Minimum Flow Decomposition for Haplotype-Aware Genome Assembly

Rael Rasa Huizenga

To obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on Thursday, 18th of September, 2025, at 10:30 AM.

Student number: 4866886
Project duration: January 6th, 2025 - September 18th 2025

Thesis committee:
Dr. Thomas Abeel, TU Delft, Thesis advisor
Dr. Jasmijn Baaijens, TU Delft, Daily supervisor
Jasper van Bemmelen, TU Delft, Daily co-supervisor
Dr. Alexandru Tomescu, Helsinki University, External Advisor
Dr. Sebastijan Dumančić, TU Delft, Committee member

An electronic version of this thesis is available at
`https://repository.tudelft.nl/`.

Abstract

Accurately reconstructing viral haplotypes from mixed sequencing samples is crucial for tracking viral evolution, detecting new clinically relevant variants, and guiding effective treatments. Existing *de novo* haplotype-aware genome assembly methods typically rely on heuristic path extraction strategies, which limit exploration of the full solution space and may fail to recover low-abundance haplotypes. In this work, we investigate the use of an Integer Linear Programming (ILP) formulation for the Minimum Flow Decomposition (MFD) based problem to reconstruct haplotypes and estimate their abundances through a contig variation graph. Our approach integrates three main components: (1) a new pipeline for constructing contig variation graphs from reads and contigs build from these reads, (2) a Minimum Path Cover (MPC) step to estimate the number of haplotypes, and (3) an MFD-based ILP to infer haplotypes and their abundances, with additional strategies to restrict the set of allowed path weights for improved tractability. We evaluate the method on simulated hepatitis C virus (HCV) and HIV datasets, comparing its assembly quality, abundance estimation accuracy, and tractability against Virus-VG and VG-flow. Results show that for low-haplotype, well-structured graphs, the MFD approach matches or exceeds the performance of existing haplotype-aware methods, with notable advantages on HIV data. However, the runtime grows exponentially with the number of haplotypes, which limits its applicability to low haplotype-count samples. Weight-restriction strategies and improving graph construction can mitigate this runtime challenge. Our findings demonstrate both the potential and current scalability limits of MFD-based assembly.

Contents

1. Introduction

Viruses mutate rapidly [39], giving rise to new genetic variants that can have significant clinical ical consequences. Such a genetically distinct variant of a virus is called a haplotype, which may differ from others in clinically relevant ways, such as how easily it spreads, the severity of the disease it causes, or its resistance to treatments [13, 49]. For example, during the COVID-19 outbreak, some haplotypes spread more rapidly, showed resistance to certain treatments or triggered different immune responses [23]. Because these properties are often specific to individual haplotypes rather than the virus species as a whole, accurately reconstructing haplotypes is crucial for better tracking viral evolution, improving diagnostics, and developing effective treatments.

Within a single infected host, viruses can accumulate mutations rapidly enough to produce multiple coexisting haplotypes. As a result, a genetic sample from an individual often contains a mixture of these closely related haplotypes, known as viral quasispecies [7]. Because these haplotypes arise from the same infection, they share a high degree of genetic similarity. In chronic infections, the number of haplotypes can range from hundreds to thousands [25]. Additionally, hosts can be simultaneously infected by multiple distinct viral haplotypes, typically resulting in two or three co-infecting variants [29, 33].

Haplotype-aware genome assembly is the process of reconstructing the genomes of all individual haplotypes present in a sequencing sample and estimating their relative abundances. Each haplotype is defined by its unique genetic sequence, and in viral populations, they are typically present at different proportions: some occur at high abundance, while others are rare. Because viral haplotypes can be closely related yet functionally distinct, accurate reconstruction and quantification of each one is essential for understanding viral diversity within a sample.

However, reconstructing haplotypes from sequencing data is challenging because samples contain mixtures of closely related sequences. The substantial overlap between haplotypes makes it difficult to distinguish them individually. Dominant haplotypes, which are more abundant in the sample, can easily mask low-abundance haplotypes, even though these less common haplotypes may be clinically significant, such as when they are resistant to treatment [42]. Additionally, sequencing errors can further complicate the process by introducing noise, which can be difficult to distinguish from actual low-abundance haplotypes. The main challenges of haplotype-aware genome assembly are therefore detecting subtle genetic differences between haplotypes and accurately estimating their relative abundance [14].

To address these challenges, different strategies for haplotype reconstruction can be used. One option is to align sequencing reads to known reference genomes. While this reference-based approach can be effective when all haplotypes are already well-characterized, it risks missing new variants because they are not in the set of known reference genomes. For highly diverse viral populations, where new haplotypes frequently arise, this limitation can be critical.

De novo haplotype assembly overcomes this by building genomes directly from the sequencing reads without relying on a reference. By identifying overlaps between reads and piecing them together, *de novo* methods can recover previously unknown haplotypes that would be missed by reference-based techniques. However, because viral haplotypes often share large portions of their genomes, distinguishing between closely related haplotypes remains a significant challenge.

A common step in methods for *de novo* reconstruction of viral haplotypes is to construct a graphical representation of sequences and their abundances in a sample. In this graph, nodes represent (sub-)sequences and each node has an associated abundance that reflects the abundance of its sequence. edges connect sequences which are known to come from the same haplotype. After the graph construction, paths with associated weights are extracted from this graph. A single path represents a reconstructed haplotype and the weight of it corresponds to the abundance of that particular haplotype. Recently, several methods have been proposed for graph-based *de novo* haplotype reconstruction, such as VG-Flow [2], Haploflow [17], and VStrains [38].

While these methods share the common step of using graphs to represent sequence relationships, they differ in how they extract haplotype paths from the graph. VG-Flow first generates a set of candidate paths in a greedy manner and then selects a subset by solving an optimization problem. Haploflow selects the most abundant paths in the graph and uses a subset of these paths to reconstruct the haplotypes. VStrains iteratively selects the longest known sequence (contig) and extends its corresponding sub-path in both directions. A limitation of these methods is their reliance on greedy path extraction strategies, which do not explore the full solution space. As a result, some haplotypes may not be reconstructed because they lie outside the considered solution space [38].

These *de novo* haplotype reconstruction methods can be seen as heuristics for solving the Minimum Flow Decomposition (MFD) problem. In this problem the goal is to decompose a graph into the smallest possible set of weighted paths. This set of paths gives a good estimation of the haplotypes and their abundances because it breaks down the observed node abundance in the graph into a small number of paths, each representing a haplotype. The weight of each path corresponds to the abundance of that haplotype. This approach is parsimonious, which means it tries to avoid creating unnecessary or artificial haplotypes, while reflecting the abundances. For this reason, an MFD is a biologically meaningful way to reconstruct haplotypes from mixed samples.

Current *de novo* haplotype reconstruction methods use greedy methods for path extraction instead of an exact MFD because finding an optimal MFD is an NP-hard problem [47]. An exception is Virus-VG [4], which considers all possible paths; however, it does not explicitly minimize the number of paths and therefore does not guarantee a parsimonious solution, and it also faces runtime issues when applied to larger genomes. Solving the MFD for genome assembly optimally was long considered computationally infeasible, given that the runtime grows exponentially with the number of paths in the decomposition [43]. Recent improvements in strategies for solving the MFD problem have led to significant speedups [11, 20], which makes it possible to solve the MFD problem several times faster than before. In addition, improvements have been made for solvers for Integer Linear programming (ILP), which can be used to solve the MFD. ILP solvers like Gurobi [22] and Highs [27] incorporate a range of internal algorithmic enhancements, enabling fast runtimes in practice, even for problems that are hard in theory. As a result, it may now be feasible to use exact MFD methods for haplotype reconstruction, which may lead to more accurate reconstructions.

1.1. Research questions and contributions

In this work, we present an MFD-based haplotype-aware assembly approach, a method for reconstructing haplotypes from sequencing data. The method is inspired by graph-based viral assembly strategies Virus-VG [4] and VG-Flow [2], but introduces an alternative formulation based on minimal flow decomposition. We apply various optimizations to solve this NP-hard problem more efficiently and we investigate the scalability of this approach on simulated data. To evaluate the method, we compare the quality of assemblies, abundance estimation and tractability of our method with Virus-VG [4], VG-Flow [2], and SAVAGE [3].

We will try to answer the following question: *How can we integrate an MFD ILP formulation to reconstruct haplotypes and their abundances from a mixed sample?*

To answer this question, we look at the following subquestions:

1. What is the maximum complexity of the graphs that we can still solve in a feasible time with an MFD?

2. How does an MFD approach for assembly compare to other methods for strain-aware assembly in terms of quality of the assembly and runtime?

3. What optimizations can we apply to reduce the runtime of the MFD ILP in larger graphs?

Here, we define graph complexity using two factors: (1) the number of haplotypes represented in the graph, and (2) the length of these haplotypes.

In this work, we investigate the integration of a minimal flow decomposition (MFD) formulation based on Integer Linear Programming (ILP) for haplotype-aware genome assembly. The goal is to assess its tractability and its potential to improve assembly accuracy. Unlike existing heuristic-based methods, our approach explores the full solution space, which may enable more accurate haplotype reconstruction and abundance estimation. To position our work within the field, we compare our method to VG-Flow [2] and Virus-VG [4], which also start from a contig variation graph and aim to reconstruct haplotype paths with associated abundances. In particular, we introduce a new method to construct variation graphs, apply a Minimum Path Cover formulation to estimate the number of haplotypes in a sample, and propose strategies to improve the MFD by restricting weights.

2. Biological terminology

2.1. Sequencing technologies and read length

Sequencing is the laboratory process of determining the nucleotide content and order of a DNA or RNA molecule. Modern sequencing machines often cannot directly read an entire genome; instead, they produce short fragments of the genome called *reads*. These reads typically overlap, allowing them to be computationally aligned and assembled into longer sequences.

The two most widely used sequencing approaches are *next-generation sequencing* (NGS) and *third-generation sequencing* (TGS). NGS produces short reads (typically ∼150 base pairs) with low error rates, making it cost-effective and highly accurate. However, the short length limits the ability to resolve large genomic structures without assembly. In contrast, TGS technologies generate much longer reads (1,000–20,000 base pairs or more) but have higher error rates [1]. While NGS remains dominant due to its accuracy and low cost per base, TGS accuracy and affordability are improving rapidly [34].

For haplotype-aware genome assembly, low sequencing error rates are essential because haplotypes often differ only slightly. In such cases, distinguishing between true mutations and sequencing errors is critical. Therefore, in this work, we focus on the use of NGS for our reconstruction pipeline.

2.2. Paired-end reads

In single-end sequencing, a DNA fragment is read from just one end, making it the simplest and most cost-effective approach among next-generation sequencing methods [28]. This is in contrast with *paired-end sequencing*, where both ends of a DNA fragment are sequenced. For a fragment with a known approximate length, the sequencer generates one read from the forward direction and one from the reverse direction. This pairing provides positional information about the reads, as their approximate distance and orientation are known, which can improve assembly accuracy. To benefit from the improved accuracy offered by paired-end sequencing, our pipeline relies on paired-end reads, which are directly used by SAVAGE [3] to create contigs.

2.3. Contigs

A *contig* (short for *contiguous sequence*) is a continuous DNA segment reconstructed from overlapping reads. In haplotype-aware assembly, where multiple highly similar viral haplotypes coexist in a sample, reconstructing full genomes directly is challenging. Therefore, most assembly tools first generate contigs as local consensus sequences, the most likely DNA sequence for

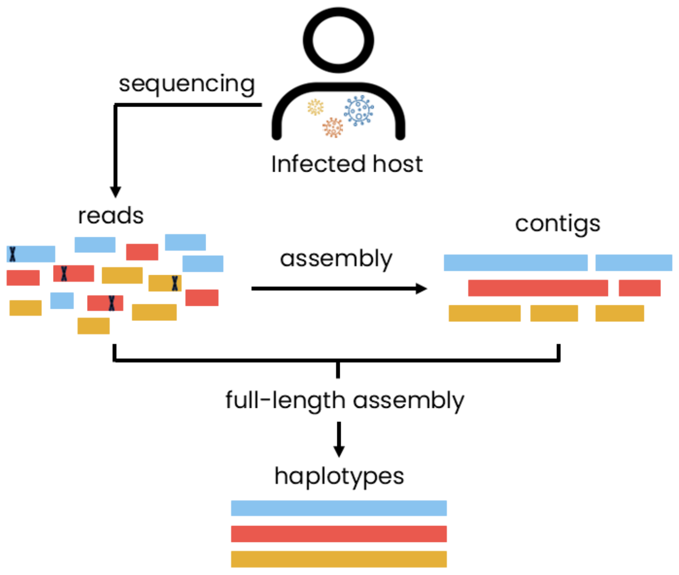


Figure 1: Overview of the steps involved with full-length *de novo* haplotype-aware genome assembly. A host is infected with multiple haplotypes. Sequencing produces reads from a sample. *De novo* reconstruction approaches use the read information to create contigs. The contigs in combination with the reads are used to create full-length haplotypes.

a genomic region, which are then linked to form full-length haplotypes (see Figure 1). State of the art tools for contig generation from mixed-haplotype samples include SPAdes [5] and SAVAGE [3]. SPAdes builds contigs by traversing a de Bruijn graph of sequencing reads, while SAVAGE uses an overlap graph to cluster reads and reconstruct haplotype-specific contigs. In our pipeline, we use SAVAGE because it is specifically designed for viral haplotypes.

2.4. Sequencing coverage

Sequencing coverage refers to the number of distinct sequencing reads that align to a specific position in the genome. Ideally, coverage is uniform across the genome, but in practice, it tends to be lower near the genome ends and exhibits variability due to sampling noise, GC-content bias, and sequencing artifacts [37]. Such uneven coverage poses challenges for *de novo* haplotype-aware assembly methods, which depend on consistent coverage to reliably connect overlapping sequences.

2.5. Reference-based vs *de novo* assembly

Haplotype-aware genome assembly can be performed using either *reference-based* or *de novo* approaches. In reference-based assembly (e.g., PredictHaplo [36], CliqueSNV [30]), reads are aligned to a known reference genome, a previously determined genetic sequence representing the organism, to reconstruct the haplotypes present in the sample. While effective when a suitable reference exists, these methods can introduce bias by assuming sample haplotypes are similar to the reference [46]. This potentially leads to missed or misassembled haplotypes, especially if the virus has undergone substantial mutations [14].

In *de novo* assembly (e.g., VG-Flow [2], VStrains [38], HaploFlow [17]), no reference genome is used. Instead, the genome is reconstructed directly from the sequencing reads. First, reads are assembled into contigs based on overlaps. Then, these contigs are then linked, often using additional information such as read-pair distances or coverage differences, to produce full-length haplotypes. In this thesis, we are looking at the *de novo* setting, which allows for unbiased reconstruction of viral haplotypes even when no suitable reference genome exists.

2.6. Haplotype-aware assembly

Haplotype-aware (or strain-aware) assembly focuses on reconstructing the distinct genomes (haplotypes) of strains within a mixed sample. While contigs can be formed from overlapping reads, distinguishing between highly similar haplotypes requires additional information beyond sequence overlap, such as differences in coverage, to link contigs and accurately reconstruct individual strains. The objective of haplotype-aware assembly methods is to resolve full-length haplotypes despite the high sequence similarity between haplotypes.

3. Methods

3.1. Overview of the approach

We present a Minimum Flow Decomposition (MFD) based approach for reconstructing individual haplotypes from mixed samples, using next-generation sequencing (NGS) reads and a set of pre-assembled contigs from these reads. The method outputs both the sequences of the haplotypes and their estimated relative abundances. This approach builds upon the approach of Virus-VG [4] and VG-flow [2], but differs in that it employs an MFD. Our approach has three main steps, illustrated in Figure 2. The first step of the approach is a new pipeline for constructing a contig variation graph by leveraging existing graph construction tools. In the second step, we formulate and solve a Minimum Path Cover (MPC) problem on the graph to estimate the number of haplotypes present in a sample. These components are necessary to enable the main contribution of this work: the use of an MFD formulation based on Integer Linear Programming (ILP) to find a global optimum solution, instead of the heuristic approaches commonly used in previous methods [2, 38, 17]. By integrating these components, we aim to investigate the effectiveness and scalability of an MFD ILP formulation to reconstruct haplotypes and their abundances from a mixed sample.

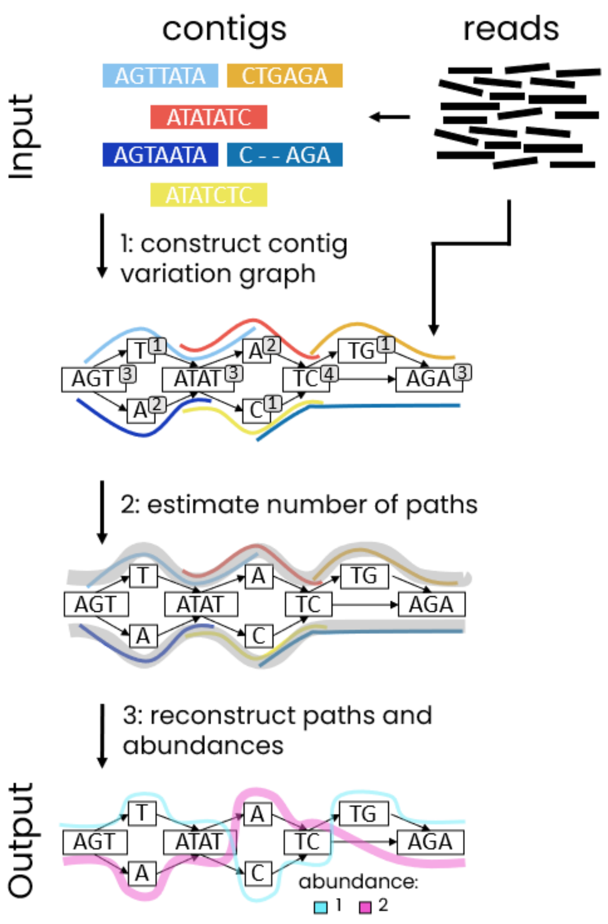


Figure 2: Overview of the steps of the approach. 1: A contig variation graph is constructed from pre-assembled contigs and the reads that were used to build the contigs. 2: The number of haplotypes is estimated using a Minimum Path Cover. 3: Haplotypes and their abundances are inferred by solving the Minimum Flow Decomposition problem.

3.2. Construction of contig variation graph

To enable subsequent haplotype reconstruction, we first build a contig variation graph that represents the diversity captured by the assembled contigs and the underlying sequencing reads. Variation graphs are data structures designed to represent genetic diversity within a population. They provide a compact encoding of a set of genetic sequences by merging shared subsequences across different haplotypes [19, 35]. We construct such a graph from a set of next-generation sequencing (NGS) reads and contigs assembled from them (Figure 2, step 1). For contig assembly, we use the specialized *de novo* viral quasispecies assembler SAVAGE [3], which uses the NGS reads to construct contigs. Contigs are longer sequences built by assembling overlapping reads and serve as error-corrected, longer sequence fragments that provide a reliable foundation for graph construction.

In a contig variation graph, nodes represent sequence segments, and edges indicate that two segments are adjacent in at least one contig. These edges preserve the co-occurrence and ordering of sequences observed in the input contigs, thereby encoding possible paths that correspond to full-length haplotypes. This structure captures the underlying variation between closely related haplotypes. Compared to De Bruijn graphs, which are used in tools such as Vstrains [38] and ViQUF [16], variation graphs constructed from contigs are typically less fragmented and contain fewer sequencing errors, making them better suited for haplotype-aware reconstruction with MFD.

We define a contig variation graph in the following way: Let C be a collection of contigs and l the length of C. We define a contig variation graph as a directed acyclic graph (DAG) $G = (V \cup \{s, t\}, E)$ with a unique source node $s \in V$, connected with an edge to all nodes without incoming edges, and a unique sink node $t \in V$, connected with an edge to all nodes without outgoing edges. It also has an associated set of subpaths $R = \{R_1, ..., R_l\}$ and a weight function $w : V \to \mathbb{R}_{\geq 0}$ which assigns a weight to every node. Each node $v \in V$ encodes a nucleotide sequence that appears as a substring of at least one contig $c \in C$. An edge $(v_1, v_2) \in E$ indicates that the concatenation of the sequences in v_1 and v_2 is also a substring of some contig $c \in C$. The weight of the nodes $\{w_v\}_{v \in V}$ corresponds to the estimated abundance of the sequence it represents. The exact calculation of these abundances is explained later in this section. For each contig $c_i \in C$, there exists a corresponding subpath $R_i \in R$ whose node sequences concatenate to exactly reconstruct c_i. These subpaths ensure that reconstructed haplotypes remain consistent with the contig structure. By representing sequence diversity in this way, the contig variation graph serves as the basis for reconstructing complete haplotypes from the assembled contigs.

Virus-VG [4] and VG-flow [2] introduced contig variation graphs as an intermediate for full-length haplotype reconstruction. These approaches make use of multiple sequence alignment with the `vg msga` tool to build the variation graph. However, `vg msga` has since been deprecated. Moreover, to our knowledge, no existing tool fully satisfies the specific requirements of our method: we start from a set of oriented contigs, aim to preserve all sequence variation without collapsing differences, and require the resulting graph to be acyclic. For this reason, we propose a new approach for constructing a contig variation graph tailored to these requirements.

After assembling contigs with SAVAGE [3] from a set of sequencing reads, we build a contig variation graph using both the constructed contigs and the reads using the following six steps:

1. **Contig filtering.** SAVAGE produces a collection of contigs, which may include erroneous low-abundance sequences. To reduce noise from these erroneous contigs, we estimate contig abundances using Kallisto [6]. Kallisto estimates the abundance of the contigs by pseudo-aligning the input reads back to them. We filter out contigs with transcript-per-million (TPM) value below two, as these are considered to be noise.

2. **All-vs-all contig alignment.** An all-vs-all alignment of the remaining contigs is performed using Minimap2 [31]. Since SAVAGE outputs contigs in a consistent forward orientation, alignments are restricted to the forward strand only.

3. **Graph induction.** SeqWish [18] is used to construct an initial variation graph from the aligned contigs. SeqWish identifies shared sequences and makes a graph that compactly represents their overlaps. We set a minimum overlap length of 20 bases to define valid overlaps. We set the `repeat-max` parameter to one, to limit each input base to participate in at most one transitive closure, preventing redundant representation of the same sequence region. The `min-repeat-distance` parameter is set to 1000 to ensure that transitive closure is not applied to bases separated by less than 1000 bases in the input, avoiding the collapse of closely spaced repetitive elements into a single node.

4. **Graph simplification.** The graph is simplified using the `vg mod` tool [19]. Long nodes are split so none exceeds 1024 bases, enabling compatibility with `vg map`. The graph is normalized by removing redundant edges and nodes. Nodes connected by a single edge are merged to reduce complexity.

5. **Node abundances estimation from read mapping.** Reads are mapped to the contig variation graph using `vg map` [19], to estimate node abundances w_v. The weight of a node v, w_v, is defined as the total number of read bases aligned to the sequence represented by v, divided by the length of the sequence stored in v. This normalization accounts for differences in sequence length and gives a per-base abundance estimate. This is the same

approach as used in Virus-VG [4] and VG-flow [2], to produce abundance-based weights in variation graphs.

6. **Low-abundance node removal.** To reduce the noise introduced by errors in contigs, nodes with abundance below 0.5% of the maximum node abundance are removed. Such low-abundance nodes are likely to be erroneous and may introduce unnecessary complexity. Without this filtering step, fewer samples were able to finish, as shown in Supplement C, Table 6.

The resulting graph can be used directly by Virus-VG and VG-flow. However, for the downstream MPC (step 2) and MFD (step 3) procedures, the graph representation must be modified so that weights are assigned to edges rather than nodes. To achieve this, we transform each original node $v \in V$ into two nodes, v_1 and v_2, connected by a new edge $e = (v_1, v_2)$. The weight originally associated with node v is transferred to the edge e. Additionally, all incoming edges of v are reconnected to v_1, and all outgoing edges are connected from v_2. This transformation produces a graph with edge weights suitable for use in the subsequent steps.

3.3. Reconstruction of haplotypes with MFD

We model haplotype abundances as a nonnegative flow f on the contig variation graph $G = (V \cup \{s,t\}, E$. For each edge $(u,v) \in E$ the flow value $(f_{uv} \geq 0$ represents the total abundance of haplotypes traversing that edge. Flow conservation enforces that, for every internal node $v \in V$, the total incoming flow equals the total outgoing flow:

$$\sum_{(u,v)\in E} f_{uv} = \sum_{(v,w)\in E} f_{vw}, \quad \forall v \in V, \tag{1}$$

Given a flow that satisfies these constraints, haplotype reconstruction is obtained with Minimum Flow Decomposition (MFD) by decomposing the flow into a minimal set of weighted s-to-t paths that explain all of the weights in the graph. Each path corresponds to a haplotype, and its associated weight provides an estimate of the corresponding haplotype's relative abundance. Ideally, we want to find the exact minimum flow decomposition in the contig variation graph. However, in practice, due to sequencing errors and inaccuracies in node weights in the contigs variation graph, an exact MFD is often not feasible. Therefore, we instead estimate the number of haplotypes with a Minimum Path Cover (MPC), and use a flow decomposition formulation that allows for error handling. In what follows, we will first introduce a formulation for the exact MFD problem on error-free graphs, which serves as the basis for our approach. Afterwards, we will show how to adapt this formulation to allow for errors in the graph.

3.3.1. Exact MFD

The exact MFD problem seeks a decomposition of the input flow into the smallest number of paths such that the sum of the weights from these paths exactly matches the observed node abundances. We solve this problem using the recently developed FlowPaths package [45], which builds on several advances in flow decomposition computation from recent literature [11, 12, 20, 40]. The flow decomposition in this package is defined as an ILP in the following way:

$$
\begin{aligned}
\min \quad & 0 && && \text{(2a)}\\
\text{s.t.} \quad & && && \\
& \sum_{(s,v)\in E} x_{svi} = 1, && \forall i \in \{1, ..., k\}, && \text{(2b)}\\
& \sum_{(u,t)\in E} x_{uti} = 1, && \forall i \in \{1, ..., k\}, && \text{(2c)}\\
& \sum_{(u,v)\in E} x_{uvi} - \sum_{(v,w)\in E} x_{vwi} = 0, && \forall i \in \{1, ..., k\}, && \text{(2d)}\\
& f_{uv} = \sum_{i\in\{1,...,k\}} \pi_{uvi}, && \forall (u,v) \in E, && \text{(2e)}\\
& \pi_{uvi} \le M x_{uvi}, && \forall (u,v) \in E, \forall i \in \{1, ..., k\}, && \text{(2f)}\\
& \pi_{uvi} \le w_i, && \forall (u,v) \in E, \forall i \in \{1, ..., k\}, && \text{(2g)}\\
& \pi_{uvi} \ge w_i - (1 - x_{uvi})M, && \forall (u,v) \in E, \forall i \in \{1, ..., k\}, && \text{(2h)}\\
& w_i \in \mathbb{Z}^+, && \forall i \in \{1, ..., k\}, && \text{(2i)}\\
& x_{uvi} \in \{0, 1\}, && \forall (u,v) \in E, \forall i \in \{1, ..., k\}, && \text{(2j)}\\
& \pi_{uvi} \in \mathbb{R}^+ \cup \{0\}, && \forall (u,v) \in E, \forall i \in \{1, ..., k\}. && \text{(2k)}
\end{aligned}
$$

Variables	
x_{uvi}	Binary variable corresponding to the usage of edge $(u, v) \in E$ in flow path $i \in \{1, ..., k\}$.
w_i	Integer variable corresponding to the weight of flow path $i \in \{1, ..., k\}$.
π_{uvi}	Integer variable corresponding to the product of the weight of flow path $i \in \{1, ..., k\}$ and the usage of edge $(u, v) \in E$ in the same flow path .
f_{uv}	The flow weight on edge uv, which corresponds to the estimated abundance of the sequence it represents.
M	A sufficiently large upper bound for any w_i, for all $i \in \{1, ..., k\}$.

Table 1: The variables and constants used in the ILP and their descriptions

A description of the variables in the formulation can be found in Table 1. Note that the ILP does not have an objective, and the goal instead is to find a feasible solution of k paths that satisfy all the constraints. With constraints 2b and 2c we model that every path starts at the source node s and ends at the sink node t. Constraints 2d make sure that the incoming flow of any node $v \in V$ matches the outgoing flow. Next, we require the sum of the weights of the paths going through an edge to match the weight of that edge with constraints 2e. Constraints 2f, 2g, 2h are used to linearize the formulation. To obtain a minimum flow decomposition, we solve the ILP iteratively for increasing values of k. The first value of k for which such a flow decomposition is feasible is the MFD.

To ensure that the reconstructed haplotype paths are consistent with the structure of the original contigs, we introduce subpath constraints into the flow decomposition. With subpath constraints, we require that each contig is covered at least once by some path in the flow decomposition. We use subpath constraints introduced by Williams et al. [48], which are formulated as follows:

$$\sum_{i \in \{1,...,k\}} r_{ij} \geq 1, \qquad \forall R_j \in R, \tag{3a}$$

$$\sum_{(u,v) \in R_j} x_{uvi} \geq |R_j| * r_{ij}, \qquad \forall i \in \{1,...,k\}, \forall R_j \in R, \tag{3b}$$

$$r_{ij} \in \{0,1\}, \qquad \forall i \in \{1,...,k\}, \forall j \in \{1,...,|R|\} \tag{3c}$$

Given a set of subpaths R, for each subpath $R_j \in R$, we introduce additional binary variables r_{ij} denoting the presence of the subpath R_j in the ith path. $r_{ij} = 1$ if and only if each edge (u,v) in R_j is covered by path i. $|R_j|$ denotes the length (i.e., number of edges) of subpath constraint R_j. With constraints 3a we enforce every subpath to be used at least once, and with constraints 3b we enforce every subpath to be full length. Finally, with constraints 3c we define r_{ij} as a binary variable. Taken together, these constraints ensure that the ILP solution corresponds to a biologically meaningful decomposition, where each haplotype path is consistent with both the graph flow and the contig-derived subpaths

In Virus-VG and VG-flow, the decomposition is restricted to use only entire contigs: each reconstructed path must correspond to one or more full-length contigs, and partial contig segments are not allowed. This is a slightly different restriction for the paths than we enforce. While Virus-VG and VG-flow enforce that reconstructed paths only use full contig paths, the constraints we use require only each that each contig is covered at least once by some path. This allows individual paths to include partial segments of contigs. Another difference is that with the constraints we use, we require all the subpaths to be used at least once. This is something that is not required for VG-flow and Virus-VG, which means that subpaths can be omitted in their implementation. We also experimented with implementing the subpath constraints used in VG-flow and Virus-VG, however, this work is not completely finished and therefore only further discussed in the discussion section.

3.3.2. Error handling MFD and estimation number of paths with MPC

An exact Minimum Flow Decomposition is only feasible when both the graph structure and the node abundance estimates are perfectly accurate. In practice, however, the contig variation graph often contains imperfections due to sequencing errors, misassemblies, and inaccuracies in the node weights derived from read alignments. These inconsistencies can make an exact decomposition impossible or overly sensitive to noise. To address this, we relax the requirement that the reconstructed flow must exactly match the observed node abundances. Instead, we allow deviations on each edge and aim to minimize the total absolute error between the reconstructed and observed abundances. We do this by removing the flow equality constraints (2e) from the exact MFD formulation and introducing the following objective function:

$$min \sum_{(u,v) \in E} |f_{uv} - \sum_{i \in \{1,...,k\}} \pi_{uvi}|. \tag{4}$$

This objective function is similar to the one used in VG-flow and Virus-VG, but with an important distinction: neither of those methods explicitly minimizes the number of paths. VG-flow applies the objective only to a restricted set of candidate paths, while Virus-VG evaluates it over all possible paths that satisfy the subpath constraints. However, because VG-flow does not consider the full path space, and both methods do not explicitly penalize the number of paths, they do not aim for the parsimonious solution, that is, the simplest set of paths that can explain the observed data, which is often the most biologically plausible.

In contrast to the exact formulation, which seeks the smallest number of paths k that explain the flow under a set of constraints, our objective is to minimize an error metric and to minimize the number of paths. This makes the concept of a uniquely defined "minimum" flow decomposition less straightforward. Previous work defined the minimum flow decomposition as the one with the smallest number of paths that achieves a reasonably low error, solving the MFD repeatedly with increasing path counts until the error stabilizes [10]. However, this iterative approach typically requires solving at least one unnecessary additional decomposition just to verify if the objective improves, which is computationally expensive because the complexity of a flow decomposition grows exponentially with the number of paths [43]. To mitigate this inefficiency, we focus on minimizing the absolute error while restricting ourselves to the minimal number of paths necessary to decompose the graph, which is also consistent with the principle of parsimony.

To determine this minimal path count, we first compute the Minimum Path Cover (MPC) of the graph (Figure 2, step 2). The MPC problem seeks the smallest set of paths that together cover all edges in the graph. Although the MPC does not consider node abundances, it provides a computationally efficient lower bound on the number of necessary paths. We extend this approach by considering a modified MPC problem, where the goal is to cover not only all edges, but also all relevant subpaths within the graph. This variant can be formulated as an Integer Linear Program (ILP). The detailed ILP formulation is provided in Supplement A.

To our knowledge, the use of MPC to estimate the number of paths in a *de novo* viral haplotype reconstruction setting has not been used before. While MPC has been applied in reference-based viral population inference [15] and transcript assembly [44], it is new to integrate it as a guiding step for an MFD.

3.3.3. Restricting the set of allowed weights MFD

The solution space of the MFD problem can be extremely large because of the many options to assign weights to paths. Exploring the full space of possible weights is computationally expensive and often impossible. To reduce the computational complexity, we therefore also test a version of the MFD where we constrain the solution space using K-means clustering to predefine a set of allowed path weights. Note that we use the capital letter K here to avoid confusion with the k we use to denote the number of paths in the flow decomposition. We construct this set of weights as follows:

1. We perform K-means clustering on the edge weights of the graph. To select K, we apply the elbow method, scanning values in the range $K \in [4, 50]$. The elbow is identified as the 'knee' point in the within-cluster variance curve, i.e., the point with the maximum change in slope, where adding further clusters yields only marginal improvements.
2. The cluster centroids are used as the set of allowed weights for the flow paths.
3. To account for potential overestimation in the number of paths (e.g., when the number of reconstructed paths exceeds the number of true haplotypes), we explicitly include one zero-weight in the candidate set, allowing the model to assign the zero-weight to a path.

After decomposition, any paths assigned a weight of zero are discarded from the final solution. Note that this variant no longer considers the full solution space of the original MFD problem. Instead, it trades off exactness for tractability by limiting the range of possible path weights. Together, these formulations allow us to reconstruct haplotypes as a parsimonious set of weighted paths.

3.4. Data

We evaluate our methods using two different types of datasets. The first dataset is designed to test the limits of the MFD step in terms of runtime. It consists of artificially constructed variation graphs that are error-free and precisely match the haplotypes they represent. This setup allows us to isolate and assess the computational complexity of the exact MFD step alone.

The second dataset is intended to evaluate the performance of our method in reconstructing haplotypes from actual mixtures. It consists of simulated reads generated from mixtures of real HIV and HCV haplotypes, providing a more realistic evaluation scenario. Using this dataset, we assess both the accuracy of the inexact MFD in reconstructing haplotypes and their abundances, as well as the computational tractability of the approach. We benchmark its performance against other haplotype-aware genome assembly methods: VG-flow [2], Virus-VG [4] and SAVAGE [3].

3.4.1. Perfect graph simulation

The process for generating perfect graphs is illustrated in Figure 3. We begin by constructing a reference haplotype as a random DNA sequence of 2000 nucleotides composed of the characters {A, C, T, G}. To create a set of closely related haplotype, we introduce random mutations into the reference haplotype at a rate of 0.005, meaning each base has a 0.5% chance of mutating into one of the other three nucleotides. To assign abundance values to each haplotype, random weights are generated by sampling from an exponential distribution. The scale parameter of the distribution is set to 2^k, where k is the number of haplotypes, causing the variance in abundances to increase exponentially with population size. This leads to a distribution where most haplotypes have low abundance values, while a few have substantially higher abundances. We do this because, in viral populations, it is common to observe one or a few dominant haplotypes alongside many low-abundance ones [13].

The generated haplotypes are used to construct a variation graph, where the weight of each node corresponds to the sum of the abundances of all haplotypes that pass through it. We create fragments of 100 nucleotides and an overlap of 20 nucleotides of the haplotypes (see Figure 3. These fragments represent the contigs and are saved as subpaths in the variation graph. The resulting perfect variation graph is used to test the computational performance of the exact MFD step.

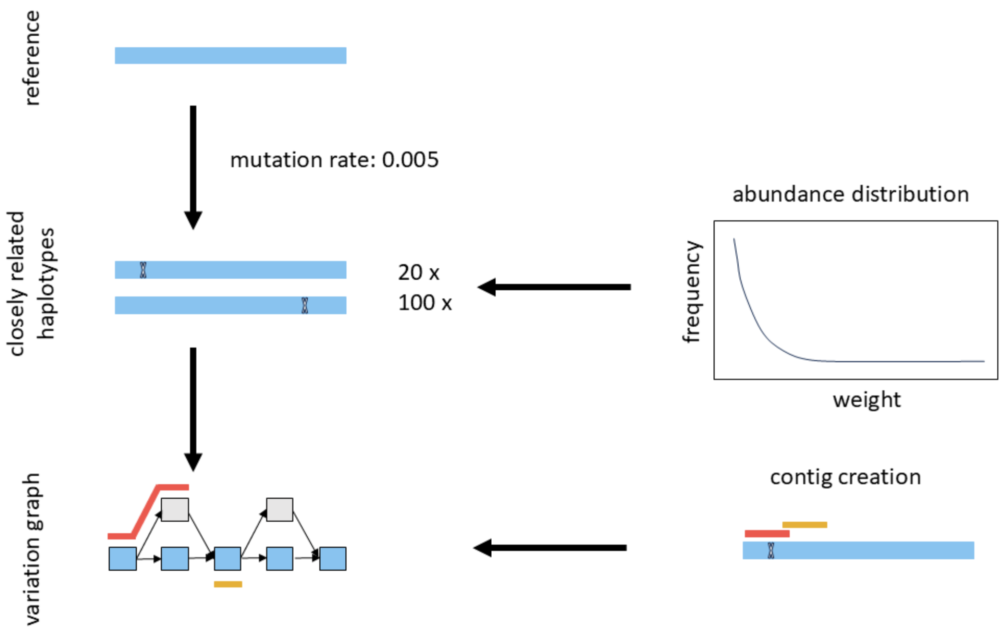


Figure 3: Perfect graph simulation: A reference haplotype is mutated to create a set of closely related haplotype. The abundance is taken from an exponential distribution. Contigs are overlapping parts of the sequences. The sequences, abundances and contigs are used to create a variation graph.

3.4.2. Data simulation mixed samples

To evaluate the performance of our method in a realistic setting, we generated synthetic datasets based on two commonly used RNA viruses: hepatitis C virus (HCV) and HIV-1. These viruses are frequently used to benchmark viral haplotype assemblers due to their high mutation rates [2, 4, 38]. We simulated ten samples for each chosen number of haplotypes. In each sample, haplotype abundances were drawn from an exponential distribution with a minimum relative abundance threshold of 5% per haplotype. The abundances were then rescaled to yield an average haplotype coverage of 750×. Reads were simulated using ART [26] with 150 bp paired-end HiSeq settings, a fragment size of 300 bp and a standard deviation of 10.

To comprehensively assess our method's performance across different viral genomic characteristics, we generated datasets for both HCV and HIV, which differ in sequence divergence, and structural features.

HCV Hepatitis C virus (HCV) is a positive-sense single-stranded RNA virus that primarily infects the liver. It is characterized by high genetic diversity, with multiple genotypes and subtypes circulating worldwide [32]. For the HCV dataset, each sample consists of three to seven haplotypes selected from the SAVAGE benchmark dataset [3]. These haplotypes have a pairwise divergence ranging from 6% to 9% and are approximately 9300 bases long.

HIV Human immunodeficiency virus type 1 (HIV-1) is a retrovirus, with a positive-sense single-stranded RNA genome. HIV-1 infects immune cells, leading to progressive immune deficiency and, if untreated, acquired immunodeficiency syndrome (AIDS) [41]. The virus exhibits extensive genetic diversity due to rapid replication, high mutation rates, and recombination, which complicates both treatment and vaccine efforts [24]. The HIV samples contain two to five haplotypes selected from five known HIV-1 reference strains [9], which have a pairwise divergence of 1% to 6%. HIV-1 contains long terminal repeats (LTRs) at its genome ends, which can introduce cycles in the contig variation graph. Since our MFD formulation assumes acyclic graphs, we remove the LTRS by retaining only the central 8000 bp of each haplotype (the coding region is approximately 8627 bp long) before read simulation.

3.5. Availability of code and data

All scripts used to run the models and perform the analysis are publicly available at: `https://github.com/Raelhuizenga/MinimumFlowDecomposition`. The perfect graphs, the simulated datasets and the corresponding contig variation graphs for the HCV and HIV experiments are available at: `https://doi.org/10.5281/zenodo.16910556`.

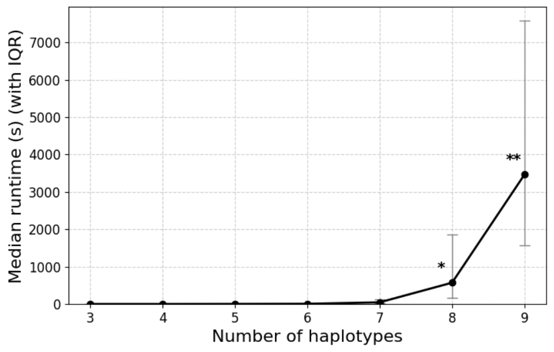
4. Results

All experiments were conducted on the Delft AI Cluster [8], using hardware specifications: 2 x Intel(R) Xeon(R) E5-2680 v4 CPUs at 2.40GHz. All methods were run with 8 threads and given a maximum runtime of 24 hours and maximum memory usage of 64 GB, unless specified otherwise.

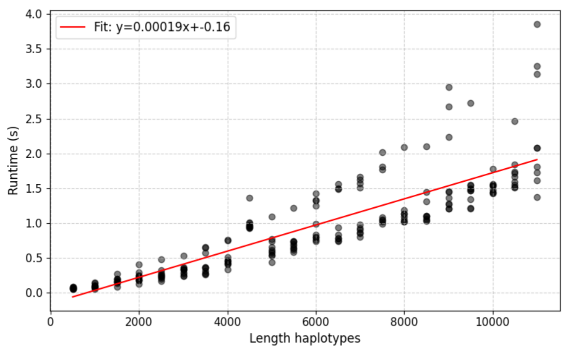
4.1. Runtime MFD grows exponentially with the number of haplotypes

To evaluate the scalability of the MFD step, we first tested it on a set of perfect graphs without sequencing errors. Because these graphs are error-free, they can be solved with an exact MFD. This enabled us to analyze the effect of the number of haplotypes as well as the respective genome lengths on the runtime of the MFD step. For each haplotype count (3–9), we simulated ten perfect graphs (Methods 3.4.1). Similarly, to study the effect of genome length, we generated ten perfect graphs per length (500–11,000 bp) with a fixed haplotype count of three. All graphs were then solved using the exact MFD, executed on four threads per sample.

As shown in Figure 4a, the runtime increases exponentially with the number of haplotypes present in the sample (See Supplement D for the corresponding figure on a log scale). For graphs with more than seven haplotypes, multiple samples failed to complete within the 24-hour time limit. These results confirm that the computational complexity of the MFD step poses a challenge when scaling to samples with more haplotypes. In contrast, the length of the haplotype sequences does not substantially affect runtime in this error-free setting. Figure 4b shows that, for a fixed number of haplotypes of three, runtime does not increase exponentially with haplotype length. Instead, the samples appear to form two distinct groups: one group exhibits a roughly linear increase in runtime as haplotype length grows, while the other group shows a faster rate of increase. However, based on this graph and the available number of samples, it is not yet clear whether the runtime in the second group grows exponentially or follows a different pattern. Overall, these results indicate that runtime is primarily driven by the number of haplotypes rather than their lengths, highlighting a key limitation of the MFD approach when scaling to highly diverse samples.



(a) Median runtime (in seconds) required to compute an exact MFD in basic simulated graphs with the IQR. The dataset consists of ten graphs per haplotype count. *: one sample with eight strains did not finish within 24 hours **: three samples with nine haplotypes did not finish within 24 hours.



(b) Scatter plot of runtime (in seconds) required to compute an exact MFD in basic simulated graphs constructed from haplotypes with varying genome lengths and a fixed number of haplotypes of three. The red line is a linear fit through these points.

Figure 4: Relation between the number of haplotypes in a sample, the length of the haplotypes and the runtime of the MFD step

4.2. MPC can accurately estimate the number of haplotypes in a variation graph

As shown in the last section (4.1) a high number of haplotypes in a sample poses a computational challenge for the MFD step. So to accelerate the MFD we introduced an intermediate step to estimate the number of haplotypes in the sample using a Minimum Path Cover (MPC) (see Methods 3.3.2). We use this estimation as the number paths required to decompose the graph. We evaluated the performance of the MPC to estimate the number of haplotypes on variation graphs constructed from simulated HCV and HIV mixed-strain samples (Methods 3.4.2), under two settings: with and without subpath constraints derived from the contigs used to construct the graph. In both of these settings, the MPC step was completed within 15 minutes for all samples.

Figure 5 shows the number of paths found by MPC compared to the true haplotype count for both the HCV and HIV samples. When subpath constraints are included (Figure 5a, c), MPC either correctly estimates or overestimates the haplotype count. This suggests that while the variation graph likely captures all true variation, some contigs may be erroneous, introducing additional subpaths that force the cover to include extra paths. This overestimation will increase the runtime of the subsequent MFD step as the flow must be decomposed into more paths.

In contrast, excluding subpath constraints (Figure 5b, d) yields estimates that more closely match the true haplotype count. However, subpath constraints remain essential for solving the MFD, as they constrain the solution space to only those path combinations consistent with the input contigs, thereby reducing the solution space to a tractable size and ensuring biological plausibility. Therefore, while MPC without subpath constraints provides a good estimate of haplotype count, the full decomposition must incorporate subpaths. Since MPC provides a lower bound on the number of paths needed for any feasible flow decomposition, and a decomposition with fewer paths would be impossible, our approach uses the MPC with subpath constraints as the basis for the subsequent flow decomposition to reconstruct haplotypes. Overall, MPC offers a fast and reliable haplotype count estimate that, when combined with subpath constraints, can be used by the subsequent flow decomposition step.

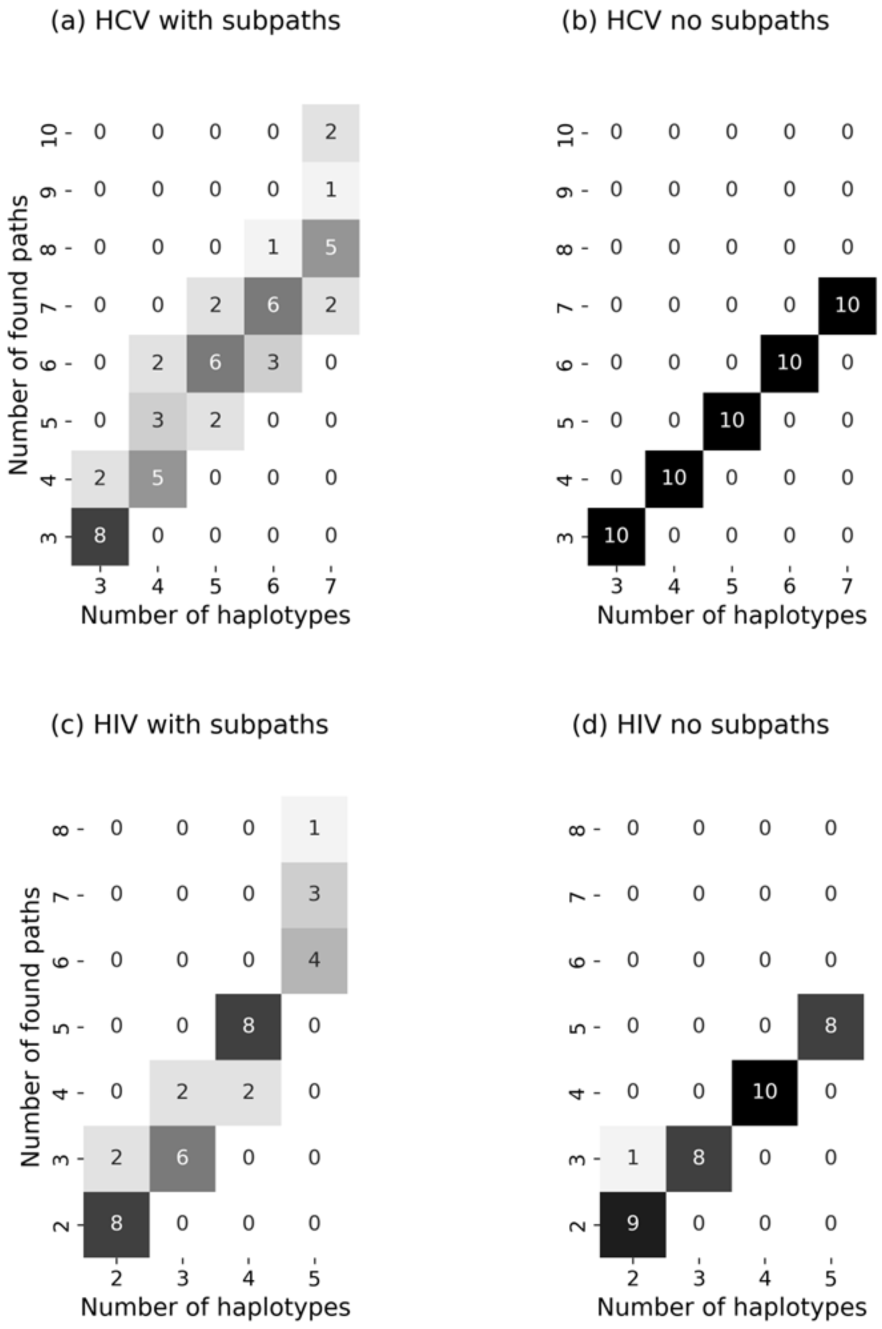


Figure 5: Heatmaps showing the number of paths found by the MPC algorithm (y-axis) versus the ground truth number of haplotypes (x-axis) on different datasets and with and without subpath constraints. For HIV, some graphs contained cycles and were excluded from the evaluation; consequently, the HIV 3-haplotype and 5-haplotype sets comprise eight graphs instead of ten. (a) HCV samples with subpath constraints. (b) HCV samples without subpath constraints. (c) HIV samples with subpath constraints. (d) HIV samples without subpath constraints.

4.3. Comparing MFD approach against other haplotype-aware assembly methods

We estimate the number of haplotypes in each sample using the MPC approach, and subsequently apply least absolute error flow decomposition to reconstruct the haplotypes. We compare this approach to two other *de novo* haplotype-aware assembly methods, VG-flow [2] and Virus-VG [4], both of which also rely on SAVAGE [3] contigs as input and start from a contig variation graph to reconstruct haplotype paths with associated abundances. These tools represent two alternative strategies for path selection in the variation graph: VG-flow uses a heuristic that greedily constructs candidate paths before optimizing abundances over this reduced set, and Virus-VG enumerates all possible paths, given the contig information, and estimates the path abundances. Virus-VG and VG-flow differ fundamentally from the MFD approach, because they do not explicitly minimize the number of paths.

In contrast, the MFD-based approach considers the entire solution space given by the subpath constraints from the contigs, while directly minimizing the number of reconstructed paths, ensuring a parsimonious set of haplotypes consistent with the input contigs. We compare this approach to both the heuristic-based method (VG-flow) and the exhaustive non-minimizing method (Virus-VG). For completeness, we also compare to the original SAVAGE output. All methods were evaluated on simulated HCV and HIV mixtures (Methods 3.4.2).

Our analysis revealed two categories of samples. In the first group, the estimated number of haplotypes was the same with and without enforcing subpath constraints in MPC. For these samples, the MFD could be solved efficiently, within two minutes, because of the ability of the solver quickly find a good initial solution. In the second group, the MPC with subpath constraints overestimated the number of haplotypes. This is probably due to erroneous contigs, which introduce wrong subpaths in the graph. Although a few of these samples were solved, the vast majority timed out. The samples that were solved, showed consistently a worse error-rate than Virus-VG and VG-flow. The full results can be found in Supplement F, Table 8.

Because the second group of samples frequently led to an overestimation of the number of haplotypes and solver timeouts, a direct comparison across all methods would not be meaningful. To ensure a fair and consistent evaluation, we therefore restricted our analysis to the first group of samples, in which almost all methods produced complete solutions. Within this subset, we assessed assembly quality using QUAST [21], based on the metrics summarized in Table 2. We present the results for low-complexity (three haplotypes for HCV and two haplotypes for HIV) and the most complex samples that the MFD could solve (seven haplotypes for HCV and four haplotypes for HIV). Results for intermediate haplotype counts are provided in Supplement E, Table 7, and exhibit largely consistent trends with those shown here.

4.3.1. MFD reconstructs haplotypes effectively when the input graphs are well-structured

For samples where the haplotype count was estimated correctly, the MFD approach is fast and all samples finished within five minutes. Overall, VG-flow was the fastest method, finishing all samples in under a minute. Virus-VG was also able to find a solution fast in most cases, though it failed to complete some of the more complex four or five haplotype HIV mixtures within the 24-hour limit. Interestingly, in one such case, MFD successfully completed while Virus-VG did not. This sample contained many short contigs, which expands Virus-VG's search space but do not substantially impact the MFD as long as the input contigs are error-free.

Table 3 summarizes the assembly performance across all evaluated methods. To assess statistical significance, we applied a one-sided Wilcoxon signed-rank test to evaluate whether the MFD approach outperforms the other methods (p-value < 0.05). MFD achieves a significantly higher genome fraction than both VG-flow and Virus-VG (see Supplement G.1 Table 9 for the p-values). This means that the MFD approach can reconstruct more complete haplotypes in a

Metric	Description
Genome fraction	The percentage of the reference genome covered by aligned bases, calculated as (aligned bases in the reference / genome size) $\times$ 100. A higher percentage is better.
Duplication ratio	The total number of aligned bases in the assembly, divided by the total number of aligned bases in the references. A value > 1 indicates redundancy.
NG50	Length for which all contigs in the assembly of at least this length together add up to at least 50% of the total target length. A number close to the true length of the haplotypes indicate a better assembly.
ER	Error rate: percentage of mismatches, indels and unambiguous bases (N's) per aligned base. A low error rate is a better.
# contigs	Number of contigs ($>$ 500 bp): count of assembled contigs that are longer than 500 bases.

Table 2: Assembly evaluation metrics and their descriptions.

sample than VG-flow and Virus-VG. It also has the best duplication ratio, but this is because we only look at the samples where we correctly estimated the number of haplotypes in the sample. Finally, every method except SAVAGE produced assemblies with NG50 values close to the true haplotype lengths, and substantially higher than those of SAVAGE. This indicates their ability to reconstruct full-length haplotypes, in contrast to SAVAGE which produces more fragmented assemblies.

Regarding the error rate, SAVAGE achieves near-zero errors across all samples. This means that the contigs used to construct the variation graphs are generally error-free. If there are errors in the contig from SAVAGE that produce low-abundance nodes, they will be filtered out during the graph construction. This results in the variation graphs being nearly error-free, except for missing information that causes the genome fraction to be less than 100%. The MFD approach has a significantly lower error rate than Virus-VG on both the HCV and the HIV dataset. Compared to VG-flow, MFD performs significantly better on the HIV dataset, while on the HCV dataset both methods maintain comparably low error rates (see Supplement G.2 Table 10). The main distinction is that MFD maintains consistently low error rates across both datasets, whereas Virus-VG and VG-flow show larger variability. In particular, VG-flow is stable on HCV but less so on HIV, while Virus-VG exhibits occasional high-error outliers in both datasets.

More specifically, we see that MFD approach maintains a low error rate, consistently below 0.022% on the HCV dataset and below 0.003 % on the HIV dataset. While Virus-VG and VG-flow perform well in many cases, their error rates vary more widely. Some samples showed notably higher error rate as 0.687% for VG-flow on a 4-strain HIV sample and 1.123% for Virus-VG on a 3-strain HIV sample, as can be seen in the boxplots in Supplement G.2 Figure 10. Overall, these results demonstrate that when the input variation graphs are well-structured, the MFD approach achieves assembly quality comparable to or better than VG-flow and Virus-VG. However, its strong performance relies on the graphs being well-structured, which remains a key limitation of the approach.

	genome fraction	duplication ratio	NG50	ER	# contigs
3-strain HCV (8 samples)					
MFD	99.863%	**1.0**	9290.5	0.005%	**3.0**
VG-flow	99.452%	1.166	9280.1	0.008%	3.5
Virus-VG	99.787%	1.501	9280.25	0.209%	4.5
SAVAGE	**99.912%**	1.236	2213.6	**0.000%**	29.9
7-strain HCV (2 samples)					
MFD	99.867%	**1.0**	9287.0	0.011%	**7.0**
VG-flow	**98.917%**	**1.0**	9263.5	0.009%	**7.0**
Virus-VG	99.693%	1.072	9269.5	0.027%	7.5
SAVAGE	95.522%	1.189	3547.0	**0.000%**	45.5
2-strain HIV (8 samples)					
MFD	**99.721%**	**1.0**	8001.25	**0.000%**	**2.0**
VG-flow	99.693%	1.063	7966.6	0.001%	2.1
Virus-VG	99.716%	1.518	8020.7	0.257%	3.1
SAVAGE	91.562%	1.212	1644.1	**0.000%**	18.3
4-strain HIV (1 sample)					
MFD	**99.816%**	**1.0**	8079.0	0.003%	**4.0**
VG-flow	99.741%	1.25	7984.0	0.687%	5.0
Virus-VG	99.722%	**1.0**	7983.0	**0.000%**	4.0
SAVAGE	97.534%	1.218	1579.0	**0.000%**	33

Table 3: Average assembly performance on the simulated HCV and HIV mixtures datasets. Only statistics from samples where all methods were able to produce a solution within the 24-hour time limit were included. The best result for each metric and configuration is highlighted in bold, except for NG50 where the 'correctness' depends on the lengths of genomes in a sample.

4.3.2. MFD does estimates abundances better than VG-flow and Virus-VG on HIV data

In addition to assembly quality, we also compared the abundance estimation accuracy of the MFD approach against Virus-VG and VG-flow. SAVAGE was excluded from this comparison, because it does not provide abundance estimates. We evaluate abundance estimation accuracy by comparing the found haplotype abundances to the ground truth. For each found haplotype, we compute pairwise sequence alignments against the ground truth haplotypes using the Biopython pairwise aligner with default scoring parameters. Each found haplotype is assigned to the ground truth haplotype to which it aligns with the highest alignment score. Based on this assignment, we aggregate the estimated abundances of all found haplotypes assigned to each ground truth haplotype. These aggregated estimates are compared to the true abundances by calculating the L1 error, defined as:

$$\text{L1 error} = \sum_{i=1}^{k} |\hat{a}_i - a_i|. \tag{5}$$

Where $\hat{a}_i$ is the estimated abundance of the i-th haplotype, a_i is its true abundance and k is the number of ground truth haplotypes. This metric quantifies the total absolute difference between the estimated and true abundance across all haplotypes in a sample. The error has a range of [0, 2], where a lower error means that the estimated abundance resembles the ground truth abundance more closely.

Figure 6 shows the L1 error per sample for the different methods. For both MFD and VG-flow, the estimated abundances consistently align closely with the ground truth. Virus-VG also achieves good accuracy in most cases, but displays a few samples with noticeably higher errors. This pattern is further illustrated in Supplement I, Figure 6, which compares estimated and true abundances per haplotype. While all three methods generally follow the diagonal, indicating accurate estimates, Virus-VG exhibits occasional outliers that deviate substantially from the ground truth.

On the HCV dataset, the MFD approach does not achieve significantly better abundance estimates than VG-flow or Virus-VG. On the HIV dataset, MFD provides significantly more accurate abundance estimates (Supplement H.1, Table 11). Because the number of samples is limited, and most samples show very similar error values across methods and haplotype counts, it is difficult to assess whether increasing the number of haplotypes consistently leads to higher estimation errors. For this reason, we focused our statistical comparisons at the method level rather than stratifying by haplotype count.

Overall, these results indicate that the MFD approach can estimate haplotype abundances with accuracy comparable to VG-flow and Virus-VG on HCV data, and significantly better on HIV data. This pattern mirrors the assembly quality results reported in Section 4.3.1, where MFD also outperformed the other methods on the HIV dataset but was similar on the HCV dataset.

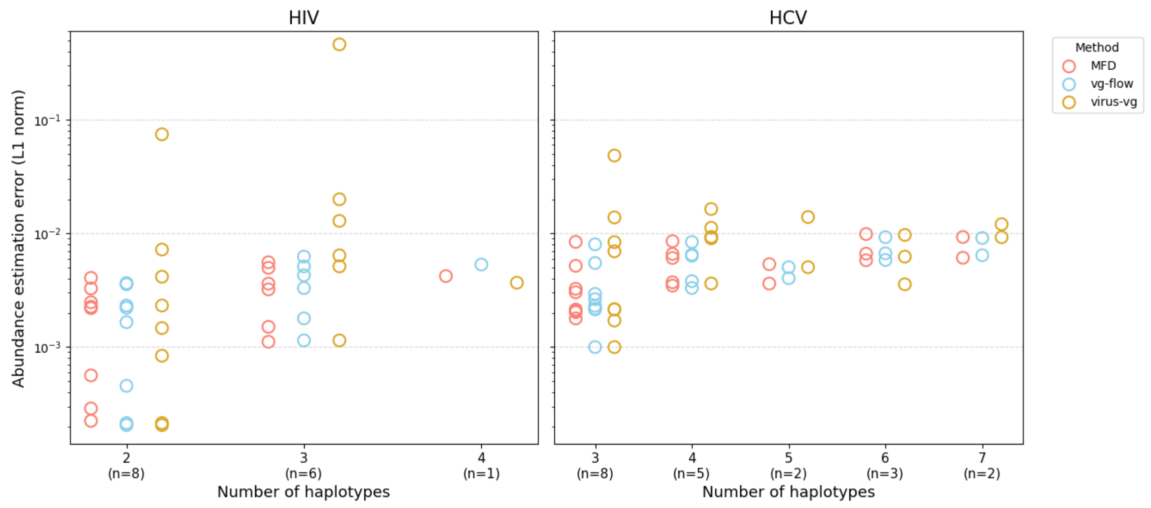


Figure 6: Absolute abundance estimation error (L1 error) of the different methods across different numbers of haplotypes per sample. Note that the y-axis is on a log scale.

4.4. Assembly results subset weights

Given that the MFD approach becomes computationally demanding for samples with many haplotypes, we next explore whether restricting the set of allowed weights can reduce runtime while maintaining assembly quality. By restricting weights to a predefined set, we limit the solution space by reducing the number of possible weight–path combinations that need to be considered. We restrict the weights in two different ways. First we give the MFD the ground truth weights appended with zero weights if the number of paths estimated with the MPC with subpath constraints was higher than the ground truth number of paths. This setup represents an idealized scenario, allowing us to test whether constraining the weights can yield high-quality solutions under perfect conditions. Second, we used K-means clustering on the weights as described in Methods 3.3.3), in order to assess whether K-means clustering provides a suitable strategy for identifying a candidate weight set.

We can see in Figure 7 that with restricting the weights, generally more samples were able to finish. By giving the perfect weights, all samples could finish within the allocated runtime, although the samples with more haplotypes took longer to finish than samples with fewer haplotypes (see Supplement K, Table 13 and 14). With the K-means clustering weights, generally more samples were able to finish than witout restricting the weights, but in some cases (5 haplotypes HCV and 3 haplotypes HIV) fewer samples could finish with the K-means clustering weights than without the weights. This shows the importance of well-chosen weights for this method to work, because with the perfect weights all samples were able to finish, while with the K-means weights sometimes samples did not finish that did finish with no weight restriction.

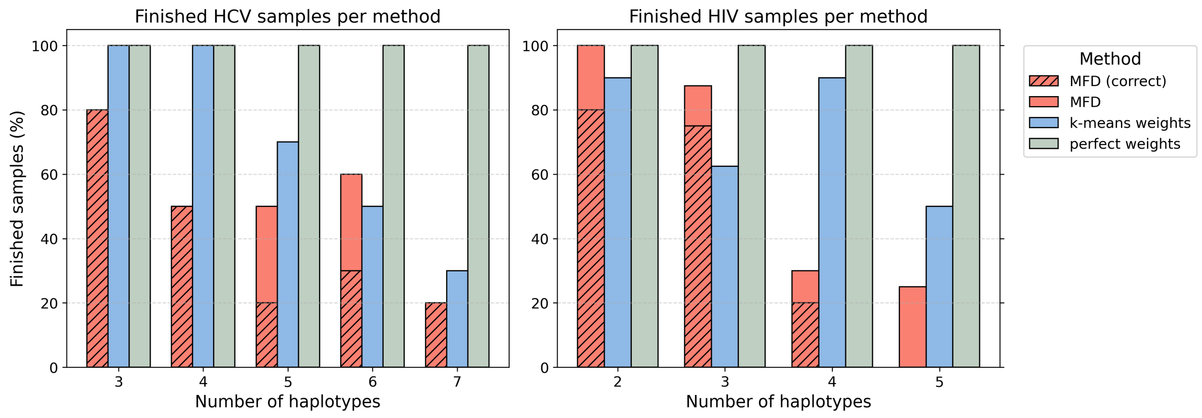


Figure 7: The percentage of finished samples per method. The MFD (correct) is the percentage of MFD samples (with no weight restrictions) that have the same number of estimated haplotypes with and without subpath constraints (i.e. those considered in 4.3).

Next, we evaluate the assembly performance of the weight-set restriction methods in comparison to VG-flow. We focus on VG-flow because, as shown in the previous section, it generally outperformed Virus-VG and successfully completed all samples, whereas Virus-VG and the standard MFD method failed to finish in some cases. We only compare the samples where K-means clustering was able to give a solution. The results are reported in Table 4 and the full table of results is in Supplement J, Table 12. Here we see that restricting the weights to the perfect weight set gives the best assembly results. It has the lowest error rate and highest genome fraction in the majority of the samples. Especially on the HIV dataset the MFD with perfect weights performs better. By using the K-means weights we see that the performance drops in most cases. Statistical tests to compare the perfect weight set to k-means clustering approach and VG-flow are provided in Supplement L, Table 15 and 16.

We note that with the K-means weight set approach, a zero weight is included. When the zero weight is applied, the corresponding haplotype is filtered out, which can result in fewer haplotypes being reconstructed than expected. This happens, for example, in some samples with seven HCV haplotypes. We hypothesize that the zero weight is used when the allowed weight set lacks sufficiently low values to capture the true abundances. This point is further discussed in the discussion section.

Overall, these results demonstrate that selecting appropriate weights provides clear benefits. In Supplement F, Table 8, we compare samples in which the number of haplotypes was not estimated correctly. Using an unrestricted MFD, fewer of these samples completed within the allocated runtime, and the resulting assemblies had poorer statistics. Applying correct weights improves these assembly metrics and increases the number of successfully completed samples. Taken together, careful weight selection not only mitigates runtime issues but also enhances overall assembly performance.

	genome fraction	duplication ratio	NG50	ER	# contigs
3-strain HCV (10 samples)					
perfect weights	**99.880%**	**1.0**	9292.2	**0.004%**	**3.0**
K-means weights	**99.880%**	**1.0**	9292.2	0.005%	**3.0**
VG-flow	99.376%	1.133	9282.5	0.006%	3.4
7-strain HCV (3 samples)					
perfect weights	**99.888%**	**1.0**	9289.3	0.042%	**7.0**
K-means weights	85.635%	**1.0**	9289.3	0.042%	6.3
VG-flow	99.095%	1.048	9270.3	**0.035%**	7.3
2-strain HIV (9 samples)					
perfect weights	**99.758%**	**1.0**	8005.7	**0.001%**	**2.0**
K-means weights	99.738%	1.056	8006.2	0.032%	2.1
VG-flow	99.736%	1.134	7968.4	0.020%	2.4
4-strain HIV (9 samples)					
perfect weights	**99.699%**	**1.0**	8007.6	**0.059%**	**4.0**
K-means weights	96.894%	1.037	7992.7	0.066%	**4.0**
VG-flow	99.417%	1.423	7980.4	0.349%	5.9

Table 4: Average assembly performance of the weight rectricted MFD methods and VG-flow on the simulated HCV and HIV mixtures datasets. Only the samples where the K-means weights MFD method produced a solution are considered. The best result for each metric and configuration is highlighted in bold, except for NG50 where the 'correctness' depends on the lengths of genomes in a sample.

5. Conclusion and discussion

The ability to reconstruct viral haplotypes *de novo* from sequencing data is essential for tracking viral evolution, detecting new clinically relevant variants, and guiding effective treatment strategies [13]. However, current methods often rely on heuristics that limit exploration of possible haplotype combinations, which risks the omission of relevant haplotypes, especially low-abundance ones. In this work, we have investigated the use of an Integer Linear Programming (ILP) formulation for Minimum Flow Decomposition (MFD) for haplotype-aware genome assembly. This approach considers all possible path combinations, constrained by subpaths derived from contigs. To enable the use of this approach, we also introduced a new pipeline for constructing a contig variation graph. We also employed a Minimum Path Cover (MPC) formulation to estimate the number of haplotypes, providing a reliable lower bound on the number of paths required for the MFD and avoiding unnecessary iterations of solving flow decompositions.

When the topology of the graph was correct, the MFD-based approach achieved assembly quality comparable to, and in some cases exceeding, that of VG-flow and Virus-VG. It also produces haplotype abundance estimates with errors as low as those from competing methods, and with consistently stable performance across datasets, which is an important property for clinical applications where reliability is critical.

However, these benefits come with notable limitations. The runtime of the MFD step grows exponentially with the number of haplotypes, making it impractical for complex mixtures. For samples with more than seven haplotypes, many samples could not be solved within a 24-hour limit. Furthermore, when contigs contain errors, the quality of the assembly declines sharply, and the ILP often fails to complete within a reasonable timeframe. To mitigate runtime issues, we explored restricting the set of allowed path weights in the MFD formulation. When chosen appropriately, this not only reduced solution time but also improved assembly results.

Overall, we demonstrate that an MFD-based ILP can be successfully integrated into a haplotype assembly pipeline, providing accurate reconstruction of haplotypes and their abundances in low-haplotype samples. While scalability remains limited for high-haplotype-count samples, the method shows promise for detecting previously missed haplotypes in simpler mixtures. Future work could explore hybrid approaches, combining heuristics for complex cases with exact MFD when feasible, to balance scalability with accuracy.

5.1. Limitations and improvements of the current contig variation graph construction

A method to construct a contig variation graph for haplotype reconstruction was first introduced in Virus-VG [4], and later used in VG-flow [2], but has since been deprecated. We therefore introduced a new pipeline to construct a contig variation graph from reads and the contigs assembled from these reads, which is a necessary first step before we can reconstruct the full-length haplotypes. Because the MFD approach works well when the contig variation graph is error-free, improving the contig variation graph would improve results for the MFD approach. Improving the contig variation graph is also beneficial for other methods that rely on this graph, such as Virus-VG and VG-flow.

We observe that one major source of graph errors is the presence of contigs that merge sequences from multiple haplotypes. Although we already reduce noise by filtering out low-abundance contigs (likely wrong contigs) and low-abundance nodes (likely sequencing errors in otherwise valid contigs), merged-haplotype contigs evade these filters. They typically have normal abundance, since reads from multiple haplotypes map to them, but introduce incorrect subpath constraints, which complicates the reconstruction with an MFD. We found that removing such erroneous contigs during graph construction, or discarding their associated subpath constraints, allows the MFD to rapidly converge to the correct solution. Thus, developing strategies to detect and remove such merged contigs could increase the number of error-free graphs and, consequently, the proportion of samples that can be successfully solved with the MFD method.

Another difficulty arises from repeated sequences, which can create cycles in the graph. Because MFD requires the graph to be acyclic, these cycles must be resolved before the decomposition. In our HIV dataset, four samples contained small cycles that we omitted from the analysis. These small cycles can be resolved by duplicating nodes, but longer repeats, such as the long terminal repeats (LTRs) present in HIV, can cause the graph to grow excessively. In our sample simulations, we avoided this issue by omitting LTRs altogether. However, in real datasets, repeats must be identified and removed before graph construction, for example by mapping reads or contigs to known repeat sequences and filtering them out, as proposed by Di Giallonardo et al. [9]. Incorporating strategies for repeat detection and cycle resolution would make the pipeline more robust and extend its applicability to a broader range of viral genomes.

Finally, a challenge arises from the structure of the variation graph that can miss information. Because the MFD step searches only for source-to-sink ($s - t$) paths, it cannot reconstruct haplotypes that start or end at different nodes. So with this approach, we are unable to correctly reconstruct haplotypes with insertions and deletions at the start and end of the sequence of a haplotype. Furthermore, the quality of the variation graph depends on the contigs generated by SAVAGE. If SAVAGE fails to assemble certain regions of a low-abundance strain, those regions are absent from the graph and cannot be recovered by the MFD step, regardless of the accuracy of the flow decomposition. Consequently, the final haplotype reconstruction will be incomplete, as missing regions in the variation graph cannot be recovered at any later stage of the pipeline.

5.2. Optimizing the set of path weights

We introduced a method to reduce the solution space of the MFD by restricting the set of allowed path weights. Our results show the critical importance of selecting these weights carefully: when the weight set consisted of the ground-truth weights, all samples completed within the allotted runtime and produced high-quality assembly statistics.

Using the K-means clustering approach, we generated sets of allowed weights, but found that this method did not accurately capture the true weight distribution. In particular, higher weights were overrepresented. This imbalance occurred because the clustering was applied to all weights, and many regions of the graph correspond to shared sequences across multiple haplotypes, and therefore accumulate into larger combined weights. To mitigate path overestimation, we also added a zero weight to the allowed set. Interestingly, this zero weight was sometimes selected even when the number of paths was correct, likely compensating for the shortage of low weights.

Across our experiments, K-means derived weight sets contained between 8 and 19 weights, but the relationship between the number and distribution of these weights and the feasibility of the MFD remains unclear. Although the clustering-based strategy enabled a few additional samples to complete that would otherwise have failed, its overall impact on performance was limited compared to cases where the allowed set closely reflected the true weight distribution.

Because we observed that a well-chosen weight set, such as the perfect weights, leads to solutions with good assembly statistics, we recommend that future research focus on developing methods that better represent low-abundance weights in the allowed set. One potential direction is to start from the Minimum Path Cover and identify edges traversed by only a single path, which would emphasize low-abundance weights. By exploring additional strategies for constructing representative allowed weight sets, it should be possible to improve both the quality of the assembly and the number of samples that can be processed successfully.

5.3. Improving the MFD

In this work, we have investigated various strategies to accelerate the MFD step to make it a feasible approach for our application. Despite these efforts, the current implementation of MFD is still limited to handling mixtures with substantially fewer haplotypes compared to VG-flow, which relies on heuristics and thus explores a reduced solution space. Nonetheless, there are still a few promising options to explore to accelerate the MFD step.

One potential improvement lies in optimizing the solver configuration. We used Gurobi with default parameters to solve the ILP formulation of the MFD. Some parameter adjustments were tested but did not lead to meaningful reductions in runtime (see Supplement **??** for details). However, many other solver settings remain unexplored and may help improve performance. For example, providing a warm start, an initial feasible solution, could guide the solver toward a solution more quickly. Using the paths obtained from the Minimum Path Cover (MPC) or VG-flow as a warm start is a promising strategy that could speed up convergence. Further experimentation with solver parameters and warm starts could enable MFD to handle more complex datasets.

We also experimented with the placement of weights in the graph. Initially, we modeled the weights on nodes instead of edges, which required a reformulation of the ILP (see Supplement N for the ILP formulation), but this slowed down the solver. To address this, we reverted to a representation where each node is split into two connected nodes with a weighted edge, preserving the original node weights. Interestingly, when weights were placed on nodes, adding symmetry-breaking constraints and objectives considerably accelerated the MFD. This effect was not observed when weights were on edges, suggesting that the solver may handle symmetry differently in that case. A better understanding of this phenomenon could help identify additional ways to reduce the solution space.

Another promising direction for improving assembly quality is to modify the ILP formulation itself. The Flowpaths package includes an alternative objective, the path error [10], which evaluates errors at the level of entire paths rather than individual edges. By aligning the optimization objective more closely with the reconstruction of full haplotypes, this formulation has the potential to improve the accuracy of assembled sequences. Dias et al. [10] already demonstrated that the path error objective outperforms previous error-handling formulations for RNA transcript assembly graphs.

In summary, improvements such as solver tuning with warm starts, alternative objective functions, and problem reformulations with symmetry-breaking constraints show promise for improving the MFD. Enhancing the efficiency of MFD will make it more scalable and competitive with heuristic methods like VG-flow and Virus-VG in terms of runtime.

5.4. Improving the subpath constraints

In our current implementation, subpath constraints are incorporated into the ILP as defined in Flowpaths, ensuring that every subpath derived from the contigs is used at least once in the final solution. While this approach integrates the contig information, it enforces the use of all contigs, including those containing errors. The presence of even a single erroneous contig can therefore prevent the ILP from converging or can degrade assembly quality.

To address this issue, we attempted to replace the Flowpaths subpath constraints with those used in VG-flow [2]. In VG-flow, instead of enforcing the use of every subpath, it only requires that each used edge belongs to a full-length subpath (see Supplement B.2 for a formal definition). This means that not all contigs have to be used in a solution. Preliminary results show that three additional samples of the HCV dataset completed successfully without changing the assembly statistics (Supplement B.2, Table 5), although we can not fully confirm the correctness of these results. This suggests that the choice of subpath constraints can improve the success rate of MFD-based assembly.

A further refinement would be to relax the subpath constraints by requiring coverage of only a subset of the subpaths, rather than all. The corresponding ILP formulation for this approach is provided in Supplement O. Such a formulation would allow the solver to exclude erroneous contigs while still enforcing sufficient coverage of the other contigs. By adopting less strict subpath constraints, we may be able to improve assembly quality, reduce runtime, and increase the number of samples for which the MFD solver produces a feasible solution.

5.5. Suitability and generalizability of MFD

An important question arising from our results is under what dataset conditions an MFD-based approach to haplotype reconstruction is most suitable. While our experiments demonstrate that MFD can produce high-quality assemblies under certain conditions, its broader applicability depends on dataset characteristics such as the number of haplotypes, their divergence, genome length, sequencing technology, and the quality of contigs used to build the variation graph.

In general, MFD can be advantageous over heuristic methods when only a small number of haplotypes are present, as in many co-infection cases [29, 33]. However, although various strategies can accelerate the MFD step, the problem remains NP-hard, making it unlikely to efficiently handle samples with very high haplotype counts, as encountered in some intra-host viral populations.

Our evaluation of the MFD approach on different viruses showed variable performance. On the HIV dataset, MFD achieved particularly low abundance error rates compared to other methods, but it scaled to fewer haplotypes than on the HCV dataset. This makes us wonder what the difference is between these datasets that explains this difference in performance. One key difference between these datasets is the pairwise divergence between haplotypes: in HCV, haplotypes differ by 6–9%, whereas in HIV they differ by only 1–6%. So a possible explanation could be that MFD performance may be influenced by divergence levels, with lower divergence resulting in better assemblies. Testing on additional viral datasets spanning a broader range of divergences could help clarify this relationship.

Beyond haplotype divergence, genome length is another important factor for generalizability. VG-flow was developed partly to address the limitation of earlier haplotype-aware reconstruction methods in handling genomes longer than 10 kbp, and it has been shown to be able to reconstruct haplotypes up to 200 kbp. In our perfect graph experiments shown in Results 4, the runtime of the MFD approach scaled well with the genome size, when the number of haplotypes was fixed, suggesting potential scalability to longer genomes in practice as well. Nevertheless, larger genomes increase the likelihood of erroneous contigs, which can severely degrade solver performance. If the impact of erroneous contigs could be mitigated through improved graph construction (Section 5.1) or refined subpath constraints (Section 5.4), then MFD could also be applied effectively to larger genomes.

The choice of sequencing technology and read length may also affect generalizability. All current experiments used 150 bp paired-end HiSeq reads. Longer reads could yield longer contigs, potentially improving graph construction, but may also introduce more sequencing errors. While some errors incorporated into contigs can be filtered from the graph, remaining erroneous contigs limit performance. So far, we have only tested the MFD approach on short-read sequencing data. However, as long-read technologies continue to improve in accuracy and cost, the method could be applied directly to long reads as well. This would make the graph construction step simpler, since we would no longer need to rely on contigs from SAVAGE.

Our current graph construction relies exclusively on contigs generated by SAVAGE, chosen for its ability to preserve all sequence variation in a *de novo* setting. However, SAVAGE requires ultra-deep sequencing coverage. And if SAVAGE fails to assemble certain regions, they will be absent from the variation graph and cannot be recovered by MFD in later steps. Alternative assemblers such as SPAdes [5], when used in careful mode, can also produce haplotype-aware contigs and have been successfully applied in VStrains [38] for full-length haplotype reconstruction. Exploring alternatives could broaden the applicability of the MFD approach to datasets where SAVAGE is impractical.

Our evaluation has so far been limited to simulated datasets. Testing the MFD approach on real sequencing samples with a known haplotype composition would be a critical next step to assess its robustness. Such experiments could reveal additional challenges not captured in simulations or confirm that the current settings are sufficient for practical applications.

Finally, we also observed that the relationship between the Maximum Path Cover (MPC) with and without subpath constraints provides a useful diagnostic for when MFD is likely to succeed. Specifically, if both MPC variants produce the same result, MFD tends to perform well. This criterion could therefore serve as a practical check before applying the method: in cases where the MPC's differ, heuristic approaches may be more reliable. At the same time, improving the design of weight sets and refining subpath constraints could help extend the range of cases where MFD remains effective.

In summary, MFD is best suited to samples with low haplotype counts, high sequencing coverage, and minimal contig assembly errors. The size of the genome appears to be less restrictive, but erroneous contigs remain a key challenge. A practical indicator of suitability is whether the Maximum Path Cover (MPC) with and without subpath constraints produces the same result: in such cases, MFD generally performs well, whereas differing MPCs may be better addressed with heuristic approaches. Future work on refining weight sets, improving subpath constraints, and testing on diverse real-world datasets will be essential to establish the full scope of MFD's applicability.

6. AI disclosure statement

For this article, artificial intelligence (ChatGPT and DeepL) was used to improve readability.

7. DAIC acknowledgment

Research reported in this work was partially or completely facilitated by computational resources and support of the Delft AI Cluster (DAIC) at TU Delft (RRID: SCR 025091), but remains the sole responsibility of the authors, not the DAIC team.

References

[1] Boluwatife A Adewale. "Will long-read sequencing technologies replace short-read sequencing technologies in the next 10 years?" In: *African journal of laboratory medicine* 9 (Nov. 2020). DOI: 10.4102/ajlm. URL: https://doi.org/10.4102/ajlm..

[2] Jasmijn A Baaijens, Leen Stougie, and Alexander Schönhuth. "Strain-aware assembly of genomes from mixed samples using flow variation graphs". In: *Research in Computational Molecular Biology* (Apr. 2020), pp. 221–222. DOI: 10.1101/645721. URL: https://doi.org/10.1101/645721.

[3] Jasmijn A. Baaijens et al. "De novo assembly of viral quasispecies using overlap graphs". In: *Genome Research* 27.5 (May 2017), pp. 835–848. ISSN: 15495469. DOI: 10.1101/gr.215038.116.

[4] Jasmijn A. Baaijens et al. "Full-length de novo viral quasispecies assembly through variation graph construction". In: *Bioinformatics* 35.24 (Dec. 2019), pp. 5086–5094. ISSN: 14602059. DOI: 10.1093/bioinformatics/btz443.

[5] Anton Bankevich et al. "SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing". In: *Journal of Computational Biology* 19.5 (May 2012), pp. 455–477. ISSN: 10665277. DOI: 10.1089/cmb.2012.0021.

[6] Nicolas L. Bray et al. "Near-optimal probabilistic RNA-seq quantification". In: *Nature Biotechnology* 34.5 (May 2016), pp. 525–527. ISSN: 15461696. DOI: 10.1038/nbt.3519.

[7] Philippe Colson et al. *From viral democratic genomes to viral wild bunch of quasispecies.* Nov. 2023. DOI: 10.1002/jmv.29209.

[8] Delft AI Cluster (DAIC). *The Delft AI Cluster (DAIC), RRID:SCR_025091.* 2024. DOI: 10.4233/rrid:scr_025091. URL: https://doc.daic.tudelft.nl/.

[9] Francesca Di Giallonardo et al. "Full-length haplotype reconstruction to infer the structure of heterogeneous virus populations". In: *Nucleic Acids Research* 42.14 (Aug. 2014). ISSN: 13624962. DOI: 10.1093/nar/gku537.

[10] Fernando H. C. Dias and Alexandru I. Tomescu. *Accurate Flow Decomposition via Robust Integer Linear Programming.* Mar. 2023. DOI: 10.1101/2023.03.20.533019. URL: http://biorxiv.org/lookup/doi/10.1101/2023.03.20.533019.

[11] Fernando H. C. Dias et al. "Fast, Flexible, and Exact Minimum Flow Decompositions via ILP". In: *Research in Computational Molecular Biology: 26th Annual International Conference, RECOMB 2022, San Diego, CA, USA, May 22–25, 2022, Proceedings.* Berlin, Heidelberg: Springer-Verlag, Jan. 2022, pp. 230–245. DOI: 10.1007/978-3-031-04749-7{_}14. URL: http://arxiv.org/abs/2201.10923%20http://dx.doi.org/10.1007/978-3-031-04749-7_14.

[12] Fernando H.C. Dias et al. "Efficient Minimum Flow Decomposition via Integer Linear Programming". In: *Journal of Computational Biology*. Vol. 29. 11. 2022. DOI: `10.1089/cmb.2022.0257`.

[13] Esteban Domingo, Julie Sheldon, and Celia Perales. "Viral Quasispecies Evolution". In: *Microbiology and Molecular Biology Reviews* 76.2 (June 2012), pp. 159–216. ISSN: 1092-2172. DOI: `10.1128/mmbr.05023-11`.

[14] Anton Eliseev et al. "Evaluation of haplotype callers for next-generation sequencing of viruses". In: *Infection, Genetics and Evolution* 82 (Aug. 2020). ISSN: 15677257. DOI: `10.1016/j.meegid.2020.104277`.

[15] Nicholas Eriksson et al. "Viral population estimation using pyrosequencing". In: *PLoS Computational Biology* 4.5 (2008). ISSN: 15537358. DOI: `10.1371/journal.pcbi.1000074`.

[16] Borja Freire et al. "ViQUF: De Novo Viral Quasispecies Reconstruction Using Unitig-Based Flow Networks". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 20.2 (Mar. 2023), pp. 1550–1562. ISSN: 15579964. DOI: `10.1109/TCBB.2022.3190282`.

[17] Adrian Fritz et al. "Haploflow: strain-resolved de novo assembly of viral genomes". In: *Genome Biology* 22.1 (Dec. 2021). ISSN: 1474760X. DOI: `10.1186/s13059-021-02426-8`.

[18] Erik Garrison and Andrea Guarracino. "Unbiased pangenome graphs". In: *Bioinformatics* 39.1 (Jan. 2023). ISSN: 13674811. DOI: `10.1093/bioinformatics/btac743`.

[19] Erik Garrison et al. "Variation graph toolkit improves read mapping by representing genetic variation in the reference". In: *Nature Biotechnology* 36.9 (Oct. 2018), pp. 875–881. ISSN: 15461696. DOI: `10.1038/nbt.4227`.

[20] Andreas Grigorjew et al. "Accelerating ILP Solvers for Minimum Flow Decompositions Through Search Space and Dimensionality Reductions". In: *Leibniz International Proceedings in Informatics, LIPIcs*. Vol. 301. Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, July 2024. ISBN: 9783959773256. DOI: `10.4230/LIPIcs.SEA.2024.14`.

[21] Alexey Gurevich et al. "QUAST: Quality assessment tool for genome assemblies". In: *Bioinformatics* 29.8 (Apr. 2013), pp. 1072–1075. ISSN: 13674803. DOI: `10.1093/bioinformatics/btt086`.

[22] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual.* 2025.

[23] William T. Harvey et al. "SARS-CoV-2 variants, spike mutations and immune escape". In: *Nature Reviews Microbiology* 19.7 (July 2021), pp. 409–424. ISSN: 17401534. DOI: `10.1038/s41579-021-00573-0`.

[24] Joris Hemelaar. *Implications of HIV diversity for the HIV-1 pandemic.* May 2013. DOI: `10.1016/j.jinf.2012.10.026`.

[25] Lewis Z. Hong et al. "BAsE-Seq: a method for obtaining long viral haplotypes from short sequence reads". In: *Genome biology* 15.11 (2014), p. 517. ISSN: 1474760X. DOI: `10.1186/s13059-014-0517-9`.

[26] Weichun Huang et al. "ART: A next-generation sequencing read simulator". In: *Bioinformatics* 28.4 (Feb. 2012), pp. 593–594. ISSN: 13674803. DOI: `10.1093/bioinformatics/btr708`.

[27] Q. Huangfu and J. A.J. Hall. "Parallelizing the dual revised simplex method". In: *Mathematical Programming Computation* 10.1 (Mar. 2018), pp. 119–142. ISSN: 18672957. DOI: `10.1007/s12532-017-0130-5`.

[28] Illumina. *Advantages of paired-end and single-read sequencing.* URL: https://emea.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html.

[29] Cassandra B. Jabara et al. "Hepatitis C Virus (HCV) NS3 sequence diversity and antiviral resistance-associated variant frequency in HCV/HIV coinfection". In: *Antimicrobial Agents and Chemotherapy* 58.10 (Oct. 2014), pp. 6079–6092. ISSN: 10986596. DOI: 10.1128/AAC.03466-14.

[30] Sergey Knyazev et al. "Accurate assembly of minority viral haplotypes from next-generation sequencing through efficient noise reduction". In: *Nucleic Acids Research* 49.17 (Sept. 2021), E102–E102. ISSN: 13624962. DOI: 10.1093/nar/gkab576.

[31] Heng Li. "Minimap2: Pairwise alignment for nucleotide sequences". In: *Bioinformatics* 34.18 (Sept. 2018), pp. 3094–3100. ISSN: 14602059. DOI: 10.1093/bioinformatics/bty191.

[32] Jane P Messina et al. "Global Distribution and Prevalence of Hepatitis C Virus Genotypes". In: (2014). DOI: 10.1002/hep.27259/suppinfo. URL: www.esri..

[33] Jose Arturo Molina-Mora et al. "Metagenomic pipeline for identifying co-infections among distinct SARS-CoV-2 variants of concern: study cases from Alpha to Omicron". In: *Scientific Reports* 12.1 (Dec. 2022). ISSN: 20452322. DOI: 10.1038/s41598-022-13113-4.

[34] Josephine B. Oehler et al. *The application of long-read sequencing in clinical settings.* Dec. 2023. DOI: 10.1186/s40246-023-00522-3.

[35] Benedict Paten et al. *Genome graphs and the evolution of genome inference.* May 2017. DOI: 10.1101/gr.214155.116.

[36] Sandhya Prabhakaran et al. "HIV haplotype inference using a propagating dirichlet process mixture model". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 11.1 (2014), pp. 182–191. ISSN: 15455963. DOI: 10.1109/TCBB.2013.145.

[37] Michael G. Ross et al. "Characterizing and measuring bias in sequence data". In: *Genome Biology* 14.5 (May 2013). ISSN: 1474760X. DOI: 10.1186/gb-2013-14-5-r51.

[38] Luo Runpeng and Yu Lin. "VStrains: De Novo Reconstruction of Viral Strains via Iterative Path Extraction from Assembly Graphs". In: *Research in Computational Molecular Biology.* Lecture Notes in Computer Science 13976.13976 (2023). Ed. by Haixu Tang, pp. 3–20. DOI: 10.1007/978-3-031-29119-7. URL: https://link.springer.com/10.1007/978-3-031-29119-7.

[39] Rafael Sanjuán and Pilar Domingo-Calap. "Mechanisms of viral mutation". In: *Cellular and Molecular Life Sciences* 73.23 (July 2016), pp. 4433–4448. ISSN: 14209071. DOI: 10.1007/s00018-016-2299-6.

[40] Francisco Sena and Alexandru I. Tomescu. "Safe Paths and Sequences for Scalable ILPs in RNA Transcript Assembly Problems". In: (Nov. 2024). URL: http://arxiv.org/abs/2411.03871.

[41] Paul M. Sharp and Beatrice H. Hahn. *The evolution of HIV-1 and the origin of AIDS.* Aug. 2010. DOI: 10.1098/rstb.2010.0031.

[42] Birgitte B. Simen et al. "Low-Abundance Drug-Resistant Viral Variants in Chronically HIV-Infected, Antiretroviral Treatment–Naive Patients Significantly Impact Treatment Outcomes". In: *Journal of Infectious Diseases* 199.5 (Mar. 2009), pp. 693–701. ISSN: 0022-1899. DOI: 10.1086/596736.

[43] Alexandru I. Tomescu et al. “Explaining a Weighted DAG with Few Paths for Solving Genome-Guided Multi-Assembly”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics.* Vol. 12. 6. Institute of Electrical and Electronics Engineers Inc., Nov. 2015, pp. 1345–1354. DOI: 10.1109/TCBB.2015.2418753.

[44] Cole Trapnell et al. “Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation”. In: *Nature Biotechnology* 28.5 (May 2010), pp. 511–515. ISSN: 10870156. DOI: 10.1038/nbt.1621.

[45] Helsinki University. *flowpaths.* Accessed: 2025-06-23. 2025. URL: https://algbio.github.io/flowpaths/.

[46] Carlos Valiente-Mullor et al. “One is not enough: On the effects of reference genome for the mapping and subsequent analyses of short-reads”. In: *PLoS Computational Biology* 17.1 (Jan. 2021). ISSN: 15537358. DOI: 10.1371/JOURNAL.PCBI.1008678.

[47] B. Vatinlen et al. “Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths”. In: *European Journal of Operational Research* 185.3 (Mar. 2008), pp. 1390–1401. ISSN: 03772217. DOI: 10.1016/j.ejor.2006.05.043.

[48] Lucia Williams, Alexandru I. Tomescu, and Brendan Mumey. “Flow Decomposition With Subpath Constraints”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 20.1 (Jan. 2023), pp. 360–370. ISSN: 15579964. DOI: 10.1109/TCBB.2022.3147697.

[49] Hyung-June Woo and Jaques Reifman. “A quantitative quasispecies theory-based model of virus escape mutation under immune selection”. In: *PNAS* 109.32 (Aug. 2012), pp. 12980–12985. DOI: 10.1073/pnas.1117201109/-/DCSupplemental. URL: www.pnas.org/cgi/doi/10.1073/pnas.1117201109.

8. Appendix

A. ILP minimum path cover

$$\sum_{(s,v)\in E} x_{svi} = 1, \qquad \forall i \in \{1,...,k\}, \tag{6a}$$

$$\sum_{(u,t)\in E} x_{uti} = 1, \qquad \forall i \in \{1,...,k\}, \tag{6b}$$

$$\sum_{(u,v)\in E} x_{uvi} - \sum_{(v,w)\in E} x_{vwi} = 0, \qquad \forall i \in \{1,...,k\}, \tag{6c}$$

$$\sum_{(s,v)\in E} x_{uvi} \geq 1, \qquad \forall i \in \{1,...,k\}, \tag{6d}$$

$$\sum_{(u,v)\in R_j} x_{uvi} \geq |R_j| * r_{ij}, \qquad \forall i \in \{1,...,k\}, \forall R_j \in R, \tag{6e}$$

$$\sum_{i\in\{1,...,k\}} r_{ij} \geq 1, \qquad \forall R_j \in R \tag{6f}$$

$$\pi_{uvi} \leq M x_{uvi}, \qquad \forall (u,v) \in E, \forall i \in \{1,...,k\}, \tag{6g}$$

$$\pi_{uvi} \leq w_i, \qquad \forall (u,v) \in E, \forall i \in \{1,...,k\}, \tag{6h}$$

$$\pi_{uvi} \geq w_i - (1 - x_{uvi})M, \qquad \forall (u,v) \in E, \forall i \in \{1,...,k\}, \tag{6i}$$

$$w_i \in \mathbb{Z}^+, \qquad \forall i \in \{1,...,k\}, \tag{6j}$$

$$x_{uvi} \in \{0,1\}, \qquad \forall (u,v) \in E, \forall i \in \{1,...,k\}, \tag{6k}$$

$$\pi_{uvi} \in \mathbb{R}^+ \cup \{0\}, \qquad \forall (u,v) \in E, \forall i \in \{1,...,k\}. \tag{6l}$$

$$r_{ij} \in \{0,1\}, \qquad \forall i \in \{1,...,k\}, \forall j \in \{1,...,|R|\} \tag{6m}$$

B. VG-flow subpath constraints

B.1. Definition

We mimicked the subpath constraints that are enforced in VG-flow [2]. Here, every edge that is used needs to be part of a subpath (contig). VG-flow uses an additional variation graph to restrict reconstructed paths to follow the contig-derived paths. In our formulation, this restriction can be enforced directly within the ILP by introducing the following constraints:

$$\sum_{\substack{j\in\{1,...,|R|\}\\(u,v)\in R_j}} r_{ij} \geq x_{uvi}, \qquad \forall i \in \{1,...,k\}, \forall (u,v) \in E \setminus \{s,t\}, \tag{7a}$$

$$\sum_{(u,v)\in R_j} x_{uvi} \geq |R_j| * r_{ij}, \qquad \forall i \in \{1,...,k\}, \forall R_j \in R, \tag{7b}$$

$$r_{ij} \in \{0,1\}, \qquad \forall i \in \{1,...,k\}, \forall j \in \{1,...,|R|\}. \tag{7c}$$

Given a set of subpaths R, for each subpath $R_j \in R$, we introduce additional binary variables r_{ij} denoting the presence of the subpath R_j in the ith path. $r_{ij} = 1$ if and only if each edge (u,v) in R_j is covered by path i. $|R_j|$ denotes the length (i.e., number of edges) of subpath constraint R_j. With the constraint 7a we enforce that if a path i uses an edge (u,v), this path should be part of at least one subpath r_{ij}. With constraint 7b we enforce the used subpaths to be full length. With the last constraint 7c we define r_{ij} as a binary variable.

B.2. Results

Three more samples could finish within the 24-hour allocated runtime with the VG-flow subpath constraints compared to the Flowpaths subpath constraints. The samples that finished with both subpath constraint strategies resulted in the same assemblies, as can be seen in Table 5.

	genome fraction	duplication ratio	NG50	ER	# contigs
5-strain HCV (3 samples)					
normal con.	99.891%	1.200	9292.3	0.682%	6.0
vg-flow con.	99.891%	1.200	9293.3	0.682%	6.0
6-strain HCV (3 samples)					
normal con.	99.870%	1.220	9291.7	0.909%	7.3
vg-flow con.	99.870%	1.220	9291.7	0.909%	7.3

Table 5: Average assembly performance on the simulated HCV mixture datasets from the samples where MFD approach overestimated the number of haplotypes with normal constraints and with the vg-flow constraints.

C. Assembly results without removing low abundance nodes

	genome fraction	duplication ratio	NG50	ER	# contigs
MFD without removing low abundance nodes					
3 strains	**99.851%**	**1.0**	**9296.5**	0.005%	**3**
4 strains	**99.781%**	**1.0**	**9298.7**	0.004%	**4**
5 strains	**99.818%**	**1.0**	**9298.0**	0.019%	**5**
6 strains	-	-	-	-	-
7 strains	**99.791%**	**1.0**	**9276**	0.015%	**7**
VG-flow					
3 strains	99.424%	1.082	9282.0	0.008%	3.25
4 strains	99.489%	1.082	9265.3	0.012%	4.3
5 strains	99.587%	1.1	9279.0	0.012%	5.5
6 strains	-	-	-	-	-
7 strains	98.793%	1.144	9248.0	0.023%	8.0

Table 6: Average assembly performance on the simulated HCV mixture datasets without removing the low abundance nodes.

D. Extended results runtime analysis

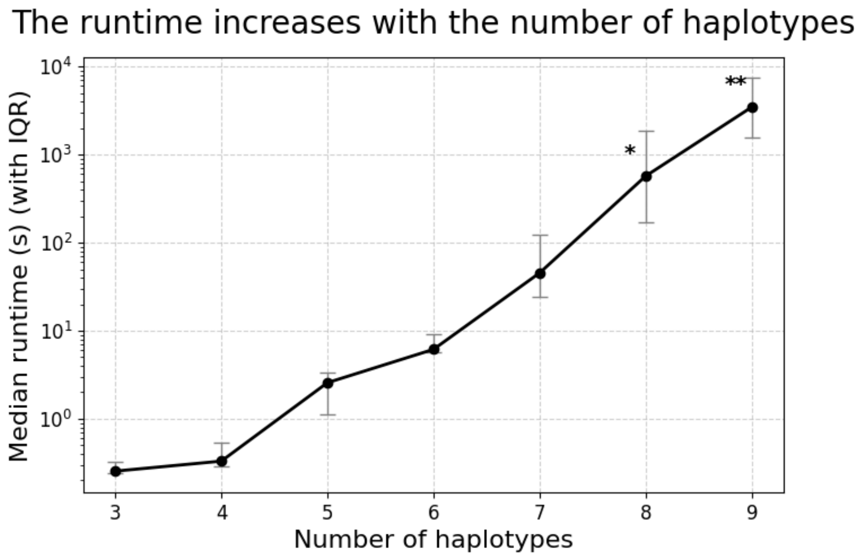


Figure 8: Runtime (in seconds) required to compute an exact MFD in basic simulated graphs. Note that the y-axis is on a log scale. *: one instance with eight haplotypes did not finish within 24 hours **: three instances with nine haplotypes did not finish within 24 hours.

E. Extended assembly results instances correct number haplotypes

	genome fraction	duplication ratio	NG50	ER	# contigs
3-strain HCV (8 samples)					
MFD	99.863%	**1.0**	9290.5	0.005%	**3.0**
VG-flow	99.452%	1.166	9280.1	0.008%	3.5
Virus-VG	99.787%	1.501	9280.3	0.209%	4.5
SAVAGE	**99.912%**	1.236	2213.6	**0.000%**	29.9
4-strain HCV (5 samples)					
MFD	**99.837%**	**1.000**	9293.2	0.003%	**4.0**
VG-flow	99.522%	1.15	9277.6	0.003%	4.6
Virus-VG	99.735%	1.302	9313.0	0.052%	5.2
SAVAGE	93.391%	1.239	1645.4	**0.001%**	37.8
5-strain HCV (2 samples)					
MFD	**99.894%**	**1.000**	9298.0	0.017%	**5.0**
VG-flow	99.586%	**1.000**	9279.0	0.010%	**5.0**
Virus-VG	99.774%	1.200	9286.5	0.033%	6.0
SAVAGE	95.289%	1.260	5210.0	**0.000%**	45.0
6-strain HCV (3 samples)					
MFD	**99.891%**	**1.000**	9289.1	0.014%	**6.0**
VG-flow	98.924%	**1.000**	9183.0	0.004%	**6.0**
Virus-VG	99.709%	1.058	9276.7	0.024%	6.3
SAVAGE	96.338%	1.201	3421.0	**0.001%**	46.3
7-strain HCV (2 samples)					
MFD	99.867%	**1.000**	9287.0	0.011%	**7.0**
VG-flow	**98.917%**	**1.000**	9263.5	0.009%	**7.0**
Virus-VG	99.693%	1.072	9269.5	0.027%	7.5
SAVAGE	95.522%	1.189	3547.0	**0.000%**	45.5
2-strain HIV (8 samples)					
MFD	**99.721%**	**1.000**	8001.3	**0.000%**	2.0
VG-flow	99.693%	1.063	7966.6	0.001%	2.1
Virus-VG	99.716%	1.518	8020.7	0.257%	3.1
SAVAGE	91.562%	1.212	1644.1	**0.000%**	18.3
3-strain HIV (6 samples)					
MFD	99.739%	**1.000**	7982.2	**0.000%**	**3.0**
VG-flow	99.722%	1.110	7983.2	0.076%	3.3
Virus-VG	**99.775%**	2.73	8060.8	0.809%	8.167
SAVAGE	88.5%	1.238	1115.2	0.001%	28.5
4-strain HIV (1 sample)					
MFD	**99.816%**	**1.000**	8079.0	0.003%	**4.0**
VG-flow	99.741%	1.250	7984.0	0.687%	5.0
Virus-VG	99.722%	**1.000**	7983.0	**0.000%**	4.0
SAVAGE	97.534%	1.218	1579.0	**0.000%**	33.0

Table 7: Average assembly performance on the simulated HCV and HIV mixtures datasets. The results are an average of the instances that produced the same MPC with and without subpath constraints.

F. Assembly results instances incorrect number haplotypes

	genome fraction	duplication ratio	NG50	ER	# contigs
5-strain HCV (3 samples)					
MFD	**99.891%**	1.200	9293.3	0.682%	6.0
prefect weights	**99.891%**	**1.000**	9293.3	0.012%	5.0
VG-flow	99.478%	1.067	9285.7	0.012 %	**5.3**
Virus-VG	99.820%	1.401	9285.7	0.149%	7.0
SAVAGE	95.118%	1.251	1932.3	**0.007%**	48.0
6-strain HCV (3 samples)					
MFD	**99.870%**	1.220	9291.7	0.909%	7.3
prefect weights	99.864%	**1.000**	9285.7	0.105%	6.0
VG-flow	99.356%	1.166	9234.7	**0.012%**	**7.0**
Virus-VG	99.729%	1.502	9286.3	0.358%	9.0
SAVAGE	94.298%	1.271	1927.3	**0.012%**	54.3
2-strain HIV (2 samples)					
MFD	**99.822%**	1.498	7991.0	0.728%	3.0
prefect weights	99.806%	**1.000**	7981	**0.000%**	**2.0**
VG-flow	99.818	1.602	7987.5	0.090%	4.0
Virus-VG	**99.822%**	2.508	8083.5	0.690%	5.0
SAVAGE	88.640%	**1.265**	690.0	**0.000%**	24.5
3-strain HIV (1 sample)					
MFD	99.608%	1.332	7986.0	0.779%	4.0
prefect weights	99.609%	**1.000**	7986.0	0.05%	**3.0**
VG-flow	**99.858%**	2.333	7992.0	0.553%	7.0
Virus-VG	99.854%	3.333	8005.0	1.043%	10.0
SAVAGE	82.029%	1.209	658.0	**0.000%**	30
5-strain HIV (2 samples)					
MFD	**99.682%**	1.199	7982.0	0.572%	6.0
prefect weights	99.596%	**1.000**	7976.5	0.044%	**5.0**
VG-flow	99.506%	1.200	7971.5	0.125%	6.0
Virus-VG	99.502%	3.304	8068.0	1.153%	16.5
SAVAGE	90.029%	1.282	1256.5	**0.006%**	49.0

Table 8: Average assembly performance on the simulated HCV and HIV mixtures datasets on the samples where the MFD approach overestimated the number of haplotypes. Note that we do not show results of the 4-strain HIV samples because for these samples Virus-VG did not give a solution in the allocated runtime.

G. Extended results assembly quality

G.1. Genome fraction

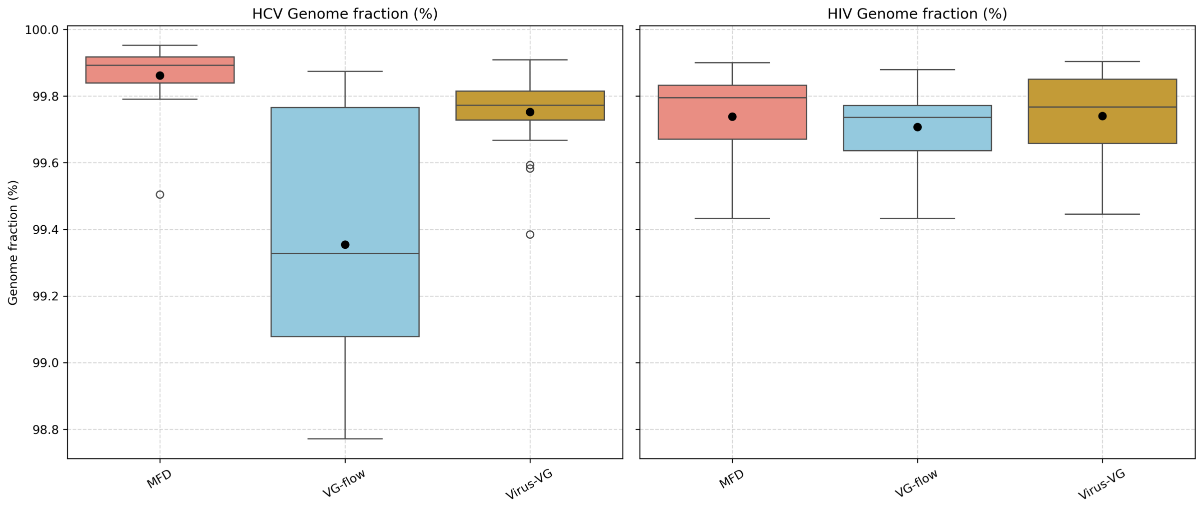


Figure 9: Boxplot of the genome fractions (%) obtained by the different methods. SAVAGE's genome fraction is always lower than the other methods, so is not shown in this figure be better able to see the range of values of the other methods.

Dataset	Method 1	Method 2	p-value
HCV	MFD	VG-flow	0.0001
HCV	MFD	Virus-VG	0.0001
HCV	MFD	SAVAGE	0.0000
HIV	MFD	VG-flow	0.0014
HIV	MFD	Virus-VG	0.6052
HIV	MFD	SAVAGE	0.0002

Table 9: Wilcoxon signed-rank test on the genome fraction of the different methods.

G.2. Error rate

Dataset	Method 1	Method 2	p-value
HCV	MFD	VG-flow	0.8570
HCV	MFD	Virus-VG	0.0007
HCV	MFD	SAVAGE	0.9998
HIV	MFD	VG-flow	0.0216
HIV	MFD	Virus-VG	0.0014
HIV	MFD	SAVAGE	0.5000

Table 10: Wilcoxon signed-rank test on the error rate of the different methods.

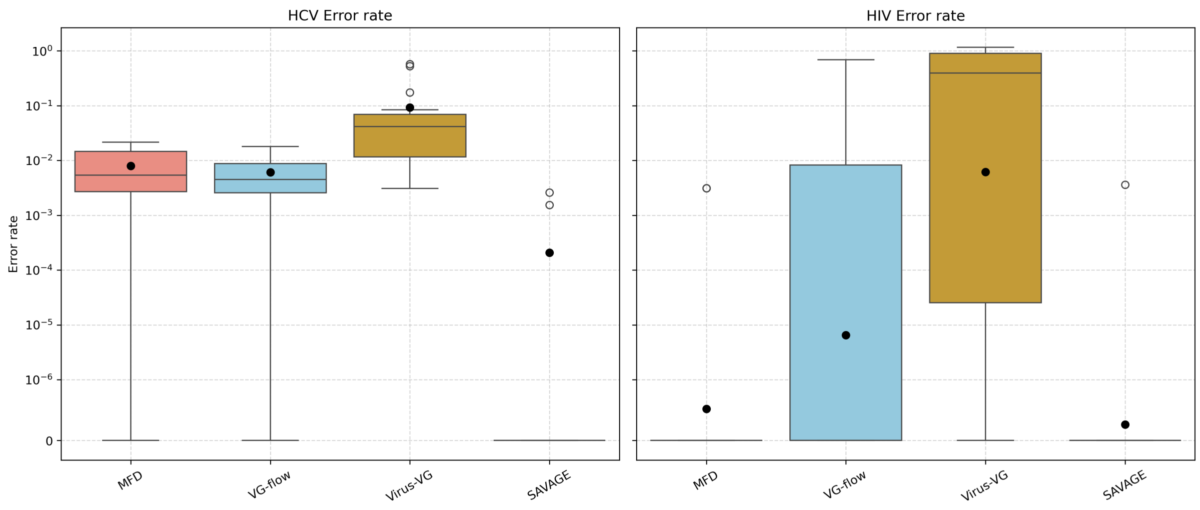


Figure 10: Error rates of the different methods. Full points are the mean. Note the y-axis is on a log-scale to accommodate the wide ranges.

H. Extended results abundance estimation

H.1. Hypothesis tests

We used the Wilcoxon signed-rank test to test if there is a significant difference in the L1 errors of the abundance estimates of MFD, VG-flow and Virus-VG. The resulst can be found in Table 11.

Dataset	Method 1	Method 2	n samples	p-value
HCV	MFD	VG-flow	20	0.773812
HCV	MFD	Virus-VG	20	0.066363
HIV	MFD	VG-flow	15	0.126190
HIV	MFD	Virus-VG	15	0.020630

Table 11: One sided Wilcoxon signed-rank test on the abundance estimates of the different methods. We test if MFD has a lower abudancen estimation error rate than the other methods.

I. Abundance estimates vs ground truth

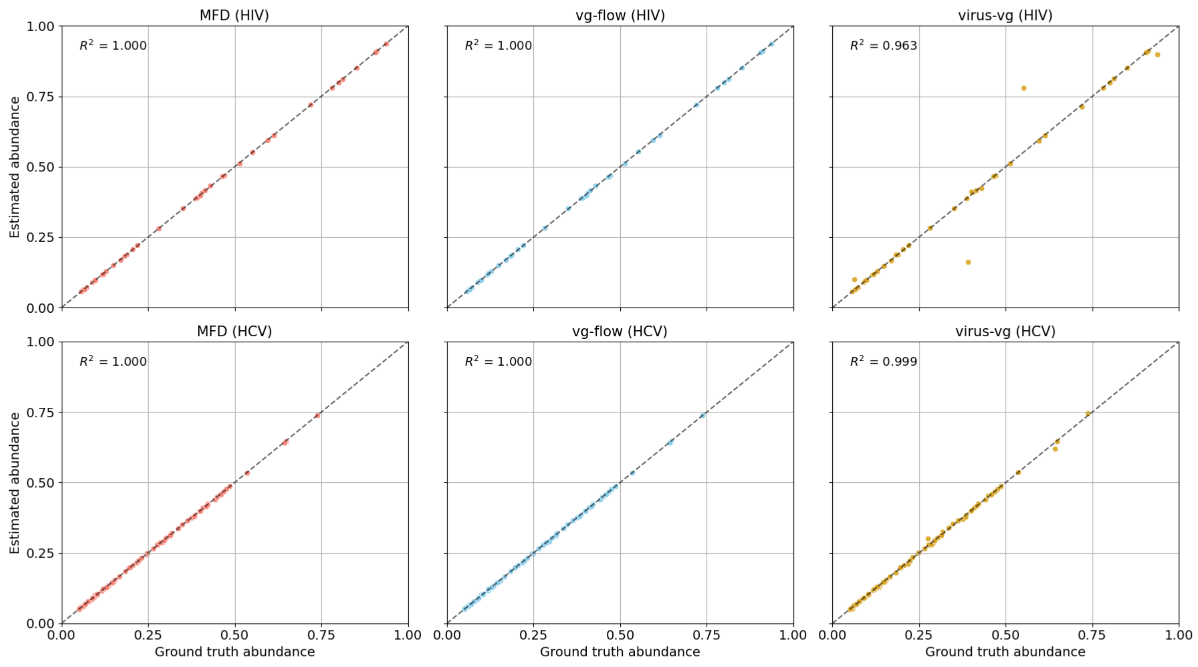


Figure 11: Abundance estimations versus the ground truth of MFD, VG-flow and Virus-VG.

J. Extended results restricted weight set

	genome fraction	duplication ratio	NG50	ER	# contigs
3-strain HCV (10 samples)					
perfect weights	**99.880%**	1.0	9292.2	**0.004%**	**3.0**
K-means weights	**99.880%**	1.0	9292.2	0.005%	**3.0**
VG-flow	99.376%	1.133	9282.5	0.006%	3.4
4-strain HCV (10 samples)					
perfect weights	**99.885%**	**1.0**	9292.1	0.219%	**4.0**
K-means weights	**99.885%**	1.05	9292.1	0.222%	4.2
VG-flow	99.541%	1.1	9282.5	**0.015%**	4.4
5-strain HCV (7 samples)					
perfect weights	**99.859%**	**1.0**	9292.7	**0.013%**	5.0
K-means weights	94.161%	1.036	9292.7	0.415%	4.9
VG-flow	99.286%	1.086	9266.0	0.024%	5.4
6-strain HCV (5 samples)					
perfect weights	**99.874%**	**1.0**	9287.4	0.020%	**6.0**
K-means weights	83.213%	**1.0**	9288.8	0.036%	5.4
VG-flow	99.207%	**1.0**	9219.4	**0.006%**	**6.0**
7-strain HCV (3 samples)					
perfect weights	**99.888%**	**1.0**	9289.3	0.042%	**7.0**
K-means weights	85.635%	**1.0**	9289.3	0.042%	6.3
VG-flow	99.095%	1.048	9270.3	**0.035%**	7.3
2-strain HIV (9 samples)					
perfect weights	**99.758%**	**1.0**	8005.7	**0.001%**	**2.0**
K-means weights	99.738%	1.056	8006.2	0.032%	2.1
VG-flow	99.736%	1.134	7968.4	0.020%	2.4
3-strain HIV (6 samples)					
perfect weights	**99.678%**	**1.0**	7982.3	**0.018%**	**3.0**
K-means weights	99.647%	**1.0**	9782.0	0.022%	**3.0**
VG-flow	98.335%	1.514	7989.5	0.220%	4.5
4-strain HIV (9 samples)					
perfect weights	**99.699%**	**1.0**	8007.6	**0.059%**	**4.0**
K-means weights	96.894%	1.037	7992.7	0.066%	**4.0**
VG-flow	99.417%	1.423	7980.4	0.349%	5.9
5-strain HIV (4 samples)					
perfect weights	**99.609%**	**1.0**	8005.5	**0.046%**	**5.0**
K-means weights	89.717%	1.124	7981.3	0.311%	**5.0**
VG-flow	99.593%	1.242	7975.0	0.213%	6.25

Table 12: Average assembly performance on the simulated HCV and HIV mixtures datasets.

K. Runtime samples perfect weights

number of haplotypes	3	4	5	6	7
average runtime (s)	2.9057	14.1576	664.3897	664.3897	12886.15

Table 13: Average runtime of the MFD step with perfect weights on samples with different haplotype counts on the HCV dataset

number of haplotypes	2	3	4	5
average runtime (s)	0.8752	2.6237	10.5043	120.7274

Table 14: Average runtime of the MFD step with perfect weights on samples with different haplotype counts on the HIV dataset

L. Statistical testing restricted weight set

Dataset	**Method 1**	**Method 2**	**p-value**
HCV	MFD perfect	VG-flow	1.826e-7
HCV	MFD perfect	MFD k-means	0.002531
HIV	MFD perfect	VG-flow	0.048750
HIV	MFD perfect	MFD k-means	0.001673

Table 15: Wilcoxon signed-rank test on the genome fraction of the different methods.

Dataset	**Method 1**	**Method 2**	**p-value**
HCV	MFD perfect	VG-flow	0.812800
HCV	MFD perfect	MFD k-means	0.015430
HIV	MFD perfect	VG-flow	0.004017
HIV	MFD perfect	MFD k-means	0.043430

Table 16: Wilcoxon signed-rank test on the error rate of the different methods.

M. Gurobi settings tried

We experimented with several Gurobi parameter configurations to improve convergence speed. However, none of these adjustments resulted in a significant reduction in runtime:

1. `model.setParam("Presolve", 2)`
2. `model.setParam("Heuristics", 0.5)`
3. `model.setParam("Cuts", 2)`
4. Tolerance set to 10^{-6}

N. ILP weights on nodes

$$\sum_{(s,v)\in E} x_{svi} = 1, \qquad \forall i \in \{1, ..., k\}, \tag{8a}$$

$$\sum_{(u,t)\in E} x_{uti} = 1, \qquad \forall i \in \{1, ..., k\}, \tag{8b}$$

$$\sum_{(u,v)\in E} x_{uvi} - \sum_{(v,w)\in E} x_{vwi} = 0, \qquad \forall i \in \{1, ..., k\}, \tag{8c}$$

$$f_v = \sum_{(u,v)\in E} \sum_{i\in\{1,...,k\}} \pi_{uvi}, \qquad \forall v \in V, \tag{8d}$$

$$\pi_{uvi} \leq M x_{uvi}, \qquad \forall (u,v) \in E, \forall i \in \{1, ..., k\}, \tag{8e}$$

$$\pi_{uvi} \leq w_i, \qquad \forall (u,v) \in E, \forall i \in \{1, ..., k\}, \tag{8f}$$

$$\pi_{uvi} \geq w_i - (1 - x_{uvi})M, \qquad \forall (u,v) \in E, \forall i \in \{1, ..., k\}, \tag{8g}$$

$$w_i \in \mathbb{Z}^+, \qquad \forall i \in \{1, ..., k\}, \tag{8h}$$

$$x_{uvi} \in \{0, 1\}, \qquad \forall (u,v) \in E, \forall i \in \{1, ..., k\}, \tag{8i}$$

$$\pi_{uvi} \in \mathbb{R}^+ \cup \{0\}, \qquad \forall (u,v) \in E, \forall i \in \{1, ..., k\}. \tag{8j}$$

O. Subpath constraints coverage

$$u_j \leq \sum_{i\in\{1,...,k\}} r_{ij}, \qquad \forall R_j \in R, \tag{9a}$$

$$\sum_{R_j\in R} u_j \geq 0.9 * |R|, \qquad \forall i \in \{1, ..., k\}, \forall R_j \in R, \tag{9b}$$

$$u_j \in \{0, 1\}, \qquad \forall R_j \in R. \tag{9c}$$

u_j is a binary variable that is one if subpath R_j is used in at least one path, and zero otherwise. With the first constraint we allow u_j only to be one if the subpath was used in a path. With the second constraint we enforce that at least 90% of the subpath constraints must be used. The 0.9 can also be replaced by another number to enforce a different proportion of the subpaths to be covered.