IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 April 2023, accepted 8 May 2023, date of publication 18 May 2023, date of current version 25 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3277755

RESEARCH ARTICLE

BinChill: A Metagenomic Binning Ensemble Method

OLIVER S. BAK, MARCUS D. JENSEN, FREDERIK M. TRUDSLEV, ANDREAS WINDFELD, AND ANDRE LAMURIAS
Department of Computer Science, Aalborg University, 9220 Aalborg, Denmark
Corresponding author: Andre Lamurias (andrel@cs.aau.dk)

This work was supported in part by VILLUM FONDEN under Grant 34299.

ABSTRACT The goal of metagenomic binning is to reconstruct genomes from a mixture of DNA sequences into genomic bins, which can be considered a clustering task. Multiple methods have been proposed for this task, such as distance-based metrics, machine learning, and ensemble approaches. We propose BinChill, a metagenomic ensemble method, based on the generic co-occurrence ensembler method, ACE. BinChill incorporates domain information in the form of Single-Copy Genes (SCG) with a co-occurrence strategy. This strategy combines multiple clustering partitions according to how often two items co-occur in the same cluster. BinChill was able to reconstruct more or equally as many high- and medium quality while having an equal or faster runtime than other metagenomics-specific methods on a smaller simulated dataset. On larger datasets, both simulated and real-world, BinChill outperformed other methods in reconstructing high-quality bins, at the cost of an increased processing time when compared to generic ensemble clustering algorithms. This is due to the domain-specific steps that our method implements. Our results show that the strengths of multiple partitions can be combined to generate a partition of higher quality.

INDEX TERMS Genomics, clustering algorithms, bioinformatics, partitioning algorithms.

I. INTRODUCTION

Metagenomics targets the study of genomes of microbial communities from real-world DNA sequencing datasets. Within this field, computational methods play an important role, as the samples are taken from complex environments, where if human supervision is necessary, the approach cannot truly be reproducible or scalable. Within metagenomics, contig binning is the process of reconstructing genomes from contigs. A contig is a DNA string obtained from a series of overlapping DNA sequences, that were sequenced from a mixture of many microbial genomes. The goal of metagenomic binning is to place contigs that correspond to the same genome into the same bins. The binned genomes can then be considered Metagenome-Assembled Genomes (MAGs), which can be analyzed according to their functions and processes [1]. The study of contig binning, which can be considered a clustering problem, is an important subject, as the quality of the reconstructed MAGs has a great effect on the results of metagenomic analysis. Within the fields of biology, metagenomics is used to deal with the problem of identifying uncultured bacteria and viruses that can only be found in microbial communities. Microbial communities have a direct impact on human health and the environment, and they are critical to achieving the Sustainable Development Goals [2], [3].

Most computational approaches performing metagenomic binning explore diverse genomics features, such as composition, abundance, and Single-Copy Genes (SCGs). Composition and abundance are properties of the DNA sequence itself, where the contigs composition is the frequency of k-mers [4], while abundance is the degree of overlapping reads. SCGs refer to specific genes, that exist in only one copy in a single genome. Each contig may contain zero or more distinct SCGs. Thus, an ideal cluster should contain only one copy of each SCG. Consequently, the presence of multiple copies of the same SCG would be considered an error, as it would mean that that cluster contains contigs of more than one genome. A MAG with a complete set of unique SCG can be considered fully recovered [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Yongqiang Cheng.

To improve the efficiency of clustering methods, these features are converted to a lower dimensional space. Multiple methods for generating contig embeddings have been proposed. This includes methods such as Principal Component Analysis (PCA) and Gaussian Mixture Model (GMM) [4]. Other binners such as VAMB [6] utilize variational autoencoders (VAE) to convert composition and abundance into an embedding vector for clustering.

Despite extensive studies, none of the individual binners perform best in all situations [7]. Therefore, ensemble methods have been used in metagenomics to improve the binning performance [8], [9], [10], [11]. Ensemble methods can either be performed using multiple different binning strategies or multiple diverse results from a single binner. Another approach would be to use ensemble methods as a binning strategy, by first generating and then ensemble diverse partitions, such as MetaBinner [8].

An ensembler benefits from combining the strengths of many individual partitions by finding a consensus, which provides an improved overall clustering [12]. To accomplish this, a consensus function is used to specify how to map the different input partitions into one singular final partition. Vega-Pons and Ruiz-Shulcloper [13] have reviewed some of the existing approaches within consensus functions, and have classified two main approaches:

The median partition approach focuses on the optimization problem of finding a *median partition*, which treats the consensus function as an optimization problem and usually tries to find a median partition or maximize similarity according to some evaluation metric. This approach has been used in the metagenomic ensemble method DAS-Tool [9].

The object co-occurrence approach focuses on determining the cluster label associated with each object in the consensus partition. This is done by counting the number of occurrences of an object or pairs of objects in the same cluster and generating a final clustering result through a voting process and object similarity [13], [14].

Tahani Alqurashi and Wenjia Wang proposed ACE [14], a generic ensemble method using a consensus function based on the object co-occurrence approach. They chose this, as it had been pointed out that the co-occurrence approach was generally only studied on a theoretical level, and thereby under-utilized in terms of ensemblers [13] and, by extension, metagenomic ensemblers. However, they did not take domain-specific information into account.

Therefore, we present the metagenomic ensembler BinChill, an ensembler that uses the object co-occurrence approach based on ACE, while implementing domain information into the similarity measurement. Furthermore, we also implement a standalone version of BinChill, where instead of relying on partitions from other approaches, it generates its own partitions, so that different techniques can be explored within our framework. We test this approach on metagenomic datasets of several types, according to the metrics taken into consideration in metagenomic studies. BinChill is freely available at https://github.com/marc391130/P6-BinChilling.

In this article, we first present the prior work to contextualize our method (Section II), then we describe the BinChill ensemble clustering method (Section III), its implementation details (Section IV), and the experiments performed (Section V). Finally, we discuss the limitations and future directions of our method (Section VI) and our main conclusions (Section VII).

II. PRIOR WORK

Many different strategies for metagenomic binning have been proposed in recent years. VAMB introduced the use of variational autoencoders to the field of metagenomic binning, which has showcased the reconstruction of more near-complete genomes in both simulated- and real datasets, than other binning methods [6]. VAMB utilizes composition and abundance as input features for a deep variational autoencoder, which is a type of neural network, trained on encoding and decoding the feature matrix. The encoded feature matrix, referred to as the latent representation, is then utilized for clustering the contigs into bins (binning). The clustering is done using an iterative medoid clustering algorithm. VAMB showed improvements in the reconstruction of near-complete strains over multiple datasets, with a performance increase of 29% to 98% on simulated datasets and 45% on real data compared to other state-of-the-art metagenomic binning methods, such as MetaBAT2, MaxBin2, and Canopy. Other recent approaches further explore deep learning algorithms to learn contig embeddings [15], [16].

DAS-Tool is an ensembler that attempts to improve on metagenomic binning by utilizing multiple binning results [9]. As input, DAS-Tool takes in multiple binning partitions, created from the same assembly data. It firstly predicts all the single-copy genes on each bin and scores the bins accordingly, after which candidate bin sets are aggregated. DAS-Tool then iteratively selects high-scoring bins and updates the remaining partial candidate bins until it can output a non-redundant bin set.

The scoring function used in DAS-Tool utilizes completeness (*com*), contamination (*con*), and a megabin penalty (*meg*):

$$S_b = com{-}b \cdot con{-}c \cdot meg \tag{1}$$

These metrics are calculated using the SCGs identified in each bin, and a set of reference SCGs expected to exist once in each MAG. Completeness refers to the fraction of reference SCGs that are in the bin, contamination refers to how many SCGs exist more than once in a bin, and the megabin penalty refers to the total number of SCGs in a bin divided by the number of reference SCGs. Both contamination and megabin penalty are accompanied by weighting factors b and c, respectively. Using this scoring function to determine which bins to have in the final partition, they were able to recover substantially more near-complete genomes than any single binning method tested against [9].

MetaBinner is an ensemble algorithm that, like DAS-Tool, attempts to improve on metagenomic binning through an ensemble of binning results [8]. MetaBinner is a stand-alone

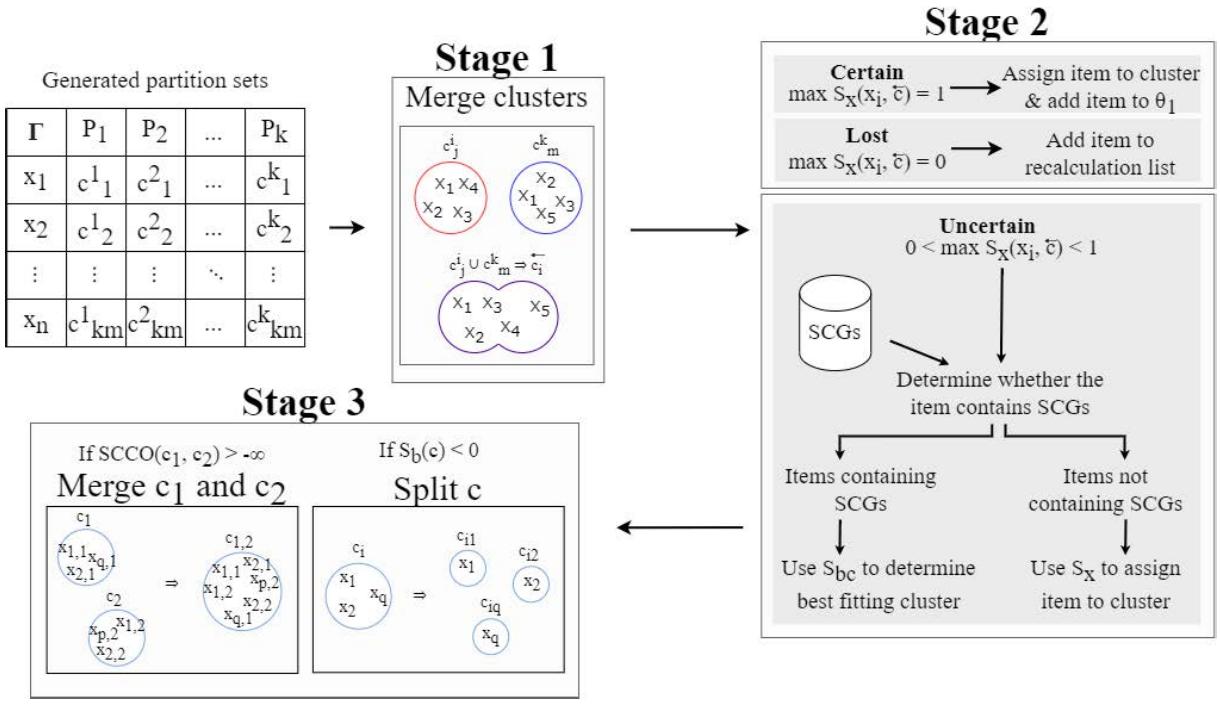

FIGURE 1. An overview of the three stages of the BinChill method.

ensembler, which means it generates the partitions itself, using composition and abundance features. A final partition is produced by combining the generated partitions, selecting the ones with the highest completeness and lowest contamination This means MetaBinner does not rely on binning results from other binners, but rather generates multiple partitions with different features and initializations. MetaBinner showed improved results compared to both single binners, such as MaxBin, MetaBAT, and VAMB, while also outperforming other ensemblers, such as DAS-Tool and MetaWrap, in both simulated- and real datasets.

ACE is a generic clustering ensembler, tested on several real-world benchmark datasets [14]. ACE introduces a novel consensus function, which employs two different similarity measures, namely cluster similarity, and membership similarity. Firstly, it constructs the membership matrix, based on the input partitions. Secondly, it adaptively merges similar clusters to reduce the number of clusters and to increase items' membership value. Lastly, it enforces hard clustering, by assigning each item to only one cluster using membership similarity and co-association. On average, ACE outperformed other state-of-the-art cluster ensemble methods.

III. BinChill ENSEMBLE METHOD

BinChill, the ensemble method proposed in this paper, is described using a framework adapted from the cluster ensembler framework used in ACE [14]. This framework was extended, as to incorporate metagenomic-related domain information, with the goal of using it as a binning ensembler.

For a generic ensemble framework, a given dataset denoted by $X = \{x_1, x_2, \ldots, x_n\}$, $|X| = n$ with x_i being the i'th contig in the dataset. Let $P_m = \{c_1^m, c_2^m, \ldots, c_{k_m}^m\}$, $|P_m| = k_m$, be a partition of k_m clusters, where m also denotes the partition a cluster is a member of. Each cluster c_i^m consists of $|c_i^m| = q$ contigs, and can therefore be defined as $c_i^m = \{x_1, x_2, \ldots, x_q\}$, where $\bigcap_{1\le i\le k_m} c_i^m = \emptyset$ and $\bigcup_{1\le i\le k_m} c_i^m = X$. The set of all clusters can thereby be defined as $\overleftarrow{C} = \bigcup_{1\le i\le k} P_i$, $\lambda = |\overleftarrow{C}|$. Additionally, $\overleftarrow{c} \in \overleftarrow{C}$ denotes a cluster not originating from an input partition. Given a set of partitions $\Gamma = \{P_1, P_2, \ldots, P_k\}$, $|\Gamma| = k$, and a consensus function F, a cluster ensembler ϕ can be defined as $\phi(F, \Gamma) = F(P_1, P_2, \ldots, P_k) = F(\Gamma)$, resulting in the final partition P^*.

As to using the generic ensemble framework as a metagenomic ensemble framework, the framework was extended, such that dataset X is the set of contigs, where each contig contains a set of Single-Copy Genes (SCGs). The function $g(x_i)$ maps a contig to the set of SCGs for that contig, such that $g(x_i) \Rightarrow \{g_1, g_2, \ldots, g_j\} \cup \emptyset$. The set of all SCGs is denoted by G.

Having established the variables of the framework, we can begin to present BinChill. The workflow of the algorithm goes through three main stages; Relaxation, Bin assignment, and Bin refinement, as can be seen in Fig. 2, 3 and 4, which shows the workflow of the algorithm. Figure 1 provides an overview of full method.

A. STAGE 1: RELAXATION

Having generated Γ, the set of k partitions, this stage transforms all clusters in $\overleftarrow{C}$ into a pairwise similarity matrix S_c using set correlation as a cluster similarity measurement. Afterward, the clusters are merged in a manner reminiscent of

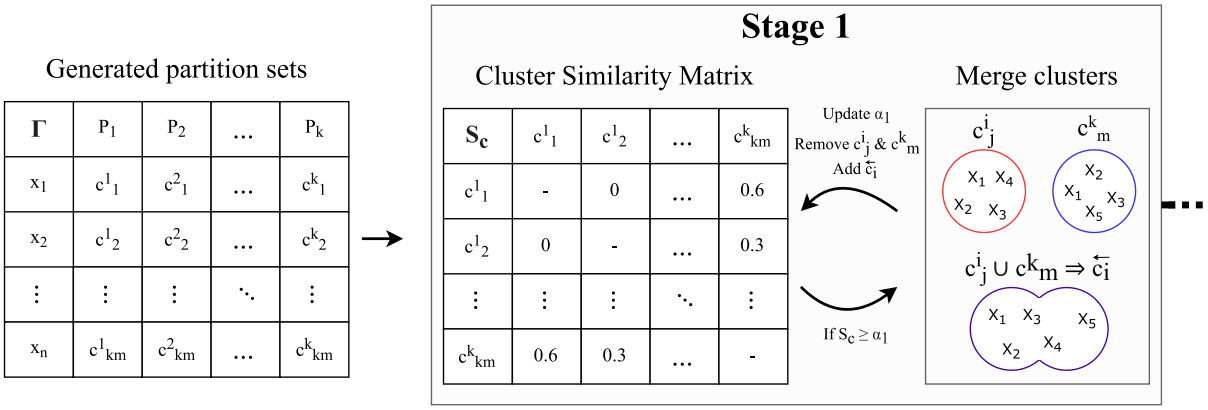

FIGURE 2. An illustrative example of Stage 1 (Relaxation) of the BinChill Method. The pairwise cluster similarity is calculated between clusters of different partitions and the most similar are merged iteratively.

relaxation (chilling), to form new ensemble clusters. An illustration of this can be found in Fig. 2.

Firstly the similarity between clusters $c^m_{j_m}, c^l_{j_l}$ is measured while also taking their sizes into account. The formula used for calculating the cluster similarity is represented as follows:

$$S_c(c^m_i, c^l_j) = \frac{|c^m_i \cap c^l_j| - \frac{|c^m_i||c^l_j|}{n}}{\sqrt{|c^m_i||c^l_i|(1-\frac{|c^m_i|}{n})(1-\frac{|c^l_j|}{n})}} \quad (2)$$

where m and l are two partition sets, $m \neq l$ and i, j are the cluster indices in m and l, with n being the number of items in X.

This results in a cluster similarity matrix that contains the similarity measure between initial cluster vectors, where $S_c(c_i, c_j) = S_c(c_j, c_i)$, hence why duplicate values are ignored. Its values are bounded in $[-1, 1]$, where a value of 1 implies that the two clusters are completely identical, and a value of -1 implies that they are a complement of each other.

Having calculated the cluster similarity matrix, we then find the most similar cluster pairs, with $S_c \geq \alpha_1$, and merge these based on the criterion in equation 3 to produce new clusters ($\overleftarrow{c_i}$), which are added to $\overleftarrow{C}$. After each iteration through the matrix S_c, the value $\alpha_1 \Leftarrow \max S_c$. This is done repeatedly, until $\alpha_1 < \alpha_{1_{min}}$. ACE does not have an implicit method for determining an optimal $\alpha_{1_{min}}$. This stage replicates stage 2 of the ACE method, which determines $\alpha_{1_{min}}$ empirically. We conduct an empirical analysis of the optimal $\alpha_{1_{min}}$ value in Section V-C.

$$\text{If } S_c(c^m_{j_m}, c^l_{j_l}) >= \alpha_1 \Rightarrow \quad c^m_{j_m} \text{ and } c^l_{j_l} \text{ are merged.} \quad (3a)$$

$$\text{If } S_c(c^m_{j_m}, c^l_{j_l}) < \alpha_1 \quad \Rightarrow \quad c^m_{j_m} \text{ and } c^l_{j_l} \text{ are not merged.} \quad (3b)$$

The process of merging continues until no clusters from S_c satisfy the merging criteria, meaning they are not similar enough. If two clusters have identical scores, then the ordering of the clusters as they appear in the input file is used as a tiebreak. This results in the set $\overleftarrow{C}$ consisting of non-merged clusters as well as several merged clusters, e.g. $\overleftarrow{C} = \{c_i, c_j, \dots\} \cup \{\overleftarrow{c_l}, \overleftarrow{c_v}, \dots\}$.

B. STAGE 2: BIN ASSIGNMENT

The aim of stage 2 is to ensure that all contigs only appear in one cluster, as can be seen in Fig. 3. To do this, the membership similarity in the new clusters should be calculated. This stage is inspired by ACE, but expands the certainty categories to include items with and without SCGs. Firstly, the membership value of each contig to the initial clusters is defined as:

$$\delta(x_i, c_j) = \begin{cases} 1 & \text{if } x_i \in c_j, \forall i = 1, \dots, n \\ 0 & \text{if } x_i \notin c_j \end{cases} \quad (4)$$

Each cluster can thereby be considered a vector of membership values, with n dimensions. Each initial cluster is defined as $c_j = [\delta(x_i, c_j) \mid x_i \in X]$, whereas a merged cluster is defined as the accumulation of the two previous clusters, such that $\overleftarrow{c_j} = c_m + c_j$. This also applies when merging two merged clusters.

We can then define the membership similarity measure, which denotes the similarity a contig has with a given cluster in $\overleftarrow{C}$. The membership similarity denoted as S_x, can be defined as;

$$S_x(x_i, \overleftarrow{c_g}) = \frac{\Theta(x_i, \overleftarrow{c_g})}{\sum_{c_u \in \overleftarrow{C}} \delta(x_i, c_u)} = \frac{\Theta(x_i, \overleftarrow{c_g})}{k} \quad (5)$$

where Θ is a function returning the cumulative membership value of item x_i in cluster $\overleftarrow{c_g}$, given by equation 4, defined as $\Theta(x_i, \overleftarrow{c_g}) = \sum_{c_u \in \overleftarrow{c_g}} \delta(x_i, c_u)$ where c_u are all clusters used to construct $\overleftarrow{c_g}$. The membership similarity matrix S_x thereby gives us the certainty of which cluster an item belongs to, in the domain [0, 1], where each contig's certainty sums to 1. We can then use these certainties throughout the following steps of this stage.

Throughout this stage, two groups are considered, θ_1 and θ_2, which represent assigned and non-assigned contigs, respectively. These groups have the property that $\theta_1 \cap \theta_2 = \emptyset$, meaning every time an item is added to θ_1, it is removed from θ_2. Initially, $\theta_1 = \emptyset$, $\theta_2 = X$. The process of this stage is

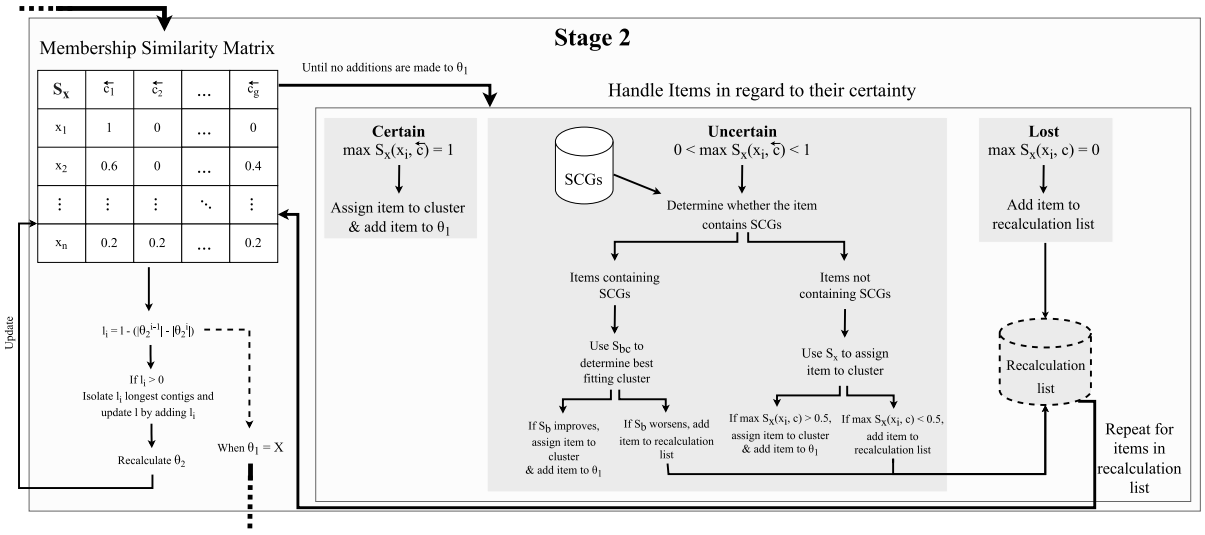

FIGURE 3. An illustrative example of Stage 2 (Bin assignment) in the BinChill method. Each item is handled according to its maximum membership similarity to newly formed clusters.

then run repeatedly, until $\theta_1 = X$. After each iteration, the membership similarity is recalculated between items in θ_1 and θ_2 using co-association [17].

1) STEP 2.1: IDENTIFY CONTIG CERTAINTY

Initially, all contigs are categorized into their respective certainty groups using S_x. For each $x \in \theta_2$, we find the maximum membership similarity in relation to all $\overleftarrow{c} \in \overleftarrow{C}$. The contig is assigned to its respective certainty group, based on the maximum membership similarity measure, with a membership similarity measure of 1 indicating that the contig is certain and 0 indicating that it is lost. The certainty group can therefore be defined as follows:

$$\begin{aligned} \max_{\overleftarrow{c} \in \overleftarrow{C}} S_x(x_i, \overleftarrow{c}) = 1 &\rightarrow \textit{certain} \\ 0 < \max_{\overleftarrow{c} \in \overleftarrow{C}} S_x(x_i, \overleftarrow{c}) < 1 &\rightarrow \textit{uncertain} \\ \max_{\overleftarrow{c} \in \overleftarrow{C}} S_x(x_i, \overleftarrow{c}) = 0 &\rightarrow \textit{lost} \end{aligned}$$

2) STEP 2.2: ASSIGN CERTAIN OBJECTS TO A SINGLE CLUSTER

Contigs that are considered certain, by definition have a member similarity measure of 1. Therefore, certain contigs can only belong to that cluster and are thus added to θ_1, which will be used to contain items that need no further calculations in this stage.

3) STEP 2.3: HANDLE UNCERTAIN ITEMS

Items that are classified as uncertain are handled in one of two ways. Items where $g(x_i) \neq \emptyset$ are handled as defined in Step 2.3.1, whereas items where $g(x_i) = \emptyset$ are handled in regards to the methods defined in section Step 2.3.2.

4) STEP 2.3.1: ASSIGN UNCERTAIN CONTIGS WITH SCGs

For uncertain items containing SCGs, we want to know whether there exists any bin, where the addition of those items will give a better result in regards to the completeness and contamination. We want to place these contigs in bins where they are not only most certain to be placed but also improve the quality of the bins in regards to the quality measure defined in Table 3. To accomplish this, we altered the scoring function described in equation 1, such that:

$$S_b(\overleftarrow{c}) = (100 \cdot com) - (100 \cdot con)^2 - \sqrt{(100 \cdot meg)} \tag{6}$$

This alteration also induces the property that only bins with $< 10\%$ contamination will have a positive score, which is consistent with the high-quality criteria (Table 3). A comparison between S_d and S_b can be seen in Table 1. This table shows that S_b heavily penalizes rises in contamination. As stage 2 is an iterative approach, it is important that a step in the wrong direction does not lead to a worse score down the line. Therefore, a small rise in contamination is heavily discouraged.

This results in only high-quality clusters and clusters with the potential to become high-quality clusters (through merging) having a score > 0. The megabin penalty is kept, although with a much lower influence, to still deter megabins.

We then use this scoring function to give a quality score for each bin. However, the similarity S_x also plays a role when assigning a contig to a bin. To determine which bin an uncertain contig containing SCGs is placed in, we use equation 7. This equation finds the bin with the highest positive change, by calculating the difference of score with and without

TABLE 1. Comparison between DASTool's score function S_d and BinChill's score function S_b, for different completeness and contamination values. Megabin penalty is ignored for simplicity, as it is often related to contamination.

Com	Con	S_d	S_b
1	0	1	100
1	0.05	0.95	75
1	0.1	0.9	0
0.9	0	0.9	90
0.9	0.05	0.85	65
0.9	0.5	0.8	-10
0.8	0.2	0.6	-320
0.5	0.02	0.48	46
0.5	0.5	0	-2450
0.01	1	-0.99	-9999

TABLE 2. Example cases of the $S_1 - S_2$ score, depending on the relation, to show how desirable a bin is, with and without a contig.

Relation	S_1	S_2	$S_1 - S_2$
$S_1 > S_2$	-1	-2	1
$S_1 > S_2$	1	-1	2
$S_1 > S_2$	1	-2	3
$S_1 > S_2$	2	1	1
$S_1 > S_2$	2	-1	3
$S_1 = S_2$	-1	-1	0
$S_1 = S_2$	1	1	0
$S_1 < S_2$	-1	1	-2
$S_1 < S_2$	-1	2	-3
$S_1 < S_2$	1	2	-1
$S_1 < S_2$	-2	-1	-1
$S_1 < S_2$	-2	1	-3

the contig.

$$S_{bc}(x_i) = \arg\max_{\overleftarrow{c_g} \in \overleftarrow{C}} (S_x(x_i, \overleftarrow{c_g}) \cdot (S_b(\overleftarrow{c_g}) - S_b(\overleftarrow{c_g} - \{x_i\}))) \tag{7}$$

Table 2 shows several different scenarios and their resulting change in score, with $S_1 = S_b(\overleftarrow{c_g})$ and $S_2 = S_b(\overleftarrow{c_g} - \{x_i\})$. Two notable cases are the $-1 - (-2) = 1$ case and $1 - 1 = 0$ case. The first case has a positive change, even though both S_1 and S_2 are negative. However, in this case, it is more desirable to have a score of S_1 than S_2, as S_2 has a worse score. The other case with a change of 0 is simpler and means that the contig does not make an impact.

With respect to the similarity, equation 7 returns the bin in which the contig has the highest positive influence. It is, however, not guaranteed that any bin has a positive score under the inclusion of the contig. In that case, it is determined if the item should be placed in θ_1 or stay in θ_2 based on whether the score of any bin is improved or not. If $S_b(\overleftarrow{c_g} \cup \{x_i\}) < 0$, where $S_{bc}(x_i) = \overleftarrow{c_g}$, then the item does not improve the score of any bins, hence why the item stays in θ_2, otherwise, the item is added to θ_1. In the context of S_{bc}, we utilize S_b as opposed to S_d to minimize contamination. Specifically, S_b only assigns positive scores to bins that have the potential to be high-quality (HQ) bins, as determined by our optimization function S_{bc}. This allows S_{bc} to iteratively construct improved bins.

5) STEP 2.3.2: ASSIGN UNCERTAIN CONTIGS WITHOUT SCGs

For uncertain contigs not containing SCGs, we look at the membership similarity S_x of the contigs. If an item has $\max_{\overleftarrow{c} \in \overleftarrow{C}} S_x(x_i, \overleftarrow{c}) > 0.5$ we assign the given item to that cluster and put it in θ_1, as it has a higher probability of being in that cluster than not. If however $\max_{\overleftarrow{c} \in \overleftarrow{C}} S_x(x_i, \overleftarrow{c}) \leq 0.5$, the item stays in θ_2 for recalculation.

6) STEP 2.4: BIN ISOLATION OF θ_2

To guarantee convergence, each iteration is ensured to assign at least l items to θ_1. Arbitrarily, $l = \lceil \sqrt{n} \rceil$ was chosen as the initial value. This is done by isolating the l_t longest contigs in θ_2, with $l_t = l - (|\theta_2^{t-1}| - |\theta_2^t|)$, where $|\theta_2^t|$ is the length of θ_2 in iteration t, since the longer a contig is the more reliable its features are [8]. A contig is isolated by placing it alone in a separate cluster, which it is added to $\overleftarrow{C}$ while the contig is added to θ_1. Initial iterations will typically have many contig assignments, which results in no more contigs being assigned through isolation. However, later iterations might have most or all assignments occur through isolation. To prevent this, each iteration that isolates contigs also increases l, such that $l = l + l_t$ iff $l_t > 0$.

7) STEP 2.5: RECALCULATION OF CONTIG SIMILARITY IN θ_2

In the last step of each iteration, the contigs of θ_2 have their similarity recalculated using co-association [17]. This calculates the average agreement between partitions in terms of the percentage of times a given pair of contigs are placed in the same cluster in each $P_m \in \Gamma$.

$$CO(x_i, x_j) = \frac{1}{k} \sum_{m=1}^{k} \delta_m(x_i, x_j) \tag{8}$$

Here, x_i and x_j are contigs, $\delta_m(x_i, x_j) = 1$ if x_i and x_j are in the same cluster in partition m, and $\delta_m(x_i, x_j) = 0$ otherwise.

Before the similarity recalculation is done, each cluster in $\overleftarrow{C}$ has all the contigs in θ_2 removed, such that $\overleftarrow{c} = \overleftarrow{c} - \{x_a\}$, $\forall x_a \in \theta_2$ and $\forall \overleftarrow{c} \in \overleftarrow{C}$. This implies that $\bigcup_{\overleftarrow{c} \in \overleftarrow{C}} \overleftarrow{c} = \theta_1$. Using co-association between contigs, we can re-determine the average similarity between contigs in θ_2 and contigs in θ_1 as:

$$ACO(x_i, \overleftarrow{c_j}) = \frac{1}{|\overleftarrow{c_j}|} \cdot \Big(\sum_{x_a \in (\overleftarrow{c_j} - \{x_i\})} CO(x_i, x_a) \Big) \tag{9}$$

We can then use $ACO(x_i, c_j)$ to calculate the average co-association between $\forall x_i \in \theta_2$ and contigs in clusters from θ_1 using the information from Γ. This way we can assign the contigs from θ_2 to clusters according to the contigs they were associated with the original partitions. Having calculated $ACO(x_i, c_j)$ for all contigs in θ_2, we then overwrite S_x such that $S_x(x_i, c_j) = ACO(x_i, c_j)$.

When $\theta_1 = X$, the stage halts, and the initial final partition is thereby $P^* = \overleftarrow{C}$, as $\bigcap_{\overleftarrow{c} \in \overleftarrow{C}} \overleftarrow{c} = \emptyset$ and $\bigcup_{\overleftarrow{c} \in \overleftarrow{C}} \overleftarrow{c} = X$

C. STAGE 3: BIN REFINEMENT

The refinement process uses the previously established set P^* and refines the bins within. This stage is run repeatedly, where bins are either split, merged, or left unchanged until no further actions can be made, as can be seen in Fig. 4. This stage is used to both increase quality of clusters in P^* and to decrease the partition size. This is often necessary as bin isolation from stage 2 can result in an explosion in partition size, i.e., a high number of bins. We developed this stage specifically for BinChill, since it was not part of the ACE or other generic ensemble clustering algorithms. For every bin, the set $\overleftarrow{C}$ (equivalent to P^* at this state), is searched in order to find an optimal partner bin to be merged with. This is done using an average co-association between bins (CCO) and bin scoring (S_b).

The Common Co-Association (CCO) of two bins is given by:

$$CCO(\overleftarrow{c_i}, \overleftarrow{c_j}) = \frac{1}{|\overleftarrow{c_i}||\overleftarrow{c_j}|} \sum_{x_k \in \overleftarrow{c_i}} \sum_{x_l \in \overleftarrow{c_j}} CO(x_k, x_l) \quad (10)$$

When searching for an optimal partner for a bin, one of two strategies will be used, depending on the type of bins being compared. If both bins contain SCGs ($|g(\overleftarrow{c})| > 0$), then a combination of bin score (S_b, equation 6) and (CCO) is utilized. However, if either bin contains no SCGs, then the bins are only compared using CCO. The function utilized for searching for an optimal pair, denoted S_{cco}, is comprised of two methods centered around scoring the compatibility between two bins. If both bins contain SCGs, then the score is based on both the combined score and CCO. The bins are only considered compatible if merging them results in a positive impact, expressed as $S_b(\overleftarrow{c_i} \cup \overleftarrow{c_j}) > max(S_b(\overleftarrow{c_i}), S_b(\overleftarrow{c_j}))$. If the combined score is not greater than the max individual score, then the two bins are not compatible.

If at least one of the bins does not contain SCGs, then only CCO is used to determine the compatibility of those two bins. In this case, we use the CCO score between them to find the optimal partner, using the constraint that combining the two bins must result in a higher score. This is also expressed as $CCO(\overleftarrow{c_i}, \overleftarrow{c_j}) \geq max(ICO(\overleftarrow{c_i}), ICO(\overleftarrow{c_j}))$. Internal co-association (ICO), is a measure of the association between contigs inside a bin, given by:

$$ICO(\overleftarrow{c}) = \frac{1}{|\overleftarrow{c}|^2 - |\overleftarrow{c}|} \sum_{x_i \in \overleftarrow{c}} \sum_{x_j \in (\overleftarrow{c} - \{x_i\})} CO(x_i, x_j) \quad (11)$$

Applying ICO to a union of two bins ($ICO(\overleftarrow{c_i} \cup \overleftarrow{c_j})$) can be reduced to $\frac{ICO(\overleftarrow{c_i}) + CCO(\overleftarrow{c_i}, \overleftarrow{c_j}) + ICO(\overleftarrow{c_j})}{3}$. When considering finding an optimal partner, only the CCO part of the equation is required to improve, as the ICO parts are static. This is also why a greater than or equal sign is used, as two separate clusters having an equal CCO to max ICO score still improves the score of the min ICO score.

The optimal partner $\overleftarrow{c_j}$ for a given bin $\overleftarrow{c_i}$ is then found by searching set $\overleftarrow{C}$ for the partner yielding the maximum score, expressed as $\overleftarrow{c_j} = \arg\max_{\overleftarrow{c} \in \overleftarrow{C}} (S_{cco}(\overleftarrow{c_i}, \overleftarrow{c}))$. These two can then be merged and added to set $\overleftarrow{C}$, such that $\overleftarrow{C} \leftarrow \overleftarrow{C} - \{\overleftarrow{c_i}, \overleftarrow{c_j}\} + \{\overleftarrow{c_i} \cup \overleftarrow{c_j}\}$

$$S_{cco}(\overleftarrow{c_i}, \overleftarrow{c_j}) \begin{cases} s_b(\overleftarrow{c_i} \cup \overleftarrow{c_j}) \cdot (CCO(\overleftarrow{c_i}, \overleftarrow{c_j}) + 1) & \text{if } Con1 \\ CCO(\overleftarrow{c_i}, \overleftarrow{c_j}) & \text{if } Con2 \\ -\infty & \text{otherwise} \end{cases}$$

$$Con1 = |g(\overleftarrow{c_i})| > 0 \;\&\; |g(\overleftarrow{c_j})| > 0 \;\& \times S_b(\overleftarrow{c_i} \cup \overleftarrow{c_j}) > max(S_b(\overleftarrow{c_i}), S_b(\overleftarrow{c_j}))$$

$$Con2 = CCO(\overleftarrow{c_i}, \overleftarrow{c_j}) \geq max(ICO(\overleftarrow{c_i}), ICO(\overleftarrow{c_j}))$$

If a bin does not match the merging criteria with any other bins, as in $S_{cco}(\overleftarrow{ci}, \overleftarrow{c_j}) = -\infty$ and $S_b(\overleftarrow{c_i}) < 0$, the bin is split into single-object bins. P^* ($\overleftarrow{C}$) is therefore updated with $\overleftarrow{c_i} = \{x_1, x_2, \ldots, x_q\} \Rightarrow \{x_1\}, \{x_2\}, \ldots, \{x_q\}$.

Bin refinement is continually applied, until no further changes are possible, meaning a local maximum is reached. After the bin refinement stage halts, we are left with the final partition P^*, which completes BinChill, such that $\phi(F, \Gamma) = \phi(BinChill, \Gamma) = P^*$.

IV. IMPLEMENTATION

As S_c is the heaviest matrix in regards to the number of entries throughout BinChill, an effort was made to limit the memory usage of this matrix. Therefore, a different sparsing strategy was chosen compared to the other matrices. In S_c, values lower than $\alpha 1_{min}$ were considered as 0. This change is made, as $S_c(c_i, c_j) = 0$ rarely occurs, meanwhile the only relevant values are that $>= \alpha 1_{min}$. This also helps improve performance when searching for entries to merge (values $>= \alpha_1$), as most entries are sparsed.

A. ANALYSIS OF TIME COMPLEXITY

A large focus of the implementation has been optimizing stage 1, as this stage initially had a lot of memory and efficiency issues. This stage has a worst-case time complexity of $O(n \cdot \lambda \cdot log\ \lambda)$, since calculating each S_c has a time complexity of $O(n)$, as it utilizes the intersection between two clusters. This function is invoked $\lambda\ log\ \lambda$ times when populating the symmetric matrix, where λ is the total number of clusters in all input partitions. The worst case is however seldom reached, as it would require a partition to consist of a single cluster for the intersection calculation to require $O(n)$ time. The bigger the clusters are (in terms of number of contigs), calculations involving these clusters become more computationally expensive, but also leaves fewer entries to be calculated.

The time complexity of stage 2, is $O(\lambda \cdot n \cdot \sqrt{n})$, assuming a distinctly worst-case, where l contigs get assigned each iteration, resulting in l never growing and the maximal possible iterations being required, namely $\sqrt{n}$. In practice however, assigning exactly l contigs each iteration is extremely unlikely, and the amount of contigs which gets assigned each iteration quickly decreases, thereby increasing l.

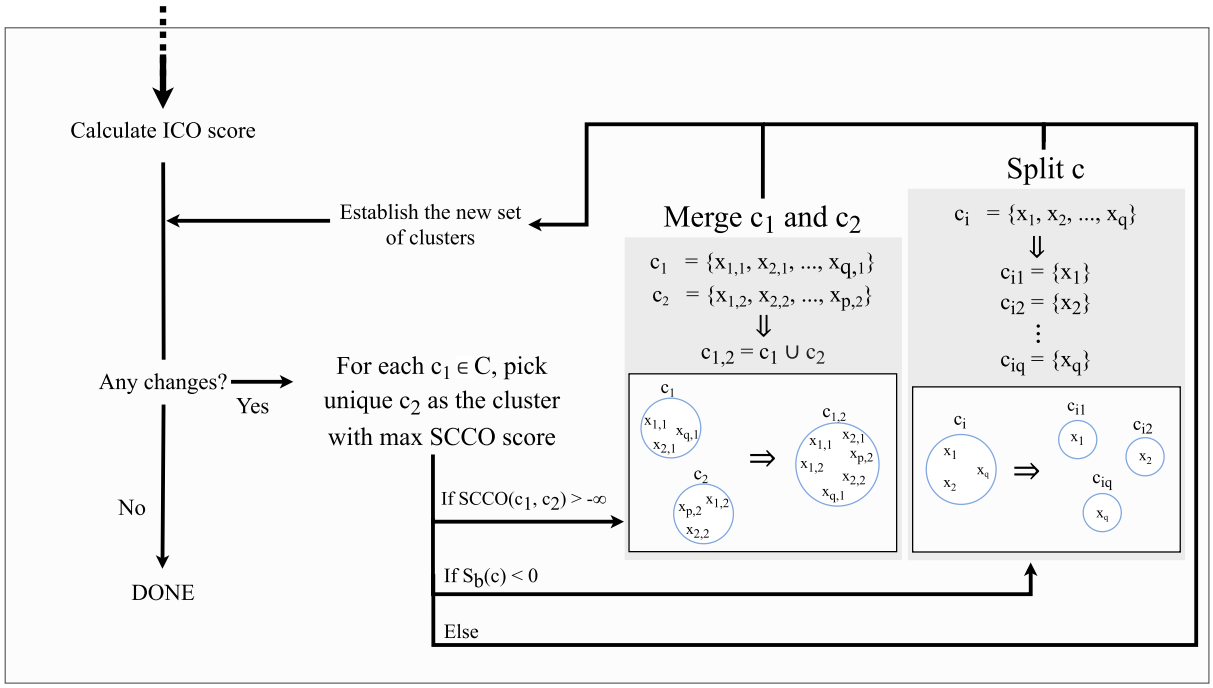

FIGURE 4. An illustrative example of Stage 3 (Bin refinement) of the BinChill method. The bins are either split, merged, or left unchanged, according to their scores.

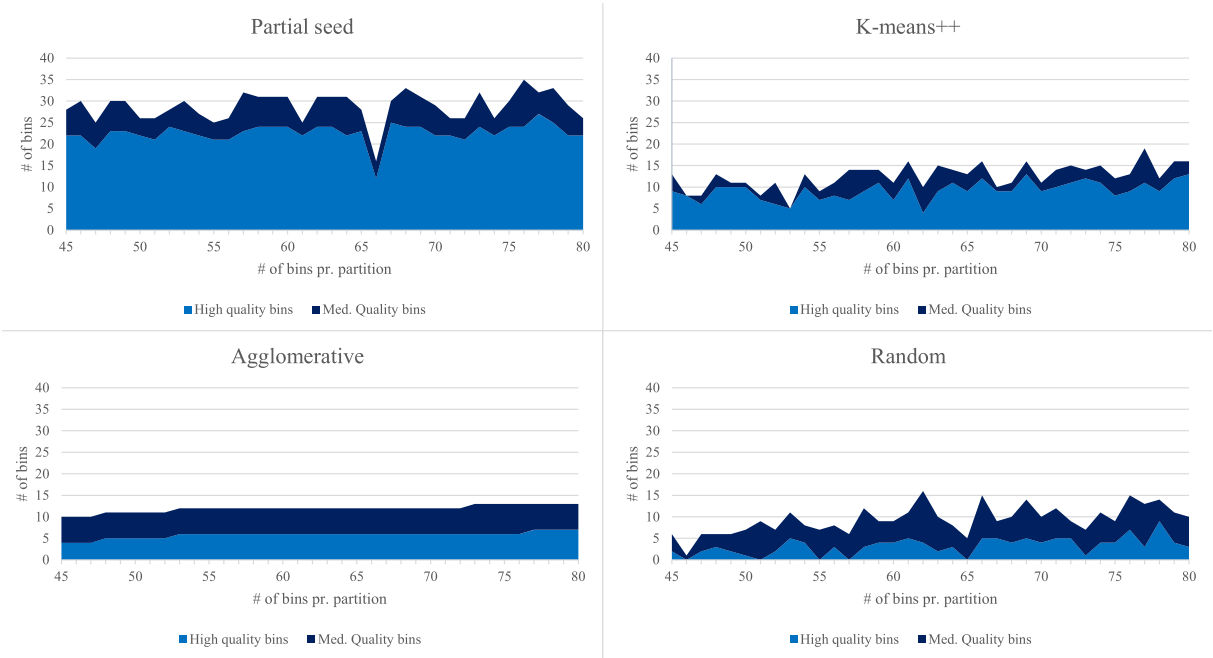

FIGURE 5. Quality assessment of bins using different clustering algorithms (Partial seed, K-means++, Agglomerative, and Random) in the BinChill Binner on Strong100. The X-axis is the number of bins for each partition, while the y-axis is the partition's quality.

Assuming a worst-case scenario for stage 3, where every cluster can be split, the maximum number of clusters becomes $O(\lambda) = n$. It will never surpass n as refining only improves the score, a contig will never be involved in a split more than once. The performance of stage 3 is heavily affected by the calculation of *CCO* and *ICO*. Both of these functions are symmetric, meaning $ICO(a, b) = ICO(b, a)$, which can help substantially speed up the implementation. Both functions can be implemented as $O(|c_1| \cdot |c_2|) = O(n^2)$ and $O(|c| \log(|c|)) = O(n \log n)$ respectively. Using these

functions to search the set $\overleftarrow{C}$ for an optimal partner can also be done symmetrically, as S_{cco} is also symmetric, meaning each iteration of the refiner takes $O(n^2 \cdot n \log n)$. This stage is iterated at most $log\ n$ times, resulting in an overall time complexity of $O(n^3 \cdot log\ n \cdot log\ n)$. This complexity is greater than that of the ACE method ($O(\lambda^2 \cdot (\lambda + n))$ [14]), which we adapted for stages 1 and 2. However, it is uncertain whether it is greater than that of other metagenomic methods, as it is seldom studied to the same degree as generic ensemble methods.

Note that this is a very strict upper bound, and the actual implementation caches ICO and S_b throughout. Likewise, subroutines can also be run concurrently, utilizing multiple threads to complete an iteration. For stage 3, searching for an optimal partner can be done in parallel, as the potential of partners can be determined concurrently (and memorized as to skip calculation later), where from the most optimal can be chosen. Stage 1 also benefits heavily from calculating S_c in parallel, while stage 2 only benefits from parallelism during recalculation (step 2.5). Stage 2 does however, benefit from caching results from subroutines, such as S_b, and S_x and CO.

B. BinChiLL AS A STANDALONE BINNER

Using a similar architecture as MetaBinner [8], we implemented a standalone binner. The binner uses MetaBinner's method for estimating the number of bins and generating partitions, using their Partial Seed method for initializing a k-means++ clustering on contig feature vectors [8]. For experimental purposes, the binner also allows for the use of other partition generation methods instead of the partial seed method. An overview of the workflow alongside a summary of the implementation of MetaBinner partition generation method can be seen in Appendix A.

V. EXPERIMENTS

In this section, we study the influence of different parameters on BinChill, alongside a comparison of its performance to standalone binners and other ensemblers. The experiments were conducted on nine different datasets, namely a smaller dataset (Strong100), consisting of 1150 contigs [16], a larger dataset (CAMI2 oral), consisting of 201k contigs [18], and seven real-world datasets. Only the Strong100 dataset was used to test the different parameters in Section V-B.

A. EVALUATION CRITERIA

The final partitions obtained with each method have been evaluated using the criteria seen in Table 3, where each bin is classified as either High-, Medium-, or Low-quality, depending on its completeness and contamination. These are the standards used in the metagenomics community, defined as the minimum information about metagenome-assembled genomes [19]. The completeness and contamination are calculated using DAS-Tools' quality assessment formula [9]. The quality of a partition is determined by the number of bins that adhere to the different quality groups, whereas the number of high-quality bins is the most significant.

TABLE 3. Quality measure of a bin using completeness and contamination.

Rank	Completeness(%)	Contamination(%)
High quality	≥ 90	≤ 5
Medium quality	≥ 50	≤ 10
Low quality	≥ 0	≤ 100

TABLE 4. Overview of different partition generation methods impact on final partition quality after running BinChill. AHPQ = Average High-Quality bins for input partitions.

Method	High	Medium	Total	AHPQ
PartialSeed	33	44	52	22.58
K-means++	32	42	58	9.25
Agglomerative	27	40	87	5.80
Random	22	41	124	3.25

B. ANALYSIS OF PARTITION QUALITY INFLUENCE

The influence of partitions with different qualities is examined with the standalone ensembler, where different clustering algorithms were used to sample the partitions. The algorithms tested were the Partial Seed method as described in [8], Agglomerative Clustering [1] with cannot-link constraints between contigs containing the same SCGs, k-means++ [20] where contig lengths are used as a weight when adjusting centroids, and Randomized embedding k-means clustering.

The MetaBinner approach was used for estimating the number of clusters [8]. This gave a lower bound estimate of 45 bins and an upper bound estimate of 80 bins. A partition was then generated for each number of clusters between 45 and 80, and for each clustering method. More details about this method can be found in Appendix B.

Table 4 contains the ensemble results, where the partitions generated with different methods were used as input. This is displayed alongside the average quality of the 36 input partitions. Each method was run using an $\alpha_{1_{min}}$ of 0.9.

A plot of the different bin qualities from the partitions can be seen in Fig. 5. The clustering algorithms differ in the similarity of their partitions measured using ARI (Adjusted Rand Index) and NMI (Normalized Mutual Information). A boxplot of all the partition's ARI and NMI scores, in relation to other partitions generated from the same method, can be seen in Fig. 6. Even though bins found through Partial seed and k-means++ had vastly different qualities, they both resulted in similar final partition quality. This may be connected to both methods having a considerably higher variety of partitions compared to the Agglomerative method, meaning that the partitions likely capture different properties of the dataset. The Agglomerative Clustering algorithm produced partitions with lower quality while having high similarities between partitions. It is uncertain whether the lower final partition quality is a consequence of either the high similarity or the low partition quality. The randomly generated bins

[1]We used the scikit learn implementation: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

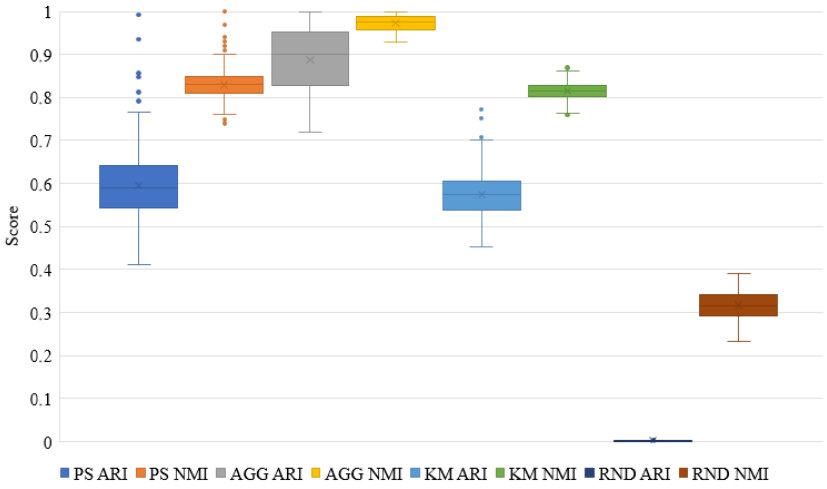

FIGURE 6. Distribution of NMI and ARI score between partitions using different clustering algorithms (Partial seed (PS), K-means++ (KM), Agglomerative (AGG), and Random (RND)) in the BinChill Binner on Strong100.

TABLE 5. Runtime of the BinChill binner on Strong100 for each clustering method.

Method	Runtime (s)
PartialSeed	23.50
K-means++	39.66
Agglomerative	25.87
Random	468.37

have a slightly lower average quality than Agglomerative while being on opposite ends of the similarity spectrum. This resulted in a lower final partition quality while taking a significantly longer time to complete, as can be seen in Table 5.

C. INFLUENCE OF EACH STAGE

Stage 1 of BinChill is run until $\alpha_1 < \alpha_{1_{min}}$. The optimal $\alpha_{1_{min}}$ is therefore influenced heavily by the partitions being ensembled. As can be seen in Fig. 7, changing the $\alpha_{1_{min}}$ value changes the resulting quality considerably, even for a small dataset such as Strong100. Therefore, a good estimate of the $\alpha_{1_{min}}$ value can be key to obtaining a good result. Furthermore, We observed that during this stage, 86.2% and 79% of the bins were merged with other bins, for the simulated and Aale dataset, respectively.

Stage 2, like stage 1, is affected by partition similarity. This stage calculates the cluster certainty of each item. This metric influences which strategy is used to assign an item to a cluster, as can be seen in Fig. 8. Agglomerative had the highest partition similarity, which correlated to it also having a larger amount of items being assigned as certain items, while the remaining methods had close to none assigned this way. As expected, the randomized partitions had a large majority of its items assigned through lost items, while a minority of the items in other methods were lost.

Lastly, stage 3 has the effect of lowering the final number of bins and increasing the number of high-quality bins. We ran the experiments from Table 4 without the refiner, to gauge its impact. The inclusion of the refiner resulted in an average improvement of 46% more HQ bins. The different results of each method can be seen in Fig. 9.

D. COMPARISON TO OTHER BINNERS AND ENSEMBLERS

This subsection evaluates the ability of BinChill to reconstruct genomes, compared to other binners and ensemblers. The ensemblers were run on five different VAMB partitions. These partitions were sampled using the same trained model, however using different seeds for the clustering, to get varied partitions. The ARI and NMI are calculated on the partitions, as to ensure that these were dissimilar enough to be used by an ensembler, as can be seen in Fig. 10. VAMB was chosen, due to its non-deterministic output and satisfactory performance relative to other binning methods. The randomization aspect of VAMB was also a factor in its selection to generate diverse partitions.

We compare the results of BinChill to the metagenomic binners and ensemblers mentioned in Section II, along with the standalone ensembler, MetaBAT2, and two generic ensemblers, namely CSPA (Cluster-based Similarity Partitioning Algorithm) [21] and HBGF (Hybrid Bipartite Graph Formulation) [22], as to give its relative performance to generic clustering ensemblers, that do not use domain-specific information. BinChill was run using an $\alpha_{1_{min}}$ score of 0.9.

All experiments were executed on the same computer, running on 4 cores, clocked at 2GHz, and 32 GB of memory. As VAMB's results varied when using different seeds, the average performance of the binner is used for evaluation. Likewise, the runtime of VAMB is the average of all five runs.

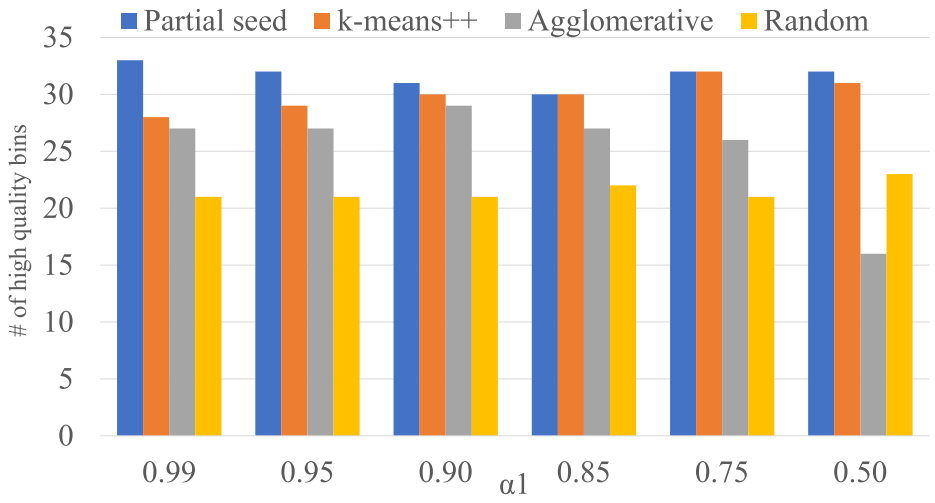

FIGURE 7. Number of high quality bins using different $\alpha_{1_{min}}$ values and clustering methods (Strong100).

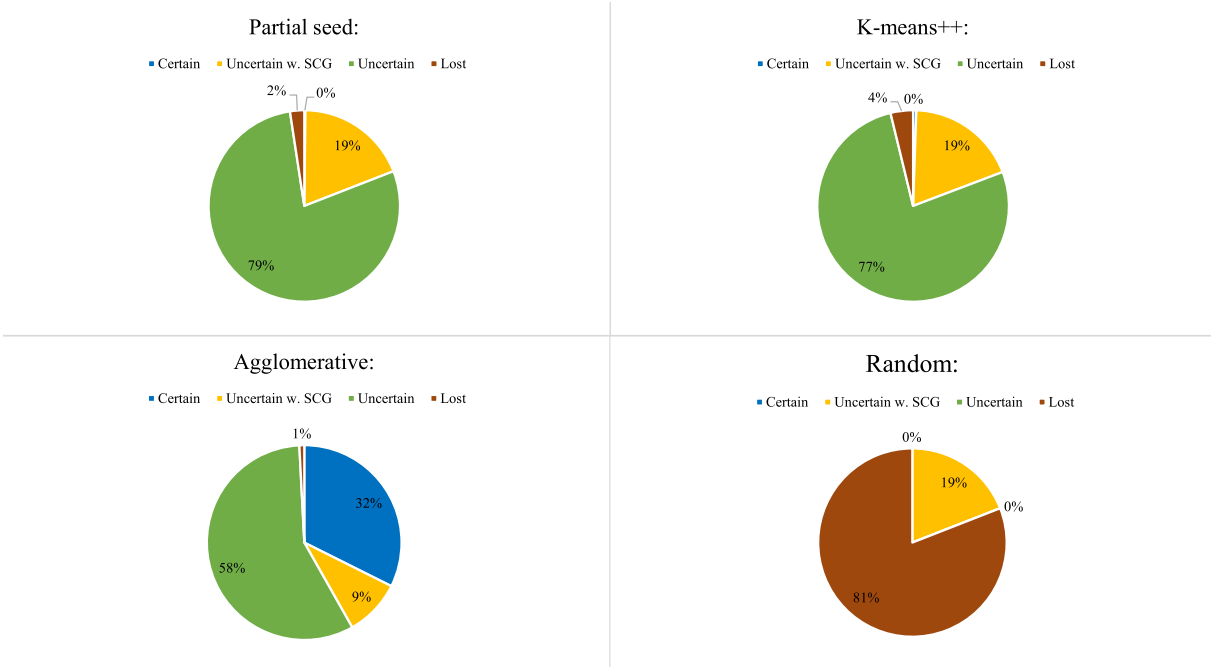

FIGURE 8. Percentage of contigs classified in regards to certainty through stage 2 of the BinChill Ensembler using different clustering algorithms (Partial seed, K-means++, Agglomerative, and Random) in the BinChill Binner on Strong100.

The results of the experiments, performed on Strong100, can be seen in Table 6. The results show that BinChill has a significantly faster runtime than other metagenomic ensemblers and binners while producing similar or better results. Compared to the generic ensemblers (HBGF, CSPA, and ACE), the runtime was similar, but BinChill outperformed them in reconstructing high-quality bins.

The results of the experiments, performed on CAMI2 oral, can be seen in Table 7. Here, BinChill was able to reconstruct more high- and medium-quality bins than any other method tested. When compared to the second best method, MetaBinner, BinChill took much less time and obtained better results as a standalone ensembler. As an ensembler of VAMB partitions, it obtained even more high-quality bins, however, this was at the expense of a longer runtime than Metabinner. Table 8 provides the runtimes of each stage of BinChill standalone binner.

We tested BinChill on real-world datasets obtained from wastewater and soil samples, sequenced with Nanopore technology [16]. Each of these datasets corresponds to a different wastewater treatment facility, except for the Soil dataset, which corresponds to a single soil sample. Soil samples

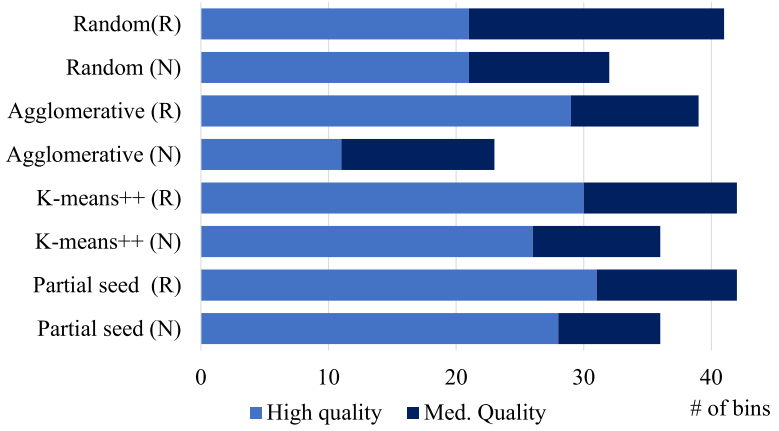

FIGURE 9. Different partitions' ensemble results with(R) and without(N) the refiner on the final result (Strong100)).

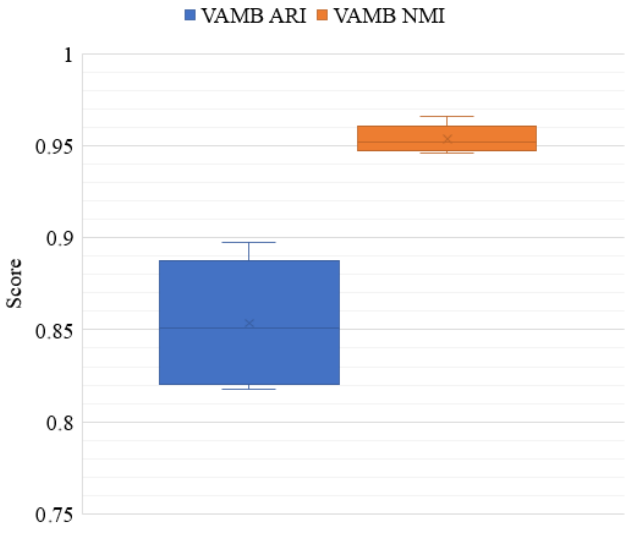

FIGURE 10. Distribution of NMI and ARI score between partitions using VAMB on Strong100.

TABLE 6. Overview of the results from the experiments performed on Strong100. S = Single binner, SE = Standalone Ensembler, E = Ensembler.

Type	Method	Runtime	High	Medium	Total
S	VAMB	0h:01m	18	21	380
S	MetaBAT2	<0h:01m	31	37	71
SE	BinChill	0h:02m	31	42	105
SE	MetaBinner	1h:43m	33	37	41
E	HBGF	<0h:01m	19	29	398
E	CSPA	<0h:01m	17	25	398
E	ACE	0h:02m	15	20	351
E	DAS-Tool	0h:12m	18	22	24
E	BinChill	<0h:01m	32	43	131

TABLE 7. Overview of the results from the experiments, run on CAMI2 oral. S = Single Binner, SE = Standalone Ensembler, E = Ensembler.

Type	Method	Runtime	High	Medium	Total
S	VAMB	3h:51m	24	45	63264
S	MetaBAT2	1h:04m	9	17	129
SE	BinChill	3h:38m	267	665	9776
SE	MetaBinner	38h:49m	236	304	378
E	HBGF	0h:08m	218	270	64105
E	CSPA	0h:06m	226	280	64105
E	ACE	72h:09m	22	45	60597
E	DAS-Tool	16h:26m	31	50	99
E	BinChill	55h:42m	333	573	28878

TABLE 8. Overview of the runtime for each step in the BinChill Standalone Binner (SE) and Ensembler (E) on the CAMI oral dataset.

Step	SE (hh:mm:ss)	E (hh:mm:ss)
Read fasta and SCG	00:02:08	00:02:32
Generate Partitions(6)	00:38:23	-
Stage 1	00:00:55	45:16:24
Stage 2	00:46:45	05:45:49
Stage 3	02:11:37	04:36:56
Total	03:38:45	55:42:39

TABLE 9. Overview of the results of the ensembler on real-world datasets.

Dataset	# items	Runtime	High	Medium	Total
Aale	45843	3:54:51	110	284	3847
Damh	38578	2:31:50	109	312	3026
Hade	79583	13:53:17	132	412	7346
Hjor	21589	0:46:12	49	151	1921
Mari	41558	2:02:29	82	267	4021
Soil	47062	13:36:35	33	224	2439
Viby	27109	0:50:25	60	226	2109

are known to be more challenging to solve than wastewater and other metagenomic datasets. We used the AalE dataset to compare the approaches in terms of high-quality bins (Table 10), while the results and runtimes of the other datasets are provided in Table 9). These results show that our approach is also effective on real-world data, and scales reasonably well on larger and more complex datasets.

We computed the similarity of the partitions generated by each method using the NMI, to observe if there was some degree of overlap between the clusters of different methods (Figure 11). We noticed that the Metabinner clusters were

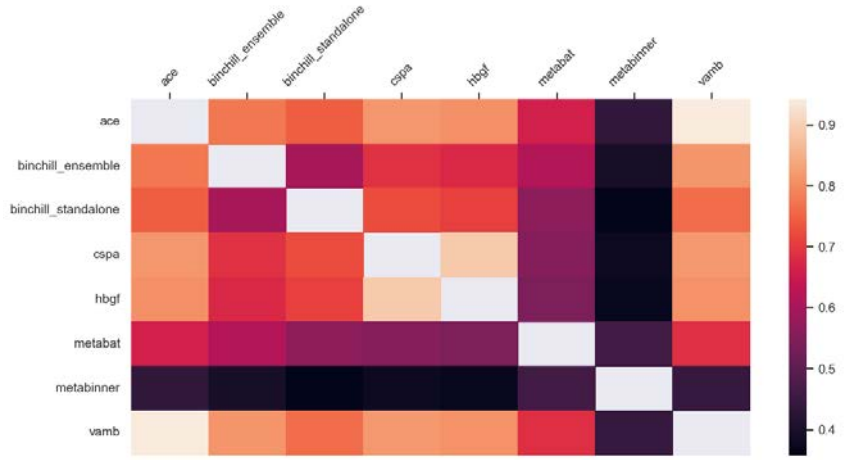

FIGURE 11. NMI scores of each method against every other method, for the AalE dataset.

TABLE 10. Overview of the results from the experiments, run on the Aale real-world dataset. S = Single Binner, SE = Standalone Ensembler, E = Ensembler.

Type	Method	Runtime	High	Medium	Total
S	VAMB	0:52:51	64	112	17549
S	MetaBAT2	0:14:33	70	105	617
SE	BinChill	0:50:23	24	370	2815
SE	MetaBinner	10:29:16	90	119	255
E	HBGF	0:01:24	3	40	17655
E	CSPA	0:00:55	3	46	17656
E	ACE	4:41:56	50	97	20635
E	DAS-Tool	2:45:50	66	115	237
E	BinChill	3:54:51	110	284	3847

the most different, compared to the other methods, followed by MetaBAT2. These two methods obtained fewer HQ bins than our BinChill method, but more than every other method. The BinChill ensemble method generated a partition that had more in common with VAMB and the other generic ensemble methods. This was expected since we used VAMB to generate partitions, and used a clustering ensemble approach to combine those partitions. Due to the low similarity of our method with MetaBAT and Metabinner, it is possible that adding those partitions to our methods would have achieved lower results, as our experiments show that the input partitions should be similar.

VI. LIMITATIONS AND FUTURE WORK

Currently, $\alpha_{1_{min}}$ is manually set by the user, which could lead to suboptimal results, as the results change based on the value. Therefore, an effort to eliminate $\alpha_{1_{min}}$ as a user parameter, utilizing domain and cluster information could be explored. We experimented with taking the average score of all clusters multiplied by their respective similarity. Whenever the average score started to decline, the value of $\alpha_{1_{min}}$ would be set. However, this approach needs further refinement to be fully utilized.

More domain information could be incorporated into BinChill, in an attempt to improve the results. This could be information such as bin size, to get a MAG that closer resembles the average genome. We tried to implement a naive scoring function to determine how close a bin is to the average genome size. However, with this, another problem arose, as the different kinds of bacteria have different average genome sizes [23]. This means that a bin could be stuck in a local maximum, where the score would have difficulties moving from one peak to another. Therefore, this idea was put on hold, as it needed more development to be functional.

Currently, the user is required to have the Single-Copy Genes to run BinChill. This is not an ideal solution, as users first need to run another program, as to obtain these SCGs, whereafter they then need to start BinChill. Therefore it could be beneficial for the workflow to incorporate the retrieving of SCGs into BinChill. This could be achieved by implementing CheckM [5] as a subprocess to the workflow.

The method used to evaluate a partition's quality is using the formulas for completeness and contamination used in DAS-Tool [9]. However, this approach only considers the bins' quality in terms of SCGs and not according to ground truth, in the cases where one would be available (simulated datasets). In those cases, we do not know how close the final partitions are to the ground truth. We do however still consider our measurement of completeness and contamination sufficient, as contigs are only binned together in the final partition, if they have a relation in the input partitions, as measured by *CO* (Equation 8). Moreover, this evaluation approach is also limited, in the aspect that not all contigs contain SCGs, meaning only some impact the quality evaluations. However, in BinChill, contigs that do not contain SCGs will still have an impact on the final partition through its relations with other contigs. Finally, an evaluation based on SCGs works on both simulated and real-world datasets.

VII. CONCLUSION

The metagenomic ensembler proposed in this paper, BinChill, provides an object co-occurrence ensemble method

using domain information, in the form of SCGs to ensemble and refine bins from multiple partitions. Member similarity alongside co-association, in combination with a bin scoring function adapted from the DAS-Tool scoring function, is utilized as a consensus function for the ensembler. Moreover, the ensembler also contains a standalone binner, exploring several clustering algorithms.

Both the binner and ensembler produced on par or better results compared to other binners and ensemblers, both in terms of quality and performance, on the small dataset, Strong100. On the CAMI Oral dataset, a large-scale simulated dataset, we obtained better results than the other methods, both with the standalone binner and with the ensembler. However, the improvement obtained with the ensembler approach came at the cost of higher runtimes, showing that the scalability of the method could be improved. Furthermore, the results on real-world datasets show how the method also outperforms on this type of data, which is in general noisier than simulated datasets. Generally, the object co-occurrence approach shows promising results, within the field of metagenomic ensemblement, and could possibly show even better results with further optimizations.

APPENDIX A
IMPLEMENTATION OF MetaBinner's PARTITION GENERATION METHOD IN THE BinChill BINNER

The feature vectors constructed in step 1 of the BinChill Binner use the normalized composition as an embedding vector in contrast to METABinner, which uses the composition, abundance, and combined composition and abundance as feature vectors. MetaBinner estimates the number of bins, by first finding the lower bound for a number of bins, k_0. This is done by measuring the number of times each SCG is present in the contigs. Sorted from least present SCG to most, the third quartile is chosen as the k_0 value. Afterward, a range of numbers larger than k_0 is tried as the k value in k-means++. The bin count with the highest silhouette score is chosen as K, which is the upper bound of the bin count.

A partition is then generated using each k in the range $k0$ to K, using MetaBinner's partial seed method for initializing k-means++. One SCG is chosen, and all contigs containing that SCG are initialized as centroids. If more centroids are required, k-means++ is used to initialize the rest. While adjusting the centroids in the k-means++ method, the contig lengths are used as weights [8]. Fig. 12 illustrates how the BinChill Binner is integrated with the ensemble method.

APPENDIX B
EXAMPLE OF BinChill METHOD

We illustrate an example of how the stages of BinChill work and how they cooperate using a simple example. This will be done adapting the toy example from the ACE paper [14], where we have a dataset of $X = \{x_1, x_2, \ldots, x_{10}\}$, of which 3 partitions are generated, as seen in Figure 13. Additionally, we can define the SCGs $G = \{g_1, g_2, g_3\}$. We then have the mapping $g(x_1) = \{g_1, g_2\}$, $g(x_2) = \{g_3\}$, $g(x_3) = \{g_1\}$, $g(x_4) = \{g_2\}$, $g(x_7) = G$ and $g(x_{10}) = \{g_3\}$.

TABLE 11. The result of θ_1 after we merge the most similar clusters, which are $\overleftarrow{c}_1 = \{c_1^1 + c_2^3\}$, $\overleftarrow{c}_3 = \{c_2^1 + c_1^3\}$, and $\overleftarrow{c}_4 = \{c_3^3 + c_3^1 + c_1^2\}$.

	c_2^2	c_2^3	$\overleftarrow{c}_1$	$\overleftarrow{c}_3$	$\overleftarrow{c}_4$
x_1	0	$\frac{1}{3}$	0	$\frac{2}{3}$	0
x_2	0	$\frac{1}{3}$	0	$\frac{2}{3}$	0
x_3	0	0	0	$\frac{1}{3}$	$\frac{2}{3}$
x_4	0	0	0	0	1
x_5	0	0	0	0	1
x_6	0	$\frac{1}{3}$	$\frac{2}{3}$	0	0
x_7	$\frac{1}{3}$	0	$\frac{2}{3}$	0	0
x_8	$\frac{1}{3}$	0	$\frac{2}{3}$	0	0
x_9	0	$\frac{1}{3}$	$\frac{2}{3}$	0	0
x_{10}	$\frac{1}{3}$	0	0	$\frac{2}{3}$	0

TABLE 12. Table showcasing the S_{bc} score for uncertain items with SCGs, with the best scores being underlined.

Assignment	Score
$x_1 \in c_2^3$	<u>22.22</u>
$x_1 \in \overleftarrow{c}_3$	20.63
$x_2 \in c_2^3$	<u>11.11</u>
$x_2 \in \overleftarrow{c}_4$	-3.85
$x_3 \in \overleftarrow{c}_3$	11.11
$x_3 \in \overleftarrow{c}_4$	<u>22.22</u>
$x_7 \in c_2^2$	20.30
$x_7 \in \overleftarrow{c}_1$	<u>66.67</u>
$x_{10} \in c_2^2$	11.11
$x_{10} \in \overleftarrow{c}_3$	<u>22.22</u>

For stage 1, we take our set of clusters $\overleftarrow{C}$, and build the initial S_c matrix, as seen in table 15. Using $\alpha_{1_{min}} = 0.7$ and an initial $\alpha_1 = 1.0$, we merge the clusters with the S_c score greater than α_1, namely c_1^1 and c_2^3 alongside c_3^1 and c_1^2, and replace them with $\overleftarrow{c_1}$ and $\overleftarrow{c_2}$ respectively. α_1 is then updated to $\max(S_c) = 0.802$. Since $\alpha_1 > \alpha_{1_{min}}$, another iteration is required. c_2^1 and c_1^3 are merged to create $\overleftarrow{c_3}$, as can be seen from the S_c matrix in table 16. Next iteration, as seen in table 17, $\alpha_1 = S_c = 0.763$, which is greater than 0.7, and we then merge c_3^3 and $\overleftarrow{c_2}$ into $\overleftarrow{c_4}$. Afterwards, $\max(S_c) < \alpha_{1_{min}}$, as $\max(S_c) = 0.356$, as can be seen in table 18, which indicates the end of stage 1. This means we end with the following soft clustering, each getting renamed to $\overleftarrow{c_i}$ to distinguish from the input clusters. At the end of this stage we have $\overleftarrow{C} = \{\{8, 10, 7\}, \{1, 2, 6, 9\}, \{6, 7, 8, 9\}, \{1, 2, 3, 10\}, \{3, 4, 5\}\}$.

For the second stage, we start by assigning each contig into one of the four certainty groups. From Table 11, we can see that x_4 and x_5 are certain, while the rest are uncertain. The uncertain category is additionally split into the groups with or without SCGs. x_1, x_2, x_3, x_7 and x_{10}, are uncertain with SCGs, while x_6, x_8 and x_9 are uncertain without SCGs. Certain contigs already only belong to a single cluster, and therefore no further action is required. For the remaining clusters, we are going to maximize $S_x \cdot S_b$ through S_{bc} or just maximize S_x.

For the uncertain contigs with SCGs, we maximise $S_x \cdot S_b$ through S_{bc}. Running this iteratively through all items in

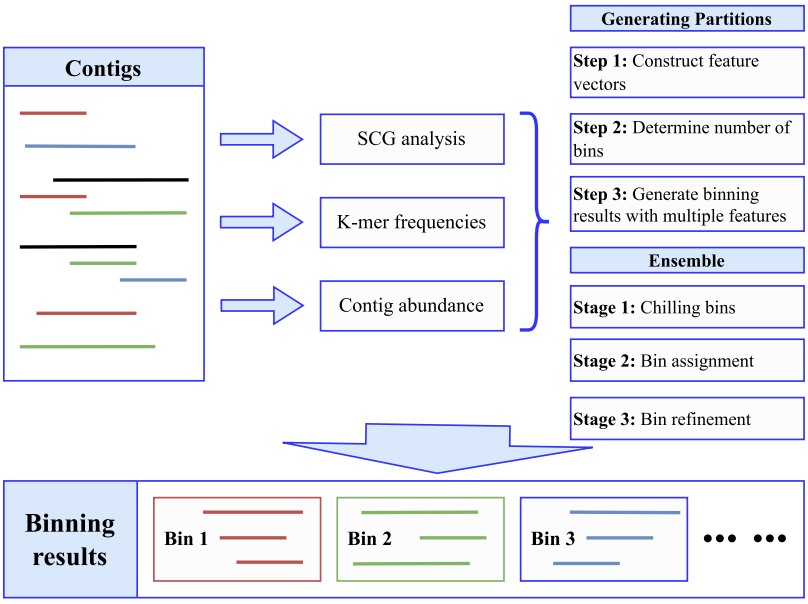

FIGURE 12. An illustrative of BinChill Binner.

objects	m_1	m_2	m_3
x_1	2	3	1
x_2	2	3	1
x_3	3	1	1
x_4	3	1	3
x_5	3	1	3
x_6	1	3	2
x_7	1	2	2
x_8	1	2	2
x_9	1	3	2
x_{10}	2	2	1

c_1^1	c_2^1	c_3^1	c_1^2	c_2^2	c_3^2	c_1^3	c_2^3	c_3^3
0	1	0	0	0	1	1	0	0
0	1	0	0	0	1	1	0	0
0	0	1	1	0	0	1	0	0
0	0	1	1	0	0	0	0	1
0	0	1	1	0	0	0	0	1
1	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	1	0
1	0	0	0	1	0	0	1	0
1	0	0	0	0	1	0	1	0
0	1	0	0	1	0	1	0	0

FIGURE 13. Example data from the ACE paper [14], which will be used to showcase how the stages of BinChill work.

TABLE 13. Stage 2 uncertain items with no SCGs, with the clusters achieving the best scores being the cluster the item is assigned to.

Assignment	Score
$x_6 \in c_2^3$	$\frac{1}{3}$
$x_6 \in \overleftarrow{c_1}$	$\frac{1}{3}$
$x_8 \in c_2^2$	$\frac{1}{3}$
$x_8 \in \overleftarrow{c_1}$	$\frac{1}{3}$
$x_9 \in c_2^3$	$\frac{1}{3}$
$x_9 \in \overleftarrow{c_1}$	$\frac{1}{3}$

TABLE 14. The CO Matrix of the contigs, compared to each other.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1	-	1	$\frac{1}{3}$	0	0	$\frac{1}{3}$	0	0	$\frac{1}{3}$	$\frac{2}{3}$
x_2	1	-	$\frac{1}{3}$	0	0	$\frac{1}{3}$	0	0	$\frac{1}{3}$	$\frac{2}{3}$
x_3	$\frac{1}{3}$	$\frac{1}{3}$	-	$\frac{2}{3}$	$\frac{2}{3}$	0	0	0	0	0
x_4	0	0	$\frac{2}{3}$	-	1	0	0	0	0	0
x_5	0	0	$\frac{2}{3}$	1	-	0	0	0	0	0
x_6	$\frac{1}{3}$	$\frac{1}{3}$	0	0	0	-	$\frac{2}{3}$	$\frac{2}{3}$	1	0
x_7	0	0	0	0	0	$\frac{2}{3}$	-	1	$\frac{2}{3}$	$\frac{1}{3}$
x_8	0	0	0	0	0	$\frac{2}{3}$	1	-	$\frac{2}{3}$	$\frac{1}{3}$
x_9	$\frac{1}{3}$	$\frac{1}{3}$	0	0	0	1	$\frac{2}{3}$	$\frac{2}{3}$	-	0
x_{10}	$\frac{2}{3}$	$\frac{2}{3}$	0	0	0	0	$\frac{1}{3}$	$\frac{1}{3}$	0	-

this group, we get the results as seen in Table 12. Assigning each contig to the cluster with the maximum positive score, we end up with the following cluster set: $\overleftarrow{C} = \{\{8\}, \{1, 2, 6, 9\}, \{6, 8, 9, 7\}, \{10\}, \{4, 5, 3\}\}$.

Afterwards, we consider the contigs uncertain group with no SCGs, we maximize only the S_x score, and only assign it if $S_x > 0.5$. The results can be seen in Table 13. This leaves us with the following cluster set: $\overleftarrow{C} = \{\{\}, \{1, 2\}, \{7, 6, 8, 9\}, \{10\}, \{4, 5, 3\}\}$. Note here, that $\overleftarrow{C}$ does not have any clusters with overlapping elements and that

TABLE 15. The Similarity Matrix S_c, which is the result of measuring the similarity between initial cluster vectors in our examples (Figure 13) using S_c measure. "-" cells indicate that this similarity is not calculated, as they are placed in the same member.

	c_1^1	c_2^1	c_3^1	c_1^2	c_2^2	c_3^2	c_1^3	c_2^3	c_3^3
c_1^1	-	-	-	-0.535	0.356	0.167	-0.667	1	-0.408
c_2^1	-	-	-	-0.429	0.048	0.356	0.802	-0.535	-0.327
c_3^1	-	-	-	1	-0.429	-0.535	-0.089	-0.535	0.764
c_1^2	-0.535	-0.429	1	-	-	-	-0.089	-0.535	0.764
c_2^2	0.356	0.048	-0.429	-	-	-	0.764	0.356	-0.327
c_3^2	0.167	0.356	-0.535	-	-	-	0.167	0.167	-0.408
c_1^3	-0.667	0.802	-0.089	-0.089	0.764	0.167	-	-	-
c_2^3	1	-0.535	-0.535	-0.535	0.356	0.167	-	-	-
c_3^3	-0.408	-0.327	0.764	0.764	-0.327	-0.408	-	-	-

TABLE 16. The Similarity Matrix S_c, which is the result of measuring the similarity between cluster vectors after first iteration has concluded using S_c measure. "-" cells indicate that this similarity is not calculated, as they are placed in the same member.

	c_1^2	c_2^2	c_2^3	c_3^1	c_3^3	$\overleftarrow{c_1}$	$\overleftarrow{c_2}$
c_1^2	-	-	0.356	0.802	-0.317	-0.535	-0.429
c_2^2	-	-	-0.535	-0.089	-0.327	0.356	-0.429
c_2^3	0.356	-0.535	-	0.167	-	0.167	-0.535
c_3^1	0.802	-0.089	0.167	-	-0.408	-0.667	-0.089
c_3^3	-0.317	-0.327	-	-0.408	-	-0.408	0.764
$\overleftarrow{c_1}$	-0.535	0.356	0.167	-0.667	-0.408	-	-0.535
$\overleftarrow{c_2}$	-0.429	-0.429	-0.535	-0.089	0.764	-0.535	-

TABLE 17. The Similarity Matrix S_c, which is the result of measuring the similarity between cluster vectors after second iteration has concluded using S_c measure. "-" cells indicate that this similarity is not calculated, as they are placed in the same member.

	c_2^2	c_2^3	c_3^3	$\overleftarrow{c_1}$	$\overleftarrow{c_2}$	$\overleftarrow{c_3}$
c_2^2	-	-0.535	-0.327	0.356	-0.429	-0.089
c_2^3	-0.535	-	-	0.167	-0.535	0.167
c_3^3	-0.327	-	-	-0.408	0.764	-0.409
$\overleftarrow{c_1}$	0.356	0.167	-0.408	-	-0.535	-0.667
$\overleftarrow{c_2}$	-0.429	-0.535	0.764	-0.535	-	-0.089
$\overleftarrow{c_3}$	-0.089	0.167	-0.409	-0.667	-0.089	-

TABLE 18. The Similarity Matrix S_c, which is the result of measuring the similarity between cluster vectors after third iteration has concluded using S_c measure. "-" cells indicate that this similarity is not calculated, as they are placed in the same member.

	c_2^2	c_2^3	$\overleftarrow{c_1}$	$\overleftarrow{c_3}$	$\overleftarrow{c_4}$
c_2^2	-	-0.535	0.356	-0.089	-0.429
c_2^3	-0.535	-	0.167	0.167	-0.535
$\overleftarrow{c_1}$	0.356	0.167	-	-0.667	-0.535
$\overleftarrow{c_3}$	-0.089	0.167	-0.667	-	-0.089
$\overleftarrow{c_4}$	-0.429	-0.535	-0.535	-0.089	-

$\overleftarrow{C}$ has an empty set, which can be removed for clarity. This concludes stage 2 where $P^* = \overleftarrow{C}$.

During stage 3, we refine P^* to achieve more optimal results using the S_{cco} functions until convergence. For this example, this is rather quick, as it does so after only one iteration. The CO matrix can be seen in Table 14. For all $\overleftarrow{c_i}, \overleftarrow{c_j} \in \overleftarrow{C}$ where $\overleftarrow{c_i} \neq \overleftarrow{c_j}$, we utilize the S_{cco} function to score the compatibility between clusters (see Section III-C). Note that S_{cco} is symmetric, and we therefore do not need to perform the calculation for any two clusters twice.

Here, $\overleftarrow{c_1}$ and $\overleftarrow{c_2}$ both only have $-\infty$ compatibility scores, but both have positive S_b scores and are therefore left unchanged. The score of $\overleftarrow{c_3}$ and $\overleftarrow{c_4}$ is 100, and are therefore merged into a new $\overleftarrow{c_3}$. This leaves $\overleftarrow{C} = \{\{1, 2\}, \{7, 6, 8, 9\}, \{3, 4, 5, 10\}\}$. After another iteration of stage 3, we see that all clusters are left unchanged, and stage 3 has therefore converged. Testing the score of the clusters, we see the following:

$$S_b(\overleftarrow{c_1}) = 100$$
$$S_b(\overleftarrow{c_2}) = 100$$
$$S_b(\overleftarrow{c_3}) = 100$$

ACKNOWLEDGMENT

(Oliver S. Bak, Marcus D. Jensen, Frederik M. Trudslev, and Andreas Windfeld contributed equally to this work.)

REFERENCES

[1] C. Yang, D. Chowdhury, Z. Zhang, W. K. Cheung, A. Lu, Z. Bian, and L. Zhang, "A review of computational tools for generating metagenome-assembled genomes from metagenomic sequencing data," *Comput. Struct. Biotechnol. J.*, vol. 19, pp. 6301–6314, Jan. 2021.

[2] A. A. Akinsemolu, "The role of microorganisms in achieving the sustainable development goals," *J. Cleaner Prod.*, vol. 182, pp. 139–155, May 2018.

[3] K. Timmis, W. M. De Vos, J. L. Ramos, S. E. Vlaeminck, A. Prieto, A. Danchin, W. Verstraete, De V. Lorenzo, S. Y. Lee, H. Brussow, and J. K. Timmis, "The contribution of microbial biotechnology to sustainable development goals, *Microbial Biotechnol.*, vol. 10, no. 5, pp. 984–987, 2017.

[4] J. Alneberg, B. S. Bjarnason, I. de Bruijn, M. Schirmer, J. Quick, U. Z. Ijaz, L. Lahti, N. J. Loman, A. F. Andersson, and C. Quince, "Binning metagenomic contigs by coverage and composition," *Nature Methods*, vol. 11, no. 11, pp. 1144–1146, Nov. 2014.

[5] D. H. Parks, M. Imelfort, C. T. Skennerton, P. Hugenholtz, and G. W. Tyson, "CheckM: Assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes," *Genome Res.*, vol. 25, no. 7, pp. 1043–1055, Jul. 2015.

[6] J. N. Nissen, J. Johansen, R. L. Allesøe, C. K. Sønderby, J. J. A. Armenteros, C. H. Grønbech, L. J. Jensen, H. B. Nielsen, T. N. Petersen, O. Winther, and S. Rasmussen, "Improved metagenome binning and assembly using deep variational autoencoders," *Nature Biotechnol.*, vol. 39, no. 5, pp. 555–560, May 2021.

[7] L.-X. Chen, K. Anantharaman, A. Shaiber, A. M. Eren, and J. F. Banfield, "Accurate and complete genomes from metagenomes," *Genome Res.*, vol. 30, no. 3, pp. 315–333, Mar. 2020.

[8] Z. Wang, P. Huang, R. You, F. Sun, and S. Zhu, "MetaBinner: A high-performance and stand-alone ensemble binning method to recover individual genomes from complex microbial communities," *Genome Biol.*, vol. 24, no. 1, p. 1, 2023.

[9] C. M. K. Sieber, A. J. Probst, A. Sharrar, B. C. Thomas, M. Hess, S. G. Tringe, and J. F. Banfield, "Recovery of genomes from metagenomes via a dereplication, aggregation and scoring strategy," *Nature Microbiol.*, vol. 3, no. 7, pp. 836–843, May 2018.
[10] M. R. Olm, C. T. Brown, B. Brooks, and J. F. Banfield, "DRep: A tool for fast and accurate genomic comparisons that enables improved genome recovery from metagenomes through de-replication," *ISME J.*, vol. 11, no. 12, pp. 2864–2868, Dec. 2017.
[11] G. V. Uritskiy, J. DiRuggiero, and J. Taylor, "MetaWRAP—A flexible pipeline for genome-resolved metagenomic data analysis," *Microbiome*, vol. 6, no. 1, pp. 1–13, Dec. 2018.
[12] A. Topchy, A. K. Jain, and W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, Dec. 2005.
[13] S. Vega-Pons and J. Ruiz-Shulcloper, "A survey of clustering ensemble algorithms," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 25, no. 3, pp. 337–372, May 2011.
[14] T. Alqurashi and W. Wang, "Clustering ensemble method," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 6, pp. 1227–1246, 2019.
[15] H. Q. Bao and T. Van Hoai, "A deep embedded clustering algorithm for the binning of metagenomic sequences," *IEEE Access*, vol. 10, pp. 54348–54357, 2022.
[16] A. Lamurias, M. Sereika, M. Albertsen, K. Hose, and T. D. Nielsen, "Metagenomic binning with assembly graph embeddings," *Bioinformatics*, vol. 38, no. 19, pp. 4481–4487, Sep. 2022.
[17] A. L. N. Fred and A. K. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, Jun. 2005.
[18] A. Bremges and A. C. McHardy, "Critical assessment of metagenome interpretation enters the second round," *mSystems*, vol. 3, no. 4, pp. 429–440, Aug. 2018.
[19] R. M. Bowers, N. C. Kyrpides, R. Stepanauskas, M. Harmon-Smith, D. Doud, T. B. K. Reddy, F. Schulz, J. Jarett, A. R. Rivers, E. A. Eloe-Fadrosh, and S. G. Tringe, "Minimum information about a single amplified genome (MISAG) and a metagenome-assembled genome (MIMAG) of bacteria and archaea," *Nature Biotechnol.*, vol. 35, no. 8, pp. 725–731, Aug. 2017.
[20] S. Vassilvitskii and D. Arthur, "K-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2006, pp. 1027–1035.
[21] F. Saeed, N. Salim, and A. Abdo, "Voting-based consensus clustering for combining multiple clusterings of chemical structures," *J. Cheminformatics*, vol. 4, no. 1, pp. 1–8, Dec. 2012.
[22] X. Z. Fern and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 36.
[23] S. Nayfach and K. S. Pollard, "Average genome size estimation improves comparative metagenomics and sheds light on the functional ecology of the human microbiome," *Genome Biol.*, vol. 16, no. 1, pp. 1–18, Dec. 2015.

OLIVER S. BAK received the B.Sc. degree in computer science from the Department of Computer Science, Aalborg University, Aalborg, Denmark, in 2022, where he is currently pursuing the M.Sc. degree in computer science. His research interests include machine learning, metagenomics, clustering, and statistical model checking.

MARCUS D. JENSEN received the B.Sc. degree in computer science from the Department of Computer Science, Aalborg University, Aalborg, Denmark, in 2022, where he is currently pursuing the M.Sc. degree in computer science. His research interests include machine learning, metagenomics, and statistical model checking.

FREDERIK M. TRUDSLEV received the B.Sc. degree in computer science from the Department of Computer Science, Aalborg University, Aalborg, Denmark, in 2022, where he is currently pursuing the M.Sc. degree in computer science. His research interests include machine learning, metagenomics, and clustering.

ANDREAS WINDFELD received the B.Sc. degree in computer science from the Department of Computer Science, Aalborg University, Aalborg, Denmark, in 2022, where he is currently pursuing the M.Sc. degree in computer science. His research interests include machine learning, metagenomics, clustering, and statistical model checking.

ANDRE LAMURIAS received the B.Sc. degree in biochemistry, the M.Sc. degree in bioinformatics and computational biology, and the Ph.D. degree in systems biology from the Faculty of Sciences, University of Lisbon, Lisbon, Portugal, in 2013, 2015, and 2019, respectively.

Since 2021, he has been a Postdoctoral Researcher with the Department of Computer Science, Aalborg University. His research interests include machine learning, bioinformatics, metagenomics, and biomedical text mining.