	24
CAML_l+metaSPAdes	150	107.42	47,707	2,251	398	23,257	25
CAML_l+SGA	150	126.43	221,379	571	221	487,322	27
CAML_l+MEGAHIT	150	121.29	67,287	1,802	473	16,398	13
CAML_m+metaSPAdes	150	174.36	90,347	1,929	871	13,586	25
CAML_m+SGA	150	87.33	28,753	3,037	481	13,394	25
CAML_m+MEGAHIT	150	310.76	535,313	580	380	285,588	128
CAML_h+metaSPAdes	150	700.81	242,341	2,891	1,572	89,161	143
CAML_h+SGA	150	424.54	448,673	946	467	46,042	130
CAML_h+MEGAHIT	150	1,374.52	1,278,070	1,075	585	62,642	162

Note that the metaSPAdes assemblies were obtained using SPAdes version 3.13.0 in the meta mode with the default parameters.

B.2 Datasets used in experiments of GraphBin2

Table B.2: Information on the datasets used for experiments of GraphBin2

Dataset	Assembler	Read length (bp)	Number of paired end reads	Total number of non-isolated contigs	Mean contig length (bp)	Number of species in ground truth
Sim-5G	metaSPAdes	300	2,000,000	516	51,723	5
	SGA	300	2,000,000	18,192	1,675	5
Sim-10G	metaSPAdes	300	6,999,998	900	47,279	10
	SGA	300	6,999,998	32,389	1,300	10
Sim-20G	metaSPAdes	300	15,000,001	1,404	48,021	20
	SGA	300	15,000,001	72,791	873	20
Sharon-1	metaSPAdes	100	14,869,863	371	17,144	12
	SGA	100	14,869,863	766	3,034	12
Sharon-All	metaSPAdes	100	135,493,567	2,730	7,689	12
	SGA	100	135,493,567	20,942	1,547	12
50G-SR	metaSPAdes	300	20,730,313	4,159	37,027	50
Lake Water	metaSPAdes	300	4,627,091	96,880	1,020	57
100G-LR	metaFlye	8,000	3,754,639	958	2,538	100

Note that the metaSPAdes assemblies were obtained using SPAdes version 3.13.0 with the default parameters.

Table B.3: Species details of the simulated short-read datasets Sim-5, Sim-10 and Sim-20 used in the experiments of GraphBin2

Dataset	Species present	Genome size	Coverage	Abundance
Sim-5G	<i>Acetobacter pasteurianus</i>	2.9 Mb	115×	28%
	<i>Aeromonas veronii</i>	4.6 Mb	72×	28%
	<i>Amycolatopsis mediterranei</i>	10.4 Mb	26×	22%
	<i>Arthrobacter arilaitensis</i>	3.9 Mb	41×	13%
	<i>Azorhizobium caulinodans</i>	5.4 Mb	20×	9%
Sim-10G	<i>Acetobacter pasteurianus</i>	2.9 Mb	357×	25%
	<i>Aeromonas veronii</i>	4.6 Mb	225×	25%
	<i>Amycolatopsis mediterranei</i>	10.4 Mb	80×	20%
	<i>Arthrobacter arilaitensis</i>	3.9 Mb	128×	12%
	<i>Azorhizobium caulinodans</i>	5.4 Mb	62×	8%
	<i>Bacillus cereus</i>	5.3 Mb	58×	7%
	<i>Bdellovibrio bacteriovorus</i>	3.8 Mb	11×	1%
	<i>Bifidobacterium adolescentis</i>	2.1 Mb	20×	1%
	<i>Brachyspira intermedia</i>	3.4 Mb	11×	1%
	<i>Campylobacter jejuni</i>	1.7 Mb	21×	1%
Sim-20G	<i>Acetobacter pasteurianus</i>	2.9 Mb	705×	23%
	<i>Aeromonas veronii</i>	4.6 Mb	445×	23%
	<i>Amycolatopsis mediterranei</i>	10.4 Mb	157×	18%
	<i>Arthrobacter arilaitensis</i>	3.9 Mb	253×	11%
	<i>Azorhizobium caulinodans</i>	5.4 Mb	123×	7%
	<i>Bacillus cereus</i>	5.3 Mb	114×	7%
	<i>Bdellovibrio bacteriovorus</i>	3.8 Mb	22×	1%
	<i>Bifidobacterium adolescentis</i>	2.1 Mb	40×	1%
	<i>Brachyspira intermedia</i>	3.4 Mb	21×	1%
	<i>Campylobacter jejuni</i>	1.7 Mb	41×	1%
	<i>Candidatus Pelagibacter ubique</i>	1.3 Mb	54×	1%
	<i>Chlamydia trachomatis</i>	1.1 Mb	64×	1%
	<i>Clostridium acetobutylicum</i>	4.0 Mb	18×	1%
	<i>Corynebacterium diphtheriae</i>	2.5 Mb	28×	1%
	<i>Cyanobacterium UCYN</i>	1.5 Mb	47×	1%
	<i>Desulfovibrio vulgaris</i>	3.6 Mb	20×	1%
	<i>Ehrlichia ruminantium</i>	1.5 Mb	47×	1%
	<i>Enterococcus faecium</i>	3.0 Mb	24×	1%
	<i>Erysipelothrix rhusiopathiae</i>	1.8 Mb	39×	1%
	<i>Escherichia coli</i>	5.0 Mb	14×	1%

Table B.4: Species details of the 50G-SR dataset used in the experiments of GraphBin2

Species present	Genome size	Coverage	Abundance
<i>Acetobacter pasteurianus</i>	2.9 Mb	773×	4%
<i>Aeromonas veronii</i>	4.6 Mb	493×	3%
<i>Amycolatopsis mediterranei</i>	10.4 Mb	175×	2%
<i>Arthrobacter arilaitensis</i>	3.9 Mb	281×	2%
<i>Azorhizobium caulinodans</i>	5.4 Mb	136×	2%
<i>Bacillus cereus</i>	5.3 Mb	126×	2%
<i>Bacillus thuringiensis</i>	5.4 Mb	35×	2%
<i>Bdellovibrio bacteriovorus</i>	3.8 Mb	25×	2%
<i>Bifidobacterium adolescentis</i>	2.1 Mb	44×	2%
<i>Bifidobacterium animalis</i>	2.0 Mb	48×	2%
<i>Brachyspira intermedia</i>	3.4 Mb	23×	2%
<i>Campylobacter jejuni</i>	1.7 Mb	47×	2%
<i>Candidatus Pelagibacter ubique</i>	1.3 Mb	59×	2%
<i>Candidatus Phytoplasma mali</i>	0.6 Mb	129×	2%
<i>Candidatus Sulcia muelleri</i>	0.3 Mb	279×	2%
<i>Chlamydia psittaci</i>	1.2 Mb	66×	2%
<i>Chlamydia trachomatis</i>	1.1 Mb	74×	2%
<i>Clostridium acetobutylicum</i>	4.0 Mb	20×	2%
<i>Clostridium botulinum</i>	2.8 Mb	28×	2%
<i>Clostridium tetani</i>	2.8 Mb	28×	2%
<i>Clostridium thermocellum</i>	3.9 Mb	20×	2%
<i>Corynebacterium diphtheriae</i>	2.5 Mb	31×	2%
<i>Corynebacterium pseudotuberculosis</i>	2.4 Mb	33×	2%
<i>Corynebacterium ulcerans</i>	2.5 Mb	31×	2%
<i>Cyanobacterium UCYN</i>	1.5 Mb	54×	2%
<i>Cyanothece</i> sp	6.2 Mb	13×	2%
<i>Desulfovibrio vulgaris</i>	3.6 Mb	22×	2%
<i>Ehrlichia ruminantium</i>	1.5 Mb	52×	2%
<i>Enterococcus faecium</i>	3.0 Mb	26×	2%
<i>Erysipelothrix rhusiopathiae</i>	1.8 Mb	43×	2%
<i>Escherichia coli</i>	5.0 Mb	16×	2%
<i>Continued to next page ...</i>			

Species present	Genome size	Coverage	Abundance
<i>Fervidicoccus fontis</i>	1.3 Mb	59×	2%
<i>Fibrobacter succinogenes</i>	3.9 Mb	20×	2%
<i>Flavobacterium branchiophilum</i>	3.6 Mb	22×	2%
<i>Francisella novicida</i>	1.9 Mb	41×	2%
<i>Francisella tularensis</i>	1.9 Mb	41×	2%
<i>Fusobacterium nucleatum</i>	2.2 Mb	36×	2%
<i>Gardnerella vaginalis</i>	1.8 Mb	45×	2%
<i>Granulicella tundricola</i>	4.4 Mb	18×	2%
<i>Haemophilus influenzae</i>	1.9 Mb	41×	2%
<i>Haemophilus parainfluenzae</i>	2.1 Mb	37×	2%
<i>Haemophilus somnus</i>	2.3 Mb	34×	2%
<i>Halobacterium</i> sp. NRC-1	2.0 Mb	38×	2%
<i>Halothiobacillus neapolitanus</i>	2.6 Mb	30×	2%
<i>Helicobacter pylori</i>	1.6 Mb	49×	2%
<i>Hyphomicrobium</i> sp. MC1	4.9 Mb	16×	2%
<i>Ignavibacterium album</i>	3.7 Mb	21×	2%
<i>Klebsiella oxytoc</i>	6.1 Mb	13×	2%
<i>Krokinobacter</i> sp	3.4 Mb	23×	2%
<i>Lactobacillus brevis</i>	2.3 Mb	34×	2%

B.3 Datasets used in experiments of MetaCoAG

Table B.5: Samples used for the real datasets in experiments of MetaCoAG

Dataset	BioProject number	NCBI accession numbers of the samples (runs) used
Sharon	PRJNA60717	SRR492197, SRR492196, SRR492195, SRR492194, SRR492193, SRR492192, SRR492191, SRR492190, SRR492189, SRR492188, SRR492187, SRR492186, SRR492185, SRR492184, SRR492183, SRR492182, SRR492066, SRR492065
COPD	PRJEB9034	ERR970477, ERR970476, ERR970475, ERR970474, ERR970473, ERR970472, ERR970471, ERR970470, ERR970407, ERR970406, ERR970405, ERR970404, ERR970403, ERR970402, ERR970401, ERR970400, ERR970399, ERR970398
Deep HMP TD	PRJNA48479	SRR1031078, SRR1031179, SRR1031181, SRR1031229, SRR1031267, SRR1031290, SRR1031684, SRR1031924

Table B.6: Information of the datasets used in experiments of MetaCoAG

Dataset	No. of samples	Read length (bp)	Assembly size (Mb)	Total no. of assembled contigs	No. of contigs longer than 1,000 bp	No. of edges in the assembly graph	Density [†] of the assembly graph	N50 (bp)
simHC+	1	301	314.23	15,729	6,706	31,199	1.984	154,030
CAMI UG	9	150	336.69	192,679	49,927	40,861	0.212	8,081
CAMI Skin	10	150	639.91	600,508	139,217	51,139	0.085	1,965
CAMI Oral	10	150	517.35	493,149	96,079	87,910	0.178	2,405
CAMI GI	10	150	507.09	255,722	75,913	39,240	0.153	9,319
CAMI Airways	10	150	664.49	729,063	142,477	49,531	0.067	1,506
Sharon	18	100	45.06	37,164	7,067	20,328	0.547	5,609
COPD	18	151	343.52	452,600	67,753	73,519	0.162	1,042
Deep HMP TD	8	101	158.01	227,635	28,164	86,250	0.379	1,123

Note that the assemblies were obtained using SPAdes version 3.15.2 with `-k 21,33,55,77,99,127`.

[†]Density of the graph is calculated as (the number of edges) / (the number of nodes).

Appendix C

simHC+ Dataset

C.1 Binning results of simHC+ Dataset

Table C.1: Information about the recovered species for the simHC+ dataset. "-" represents that the species was not recovered by the binning tool.

Species	NCBI accession number	Relative Abundance (%)	Coverage (×)	MaxBin2 F1-score (%)	MetaBAT2 F1-score (%)	MetaCoAG F1-score (%)
Acetobacter pasteurianus	AP011170.1	2.30	1533	96.92	-	98.85
Aeromonas veronii	NC_015424.1	1.90	979	96.52	22.16	97.18
Amycolatopsis mediterranei	CP003729.1	1.30	348	97.21	63.99	97.33
Arthrobacter arilaitensis	NC_014550.1	1.20	557	97.65	-	98.99
Azorhizobium caulinodans	NC_009937.1	1.20	269	85.55	67.81	85.65
Bacillus cereus	NC_011658.1	1.10	251	94.33	-	88.56
Bacillus thuringiensis	NC_014171.1	1.10	69	47.56	-	-
Bdellovibrio bacteriovorus	NC_005363.1	1.10	49	99.81	99.95	99.83
Bifidobacterium adolescentis	NC_008618.1	1.10	88	98.40	-	96.52
Bifidobacterium animalis	CP001892.1	1.10	95	94.58	46.89	92.23
Brachyspira intermedia	CP002874.1	1.10	46	99.41	-	99.90
Campylobacter jejuni	NC_002163.1	1.10	94	99.88	-	99.88
Candidatus Pelagibacter ubique	NC_007205.1	1.10	117	99.17	-	99.31
Candidatus Phytoplasma mali	NC_011047.1	1.10	255	93.69	-	95.25
Candidatus Sulcia muelleri	NC_013123.1	1.10	554	88.65	-	91.46
Chlamydia trachomatis	CP002054.1	1.10	147	95.25	-	97.68
Chlamydophila psittaci	CP002807.1	1.10	131	98.49	74.33	84.43
Clostridium acetobutylicum	CP002118.1	1.10	39	-	64.09	65.26
Clostridium botulinum	NC_015425.1	1.10	55	-	-	84.74
Clostridium tetani	NC_004557.1	1.10	55	-	-	90.00
Clostridium thermocellum	NC_009012.1	1.10	40	-	-	96.04
Corynebacterium diphtheriae	NC_016788.1	1.10	62	65.76	43.14	55.56
Corynebacterium pseudotuberculosis	NC_017462.1	1.10	66	92.29	-	93.23
Corynebacterium ulcerans	CP002790.1	1.10	61	86.39	-	54.43

Continued to next page ...

Species	NCBI accession number	Relative Abundance (%)	Coverage (×)	MaxBin2 F1-score (%)	MetaBAT2 F1-score (%)	MetaCoAG F1-score (%)
Cyanobacterium UCYN	NC_013771.1	1.10	106	96.72	61.16	85.86
Cyanothece sp	NC_014501.1	1.10	25	99.77	-	92.77
Desulfovibrio vulgaris	NC_002937.3	1.10	25	99.81	33.14	99.75
Ehrlichia ruminantium	NC_006831.1	1.10	102	96.90	-	98.38
Enterococcus faecium	CP003351.1	1.10	52	95.85	-	97.81
Erysipelothrix rhusiopathiae	NC_015601.1	1.00	86	98.17	-	86.43
Escherichia coli	NC_011415.1	1.00	31	-	-	-
Fervidicoccus fontis	NC_017461.1	1.00	116	99.63	80.62	-
Fibrobacter succinogenes	CP002158.1	1.00	40	99.93	51.60	75.97
Flavobacterium branchiophilum	NC_016001.1	1.00	43	98.24	-	99.20
Francisella novicida	NC_008601.1	1.00	80	62.68	-	55.32
Francisella tularensis	NC_009749.1	1.00	81	24.67	-	-
Fusobacterium nucleatum	NC_003454.1	1.00	71	98.24	-	99.05
Gardnerella vaginalis	CP002725.1	1.00	89	96.98	54.71	98.36
Granulicella tundricola	NC_015064.1	1.00	36	98.52	24.17	98.59
Haemophilus influenzae	NC_009567.1	1.00	81	-	-	97.37
Haemophilus parainfluenzae	FQ312002.1	1.00	74	-	-	95.05
Haemophilus somnus	NC_010519.1	1.00	68	-	-	97.87
Halobacterium sp	AE004437.1	1.00	76	98.47	-	-
Halothiobacillus neapolitanus	NC_013422.1	1.00	59	99.51	-	99.36
Helicobacter pylori	CP001680.1	1.00	98	99.78	-	99.82
Hyphomicrobium sp	NC_015717.1	1.00	32	90.25	72.49	90.49
Ignavibacterium album	NC_017464.1	1.00	42	99.43	27.36	99.04
Klebsiella oxytoca	NC_016612.1	1.00	26	52.76	-	80.72
Krokinobacter sp	NC_015496.1	1.00	45	99.43	60.11	99.45
Lactobacillus brevis	NC_008497.1	1.00	67	79.75	-	98.26
Lactobacillus casei	CP002618.1	1.00	50	93.29	-	98.91
Lactobacillus delbrueckii	NC_008054.1	1.00	82	97.31	-	97.27
Lawsonia intracellularis	NC_008011.1	1.00	105	75.83	98.17	73.93
Legionella pneumophila	NC_014125.1	1.00	44	96.27	70.13	95.55
Metallosphaera cuprina	NC_015435.1	0.90	83	99.63	65.17	-
Methanocorpusculum labreanum	NC_008942.1	0.90	85	99.61	-	-
Methanosarcina acetivorans	AE010299.1	0.90	27	74.10	-	-
Methanosarcina barkeri	NC_007355.1	0.90	32	56.89	-	-
Micrococcus luteus	NC_012803.1	0.90	61	85.81	-	84.35
Mycobacterium bovis	CP002095.1	0.90	35	-	-	96.85
Mycobacterium sp	NC_008146.1	0.90	27	42.07	-	86.88
Mycoplasma gallisepticum	CP003512.1	0.90	157	97.28	-	97.77
Mycoplasma hyorhinis	CP002669.1	0.90	185	97.81	-	99.39

Continued to next page ...

Species	NCBI accession number	Relative Abundance (%)	Coverage (×)	MaxBin2 F1-score (%)	MetaBAT2 F1-score (%)	MetaCoAG F1-score (%)
<i>Neisseria meningitidis</i>	CP001561.1	0.90	68	97.51	-	97.13
<i>Nitrosococcus watsonii</i>	NC_014315.1	0.90	46	99.56	29.08	99.33
<i>Nitrosomonas</i> sp	NC_015222.1	0.90	48	93.71	-	92.12
<i>Nocardia farcinica</i>	NC_006361.1	0.90	26	42.18	31.56	69.90
<i>Odoribacter splanchnicus</i>	NC_015160.1	0.90	35	96.13	-	99.05
<i>Paenibacillus mucilaginosus</i>	NC_017672.1	0.90	14	88.49	13.08	93.45
<i>Paenibacillus</i> sp	NC_013406.1	0.90	17	96.00	58.24	93.77
<i>Photobacterium profundum</i>	NC_006370.1	0.90	30	94.56	-	95.72
<i>Prochlorococcus marinus</i>	NC_009091.1	0.90	75	94.81	-	98.81
<i>Pseudogulbenkiania</i> sp	NC_016002.1	0.90	28	97.82	-	98.23
<i>Pseudomonas putida</i>	CP002290.1	0.90	21	59.78	-	99.56
<i>Rhizobium leguminosarum</i>	NC_008380.1	0.90	24	93.22	35.11	94.09
<i>Rhodococcus jostii</i>	NC_008268.1	0.90	16	-	-	92.09
<i>Rickettsia prowazekii</i>	CP003391.1	0.90	111	95.64	-	94.92
<i>Rickettsia rickettsii</i>	NC_016909.1	0.90	97	-	-	-
<i>Rickettsia slovaca</i>	NC_016639.1	0.90	96	-	-	67.69
<i>Ruegeria</i> sp	NC_008044.1	0.90	38	97.11	62.75	97.11
<i>Salmonella enterica</i>	NC_011083.1	0.90	25	66.15	-	76.74
<i>Sealdella termitidis</i>	NC_013517.1	0.90	28	99.16	39.46	99.40
<i>Shewanella</i> sp	NC_008322.1	0.90	26	99.21	-	99.25
<i>Shigella flexneri</i>	NC_004741.1	0.90	27	73.77	-	78.92
<i>Sodalis glossinidius</i>	NC_007712.1	0.90	29	-	-	75.67
<i>Staphylococcus aureus</i>	CP001844.2	0.90	44	99.29	-	99.29
<i>Streptococcus pneumoniae</i>	NC_010582.1	0.90	56	46.51	-	93.40
<i>Streptococcus pyogenes</i>	NC_004606.1	0.90	65	98.55	-	97.30
<i>Streptococcus suis</i>	CP002570.1	0.90	60	72.39	-	95.95
<i>Streptococcus thermophilus</i>	NC_008532.1	0.90	50	49.97	-	95.67
<i>Streptomyces scabiei</i>	NC_013929.1	0.90	9	66.74	-	84.41
<i>Symbiobacterium thermophilum</i>	NC_006177.1	0.90	26	98.36	-	98.63
<i>Thermoanaerobacter brockii</i>	NC_014964.1	0.90	39	-	-	-
<i>Thermoanaerobacter</i> sp	NC_014538.1	0.90	38	62.12	-	54.04
<i>Thermococcus sibiricus</i>	NC_012883.1	0.80	50	99.00	95.95	-
<i>Variovorax paradoxus</i>	NC_012792.1	0.80	82	98.44	99.48	-
<i>Weeksella virosa</i>	CP002455.1	0.80	41	98.53	-	98.86
<i>Wolbachia</i> sp	NC_012416.1	0.80	64	63.15	-	61.90
<i>Xanthobacter autotrophicus</i>	NC_009720.1	0.70	17	90.11	-	90.35
<i>Yersinia pestis</i>	NC_010159.1	0.40	20	59.47	-	91.93

Appendix D

Commands

D.1 Assembly Tools

metaSPAdes

```
spades --meta -1 reads_1.fastq -2 reads_2.fastq -o /path/output_folder -t 20
spades --meta -1 reads_1.fastq -2 reads_2.fastq -k 21,33,55,77,99,127
-o /path/output_folder -t 20
```

SGA

```
sga preprocess -o reads.fastq --pe-mode 1 reads_1.fastq reads_2.fastq
sga index -a ropebwt -t 16 --no-reverse reads.fastq
sga correct -k 41 --learn -t 16 -o reads.k41.fastq reads.fastq
sga index -a ropebwt -t 16 reads.k41.fastq
sga filter -x 2 -t 16 reads.k41.fastq
sga fm-merge -m 45 -t 16 reads.k41.filter.pass.fa
sga index -t 16 reads.k41.filter.pass.merged.fa
sga overlap -m 55 -t 16 reads.k41.filter.pass.merged.fa
sga assemble -m 95 reads.k41.filter.pass.merged.asqg.gz
```

MEGAHIT

```
megahit -1 reads_1.fastq -2 reads_2.fastq -o /path/output_folder -t 56
megahit_toolkit contig2fastg 141 final.contigs.fa > final.fastg
```

metaFlye

```
flye --meta --pacbio-raw reads.fastq -t 56 -g 500m -o /path/output_folder
```

D.2 Binning Tools

MetaWatt

```
java -jar MetaWatt-3.5.3/dist/MetaWatt-3.5.3.jar --run /path/input_folder
--threads 8 --skip-database-update
```

MaxBin2

```
perl MaxBin-2.2.5/run_MaxBin.pl -contig contigs.fasta -abund abundance.abund
-thread 8 -out /path/output_folder
```

Note: abundance.abund is a tab separated file with contig ID and the coverage for each contig in the assembly. metaSPAdes provides the coverage of each contig in the contig identifier of the final assembly. We can directly extract these values to create the abundance.abund file. However, no such information is provided for contigs produced by SGA. Hence, reads should be mapped back to contigs in order to determine the coverage of SGA contigs.

MetaBAT2

```
jgi_summarize_bam_contig_depths --outputDepth depth.txt *.bam
metabat2 -i contigs.fasta -a depth.txt -m 1500 -t 8 -o /path/output_folder/bin
```

SolidBin

```
python scripts/gen_kmer.py contigs.fasta 1000 4
sh gen_cov.sh
python SolidBin.py --contig_file contigs.fasta --composition_profiles kmer_4.csv
--coverage_profiles cov_inputtableR.tsv --output /path/output_folder/result.tsv
--log /path/output_folder/log.txt --use_sfs
```

BusyBee Web

The default parameters provided by the BusyBee Web application were used to bin the contigs of all the data sets. Moreover, BusyBee Web has restrictions on the input file size where the maximum file size for the contigs file allowed is 200MB.

CONCOCT

```
cut_up_fasta.py contigs.fasta -c 10000 -o 0 --merge_last -b
contigs_10K.bed > contigs_10K.fa
concoct_coverage_table.py contigs_10K.bed aln-pe.sorted.bam > coverage_table.tsv
concoct --composition_file contigs_10K.fa --coverage_file coverage_table.tsv
-b /path/output_folder -t 8
merge_cutup_clustering.py /path/output_folder/clustering_gt1000.csv >
/path/output_folder/clustering_merged.csv
extract_fasta_bins.py contigs.fasta /path/output_folder/clustering_merged.csv
--output_path /path/output_folder/fasta_bins
```

Vamb (Co-assembly mode)

```
minimap2 -d catalogue.mmi contigs.fasta
minimap2 -t 8 -N 50 -ax sr catalogue.mmi reads_1.fastq reads_2.fastq | samtools view
-F 3584 -b --threads 8 > reads.bam
vamb --outdir /path/output_folder --fasta contigs.fasta --bamfiles *.bam
--minfasta 200000
```

MetaCoAG

```
./MetaCoAG --assembler spades --contigs contigs.fasta
--graph assembly_graph_with_scaffolds.gfa --paths contigs.paths
--abundance abundance.tsv --output /path/output_folder
```

D.3 Bin Refinement Tools

GraphBin

metaSPAdes version

```
python graphbin.py --assembler spades --graph assembly_graph_with_scaffolds.gfa
--paths contigs.paths --binned binning_result.csv
--output /path/output_folder
```

SGA version

```
python graphbin.py --assembler sga --graph default-graph.asqg
--binned binning_result.csv --output /path/output_folder
```

MEGAHIT version

```
python graphbin.py --assembler megahit --graph final.gfa
--binned binning_result.csv --output /path/output_folder
```

GraphBin2

metaSPAdes version

```
python graphbin2.py --assembler spades --graph assembly_graph_with_scaffolds.gfa
--contigs contigs.fasta --paths contigs.paths
--abundance abundance.abund --binned binning_result.csv
--output /path/output_folder
```

SGA version

```
python graphbin2.py --assembler sga --graph default-graph.asqg
--contigs contigs.fasta --abundance abundance.abund
--binned binning_result.csv --output /path/output_folder
```

Flye version

```
python graphbin2.py --assembler flye --contigs edges.fasta
--abundance abundance.abund --graph assembly_graph.gfa
--binned binning_result.csv --output /path/output_folder
```