

A multi-flow approach for binning circular plasmids from short-reads assembly graphs

Victor Epain¹[0000–0003–0049–0954], Aniket Mane¹[0000–0002–0130–7149], Gianluca Della Vedova²[0000–0001–5584–3089], Paola Bonizzoni²[0000–0001–7289–4988], and Cedric Chauve¹[0000–0001–9837–1878]

¹ Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada
`{victor_epain,aniket_mane,cedric_chauve}@sfu.ca`
² Università degli Studi di Milano-Bicocca, Milan, Italy
`{gianluca.dellavedova,paola.bonizzoni}@unimib.it`

Abstract. We address the problem of plasmid binning, that aims to group contigs — from a draft short-read assembly for a bacterial sample — into bins each expected to correspond to a plasmid present in the sequenced bacterial genome. We formulate the plasmid binning problem as a network multi-flow problem in the assembly graph and describe a Mixed-Integer Linear Program to solve it. We compare our new method, **PlasBin-HMF**, with state-of-the-art methods, **MOB-recon**, **gplasCC**, and **PlasBin-flow**, on a dataset of more than 500 bacterial samples, and show that **PlasBin-HMF** outperforms the other methods, by preserving the explainability.

Keywords: Plasmid binning · Mixed Integer Linear Programming · Flow network

1 Introduction

Plasmids are extra-chromosomal Mobile Genetic Elements (MGE) present in bacterial genomes that can easily transfer between bacterial cells, possibly of different bacterial species, and contribute widely to the propagation of genes responsible for Antimicrobial Resistance (AMR) [15,6]. Due to the importance of AMR in public health [18], an important aspect of epidemiological surveillance focuses on detecting plasmids in bacterial sequencing data [5]. Short-read sequencing is still the most commonly used data acquisition technology for epidemiological surveillance. As a consequence, the development of bioinformatics methods for the detection and characterization of plasmids in bacterial draft assemblies obtained from short-read sequencing data is an active research area [1,17,3,16,13,19,23,10,20,14].

Detecting plasmids from a short-read draft assembly can be addressed at three different levels: contigs classification, plasmid binning and plasmid assembly. *Contigs classification* aims to separate contigs between plasmidic contigs and chromosomal contigs, although some contigs can also be classified as ambiguous if they are repeats present both in the chromosome and in some plasmid. There is

a large corpus of classification methods, most of them based on machine-learning approaches [22]. As a bacterial genome can contain several plasmids, contigs classification does not provide a precise view of its plasmid content. This is addressed by *plasmid binning*, that computes groups of contigs (*plasmid bins*) with the aim that each plasmid bin contains the contigs constituting a single plasmid. *Plasmid assembly* can be seen as a refinement of plasmid binning where contigs in each plasmid bin are ordered and oriented.

MOB-recon [16] is one of the most popular plasmid binning methods, based on aligning contigs to a database of known plasmid-specific genes and a curated database of complete plasmid sequences. To the best of our knowledge, all other plasmid binning methods rely on a different approach that makes use of the *assembly graph*, a graph provided together with contigs by assemblers widely used for bacterial data, SPAdes [4], Unicycler [24] and SKESA [21]. The first methods of this kind were plasmidSPAdes [1] and Recycler [17]. Both rely on the expectation that contigs from the same plasmid (forming a true plasmid bin) should be co-located in the assembly graph and should have read coverage that is different from the expected coverage of chromosomal contigs and uniform within a plasmid bin. Unlike MOB-recon, plasmidSPAdes and Recycler are *de novo* methods that do not rely on homology with known plasmids. HyAsP [13] introduced the notion of scoring plasmid bins based on several features including the coverage of contigs by known plasmid genes, the uniformity of read coverage and of the GC content, and integrated this scoring scheme in an iterative greedy heuristic searching for high-score walks in the assembly graph, each such path forming a plasmid bin. gplas [2], similarly to HyAsP, is a greedy walk detection heuristic but it relies on a different scoring function and introduced the idea of using plasmid contigs classification results as an important signal to detect plasmid bins; the latest development of this method is gplasCC [14]. The HyAsP scoring function served as a basis for PlasBin [9], the first approach that introduced a rigorous combinatorial optimization approach to plasmid binning; PlasBin iteratively detected the plasmid bin of maximum score using a Mixed-Integer Linear Program (MILP), peeled the corresponding subgraph out of the assembly graph, and repeated until no good plasmid bin could be found. PlasBin-flow [10] improved upon PlasBin by using the concept of network flow to define plasmid bins as connected subgraphs that would optimize a function combining the features used in PlasBin and the flow value, considered as a proxy for the copy number of the hypothetically detected plasmid; however, similarly all other graph-based methods discussed above, PlasBin-flow is an iterative method, searching for one plasmid bin at a time.

Here we propose a novel plasmid binning method, PlasBin-HMF (standing for Plasmid Binning Hierarchical Multi-Flow), that builds upon the high-level principles of PlasBin-flow but does not search for plasmid bins one at a time and, instead, relies on the concept of multi-flow to detect a set of plasmid bins at once. To the best of our knowledge, PlasBin-HMF is the first plasmid binning method that defines the problem rigorously from a combinatorial optimization point of view and solves it exactly using an MILP. In Section 2, we formulate

the plasmid binning problem as a multi-flow combinatorial optimization problem and describes how it can be solved using an MILP. In Section 3 we apply MOB-recon, gplasCC, PlasBin-flow and PlasBin-HMF to a dataset of more than 500 bacterial samples, and we show that PlasBin-HMF outperforms the other methods. We conclude by discussing avenues for improving the multi-flow model that serves as a basis for PlasBin-HMF.

2 Methods

Informally, given sequencing data from a bacterial sample and its assembly, plasmid binning aims to determine the contig content (a set of contigs) of all plasmids in the sample, without prior knowledge about the number of plasmids or their genomic sequences. From now on, we refer to a set of contigs as a *bin*.

In this section we describe a combinatorial optimization model to detect plasmid bins; we focus on describing the mathematical model, and postpone the description of the MILP formulation to the Supplementay Material. In the description that follows, we assume that plasmids are circular molecules, which is the case in the vast majority of cases.

2.1 Input data

Assembly graph. The assembly of short-read sequencing data results in a (unoriented) contig set \mathcal{C} and a link set \mathcal{L} , a link being an ordered pairs of oriented contigs, $\mathcal{L} \subset (\mathcal{C} \times \{+, -\})^2$, where $+$ and $-$ respectively stand for the forward and the reverse orientations. Note that for each link $(c, d) \in \mathcal{L}$, its reverse $(\overleftarrow{d}, \overleftarrow{c})$ is also in \mathcal{L} , where \overleftarrow{c} and \overleftarrow{d} the reverse of the oriented contigs c and d . The link set naturally defines a directed graph, called the *assembly graph*, where every vertex corresponds to an oriented contig (see Figure S1 in Appendix A).

Contigs length and coverage. For a contig $c \in \mathcal{C}$, we denote by len_c its length and by cov_c its *normalized coverage*, i.e. its average read coverage divided by the mean read coverage over all contigs. The normalized coverage serves as a proxy for the expected number of copies of the contig sequence in the sequenced genome. The widely used assembler Unicycler provides this information by default.

Plasmidness score and plasmid seeds. We also assume that each contig is provided with a *plasmidness score* $\text{plm}_c \in [-1, 1]$ that estimates the nature of the contig as either chromosomal ($\text{plm}_c < 0$) or plasmid-derived (plasmidic) ($\text{plm}_c > 0$). Accurate plasmidness scores would assign a score of -1 to purely *chromosomal* contigs, 1 to purely *plasmidic* contigs, and 0 to contigs appearing both in the chromosome and in some plasmid(s) (*ambiguous* contigs). Finally, we assume that we are provided with a subset $\mathcal{C}_{\text{seed}} \subset \mathcal{C}$ of *seed* contigs, which are estimated to be plasmid-derived with high confidence. We describe in Section 3 how we use the plasmid contigs classifications tools RFPlasmid [23], plasmidCC [14] and Platon [19] to compute plasmidness scores and determine plasmid seeds.

2.2 Overview

Most plasmid binning methods that rely on exploring the assembly graph are based on the following principle: if we assume (1) uniform sequencing depth, (2) an accurate assembly graph, and (3) an accurate classification of the contigs as chromosomal, plasmidic or ambiguous, then a circular plasmid appears in the assembly graph as a circuit composed of plasmidic (and possibly ambiguous) contigs, with uniform normalized coverage that approximates well the copy number of the plasmid. The contig content of this circuit forms the (true) plasmid bin for the corresponding plasmid.

PlasBin-HMF is based on the same high-level principle, accounting for expected inaccuracies in the input data. Indeed, in practice, (1) sequencing depth is not uniform, (2) assembly graphs may exhibit spurious or missing links, and (3) contigs classification methods are prone to errors. **PlasBin-HMF** allows to define a (plasmid) bin as a connected subgraph of the assembly graph (instead of a circuit, to handle point (2)), whose contigs' normalized coverage is consistent with a flow through this subgraph (point (1)). **PlasBin-HMF** allows a bin to contain contigs classified as chromosomal but integrates in its scoring function the plasmidness score such that it assigns a better score to subgraphs that are denser in plasmidic contigs (point (3)). Unlike iterative methods, one of the main novelties of **PlasBin-HMF** is that it simultaneously identifies in the assembly graph multiple subgraphs and flows, each corresponding to a single plasmid bin.

2.3 Multi-flow modelling of plasmid bins

We first describe the network derived from the assembly graph, on which we model plasmid bins as network flows ([Definition 1](#)). As we potentially search for several bins, we model one flow per bin ([Definition 2](#), [Figure 1](#)). We focus our search to circular bins in the network ([Definition 3](#)), and complete the list of bins with partially circular ones in the case some links are missing ([Definition 4](#)). We then model the plasmid binning as a combinatorial optimization problem on the network, using a multi-flow formulation framework, that we formalize as an MILP ([Appendix B.2](#)).

Definition 1 (Flow network). *Let $G = (V, A, \mathbf{s}, \mathbf{t})$ be the flow network defined as follows:*

- *the set V of vertices contains the vertices of the assembly graph (associated to the oriented contigs), the source \mathbf{s} and the sink \mathbf{t} : $V = V_C \cup \{\mathbf{s}, \mathbf{t}\}$, where $V_C = \mathcal{C} \times \{+, -\}$;¹*
- *the set A of arcs contains the link-arcs of the assembly graph and the arcs from the source and to the sink: $A = \mathcal{L} \cup \{(\mathbf{s}, v) \mid v \in V_C\} \cup \{(v, \mathbf{t}) \mid v \in V_C\}$;¹*

¹ $V_C = \mathcal{C} \times \{+, -\}$ is an abuse of notation. Rigorously, one should define V_C as a regular vertex set in bijection with $\mathcal{C} \times \{+, -\}$. The same remark holds for the arcs where one should define a set of link-arcs \mathcal{A}_L in bijection with \mathcal{L} .

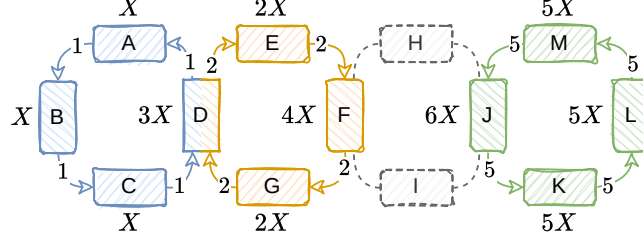


Fig. 1: Example of an assembly graph with three circular plasmid bins, $\{A, B, C, D\}$, $\{D, E, F, G\}$ and $\{J, K, L, M\}$, with respective copy numbers 1, 2 and 5. The contig coverages are written as kX , $k \in \mathbb{N}$. The positive flows are numbers on the arcs. Contig D is shared between the blue and the orange plasmids. The total incoming flow in D equals its coverage ($1 + 2 = 3$). The coverages of F and J are not fully explained by the flows, so one could assume they also belong to other elements of the genome than plasmids.

Retrieving the contig from a vertex and conversely is enabled thanks to two functions $vtoc: V_C \rightarrow \mathcal{C}$ and $ctov: \mathcal{C} \rightarrow V_C^2$. For each vertex v , $V^-(v)$ and $V^+(v)$ denote respectively the set of predecessors and successors of v in the network. For a vertex $v \in V_C$, \overleftarrow{v} denotes its reverse (the same contig but with a reversed orientation, so $vtoc(v) = vtoc(\overleftarrow{v})$).

Definition 2 (Coverage-constrained multi-flow). A multi-flow set of size $n \in \mathbb{N}_{\geq 1}$ is a set of n functions $\{f_1, \dots, f_n\}$ such that:

- $\forall 1 \leq k \leq n, f_k: A \rightarrow \mathbb{R}_{\geq 0}$
- $\forall c \in \mathcal{C}$:

$$\sum_{k=1}^n \text{expcov}(c, k) \leq \text{cov}_c \quad (1)$$

where

$$\text{expcov}(c, k) = \sum_{u \in V^-(v)} f_k(u, v) + \sum_{u \in V^-(\overleftarrow{v})} f_k(u, \overleftarrow{v}), \quad \text{ctov}(c) = (v, \overleftarrow{v}) \quad (2)$$

- $\forall v \in V_C, 1 \leq k \leq n$:

$$\sum_{u \in V^-(v)} f_k(u, v) = \sum_{w \in V^+(v)} f_k(v, w) \quad (3)$$

- $\forall 1 \leq k \leq n$:

$$\sum_{w \in V^+(s)} f_k(s, w) = \sum_{u \in V^-(t)} f_k(u, t) \quad (4)$$

Modelling plasmids as multi-flows As discussed in [Section 2.2](#), a bin is the set of contigs of a subgraph of the assembly graph. In PlasBin-HMF, each flow in a multi-flow set induces a non-empty subgraph, and thus a bin. We first search for a multi-flow composed only of circular flows ([Definition 3](#)), that models putative plasmids that appear as circuits in the assembly graph. Then we relax the circularity constraint ([Definition 4](#)) to account for possibly missing links in the assembly graph. [Figure 2](#) illustrates the two kinds of structures we are looking for.

Definition 3 (Circular coverage-constrained flow). *A function f in the coverage constrained multi-flow set is circular if the set of strictly positive flow links $\{a \in \mathcal{L} \mid f(a) > 0\}$ is defining a circuit in G .*

Definition 4 (Partially circular coverage-constrained flow). *A function f in the coverage constrained multi-flow set is partially circular if:*

- *the set of strictly positive flow arcs augmented with an artificial arc from the sink to the source $\{a \in A \mid f(a) > 0\} \cup (t, s)$ is defining a circuit in G ;*
- *the subgraph without s and t is connected;*
- *each vertex connecting the source, respectively the sink, has no other positive-flow predecessor, resp. successor, i.e.*

$$\forall v \in V_C \mid f(s, v) > 0, \quad \{u \in V^-(v) \mid f(u, v) > 0\} = \{s\} \quad (5)$$

$$\forall v \in V_C \mid f(v, t) > 0, \quad \{w \in V^+(v) \mid f(v, w) > 0\} = \{t\} \quad (6)$$

[Definition 4](#) ensures the contigs connected by the source and the sink are extremities of the “cut circuit”. It prevents to artificially connecting contigs from other plasmids or misidentified non-plasmidic contigs to the extremities of the cut circuit.

Last, we describe how we make use of the plasmidness score of contigs in defining plasmid bins. Each flow function f_k in the multi-flow set is associated to a plasmidness score ([Definition 5](#)). The more the flow uses the coverage of a long plasmidic (non-plasmidic) contig, the better (worse) the score.

Definition 5 (Plasmidness score). *The plasmidness score of a flow function f_k in a multi-flow set is defined as:*

$$PlasmidnessScore_k = \sum_{c \in \mathcal{C}} \text{len}_c \text{plm}_c \text{expcov}(c, k) \quad (7)$$

In what follows, we require that any flow that would define a plasmid bin has a positive plasmidness score.

Scoring a multi-flow set We describe how we associate with a multi-flow set a linear *explanation score* ([Definition 6](#)), where a higher scores implies a better explanation of the bins defined by the flows in terms of the input data. PlasBin-HMF aims to search for a multi-flow set maximizing the explanation

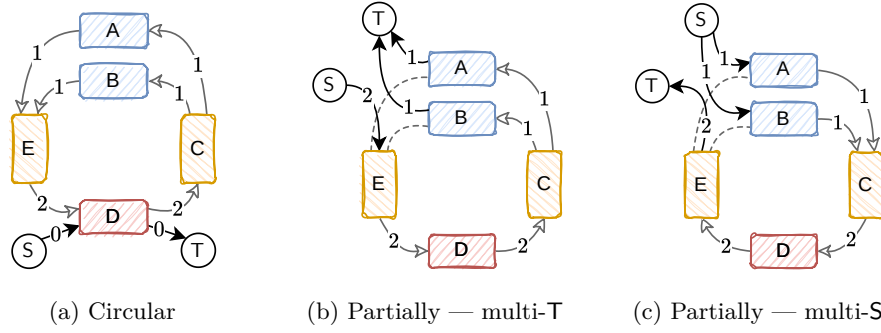


Fig. 2: Fully and partially circular flows. In each subfigure, the red contig D is a seed, orange and red contigs have a coverage of $2X$ while the blue contigs A and B have coverage X . The numbers on the arcs are the flow values. For the ease of read, we do not show other links between other plasmids or chromosomes and suppose we cannot merge C, D and E in one unitig. The links are oriented according to the orientation of the positive flow. **(a)** No link is missing such that we can find a circuit using all the contig coverages. **(b)** — **(c)** Dashed links are missing links. The source S and the sink T simulate the circularity by connecting the extremities of the partial circuit.

score. On one hand, we favour multi-flow sets that maximizes the use of plasmidic contigs, avoiding the ones which are not (*PlasmidnessScores* term). On the other hand, we penalize multi-flow sets when they fail to fully explain the coverages of plasmidic contigs (*PosPlmCovPenalty* term). We model the search of the best multi-flow set of size n through an MILP that we describe formally in [Appendix B.2](#).

Definition 6 (Explanation score). *Given a multi-flow set of size n , its explanation score is defined as follows:*

$$ExplanationScore = PlasmidnessScores + PosPlmCovPenalty \quad (8)$$

where:

$$PlasmidnessScores = \sum_{k=1}^n PlasmidnessScore_k \quad (9)$$

$$PosPlmCovPenalty = \sum_{\substack{c \in \mathcal{C} \\ plm_c > 0}} len_c plm_c \left(\left(\sum_{k=1}^n expcov(c, k) \right) - cov_c \right) \quad (10)$$

Deciding the number of flows In what precedes, we always assumed that the number of flows in a multi-flow set was given. As the number of plasmid bins is exactly the number of flows in a multi-flow set, the question of choosing the

size (number of flows) that is searched by **PlasBin-HMF** is an important question. Informally, we address this problem by computing optimal multi-flow sets of increasing size until a parsimony stopping criterion, based on the comparison among the explanation scores of successive multi-flow sets, is met.

Assume that during this process, at iteration m , we aim to find the best multi-flow set of size m . Each time we increment the number of flows, we penalize the next explanation score with an *additional flow penalty* (Definition 7). Given two parameters \underline{L} (minimum length of a plasmid bin) and \underline{cov} (minimum flow allowed to define a plasmid bin), the additional flow penalty is the plasmidness score of a flow defining a plasmid bin with a single contig, of length \underline{L} , plasmidness of 1 and normalized coverage \underline{cov} . We stop incrementing the number of flows if (1) either we reach the maximum number of flows $n \in \mathbb{N}_{\geq 1}$ we allowed to search for (see next paragraph) (2) the search is infeasible (3) or the new explanation score plus the additional flow penalty is not (strictly) greater than the previous explanation score (if $ExplanationScore_{m+1} - FlowPenalty \leq ExplanationScore_m$, $1 \leq m < n$).

Definition 7 (Additional flow penalty). *Given two parameters $\underline{L} \in \mathbb{N}$ and $\underline{cov} \in \mathbb{R}_{>0}$ the additional flow penalty is defined by:*

$$FlowPenalty = \underline{L} \cdot \underline{cov} \quad (11)$$

Search structure and parameters A multi-flow set is associated with a plasmid structure (fully or partially circular), which must be described by its flows. Additionally to the subgraph topology, the multi-flow set must respect a seed constraint: we first search for subgraphs containing at least one seed contig, and then relax the constraint. We first search for circular bins with seeds and then without seeds, and repeat the same procedure for partially circular bins, first with seeds and then without. The seed constraint provides us the maximum number of flows a multi-flow set can contain. When each bin must contain a seed, the maximum number of flows is the number of seeds ($n = |\mathcal{C}_{seed}|$). Otherwise, the upper bound equals to a user parameter $\bar{n}_c \in \mathbb{N}$, respectively $\bar{n}_{pc} \in \mathbb{N}$ when we search for circular flows, resp. for partially circular flows. Algorithms B.1 and B.2 detail the search prioritisation.

Finally, given a multi-flow set with a circularity and a seed constraints, we require that each flow function f_k (1) has a positive plasmidness score (Definition 5), (2) equals at least the normalized coverage parameter \underline{cov} when it positively passes through an arc (i.e. $f_k: A \rightarrow \{0\} \cup [\underline{cov}, \infty]$), and (3) describes a bin such that the sum of its contigs lengths at least equals the parameter \underline{L} .

3 Results and Discussion

We implemented the method in **Python3** as a submodule of a larger package <https://github.com/AlgoLab/pangebin>². We evaluated **PlasBin-HMF** against **MOB-**

² Pending the submodule becoming a stand-alone programme, **PlasBin-HMF** can be launched via the command `pangebin asm-pbf hmf --help` to output a file in the same format as **PlasBin-flow**.

recon, **PlasBin-flow** and **gplasCC** on 581 bacterial samples for which both short-read and long-read are publicly available on NCBI. In particular, long-read data was used only to define the ground truth plasmid bins. We detail in [Appendix C](#) how we obtained these 581 samples.

3.1 Data preparation and experiments

Input data preparation For each sample, we assembled the Illumina short reads with **Unicycler**, a read assembler aiming the completion of the circularity. Contigs less than 100bp were removed from the assembly graph and transformed into as many links required to connect their neighbours. The plasmidness of contigs was computed with the classification tools **RFPlasmid**. The plasmidness in **PlasBin-flow** are between 0 and 1, while those of **PlasBin-HMF** are between -1 and 1 (affine transformation). Internally, **PlasBin-HMF** raises the plasmidness of the seed contigs by computing the mean between one and the original input plasmidness. The seeds for **PlasBin-flow** and **PlasBin-HMF** are the contigs identified as plasmid contigs by **Platon** (binary classification).

Ground truth For each sample, we produced a hybrid assembly using both the short and the (Oxford Nanopore) long reads. If a hybrid contig was longer than 1Mbp, we annotated it as chromosomal, if it is circular and less than 1Mbp we considered the contig to be a plasmid, otherwise we declared the contig unlabelled. Here, all the 581 samples have a ground truth with hybrid contigs labelled as plasmids or chromosomal, and no contig is unlabelled. For each sample, we mapped the short contigs against the hybrid ones with **minimap2** [8]. We labeled a short contig as plasmidic if it mapped to a plasmidic hybrid contig, and as non-plasmidic otherwise (i.e., if it mapped only to chromosomal hybrid contigs or to no hybrid contig). To each plasmidic hybrid contig corresponds a ground truth bin, containing the plasmidic short contig that mapped to the hybrid contig. Note that, contrary to the recent plasmid binning method benchmark study [22], a short contig can belong to multiple bins in our case.

Experiments We use the default option values for **MOB-recon**, **PlasBin-flow** and **gplasCC**. For **PlasBin-HMF**, we searched for fully and partially circular bins with seeds, and complete the results with no more than two circular bins without seeds. We also fixed the minimum cumulative contig length a bin reach to $\underline{L} = 1000$, and a minimum of flow $\underline{cov} = 0.3$. As **gplasCC** outputs bins that exclusively contain contigs classified as plasmidic (even if chromosomal classified contigs participate in the walks connecting the plasmidic ones) we add to the comparison a modified version of **PlasBin-flow** and of **PlasBin-HMF** that mimic the **gplasCC** filtering behaviour. Thus, for each bin, we only keep its seeds and its contigs with a plasmidness greater than 0.5 for **PlasBin-flow** and greater than 0 for **PlasBin-HMF**.

We ran the tools and **PlasEval** on the Digital Alliance Canada high-performance computing clusters Cedar and Fir. Each sample was processed using with

16 CPUs, 32GB of RAM, with a time limit set to 10 hours. **PlasBin-flow** and **PlasBin-HMF** ran with the MILP solver Gurobi.

3.2 Accuracy Measures

We evaluate the methods according to two accuracy measures. The first has been introduced in **PlasBin-flow** [10] and is an adaptation of the recall, precision and the F1 statistics, weighted by contig length. For each sample, they evaluate the best pairs of bins between a list of predicted bins P and a list of ground truth bins T . In Equations (12) and (13), $overlap(p, t)$ is the sum of length of contigs in both the predicted bin p and the ground truth one v , while $size(b)$ is the lengths sum of contigs in bin b . The F1 measure is the harmonic mean between *Prec* and *Recall*.

$$Prec = \frac{\sum_{p \in P} \max_{t \in T} overlap(p, t)}{\sum_{p \in P} size(p)} \quad (12)$$

$$Recall = \frac{\sum_{t \in T} \max_{p \in P} overlap(p, t)}{\sum_{t \in T} size(t)} \quad (13)$$

The second evaluation measures have been introduced in the plasmid binning evaluation tool **PlasEval** [11]. For each sample, **PlasEval** provides the costs associated to the minimum number of cut and join operations required to transform the predicted bins into the ground truth ones. The join and cut costs are linearly composed with the costs of extra predicted contigs and missing ground truth contigs to define a dissimilarity score, normalized between 0 and 1. In the next sections, we set the alpha **PlasEval** parameter to 0.5. While we aim maximizing the F1 score, we aim to minimize the dissimilarity score.

These measures are relevant to evaluate a binning result. Indeed, the binning results in a list of bins that may share some contigs. Consequently, classical clustering or partitioning evaluation measures do not handle this list. Furthermore, the adapted F1 and the dissimilarity measures are not affected by true chromosomal contigs that do not belong to any predicted bins.

3.3 Results

For the best prediction-ground truth pairing evaluation, **Figure 3** shows that **PlasBin-HMF** is slightly better (mean 0.61, median of 0.78, before **MOB-recon**, with 0.58 and 0.76), however the filtered version is the best (respectively 0.66 and 0.85, with a higher Q1 of 0.30 versus 0.07 for **MOB-recon**).

Figure 4 shows the dissimilarity scores distributions. **PlasBin-HMF** and its filtered version are the best (mean and median equal to 0.3 and 0.27, versus 0.39 and 0.3 for **gplasCC**), with Q1 quartile that reaches 0 (0.05 for **gplasCC**) and Q3 0.44 (0.66 for **gplasCC**). However, the conclusions vary depending on the sequenced bacteria species. **PlasBin-HMF** (especially filtered) keeps the best place for the *Staphylococcus aureus*, *Acinetobacter baumannii* and *Pseudomonas*

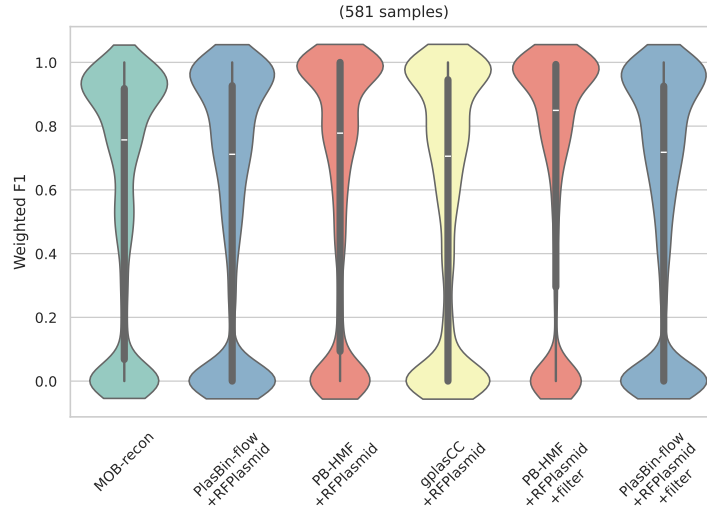


Fig. 3: Weighted F1 scores

aeruginosa species. While **MOB-recon** is not using an assembly graph, it is still efficient for *Escherichia coli* species for which it has been originally developed.

One of the main hypothesis on the encouraging results of **PlasBin-HMF** is its high recall (and low missing contigs costs, Figures S3 and S5) combined with a moderate raise of the cut costs (the cost of splitting a bin that corresponds to several ground truth bins, Figure S6). On the one hand, we penalize the plasmidic coverages not consumed by the flows (Equation (10)). On the other hand, we search for circuits in the assembly graph, and due to the flow conservation constraints (Equation (3)), if two plasmids share contigs and their circularity is preserved, our model artificially tends to merge the two circular flows in one bigger circular flow. In that case, what was a shared contig between two plasmids is now a repeat in the artificial plasmid containing the two true ones. Although we have this bias by design, the plasmid subgraphs, while still connected, appear sufficiently dispersed in the assembly graphs such that we do not merge them. In other words, the assembler **Unicycler** correctly untangled the shared subsequences between the plasmids.

4 Conclusion

In this paper, we present the first hierarchical multi-flow approach to address the problem of recovering an unknown number of plasmid bins from a short-read assembly graph. Our methodology introduces a novel plasmid binning strategy that exploits the circular structure of most plasmids. We model the binning task as a combinatorial optimisation problem: we use a mixed-integer linear

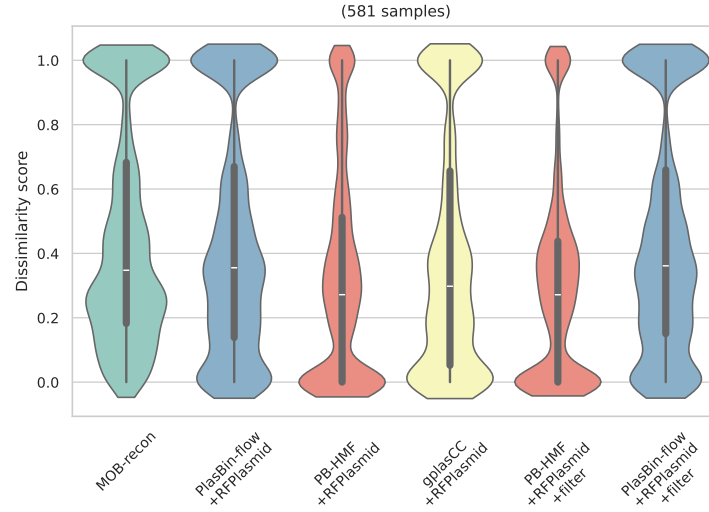


Fig. 4: Dissimilarity scores

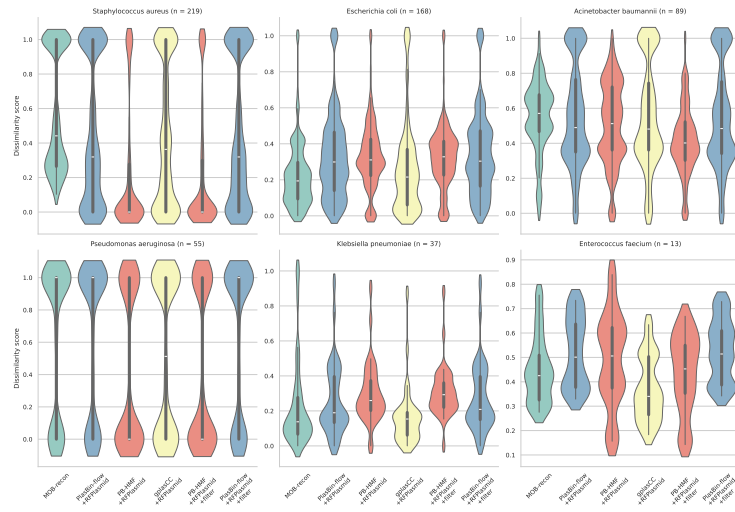


Fig. 5: Dissimilarity scores per species.

programming (MILP) formulation to search for a fixed, yet flexible, number of circular flows. Indeed, such number is increased during the process of computing flows. In a second step, we also consider partially circular flows to account for cases where missing links in the data disrupt plasmid circularity. To the best of our knowledge, **PlasBin-HMF** is the first method that simultaneously searches for multiple circular plasmids. We present experimental results comparing **PlasBin-HMF** with state-of-the-art binning tools, showing that our approach achieves the best performance on most bacterial samples.

Here we compared the exclusively binning tools (**gplasCC**, **PlasBin-flow** and **PlasBin-HMF**) with a classifier, **RFPlasmid**. Future comparisons would require us to use different classifiers, such as **plasmidCC**, the default classifier in **gplasCC**. Overall, the current results indicate that **PlasBin-HMF** has strong potential to substantially improve upon existing binning tools. However, several directions remain for further enhancing the model.

One of the main limitations of our approach is an inherent bias toward merging two circular plasmids when they share some contigs. One possible solution is to extend the MILP formulation with constraints on specific genetic content that should not be duplicated within a single plasmid. Such constraints must be carefully selected based on the plasmid genomics literature. Plasmid typing, which typically follows binning, could provide useful information for defining these constraints.

Alternatively, without introducing additional biological constraints and in order to keep the model minimalist, we could limit the amplitude of copy numbers (i.e., explained coverage variation) within each bin subgraph. This may help remove short, low-coverage chromosomal contigs that connect two plasmids and lead the model to incorrectly merge them. However, such a constraint may be less effective when separating plasmids with similar copy numbers. Similar challenges also arise in the haplotype assembly problem.

Another important direction concerns the number of flows in the multi-flow formulation, which corresponds to the unknown number of plasmids in the sample. Since this number is not known a priori, we currently implement a greedy iterative optimisation procedure to determine the appropriate number of flow functions. However, this strategy does not always scale well, as the branch-and-bound process may spend substantial time proving that the final multi-flow solution is not better than the previous one, leading to long runtimes before reaching an optimality certificate. In future work, rather than iteratively increasing the size of the multi-flow set, we aim to allow the MILP solver to directly decide the number of flows.

Acknowledgments. This study was funded by the Simon Fraser University Project N001347 Grant and V. E., P.B and G.D.V. have received funding from the European Unions Horizon 2020 Innovative Training Networks programme under the Marie Skłodowska-Curie grant agreement No. 956229. P.B and G.D.V have been funded by the Horizon Europe program under grant agreement No. 101160008 (FORGENOM II).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Antipov, D., Hartwick, N., Shen, M., Raiko, M., Lapidus, A., Pevzner, P.A.: plasmidSPAdes: Assembling plasmids from whole genome sequencing data. *Bioinformatics* **32**(22), 3380–3387 (Nov 2016). <https://doi.org/10.1093/bioinformatics/btw493>
2. Arredondo-Alonso, S., Bootsma, M., Hein, Y., Rogers, M.R.C., Corander, J., Willems, R.J.L., Schürch, A.C.: Gplas: A comprehensive tool for plasmid analysis using short-read graphs. *Bioinformatics* **36**(12), 3874–3876 (Jun 2020). <https://doi.org/10.1093/bioinformatics/btaa233>
3. Arredondo-Alonso, S., Rogers, M.R.C., Braat, J.C., Verschuuren, T.D., Top, J., Corander, J., Willems, R.J.L., Schürch, A.C.: Mlplasmids: A user-friendly tool to predict plasmid- and chromosome-derived sequences for single species. *Microbial Genomics* **4**(11), e000224 (2018). <https://doi.org/10.1099/mgen.0.000224>
4. Bankevich, A., Nurk, S., Antipov, D., Gurevich, A.A., Dvorkin, M., Kulikov, A.S., Lesin, V.M., Nikolenko, S.I., Pham, S., Prjibelski, A.D., Pyshkin, A.V., Sirotkin, A.V., Vyahhi, N., Tesler, G., Alekseyev, M.A., Pevzner, P.A.: SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology* **19**(5), 455–477 (May 2012). <https://doi.org/10.1089/cmb.2012.0021>
5. Castañeda-Barba, S., Top, E.M., Stalder, T.: Plasmids, a molecular cornerstone of antimicrobial resistance in the One Health era. *Nature Reviews Microbiology* **22**(1), 18–32 (Jan 2024). <https://doi.org/10.1038/s41579-023-00926-x>
6. De Oliveira, D.M.P., Forde, B.M., Kidd, T.J., Harris, P.N.A., Schembri, M.A., Beatson, S.A., Paterson, D.L., Walker, M.J.: Antimicrobial Resistance in ESKAPE Pathogens. *Clinical Microbiology Reviews* **33**(3), 10.1128/cmr.00181–19 (May 2020). <https://doi.org/10.1128/cmr.00181-19>
7. Epain, V., Thuillier, K.: Mixed Integer Linear Programming Framework for SAT Variants of the Connected Flow Problem (Jun 2025)
8. Li, H.: Minimap2: Pairwise alignment for nucleotide sequences. *Bioinformatics* **34**(18), 3094–3100 (Sep 2018). <https://doi.org/10.1093/bioinformatics/bty191>
9. Mane, A., Faizrahnemoon, M., Chauve, C.: A Mixed Integer Linear Programming Algorithm for Plasmid Binning. In: Jin, L., Durand, D. (eds.) *Comparative Genomics*. pp. 279–292. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-06220-9_16
10. Mane, A., Faizrahnemoon, M., Vinař, T., Brejová, B., Chauve, C.: PlasBinflow: A flow-based MILP algorithm for plasmid contigs binning. *Bioinformatics* **39**(Supplement_1), i288–i296 (Jun 2023). <https://doi.org/10.1093/bioinformatics/btad250>
11. Mane, A., Sanderson, H., White, A.P., Zaheer, R., Beiko, R., Chauve, C.: Plaseval: A framework for comparing and evaluating plasmid detection tools. *BMC Bioinformatics* **25**(1), 365 (Nov 2024). <https://doi.org/10.1186/s12859-024-05941-0>
12. Mane, A.C.: Optimization-Based Methods for Plasmid Analysis. Ph.D. thesis, Simon Fraser University (Aug 2025)
13. Müller, R., Chauve, C.: HyAsP, a greedy tool for plasmids identification. *Bioinformatics* **35**(21), 4436–4439 (Nov 2019). <https://doi.org/10.1093/bioinformatics/btz413>
14. Paganini, J.A., Kerkvliet, J.J., Teunis, G., Jordan, O., Plantinga, N.L., Meneses, R., Willems, R.J.L., Arredondo-Alonso, S., Schürch, A.C.: gplasCC: Classification and reconstruction of plasmids from short-read sequencing data for any bacterial species (Dec 2024). <https://doi.org/10.1101/2024.11.28.625923>

15. Partridge, S.R., Kwong, S.M., Firth, N., Jensen, S.O.: Mobile Genetic Elements Associated with Antimicrobial Resistance. *Clinical Microbiology Reviews* **31**(4), 10.1128/cmr.00088–17 (Aug 2018). <https://doi.org/10.1128/cmr.00088-17>
16. Robertson, J., Nash, J.H.E.: MOB-suite: Software tools for clustering, reconstruction and typing of plasmids from draft assemblies. *Microbial Genomics* **4**(8), e000206 (2018). <https://doi.org/10.1099/mgen.0.000206>
17. Rozov, R., Brown Kav, A., Bogumil, D., Shterzer, N., Halperin, E., Mizrahi, I., Shamir, R.: Recycler: An algorithm for detecting plasmids from de novo assembly graphs. *Bioinformatics* **33**(4), 475–482 (Feb 2017). <https://doi.org/10.1093/bioinformatics/btw651>
18. Sanderson, H., Gray, K.L., Manuele, A., Maguire, F., Khan, A., Liu, C., Navanekere Rudrappa, C., Nash, J.H.E., Robertson, J., Bessonov, K., Oloni, M., Alcock, B.P., Raphenya, A.R., McAllister, T.A., Peacock, S.J., Raven, K.E., Gouliouris, T., McArthur, A.G., Brinkman, F.S.L., Fink, R.C., Zaheer, R., Beiko, R.G.: Exploring the mobilome and resistome of *Enterococcus faecium* in a One Health context across two continents. *Microbial Genomics* **8**(9), 000880 (2022). <https://doi.org/10.1099/mgen.0.000880>
19. Schwengers, O., Barth, P., Falgenhauer, L., Hain, T., Chakraborty, T., Goesmann, A.: Platon: Identification and characterization of bacterial plasmid contigs in short-read draft assemblies exploiting protein sequence-based replicon distribution scores. *Microbial Genomics* **6**(10), e000398 (2020). <https://doi.org/10.1099/mgen.0.000398>
20. Sielemann, J., Sielemann, K., Brejová, B., Vinař, T., Chauve, C.: plASgraph2: Using graph neural networks to detect plasmid contigs from an assembly graph. *Frontiers in Microbiology* **14** (Oct 2023). <https://doi.org/10.3389/fmicb.2023.1267695>
21. Souvorov, A., Agarwala, R., Lipman, D.J.: SKESA: Strategic k-mer extension for scrupulous assemblies. *Genome Biology* **19**(1), 153 (Oct 2018). <https://doi.org/10.1186/s13059-018-1540-z>
22. Teixeira, M., Souque, C., Worby, C.J., Shea, T., Commins, N., Smith, J.T., Miklos, A.M., Abeel, T., Earl, A.M., Manson, A.L.: Circling in on plasmids: Benchmarking plasmid detection and reconstruction tools for short-read data from diverse species. *Briefings in Bioinformatics* **26**(6), bbaf589 (Nov 2025). <https://doi.org/10.1093/bib/bbaf589>
23. van der Graaf-van Bloois, L., Wagenaar, J.A., Zomer, A.L.: RFPlasmid: Predicting plasmid sequences from short-read assembly data using machine learning. *Microbial Genomics* **7**(11), 000683 (2021). <https://doi.org/10.1099/mgen.0.000683>
24. Wick, R.R., Judd, L.M., Gorrie, C.L., Holt, K.E.: Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads. *PLOS Computational Biology* **13**(6), e1005595 (Jun 2017). <https://doi.org/10.1371/journal.pcbi.1005595>

A Supplementary definitions

Figure S1 gives two ways of visualizing the assembly graph.



Fig. S1: Assembly graph representations. The two subfigures represent the same information: two contigs A (in blue) and B (in orange), and a link between them ($A+$, $B-$) (in green). **(a)** The arrows are the oriented contigs, the green line is the link. **(b)** The directed graph structure is an explicit representation of the oriented contigs and their (oriented) links. Each vertex is one contig in a fixed orientation. One link and its reverse are the arcs.

B Supplementary method

B.1 Method overview

The multi-flow binning approach searches several bins with the same properties at the same time. The properties are of two types:

Topology The bin is circular ☺, otherwise partially circular ☹

Seed constraint Each bin must contain at least one seed 🌱, otherwise, it can be free of seeds ☹.

► Algorithm B.1: Hierarchical Multi-Flow binning (HMF)

Require: A network G (see Definition 1).

Ensure: Extract bins from the network graph.

```

1: function EXTRACTBINS( $G$ )
2:    $prop \leftarrow \text{☺ and 🌱}$ 
3:    $I \leftarrow \text{CONNECTEDCOMP}(G) \times \{prop\} \triangleright \text{Connected subgraphs of } G$ 
      $\text{each one associated with the flow properties } prop$ 
4:   while  $I \neq \emptyset$  do
5:      $(G', prop) \leftarrow \text{EXTRACT}(I)$ 
6:      $F \leftarrow \text{GETBESTMULTIFLOWSET}(G', prop) \quad \triangleright \text{Algorithm B.2}$ 
```



```

7:   if  $F \neq \emptyset$  then
8:     output the bins described by  $F$ 
9:      $prop \leftarrow \text{GETNEXTFLOWPROP}(prop)$ 
10:    if  $prop \neq \text{null}$  then
11:       $G_{\setminus F} \leftarrow \text{REMOVECOVERAGES}(G, F)$ 
12:       $I \leftarrow I \cup \text{CONNECTEDCOMP}(G_{\setminus F}) \times \{prop\}$ 

```

Require: A network G (Definition 1).

Ensure: Output the list of connected subgraphs of G .

```

13: function  $\text{CONNECTEDCOMP}(G)$ 
14:    $G' \leftarrow G_{\setminus \{s, t\}} \triangleright G$  without the source and the sink vertices and their arcs
15:    $ccs \leftarrow$  connected components of  $G'$ 
16:   return  $\{G''_{\cup \{s, t\}} \mid G'' \in ccs\} \triangleright$  The subgraphs completed with the
    source and the sink and their links

```

Require: A network G (Definition 1).

Ensure: Give the next multi-flow set circularity and seed constraints.

```

17: function  $\text{NEXTFLOWPROP}(prop)$ 
18:   match  $prop$ 
19:     case  $\odot$  and  $\heartsuit$  do  $\triangleright$  Circular with seeds
20:       return  $\odot$  and  $\heartsuit$ 
21:     case  $\odot$  and  $\boxtimes$  do  $\triangleright$  Circular without seeds
22:       return  $\boxtimes$  and  $\heartsuit$ 
23:     case  $\otimes$  and  $\heartsuit$  do  $\triangleright$  Partially circular with seeds
24:       return  $\otimes$  and  $\heartsuit$ 
25:     case  $\otimes$  and  $\boxtimes$  do  $\triangleright$  Partially circular without seeds
26:       return  $\text{null}$ 

```

Require: A network G (Definition 1) and a multi-flow set of size n (Definition 2).

Ensure: Give the subgraph of G without the explained coverages of the multi-flow set.

```

27: function  $\text{REMOVECOVERAGES}(G = (V, A), \{f_1, \dots, f_n\})$ 
28:    $V' \leftarrow \{s, t\}$ 
29:   for  $c \in \mathcal{C}[V_c]$  do  $\triangleright$  Contigs represented by  $G$ 
30:      $\text{cov}_c \leftarrow \text{cov}_c - \sum_{k=1}^n \text{expcov}(c, k)$ 
31:     if  $\text{cov}_c > 0$  then
32:        $V' \leftarrow V' \cup \{v, \overleftarrow{v} \mid \text{ctov}(c) = (v, \overleftarrow{v})\}$ 
33:     else if  $c \in \mathcal{C}_{\text{seed}}$  then
34:        $\mathcal{C}_{\text{seed}} \leftarrow \mathcal{C}_{\text{seed}} \setminus \{c\}$ 
35:   return  $G[V']$   $\triangleright$  Subgraph  $G$  defined by vertices in  $V'$ 

```

► Algorithm B.2: Find the best multi-flow set with the best size

Require: A network graph G (Definition 1) such that $G_{\setminus\{s,t\}}$ is connected.

Ensure: Extract bins respecting the properties from the network graph.

```

1: function GETBESTMULTIFLOWSET( $G, prop$ )
2:    $n \leftarrow \text{MULTIFLOWSETSIZEUPPERBOUND}(G, prop)$ 
3:    $m \leftarrow 1$ 
4:    $infeasible \leftarrow \text{false}$ 
5:    $regression \leftarrow \text{false}$ 
6:    $F \leftarrow \emptyset$ 
7:    $z \leftarrow -\infty$  ▷ Best objective function value
8:   while  $m \leq n$  and not  $infeasible$  and not  $regression$  do
9:     Find the best multi-flow set of size  $m$  with the properties  $prop$ 
       (Appendix B.2), objective value is  $z_m \in \mathbb{R} \cup \{\text{null}\}$ 
10:    if The problem is infeasible then
11:       $infeasible \leftarrow \text{true}$ 
12:    else if  $z_m \leq z$  then
13:       $regression \leftarrow \text{true}$ 
14:    else
15:       $F \leftarrow \text{MILP multi-flow solution}$ 
16:       $z \leftarrow z_m$ 
17:       $m \leftarrow m + 1$ 
18:  return  $F$ 

```

Require: A network G (Definition 1) and the multi-flow set circularity and seed constraints.

Ensure: Give the upper-bound size of the multi-flow set.

```

19: function MULTIFLOWSETSIZEUPPERBOUND( $G = (V, A), prop$ )
20:   match  $prop$ 
21:     case  $(\odot)$  or  $(\otimes)$  and  $\heartsuit$  do
22:       return  $|\mathcal{C}_{seed} \cap \mathcal{C}[V_c]|$  ▷ Number of seeds in  $G$ 
23:     case  $\odot$  and  $\otimes$  do
24:       return  $\bar{n}_c$  ▷ User parameter
25:     case  $\otimes$  and  $\otimes$  do
26:       return  $\bar{n}_{pc}$  ▷ User parameter

```

B.2 Mixed Integer Linear Programming model

In this section we present the MILP model for the multi-binning flow approach, where one flow induces one bin. The MILP formulation consists in a set of constraints where the right inequality constant arm can change to make correspond the model to the flow/plasmid properties. Let $n \in \mathbb{N}$ be the maximum number of bins we can or want to model, and $k \in \{1, \dots, n\}$ thus representing the k^{th} flow. The changes of the constraints' constant parts must cover the following states:

- Flow k is inactive \mathbf{O} (no bin induced).

- Flow k is active \bullet (a bin is induced), with the following properties:
 - it is circular \odot or partially circular \otimes ;
 - it contains at least one seed \spadesuit or can be free of seeds \otimes .

We associate a symbol to each state to easily identify to which constant constraint right arm it corresponds to. The flow property symbols \odot , \otimes , \spadesuit and \otimes imply the flow to be active \bullet so we do not repeat the last symbol. The additional symbol \lightning means the constraint only applies for performance optimization.

In the following sections, by $A_{\mathcal{L}}$ we denote the set of link-arcs such that each link-arc corresponds to a link in \mathcal{L} .

Variables Table S1 lists all the variables used in the MILP model.

Table S1: MILP variables. Each of the following variables participates in at least one of the MILP model. Some of them participate in all the model, others are necessary for only one model. For the ease of read, we categorize the variables in three categories. Each section describing a model precise which variables are participating for each category.

Variable	Domain	Codomains		Meaning
		Practice	Relax	
For the whole multi-flow set				
$contig_c$	\mathcal{C}	$\{0, 1\}$	$[0, 1]$	Denoting whether the contig c is active in at least one bin or inactive in all the bins.
For each flow $1 \leq k \leq n$				
Decision variables				
x_v^k	$V_{\mathcal{C}}$	$\{0, 1\}$	$\mathbb{R}_{\geq 0}$	Denoting whether the vertex v is active or not.
y_a^k	A	$\{0, 1\}$		Denoting whether the arc a is active or not.
$contig_c^k$	\mathcal{C}	$\{0, 1\}$	$[0, 1]$	Denoting whether contig c is active or not.
Flow variables				
f_a^k	A	$\mathbb{R}_{\geq 0}$		The flow amount passing through the arc a .
F^k		$\mathbb{R}_{\geq 0}$		The total flow.
Connected component variables				

(the table continues on the next page)

Table S1, continued

Variable	Domain	Codomains		Meaning
		Practice	Relax	
β_a^k	$A \cup A_{\mathcal{L}}^{\leftarrow}$	$\{0, \dots, V \}$	$\mathbb{R}_{\geq 0}$	Is strictly positive if the link-arc a participates in one of the solution subgraph's exploration tree. The value corresponds to the depth of the subtree with root the successor. Note that here we consider the link-arcs undirected with $A_{\mathcal{L}}^{\leftarrow} = \{(v, u) (u, v) \in A_{\mathcal{L}}\}$.
r_v^k	$V^+(\mathbf{s})$	$\{0, 1\}$		Denoting whether the vertex v connects the source in the exploration tree or not.

Constraints for the whole bins This section gives the constraints for all the bins without distinction. The next section provides the constraints for one bin.

The cumulative incoming flow among all the bins must not exceed the coverage of the contig:

$$\sum_{k=1}^n \text{inflow}(c, k) \leq \text{cov}_c \quad \forall c \in \mathcal{C} \quad (\text{C1})$$

Where

$$\text{inflow}(c, k) = \sum_{u \in V^-(v)} f_{uv}^k + \sum_{u \in V^-(\overleftarrow{v})} f_{u\overleftarrow{v}}^k \quad (v, \overleftarrow{v}) = \text{ctov}(c)$$

Here we define the behaviour of the variable contig_c equals to 1 if the contig is active in at least one bin, otherwise 0:

$$\text{contig}_c \leq \sum_{k=1}^n \text{contig}_c^k \quad \forall c \in \mathcal{C} \quad (\text{C2})$$

$$\text{contig}_c \geq \text{contig}_c^k \quad \forall c \in \mathcal{C}, 1 \leq k \leq n \quad (\text{C3})$$

When the number of flows to be active is greater than one ($2 \leq m \leq n$), the explanation score ($\text{ExplanationScore}_m$) penalized by the flow cost must be greater or equal to the previous best objective value $\text{ExplanationScore}_{m-1}^*$ (see [Appendix B.2](#)).

$$\text{ExplanationScore}_m \geq \text{ExplanationScore}_{m-1}^* + \text{FlowPenalty} \quad \forall 2 \leq m \leq n \quad (\text{C4})$$

Where $\text{FlowPenalty} = \underline{\mathbf{L}} \cdot \text{cov}$ corresponds to the penalty cost of explaining the data with an additional flow. Remind that $\underline{\mathbf{L}}$ is the minimum cumulative contig length, and cov is the minimum amount of flow passing through a link-arc. The term is multiplied by 1 which corresponds to the best plasmidness.

When the number of bins to be active is greater than one, we define a plasmidness score order between the bins to avoid bins permutation:

$$PlasmidnessScore_k \geq PlasmidnessScore_{k+1} \quad \forall 1 \leq k < n \quad \text{⚡} \quad (C5)$$

Constraints for one specific flow Let $k \in \{1, \dots, n\}$ be the flow number.

Decision variables relationships A contig is active if, and only if at least one of its corresponding vertices is active:

$$contig_c^k \geq x_v^k \quad \forall c \in \mathcal{C} \quad v \in ctov(c) \quad (C6)$$

$$contig_c^k \leq x_v^k + x_{\overleftarrow{v}}^k \quad \forall c \in \mathcal{C} \quad (v, \overleftarrow{v}) = ctov(c) \quad (C7)$$

The vertices involved in an active arc must also be active:

$$y_{uv}^k \leq \begin{cases} x_v^k & \text{if } u = \mathbf{s} \\ x_u^k & \text{if } v = \mathbf{t} \\ \min\{x_u^k, x_v^k\} & \text{otherwise} \end{cases} \quad \forall (u, v) \in A \quad (C8)$$

An active vertex implies at least one active arc incoming to it:

$$x_v^k \leq \sum_{u \in V^-(v)} y_{uv}^k + \sum_{w \in V^+(v)} y_{vw}^k \quad \forall v \in V_{\mathcal{C}} \quad (C9)$$

Flow constraints for each contig vertex, the incoming flow equals the outgoing flow:

$$\sum_{u \in V^-(v)} f_{uv}^k = \sum_{w \in V^+(v)} f_{vw}^k \quad \forall v \in V_{\mathcal{C}} \quad (C10)$$

The total flow equals both the source outgoing flow and the sink incoming flow:

$$F^k = \sum_{w \in V^+(\mathbf{s})} f_{sw}^k \quad (C11)$$

$$F^k = \sum_{u \in V^-(\mathbf{t})} f_{ut}^k \quad (C12)$$

Each arc flow cannot exceed the coverage of the contigs involved. Also, the flow of an active arc is lower bounded by $\underline{\text{cov}} > 0$, except the source and sink

active arcs in the circularity context, where their flow equal 0:

$$y_{sv}^k - \frac{1}{\underline{\text{cov}}} f_{sv}^k \leq \begin{cases} 1 & \textcircled{\checkmark} \\ 0 & \textcircled{\times} \end{cases} \quad \forall v \in V^+(\mathbf{s}) \quad (\text{C13})$$

$$f_{sv}^k \leq \begin{cases} 0 & \textcircled{\checkmark} \\ \text{cov}_d & \textcircled{\times} \end{cases} \quad \forall v \in V^+(\mathbf{s}) \quad d = \text{vtoc}(v) \quad (\text{C14})$$

$$y_{uv}^k - \frac{1}{\underline{\text{cov}}} f_{uv}^k \leq 0 \quad \forall (u, v) \in A_{\mathcal{L}} \quad (\text{C15})$$

$$f_{uv}^k \leq \min\{\text{cov}_c, \text{cov}_d\} \quad \forall (u, v) \in A_{\mathcal{L}} \quad \begin{matrix} c = \text{vtoc}(u) \\ d = \text{vtoc}(v) \end{matrix} \quad (\text{C16})$$

$$y_{ut}^k - \frac{1}{\underline{\text{cov}}} f_{ut}^k \leq \begin{cases} 1 & \textcircled{\checkmark} \\ 0 & \textcircled{\times} \end{cases} \quad \forall u \in V^-(\mathbf{t}) \quad (\text{C17})$$

$$f_{ut}^k \leq \begin{cases} 0 & \textcircled{\checkmark} \\ \text{cov}_c & \textcircled{\times} \end{cases} \quad \forall u \in V^-(\mathbf{t}) \quad c = \text{vtoc}(u) \quad (\text{C18})$$

The cumulative incoming flow cannot exceed the coverage of the contig:

$$\text{inflow}(c, k) \leq \text{cov}_c \quad \forall c \in \mathcal{C} \quad (\text{C19})$$

Connectivity constraints To avoid the solution graph without the source and the sink (the intermediate solution graph) to have several connected components, we inspire the Connected Intermediate Flow problem in [7] and adapt the constraints to the nature of our network. Modelling the connectivity implies describing an exploration tree.

An active arc can participate in the tree. Also, an active link-arc in $A_{\mathcal{L}}$ can reversely participate in the tree ($A_{\mathcal{L}}^{\leftarrow}$):

$$\beta_a^k \leq y_a^k |V| \quad \forall a \in A \quad (\text{C20})$$

$$\beta_{vu}^k \leq y_{uv}^k |V| \quad \forall (u, v) \in A_{\mathcal{L}} \quad (\text{C21})$$

Note that in **Constraint C21**, $(v, u) \in A_{\mathcal{L}}^{\leftarrow}$.

In the circular case, we do not need to model the reverse link-arc in the tree:

$$\beta_{vu}^k \leq \begin{cases} 0 & \textcircled{\checkmark} \\ |V| & \textcircled{\times} \end{cases} \quad \forall (u, v) \in A_{\mathcal{L}} \quad \text{⚡} \quad (\text{C22})$$

To define the whole tree, we must define each subtree each arc (or reversed link-arc) defines. We must be careful of the validity of the constraints for the

source and the sink when the bin is inactive.

$$\sum_{v \in V_C} x_v^k - \sum_{w \in V^+(\mathbf{s})} \beta_{sw}^k = \begin{cases} -1 & \bullet \\ 0 & \bullet \end{cases} \quad (\text{C23})$$

$$\sum_{u \in V^-(v) \cup V^+(v)} \beta_{uv}^k - \sum_{w \in V^+(v) \cup V^-(v)} \beta_{vw}^k = x_v^k \quad \forall v \in V_C \quad (\text{C24})$$

$$\sum_{u \in V^-(\mathbf{t})} \beta_{ut}^k = \begin{cases} 1 & \bullet \\ 0 & \bullet \end{cases} \quad (\text{C25})$$

Only one contig vertex connects the source in the tree. It is a key constraint to model the connectivity of the intermediate solution graph:

$$\sum_{v \in V^+(\mathbf{s})} r_v^k = \begin{cases} 1 & \bullet \\ 0 & \bullet \end{cases} \quad (\text{C26})$$

When the bin is circular and must contain a seed, as the benefit of the performance, we only authorize the seed vertices to connect the source in the graph, so in the tree:

$$r_v^k \leq \begin{cases} 0 & \odot \wedge \heartsuit \\ 1 & \text{otherwise} \end{cases} \quad \forall v \in V^+(\mathbf{s}) \setminus V_{seed} \quad \blacklightning \quad (\text{C27})$$

Only the contig vertex chosen to be the root can connect the source in the tree:

$$\beta_{sv}^k \leq r_v^k |V| \quad \forall v \in V^+(\mathbf{s}) \quad (\text{C28})$$

Several contig vertices can connect the source in the solution graph. But only one (the root) connects the source in the tree. For performance purposes, we can arbitrarily give an order on the root choice.

Let \mathbf{v}_s to be an arbitrary vector containing once all the vertices connected to the source (order of $V^+(\mathbf{s})$). Then:

$$r_w \leq r_v + (1 - y_{sv}^k) \quad \forall 1 \leq i < |\mathbf{v}_s| \quad \blacklightning \quad (\text{C29})$$

$$v, w = \mathbf{v}_s[i], \mathbf{v}_s[i+1]$$

Thus, the first vertex in \mathbf{v}_s that connects the source in the solution graph is the one which connects the source in the tree.

Plasmid property The cumulative contig length is above a given threshold:

$$\begin{pmatrix} \bullet \\ \bullet \end{pmatrix} \begin{pmatrix} \underline{L} \\ 0 \end{pmatrix} \leq \sum_{c \in C} \text{len}_c \text{contig}_c^k \quad \underline{L} \geq 0 \quad (\text{C30})$$

Typically, we choose $\underline{L} = 1000$.

The incoming flow weighted plasmidness is above its γ coefficient dynamic mean:

$$0 \leq \sum_{c \in \mathcal{C}} \text{len}_c(\text{plm}_c - \gamma) \text{inflow}(c, k) \quad (\text{C31})$$

Typically, we choose $\gamma = 0$ to ensure the plasmidness of the bin to be positive.

To be circular or to be partially circular The circular bin is defined by only one source-arc, respectively one sink-arc, while the partially circular bin can be defined by several source-arcs, respectively several sink-arcs:

$$\left. \begin{array}{l} \sum_{v \in V^+(\text{s})} y_{sv}^k \\ \sum_{u \in V^-(\text{t})} y_{ut}^k \end{array} \right\} \leq \begin{cases} 1 & \textcircled{\checkmark} \\ |V_{\mathcal{C}}| & \textcircled{\times} \\ 0 & \textcircled{\bullet} \end{cases} \quad (\text{C32})$$

When the bin is circular, each active contig must follow and precede another contig:

$$x_v^k - \left\{ \begin{array}{l} \sum_{u \in V_{\mathcal{C}}^-(v)} y_{uv}^k \\ \sum_{w \in V_{\mathcal{C}}^+(v)} y_{vw}^k \end{array} \right\} \leq \begin{cases} 0 & \textcircled{\checkmark} \\ 1 & \textcircled{\times} \end{cases} \quad \forall v \in V_{\mathcal{C}} \quad (\text{C33})$$

When the bin is partially circular, an active contig that connects the source must not follow another contig:

$$\sum_{u \in V_{\mathcal{C}}^-(v)} y_{uv}^k + y_{sv}^k |V^-(v)| \leq \begin{cases} |V^-(v)| & \textcircled{\times} \\ 2|V^-(v)| & \textcircled{\checkmark} \end{cases} \quad \forall v \in V_{\mathcal{C}} \quad (\text{C34})$$

Analogously to the previous constraint, when the bin is partially circular, an active contig that connects the sink must not precede another contig:

$$\sum_{w \in V_{\mathcal{C}}^+(v)} y_{vw}^k + y_{vt}^k |V^+(v)| \leq \begin{cases} |V^+(v)| & \textcircled{\times} \\ 2|V^+(v)| & \textcircled{\checkmark} \end{cases} \quad \forall v \in V_{\mathcal{C}} \quad (\text{C35})$$

When the bin is circular, we can fix the same vertex to connect both the source and the sink:

$$\left\{ \begin{array}{cc} \textcircled{\checkmark} & 0 \\ \textcircled{\times} & -1 \end{array} \right\} \leq y_{sv}^k - y_{vt}^k \leq \left\{ \begin{array}{cc} 0 & \textcircled{\checkmark} \\ 1 & \textcircled{\times} \end{array} \right\} \quad \forall v \in V^+(\text{s}) \cap V^-(\text{t}) \quad \text{⚡} \quad (\text{C36})$$

Model whether the bin must have a seed or can be free of seeds Set the minimum number of seed contigs in the bin:

$$\sum_{c \in \mathcal{C}_{\text{seed}}} \text{contig}_c^k \geq \begin{cases} 1 & \text{⚡} \\ 0 & \text{⌘} \vee \text{⌘} \end{cases} \quad (\text{C37})$$

When the bin is circular, and must contain a seed, we can forbid the non-seed vertices to connect the source and the sink:

$$y_{sv}^k \leq \begin{cases} 0 & \textcircled{\times} \wedge \text{⚡} \\ 1 & \text{otherwise} \end{cases} \quad \forall v \in V^+(s) \setminus V_{seed} \quad \text{⚡} \quad (\text{C38})$$

$$y_{ut}^k \leq \begin{cases} 0 & \textcircled{\times} \wedge \text{⚡} \\ 1 & \text{otherwise} \end{cases} \quad \forall u \in V^-(t) \setminus V_{seed} \quad \text{⚡} \quad (\text{C39})$$

Objectives The MILP objective follows [Definition 6](#).

Definition 8 (MILP objective).

$$\max \text{ExplanationScore} \quad (\text{O1})$$

Where

$$\begin{aligned} \text{ExplanationScore} = & \text{PlasmidnessScores} \\ & + \text{PosPlmCovPenalty} \end{aligned}$$

such that:

$$\begin{aligned} \text{PlasmidnessScores} &= \sum_{c \in \mathcal{C}} \text{len}_c \text{plm}_c \sum_{k=1}^n \text{inflow}(c, k) \\ \text{PosPlmCovPenalty} &= \sum_{\substack{c \in \mathcal{C} \\ \text{plm}_c > 0}} \text{len}_c \text{plm}_c \left(\left(\sum_{k=1}^n \text{inflow}(c, k) \right) - \text{cov}_c \right) \end{aligned}$$

C Supplementary results

We select the same 1241 bacteria sequencing projects samples from various bacteria species on the NCBI as in [\[12\]](#)³. Among the 1241 samples, 836 have a ground truth without unlabelled hybrid contigs. Of the 836 samples, a method can fail to return bins, either because it ends without error and with no bins, or because it reaches the time limit before having the time to output any bin. Even if a tool outputs bins, **PlasEval** dissimilarity computation (**comp** command) can fail to return an evaluation because the branch and bound it relies on has reached an upper-bound number of iterations, or because the script reaches the **sbatch** time limit (3 hours). We selected the samples for which every tools we compare output bins and have a **PlasEval** evaluation. In the following, for each figure, the numbers under parenthesis are the numbers of samples for which there is a **PlasEval** evaluation.

³ We reached out to the authors of the recent plasmid classification and binning tools benchmarking article [\[22\]](#) to gain access to their data, but have not received any response as of now.

C.1 Precision, recall and F1

Precision (Figure S2), recall (Figure S3) and F1 score (Figures 3 and S4 and table S2) are obtained with `PlasEval eval` command.

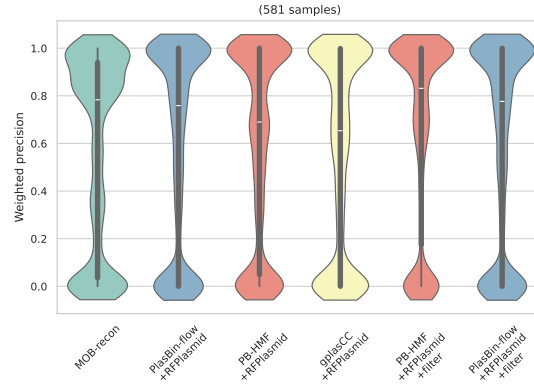


Fig. S2: Precision.

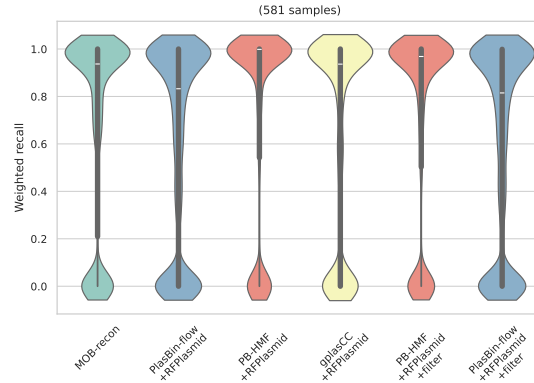


Fig. S3: Recall.

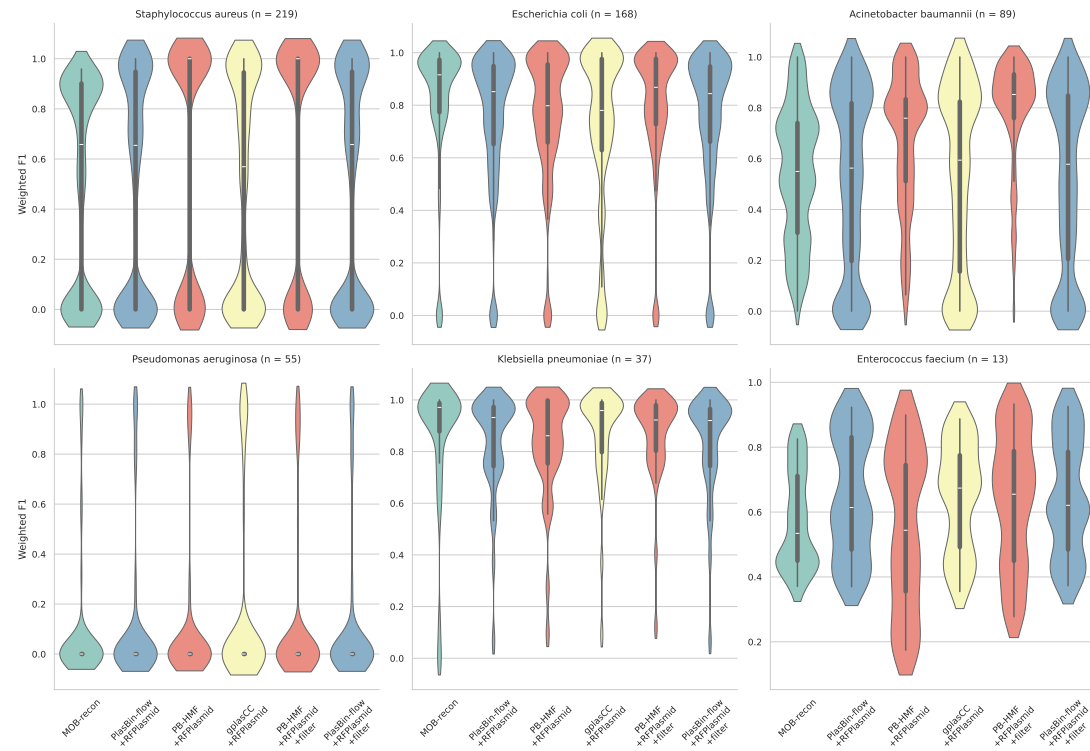


Fig. S4: F1 per species.

Table S2: F1 statistics.

Method	mean	std	min	25%	50%	75%	max
MOB-recon	0.58	0.39	0.00	0.07	0.76	0.92	1.00
PlasBin-flow+RFPlasmid	0.56	0.40	0.00	0.00	0.71	0.93	1.00
PB-HMF+RFPlasmid	0.61	0.40	0.00	0.09	0.78	1.00	1.00
gplasCC+RFPlasmid	0.56	0.40	0.00	0.00	0.71	0.95	1.00
PB-HMF+RFPlasmid+filter	0.66	0.40	0.00	0.30	0.85	0.99	1.00
PlasBin-flow+RFPlasmid+filter	0.57	0.40	0.00	0.00	0.72	0.93	1.00

C.2 Dissimilarity scores

The dissimilarity score (Figures 4 and 5 and table S3), a linear composition of cut (Figure S6), join and extra and missing contigs costs (Figure S5), is obtained with PlasEval comp command.

Table S3: Dissimilarity statistics.

Method	mean	std	min	25%	50%	75%	max
MOB-recon	0.45	0.34	0.00	0.18	0.35	0.68	1.00
PlasBin-flow+RFPlasmid	0.44	0.36	0.00	0.14	0.36	0.67	1.00
PB-HMF+RFPlasmid	0.33	0.33	0.00	0.00	0.27	0.51	1.00
gplasCC+RFPlasmid	0.39	0.37	0.00	0.05	0.30	0.66	1.00
PB-HMF+RFPlasmid+filter	0.30	0.31	0.00	0.00	0.27	0.44	1.00
PlasBin-flow+RFPlasmid+filter	0.44	0.35	0.00	0.15	0.36	0.66	1.00

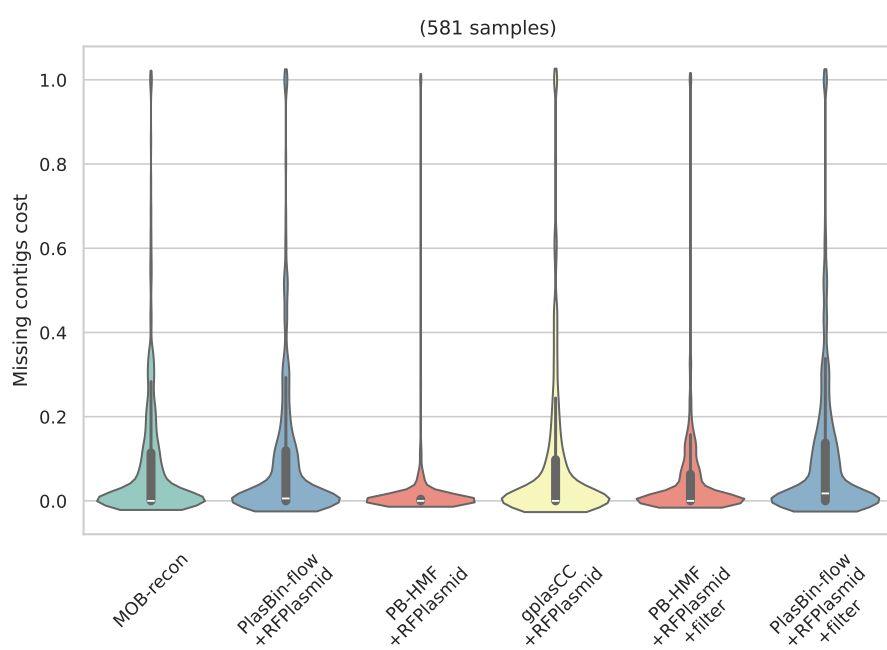


Fig. S5: Missing contigs costs.

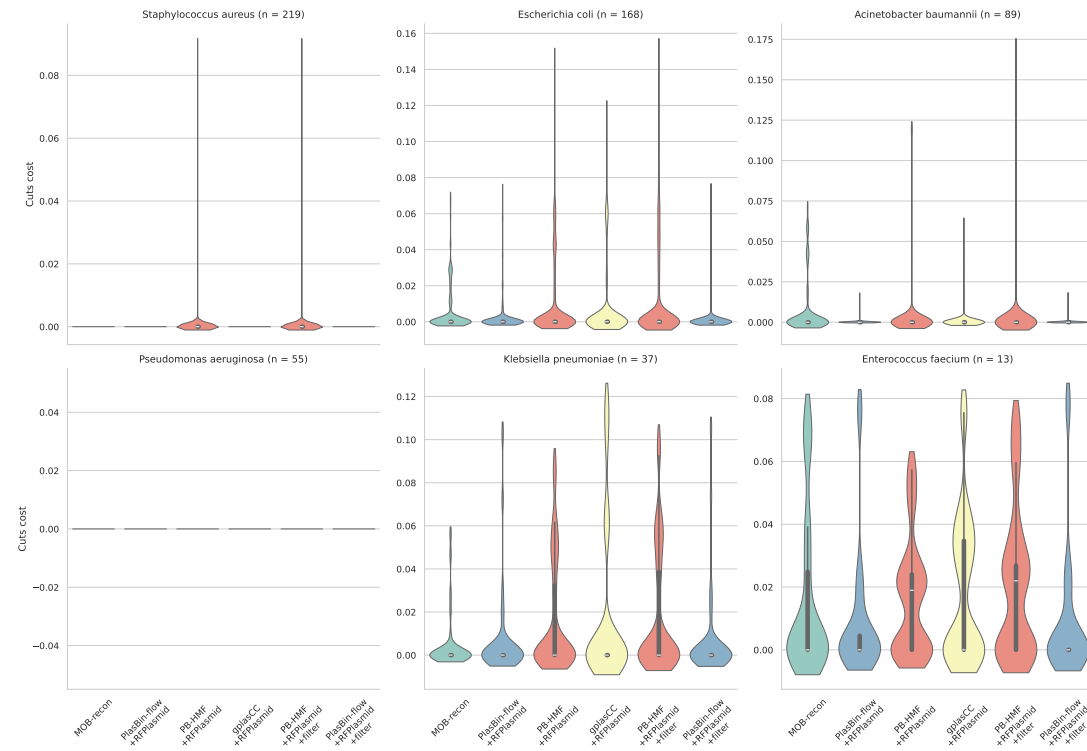


Fig. S6: Cut costs per species.