Bioinformatics, 36(11), 2020, 3307–3313
doi: 10.1093/bioinformatics/btaa180
Advance Access Publication Date: 13 March 2020
Original Paper

Genome analysis

GraphBin: refined binning of metagenomic contigs using assembly graphs

Vijini Mallawaarachchi*, Anuradha Wickramarachchi and Yu Lin*

Research School of Computer Science, College of Engineering and Computer Science, Australian National University, Canberra ACT 0200, Australia

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on April 30, 2019; revised on February 18, 2020; editorial decision on March 8, 2020; accepted on March 10, 2020

Abstract

Motivation: The field of metagenomics has provided valuable insights into the structure, diversity and ecology within microbial communities. One key step in metagenomics analysis is to assemble reads into longer contigs which are then binned into groups of contigs that belong to different species present in the metagenomic sample. 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.

Results: We propose 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. We show that GraphBin can make use of the assembly graphs constructed from both the de Bruijn graph and the overlap-layout-consensus approach. Moreover, we demonstrate improved experimental results from GraphBin in terms of identifying mis-binned contigs and binning of contigs discarded by existing binning tools. To the best of our knowledge, this is the first time that the information from the assembly graph has been used in a tool for the binning of metagenomic contigs.

Availability and implementation: The source code of GraphBin is available at https://github.com/Vini2/GraphBin.

Contact: vijini.mallawaarachchi@anu.edu.au or yu.lin@anu.edu.au

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Metagenomics involves the study of various genetic material obtained from the genomes found in microbial communities (Riesenfeld *et al.*, 2004). A microbial community such as forest soil or seawater can contain a large number of organisms. We need to identify the species present in order to investigate the taxonomic structure of a given metagenomic sample. Reads can be binned prior to assembly (Cleary *et al.*, 2015; Girotto *et al.*, 2016; Li *et al.*, 2019; Luo *et al.*, 2018; Ounit *et al.*, 2015; Schaeffer *et al.*, 2017; Vervier *et al.*, 2016; Wang *et al.*, 2012). Although binning reads can simplify the assembly process by reducing the computational costs, it is less reliable due to their short read lengths (Nolla-Ardèvol *et al.*, 2015; Wang *et al.*, 2018). Therefore, many metagenomics analysis pipelines first assemble reads into longer contigs which contain more genomic information and then bin the contigs into groups that belong to different taxonomic groups such as species, genera or higher levels (Sedlar *et al.*, 2017). Once we obtain the binned contigs, they can be used to construct the partial or complete genomes of the different species found in the metagenomic sample.

Existing metagenomics binning methods for contigs can mainly be divided into three categories: (i) composition based, (ii) abundance based and (iii) composition and abundance based (Sangwan *et al.*, 2016). Composition-based methods are 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). In order to make the process more computationally feasible, the sequence composition information is converted to numerical feature vectors. The commonly used features are the normalized frequencies of oligonucleotides (referred to as *k-mers*) of a particular size (*k*) (Dick *et al.*, 2009). Studies have shown that the frequency at which oligonucleotides occur is unique between species and it is conserved within species (Dick *et al.*, 2009; Karlin *et al.*, 1997). Moreover, the guanine-cytosine (GC) content is also used for comparison as studies have shown that GC content varies between different species (Dick *et al.*, 2009). For example, the tool 2T-binning (Saeed *et al.*, 2012) first groups sequences based on GC content and then further divides them into finer bins based on *k-mer* frequencies. Most of the tools employ conventional machine learning-based approaches such as interpolated Markov models (used in SCIMM (Sequence Clustering with Interpolated Markov Models); Kelley and Salzberg, 2010) and neural network-based approaches such as self-organizing maps (used in Databionic ESOM Tools; Ultsch and Mörchen, 2005).

Abundance-based binning methods are based on the assumption that abundance profiles of metagenomic sequences either follow the Lander–Waterman (Lander and Waterman, 1988) model 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. It uses an expectation–maximization (EM) algorithm to find parameters for the 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 methods make use of both the variation of oligonucleotide frequencies and coverage information. They have often outperformed previous methods. These methods employ techniques such as principal component analysis (PCA), probabilistic models and EM algorithms. CompostBin (Chatterji *et al.*, 2008) and CONCOCT (Alneberg *et al.*, 2014) make use of PCA for dimensionality reduction. MaxBin (Wu *et al.*, 2014) and MaxBin 2.0 (Wu *et al.*, 2016) adapt the Lander–Waterman (Lander and Waterman, 1988) model and make use of an EM algorithm to iteratively perform clustering. MetaBAT (Kang *et al.*, 2015) and MetaBAT2 (Kang *et al.*, 2019) calculate probabilistic distances between pairs of sequences based on *k-mer* frequencies and abundance.

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 *et al.*, 2018) utilizes 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. d2S Bin (Wang *et al.*, 2017) makes use of the result of existing binning tools and adjusts the contigs among bins by measuring their dissimilarity.

However, the existing binning methods still face problems when binning short sequences (e.g. shorter than 1000 bp), sequences of low abundance species and sequences of closely related species. Short sequences have a small number of *k-mers* resulting in a sparse vector with limited *k-mer* frequency information. Sequences belonging to low abundance species would form small clusters that could be mistaken to belong to a larger bin of a highly abundant species (Sedlar *et al.*, 2017). Sequences of closely related species may be binned into a single bin as they show high similarity in composition (Sczyrba *et al.*, 2017). Hence, we would like to tackle these problems using additional information on contigs available in the assembly process.

In metagenomics, the assembly process is to assemble reads into longer contigs which are then used for the binning process. Most existing assemblers use graphs as the core data structure to capture overlaps between reads, including *de Bruijn graphs* (Lin *et al.*, 2016; Pevzner *et al.*, 2001) and *overlap-layout-consensus graphs* (Kececioglu and Myers, 1995) [or similar *string graphs* (Myers, 2005; Simpson and Durbin, 2012)]. The *assembly graphs* first link reads into paths and later output non-branching paths as contigs. Contigs connected to each other in an assembly graph are more likely to belong to the same taxonomic group and previous studies have shown an improved binning result through manual refinement after visualizing the assembly graph (Barnum *et al.*, 2018).

In our work, we exploit the potential of using the assembly graph to refine the binning results from existing tools. We utilize 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.

2 Materials and methods

Figure 1 shows a flow diagram of GraphBin, to refine the binning of metagenomic contigs using assembly graphs. During the pre-processing 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 Fig. 1). Finally, GraphBin performs label propagation and refinement (Step 2 in Fig. 1) and outputs the refined binning result.

2.1 Pre-processing

2.1.1 Assemble reads into contigs and construct the assembly graph

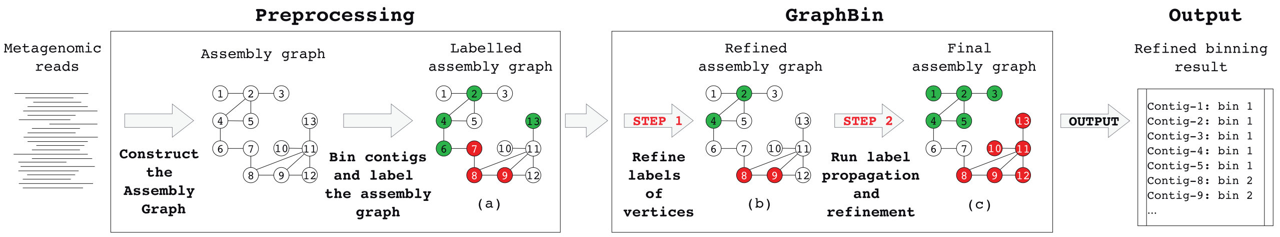
In general, a metagenomic assembly tool first constructs the assembly graph using the connection (overlapping) information between reads, and then traverses through these connections in such a way, that each read is visited in the correct order, thereby linking them together to form a contiguous sequence 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 article, we consider that all the edges have the same weight. Figure 1 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 visualizing 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 *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 *et al.*, 2015).

2.1.2 Bin contigs using an existing binning tool and label the assembly graph

In this article, we selected MetaWatt (Strous *et al.*, 2012) and MaxBin2 (Wu *et al.*, 2016) 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 *et al.*, 2019), SolidBin (Wang *et al.*, 2019) and BusyBee Web (Laczny *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 1000 bp. MetaBAT2 employs a graph clustering approach, where the graph construction is done using tetranucleotide frequency scores. However, MetaBAT2 only bins contigs of length 1500 bp or longer. SolidBin makes use of a spectral clustering approach with additional biological information. However, SolidBin only bins contigs which are longer than 1000 bp. BusyBee Web makes use of a bootstrapped supervised binning approach to bin contigs. By default, it bins contigs which are 500 bp or



```
Contig-1: bin 1
Contig-2: bin 1
Contig-3: bin 1
Contig-4: bin 1
Contig-5: bin 1
Contig-8: bin 2
Contig-9: bin 2
…
```

Fig. 1. The workflow of GraphBin. During the pre-processing 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

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 1a, vertices are illustrated in different colours (labels) to represent contigs belonging to different bins.

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 vertices i and 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. $CLV(i) = \{j | d(i, j) = d_L(i)$ and $L(j) = 1\}$. Note that $CLV(i) = \varnothing$ when $d_L(i) = \infty$. For example, in Figure 1a, $d_L(4) = 1$ 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. $CLV(i) = \varnothing$]. 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.

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 1a, vertex 13 is ambiguous because one of its closest labelled vertices [vertices 8 or 9 in $CLV(13)$] has a red label while vertex 13 itself has a green label. Similarly, vertices 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 follow-up label propagation process. Please refer to Step 1 in Figure 1 for an example. Labels of vertices 6, 7 and 13 are removed. Figure 1b denotes the graph after refining the labels.

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 $i$
function getCLV($G, i, L$)
    $SET_{current} \leftarrow G.neighbours(i)$
    mark $i$ as visited
    $CLV_{temp} \leftarrow \emptyset$ and $SET_{next} \leftarrow \emptyset$
    while $SET_{current} \neq \emptyset$ and $CLV_{temp} = \emptyset$ do
        forall $v \in SET_{current}$ do
            mark $v$ as visited
            if $L(v) = 1$ then
                $CLV_{temp} \leftarrow CLV_{temp} \cup \{v\}$
            else
                forall $r \in G.neighbours(v)$ do
                    if $L(r) = 0$ then
                        $SET_{next} \leftarrow SET_{next} \cup \{r\}$
        $SET_{current} \leftarrow SET_{next}$
    $CLV(i) \leftarrow CLV_{temp}$
    return $CLV(i)$
```

2.3 Run label propagation and refinement

Label propagation approaches have been applied to cluster metagenomic sequences (Kang *et al.*, 2019; Li *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 initializes 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 (2002) was applied to the assembly graph in order to predict the labels of the unlabelled vertices. Details about the algorithm can be found in Supplementary Material S1, Section 1.1. 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 1c. Please refer to Step 2 in Figure 1 as an example. Labels of vertices 6 and 7 will be removed after the label propagation process as they are ambiguous (as illustrated in Fig. 1c).

3 Experimental design

3.1 Datasets

3.1.1 Simulated datasets

We simulated several metagenomic datasets according to a pre-born 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 two species (*E.faecalis* and *S.aureus*, referred as **ES**) and three species (*E.faecalis*, *S.aureus* and *C.avidum*, referred as **ESC**) respectively. Paired-end reads were simulated using the tool InSilicoSeq (Gourlé *et al.*, 2019) modelling a MiSeq instrument with 300 bp mean read length. Please refer Supplementary Material S1, Section 2 for further details.

3.1.2 Sharon dataset

The **Sharon** dataset (Sharon *et al.*, 2013; available on the NCBI Sequence ReadArchive under accession number *SRA052203*) is composed of a time-series of 11 faecal 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. Please refer Supplementary Material S1, Section 2 for further details.

3.1.3 CAMI datasets

We selected three publicly available datasets from the first **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 Supplementary Material S1, Section 2 for further details about the CAMI datasets.

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 *et al.*, 2015) as discussed in Section 2.1.1. The commands used for assembly can be found in Supplementary Material S1, Section 8.1. The resulting six simulated datasets are referred as **ES+metaSPAdes**, **ES+SGA**, **ES+MEGAHIT**, **ESC+metaSPAdes**, **ESC+SGA** and **ESC+MEGAHIT**. The Sharon datasets are referred as **Sharon+metaSPAdes**, **Sharon+SGA** and **Sharon+MEGAHIT**. The CAMI_l datasets are referred as **CAMI_l+metaSPAdes**, **CAMI_l+SGA** and **CAMI_l+MEGAHIT**. The CAMI_m datasets are referred as **CAMI_m+metaSPAdes**, **CAMI_m+SGA** and **CAMI_m+MEGAHIT**. The CAMI_h datasets are referred as **CAMI_h+metaSPAdes**, **CAMI_h+SGA** and **CAMI_h+MEGAHIT**. MetaWatt 3.5.3 (Strous *et al.*, 2012), MaxBin 2.2.5 (Wu *et al.*, 2016), BusyBee Web (Laczny *et al.*, 2017) with their default parameters, SolidBin 1.3 (Wang *et al.*, 2019) in `SolidBin-SFS` mode, and MetaBAT2 (Kang *et al.*, 2019) with minimum contig length set to 1500 bp were selected to obtain the initial binning results as discussed in Section 2.1.2. The commands used for binning can be found in Supplementary Material S1, Section 8.2.

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 (i) *Precision*, (ii) *Recall*, (iii) *F1-score* and (iv) *Adjusted Rand Index (ARI)* that have been used in previous binning studies (Alneberg *et al.*, 2014; Herath *et al.*, 2017; Wang *et al.*, 2017) at the species level. The definitions of these criteria can be found in Supplementary Material S1, Section 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 *et al.*, 2016), MetaBAT2 (Kang *et al.*, 2019), SolidBin (Wang *et al.*, 2019) and BusyBee Web (Laczny *et al.*, 2017). Note that three different assemblers metaSPAdes (Nurk *et al.*, 2017), SGA (Simpson and Durbin, 2012) and MEGAHIT (Li *et al.*, 2015) were used to build the assembly graphs. Overall, GraphBin has been shown to improve the binning results of different combinations of the above assemblers and binning tools including MetaWatt (Fig. 2 and Supplementary Figs S1 and S2), MaxBin2 (Supplementary Figs S3–S5), MetaBAT2 (Supplementary Fig S6–S8), SolidBin (Supplementary Figs S9–S11) and BusyBee Web (Supplementary Figs S12–S14). 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. Figure 2 denotes a representative result of GraphBin on improving the binning result of MetaWatt based on the SGA assembler 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 Supplementary Figures S1–S14. The exact values for all the evaluation criteria can be found in the Supplementary Material S2.

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* 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 1000 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 1500 bp while MetaWatt discards short contigs as the estimation of their relative coverage and tetranucleotide frequency may not be accurate. For example, Figure 2a 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 Fig. 2b). Further results can be found Supplementary Figures S1–S14.

4.2 Results on Sharon datasets

We also tested GraphBin using the **Sharon** dataset (Sharon *et al.*, 2013). As shown in Figure 2c, 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 Fig. 2c). Moreover, similar improvements on *recall* and *F1-score* can be seen from GraphBin with other binning tools (Supplementary Figs S3–S14). Note that the increase of *recall* and *F1-score* of

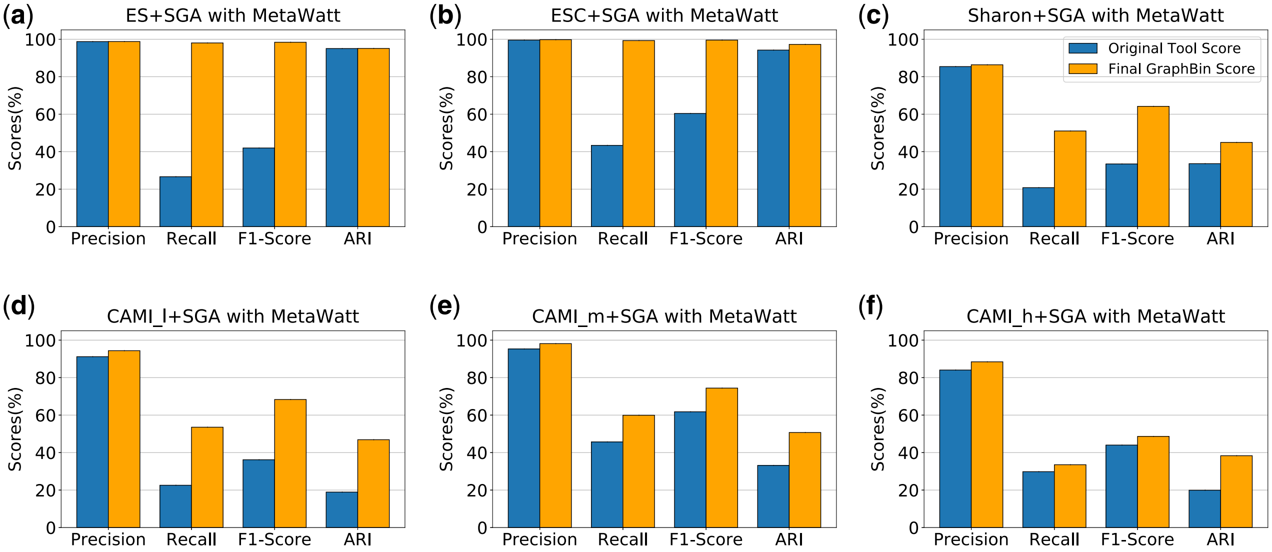


Fig. 2. Refined binning results of MetaWatt with GraphBin for the datasets (**a**) ES+SGA, (**b**) ESC+SGA, (**c**) Sharon+SGA, (**d**) CAMI_l+SGA, (**e**) CAMI_m+SGA and (**f**) CAMI_h+SGA. Each graph denotes the precision, recall, F1-score and ARI values at the species level of the original tool (MetaWatt) 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

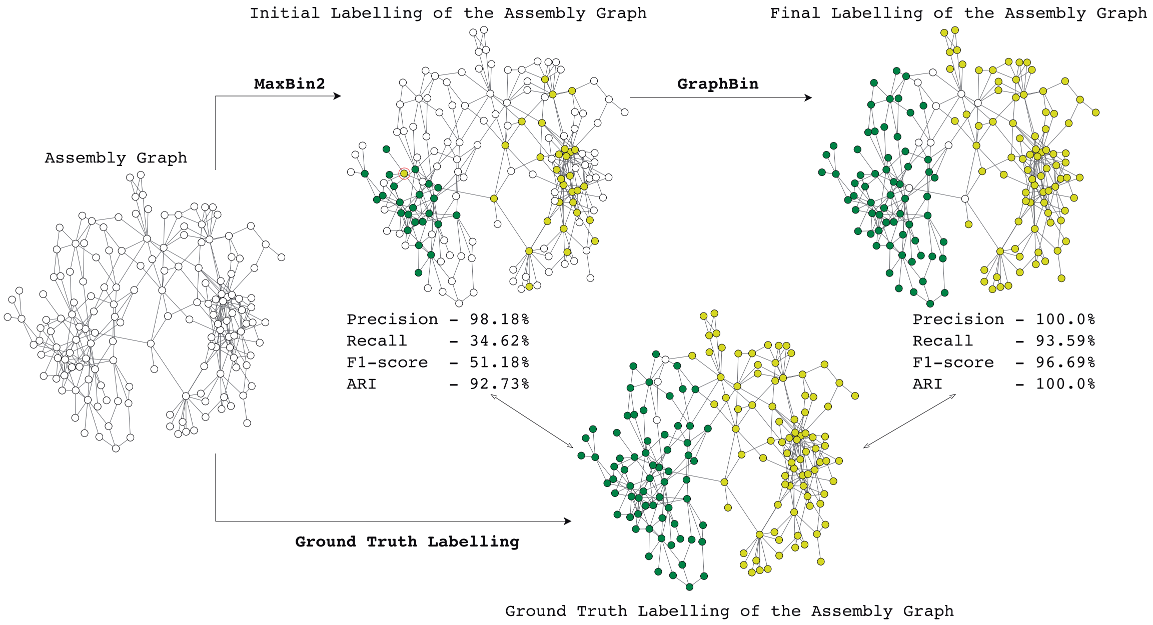


Fig. 3. The labelling of the assembly graph of ES+metaSPAdes dataset based on the initial MaxBin2 result and GraphBin result. Contigs belonging to *E.faecalis* and *S.aureus* are coloured in green and yellow respectively. Contigs which were not considered in the classification and discarded during the binning process are coloured in white. 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. (Color version of this figure is available at *Bioinformatics* online.)

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.

4.3 Results on CAMI datasets

We further tested GraphBin using the CAMI datasets of different complexities (Sczyrba *et al.*, 2017). As shown in Figure 2(d–f), GraphBin has improved all the evaluation criteria for binning results from MetaWatt for the SGA assemblies of three CAMI datasets. Similarly, GraphBin has 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 (Supplementary Figs S1–S14). 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 Supplementary Table S5, 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.

4.4 Visualization of the assembly graph

Figure 3 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. Detailed visualizations of the assembly graphs of **ES+metaSPAdes** and **ESC+metaSPAdes** datasets are provided in Supplementary Material S1, Section 6.

4.5 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, 16 G memory and Intel(R) Core(TM) i7-7700 CPU @ 3.60 GHz with 4 CPU cores. The running times for assembly and binning using MaxBin2, MetaWatt, MetaBAT2, SolidBin and GraphBin were recorded. All the running times can be found in Supplementary Table S4.

5 Conclusion

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 1000 bp) due to their limited *k-mer* frequency information.

Hence, we proposed GraphBin, a new method to refine the binning result 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. 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 the 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.

Naturally, much work remains to be done to improve metagenomics binning with the help of assembly graphs. Our method currently 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 vertices on species boundaries. This can lead to mis-binned contigs and reduce the precision of the final GraphBin result. With the advancement of third-generation sequencing, GraphBin will benefit from more reliable connectivity information of the assembly graph built from long reads. We also intend to improve the label propagation algorithm by introducing probabilistic models and more genomic features such as overlap length and coverage information. Moreover, the assembly process of complex metagenomic datasets may produce mis-assembled contigs (e.g. concatenation of segments from different genomes) as well as unreliable connections (e.g. due to chimeric reads) in the assembly graph. It is thus worth exploring how to make use of noisy assembly graphs in improving metagenomics binning.

Acknowledgements

We thank Dr Benjamin Kaehler and Prof. Gavin Huttley for various suggestions. Furthermore, this research was undertaken with the assistance of resources and services from the National Computational Infrastructure (NCI Australia), an NCRIS enabled capability supported by the Australian Government.

Funding

The authors received no specific funding for this work.

Conflict of Interest: none declared.

References

Alneberg,J. *et al.* (2014) Binning metagenomic contigs by coverage and composition. *Nat. Methods*, **11**, 1144–1146.

Bankevich,A. *et al.* (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.*, **19**, 455–477.

Barnum,T.P. *et al.* (2018) Genome-resolved metagenomics identifies genetic mobility, metabolic interactions, and unexpected diversity in perchlorate-reducing communities. *ISME J.*, **12**, 1568–1581.

Chatterji,S. *et al.* (2008). CompostBin: a DNA composition-based algorithm for binning environmental shotgun reads. In: Vingron,M. and Wong,L. (eds) *Research in Computational Molecular Biology*. Springer, Berlin, Heidelberg, pp. 17–28.

Cleary,B. *et al.* (2015) Detection of low-abundance bacterial strains in metagenomic datasets by eigengenome partitioning. *Nat. Biotechnol.*, **33**, 1053–1060.

Dick,G.J. *et al.* (2009) Community-wide analysis of microbial genome sequence signatures. *Genome Biol.*, **10**, R85.

Girotto,S. *et al.* (2016) MetaProb: accurate metagenomic reads binning based on probabilistic sequence signatures. *Bioinformatics*, **32**, i567–i575.

Gourlé,H. *et al.* (2019) Simulating Illumina metagenomic data with InSilicoSeq. *Bioinformatics*, **35**, 521–522.

Herath,D. *et al.* (2017) CoMet: a workflow using contig coverage and composition for binning a metagenomic sample with high precision. *BMC Bioinformatics*, **18**(Suppl. 16), 571.

Kang,D.D. *et al.* (2015) MetaBAT, an efficient tool for accurately reconstructing single genomes from complex microbial communities. *PeerJ.*, **3**, e1165.

Kang,D.D. *et al.* (2019) MetaBAT 2: an adaptive binning algorithm for robust and efficient genome reconstruction from metagenome assemblies. *PeerJ.*, **7**, e7359.

Karlin,S. *et al.* (1997) Compositional biases of bacterial genomes and evolutionary implications. *J. Bacteriol.*, **179**, 3899–3913.

Kececioglu,J.D. and Myers,E.W. (1995) Combinatorial algorithms for DNA sequence assembly. *Algorithmica*, **13**, 7–51.

Kelley,D. and Salzberg,S. (2010) Clustering metagenomic sequences with interpolated Markov models. *BMC Bioinformatics*, **11**, 544.

Laczny,C.C. *et al.* (2017) BusyBee Web: metagenomic data analysis by bootstrapped supervised binning and annotation. *Nucleic Acids Res.*, **45**, W171–W179.

Lander,E.S. and Waterman,M.S. (1988) Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, **2**, 231–239.

Li,D. *et al.* (2015) MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics*, **31**, 1674–1676.

Li,K. *et al.* (2019). Deconvolute individual genomes from metagenome sequences through read clustering. *bioRxiv.*

Lin,Y. *et al.* (2016) Assembly of long error-prone reads using de Bruijn graphs. *Proc. Natl. Acad. Sci. USA*, **113**, E8396–E8405.

Lu,Y.Y. *et al.* (2016) COCACOLA: binning metagenomic contigs using sequence COmposition, read CoverAge, CO-alignment and paired-end read LinkAge. *Bioinformatics*, **33**, 791–798.

Luo,Y. *et al.* (2019) Metagenomic binning through low-density hashing. *Bioinformatics*, **35**, 219–226.

Myers,E.W. (2005) The fragment assembly string graph. *Bioinformatics*, **21**(Suppl. 2), ii79–ii85.

Nielsen,H.B. *et al.* (2014) Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes. *Nat. Biotechnol.*, **32**, 822–828.

Nolla-Ardèvol,V. *et al.* (2015) Metagenome from a Spirulina digesting biogas reactor: analysis via binning of contigs and classification of short reads. *BMC Microbiol.*, **15**, 277.

Nurk,S. *et al.* (2017) metaSPAdes: a new versatile metagenomic assembler. *Genome Res.*, **27**, 824–834.

Ounit,R. *et al.* (2015) CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics*, **16**, 236.

Pevzner,P.A. *et al.* (2001) An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. USA*, **98**, 9748–9753.

Riesenfeld,C.S. *et al.* (2004) Metagenomics: genomic analysis of microbial communities. *Annu. Rev. Genet.*, **38**, 525–552.

Saeed,I. *et al.* (2012) Unsupervised discovery of microbial population structure within metagenomes using nucleotide base composition. *Nucleic Acids Res.*, **40**, e34.

Sangwan,N. *et al.* (2016) Recovering complete and draft population genomes from metagenome datasets. *Microbiome*, **4**, 8.

Schaeffer,L. *et al.* (2017) Pseudoalignment for metagenomic read assignment. *Bioinformatics*, **33**, 2082–2088.

Sczyrba,A. *et al.* (2017) Critical Assessment of Metagenome Interpretation–a benchmark of metagenomics software. *Nat. Methods*, **14**, 1063–1071.

Sedlar,K. *et al.* (2017) Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics. *Comput. Struct. Biotechnol. J.*, **15**, 48–55.

Sharon,I. *et al.* (2013) Time series community genomics analysis reveals rapid shifts in bacterial species, strains, and phage during infant gut colonization. *Genome Res.*, **23**, 111–120.

Simpson,J.T. and Durbin,R. (2012) Efficient de novo assembly of large genomes using compressed data structures. *Genome Res.*, **22**, 549–556.

Strous,M. *et al.* (2012) The binning of metagenomic contigs for microbial physiology of mixed cultures. *Front. Microbiol.*, **3**, 410.

Ultsch,A. and Mörchen,F. (2005). ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM. *Technical report*. Databionics Research Group, University of Marburg, Germany.

Vervier,K. *et al.* (2016) Large-scale machine learning for metagenomics sequence classification. *Bioinformatics*, **32**, 1023–1032.

Vollmers,J. *et al.* (2017) Comparing and evaluating metagenome assembly tools from a microbiologist's perspective - not only size matters! *PLoS One*, **12**, e0169662–31.

Wang,J. *et al.* (2018) BMC3C: binning metagenomic contigs using codon usage, sequence composition and read coverage. *Bioinformatics*, **34**, 4172–4179.

Wang,Y. *et al.* (2012) MetaCluster 4.0: a novel binning algorithm for NGS reads and huge number of species. *J. Comput. Biol.*, **19**, 241–249.

Wang,Y. *et al.* (2017) Improving contig binning of metagenomic data using d2S oligonucleotide frequency dissimilarity. *BMC Bioinformatics*, **18**, 425.

Wang,Z. *et al.* (2019) SolidBin: improving metagenome binning with semi-supervised normalized cut. *Bioinformatics*, **35**, 4229–4238.

Wu,Y.-W. and Ye,Y. (2011) A novel abundance-based algorithm for binning metagenomic sequences using l-tuples. *J. Comput. Biol.*, **18**, 523–534.

Wu,Y.-W. *et al.* (2014) MaxBin: an automated binning method to recover individual genomes from metagenomes using an expectation-maximization algorithm. *Microbiome*, **2**, 26.

Wu,Y.-W. *et al.* (2016) MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics*, **32**, 605–607.

Zhu,X. and Ghahramani,Z. (2002). Learning from labeled and unlabeled data with label propagation. *Technical report*. School of Computer Science, Carnegie Mellon University.