Bioinformatics, 37(10), 2021, 1360–1366
doi: 10.1093/bioinformatics/btaa977
Advance Access Publication Date: 13 January 2021
Original Paper

Sequence analysis

On the complexity of haplotyping a microbial community

Samuel M. Nicholls [1,2,3,4,*], Wayne Aubrey [1], Kurt De Grave[2,5], Leander Schietgat[2,6], Christopher J. Creevey[3,7,†] and Amanda Clare[1,†]

[1]Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, UK, [2]Department of Computer Science, Katholieke Universiteit Leuven, 3001 Leuven, Belgium, [3]Institute of Biological, Rural and Environmental Sciences, Aberystwyth University, Aberystwyth SY23 3DA, UK, [4]Institute of Microbiology and Infection, School of Biosciences, University of Birmingham, Birmingham B15 2TT, UK, [5]Flanders Make, 3920 Lommel, Belgium, [6]Artificial Intelligence Lab, Vrije Universiteit Brussel, 1050 Ixelles, Belgium and [7]Institute of Global Food Security, School of Biological Sciences, Queen's University, Belfast BT9 5DL, UK

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the last two authors should be regarded as Joint Last Authors.

Associate Editor: Pier Luigi Martelli
Received on August 17, 2020; revised on November 4, 2020; editorial decision on November 7, 2020; accepted on November 9, 2020

Abstract

Motivation: Population-level genetic variation enables competitiveness and niche specialization in microbial communities. Despite the difficulty in culturing many microbes from an environment, we can still study these communities by isolating and sequencing DNA directly from an environment (metagenomics). Recovering the genomic sequences of all isoforms of a given gene across all organisms in a metagenomic sample would aid evolutionary and ecological insights into microbial ecosystems with potential benefits for medicine and biotechnology. A significant obstacle to this goal arises from the lack of a computationally tractable solution that can recover these sequences from sequenced read fragments. This poses a problem analogous to reconstructing the two sequences that make up the genome of a diploid organism (i.e. haplotypes) but for an unknown number of individuals and haplotypes.

Results: The problem of single individual haplotyping was first formalized by Lancia *et al.* in 2001. Now, nearly two decades later, we discuss the complexity of 'haplotyping' metagenomic samples, with a new formalization of Lancia *et al.*'s data structure that allows us to effectively extend the single individual haplotype problem to microbial communities. This work describes and formalizes the problem of recovering genes (and other genomic subsequences) from all individuals within a complex community sample, which we term the metagenomic individual haplotyping problem. We also provide software implementations for a pairwise single nucleotide variant (SNV) co-occurrence matrix and greedy graph traversal algorithm.

Availability and implementation: Our reference implementation of the described pairwise SNV matrix (Hansel) and greedy haplotype path traversal algorithm (Gretel) is open source, MIT licensed and freely available online at github.com/samstudio8/hansel and github.com/samstudio8/gretel, respectively.

Contact: s.nicholls.1@bham.ac.uk

1 Introduction

The problem of single individual haplotyping (SIH) was first described by Lancia *et al.* at Celera Genomics in 2001. In the wake of the announcement of Celera's first human genome, it became clear that the next big research problem was not only to analyse the millions of single point variants that populate our genomes, but how to assemble the two haplotypes that make up a single individual's genome (Lancia *et al.*, 2001). This 2001 work introduced the first terminology and notation for 'computational SNPology', a phrase that did not seem to catch on in the literature. A full discussion of the history of human (and by extension, diploid) haplotyping is outside the scope of this work [although for an expanded discussion see Nicholls (2018)], but it is important to introduce this first description of the problem, as it formed the foundation for many other approaches and algorithms that followed.

The computational problems involved with haplotyping arise because genomic assembly algorithms typically achieve accuracy through the availability of high sequencing coverage. Although it is now possible to assemble high quality, contiguous sequences of

entire genomes, even for a complex community sample (Nicholls *et al.*, 2019), such sequences only represent a consensus of the true haplotypes that exist in a sample. This is problematic for the study of metagenomes where the objective of consensus generating assembly methods is at odds with our desire to explore the individual haplotypes that provide the population-level diversity of natural microbiomes. Specialized metagenomic assemblers like flye (Kolmogorov *et al.*, 2019) and metaSPAdes (Nurk *et al.*, 2017) do not aim to reconstruct haplotypes from a microbial community. Not only it is technically challenging to sequence every true haplotype to a depth sufficient for confident assembly, but also even high-quality, single-molecule, long-read sequencing platforms fail to achieve perfect recall and perfect precision on the single read level, and are therefore not capable of single-molecule haplotype identification (Ebler *et al.*, 2019). Even high-accuracy sequencing techniques such as circular consensus sequencing still produce a per-base error rate of around 1% (and a higher error rate for insertions and deletions) that will be indistinguishable from low-frequency haplotypes (Wenger *et al.*, 2019). Ideally, large-scale culturing projects would aim to culture and sequence every genome from every individual in a microbial community. However, such endeavors would be incredibly laborious and even large international projects such as the Hungate Collection (Seshadri *et al.*, 2018) and Human Gastrointestinal Bacteria Culture Collection (Forster *et al.*, 2019) generally produce a subset of representative genomes for each species. Despite these difficulties, in this article, we propose a method to reconstruct haplotypes. First, we must define the problem, then explain how solutions can be generated efficiently. Lancia *et al.*'s work first described the problem *for a diploid individual* as:

> 'Given a set of fragments obtained by DNA sequencing from the two copies of a chromosome, reconstruct two haplotypes that would be compatible with all the fragments observed.'—Lancia *et al.* (2001)

It is important to note that even in an ideal scenario with error-free read fragments, lack of coverage across the haplotypes (be it a chromosome, or region of interest) necessitates a problem definition where at best, one can only recover the two haplotypes that are most 'compatible' with the observed fragments. Perhaps more importantly for the context of this work, Lancia also defined a common notation to formally describe the problem of single individual haplotyping. The 'SNP matrix' (typically denoted M) is an $m \times n$ matrix encoding the binary allele observed at each SNP site $1, \ldots, n$ on each read fragment $1, \ldots, m$. That is, $M[i][j]$ is one of two possible alleles (typically labelled 0, 1 or A, B), or a gap (denoted –) observed at the jth SNP site on the ith read fragment (Fig. 1).

Although unstated, it would appear the inspiration for Lancia's SNP matrix in 2001 is likely from a data structure introduced by Churchill and Waterman (1992). Churchill and Waterman described a matrix with m read fragment rows and n base position columns. An element in this matrix (which given the benefit of hindsight, we will denote as M), $M[i, j]$ refers to the nucleotide observed 'on a gel' at position j on the ith read fragment. Of course, the problem a decade prior considers the accuracy of only one sequence: that of the true DNA molecule that was sequenced, rather than untangling diploid sequence data to recover a solution of two distinct haplotypes. Lancia *et al.*'s work additionally define three optimization problems to solve SIH, which involve identifying and mitigating different types of 'conflicts' in the SNP matrix. A pair of read fragments r_i, r_j are said to be in fragment conflict if they have opposing alleles on at least one SNP. Similarly, a pair of SNPs s_k, s_l are said to be in SNP conflict if reads r_u and r_v are heterozygous at one SNP and homozygous at the other. Although a conflict might indicate that a pair of fragments originate from different haplotypes, Lancia *et al.* focus on the idea that conflicts arise due to errors in the underlying sequence data, arguing that 'experiments in molecular biology are never error-free'. An SNP matrix M which contains any conflicts is *infeasible*, and the three optimization methods aim to resolve these conflicts to yield a pair of feasible haplotypes. This SNP matrix influenced almost all haplotyping algorithms for the next twenty years (Lancia, 2016; Nicholls, 2018).

Lancia *et al.*'s definition of the SNP matrix only permits binary symbols (and gaps –) and so conflict resolution strategies that operate on this matrix are only valid under a diploid assumption, or polyploid cases where only the major and minor alleles are considered (Aguiar and Istrail, 2013; Schrinner *et al.*, 2020). Polyploid haplotyping algorithms also make the assumption that the ploidy is fixed and known in advance, and that the ploidy number can be used to remove or correct conflicting data (He *et al.*, 2018; Moeinzadeh *et al.*, 2020). These assumptions will not hold when analysing a microbial community, where a diverse population of individual organisms can specialize to produce an unknown number of variants of the same gene (i.e. haplotypes) (Rubino *et al.*, 2017). Reconstructing this population-level diversity would be a significant step towards a complete understanding of the evolutionary, ecological and functional importance of microbial communities (Kuleshov *et al.*, 2016; Stewart *et al.*, 2019; Zhang and Kim, 2010), so there is a need to reformulate the problem of haplotyping such that it can be applied to a microbial community. To do this, we take inspiration from Lancia *et al.*'s seminal work and offer a new data structure and algorithmic solution to generalize the haplotype recovery problem to individuals in a microbial community. We term this the metagenome individual haplotyping (MIH) problem.

2 Theory

We begin with a naive *de novo* formulation of the MIH problem, where the input is a collection of reads generated by a DNA sequencer from an environmental sample, with an unknown number of organisms. The ideal output from a solution to the MIH problem is the collection of whole-genome sequences representing all the

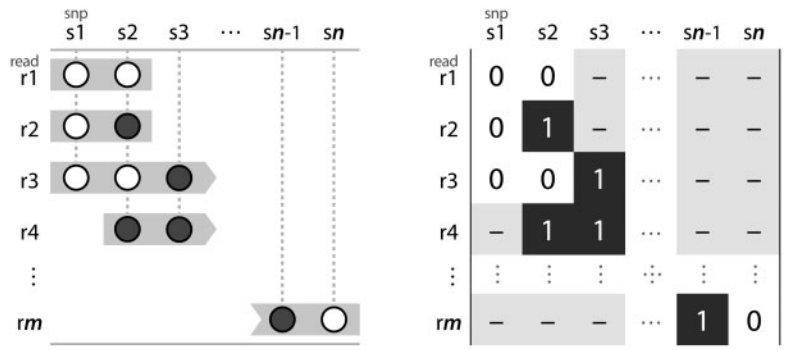

Fig. 1. An example SNP matrix. Read fragments represented by grey boxes (left) are aligned to some reference with known SNP loci. The alleles at the SNP loci are represented by white and grey circles. These reads can be alternatively represented by an $m \times n$ SNP matrix (right). Each row of the matrix models one of the m read fragments and each column corresponds to one of the n SNPs. Elements encode the allele at a given SNP for a particular read fragment as a 0 or 1, or a—if the read does not cover that position. A column containing only one element indicates the corresponding SNP site is homozygous, otherwise it is heterozygous

individual organisms in a microbial community. To illustrate further, we require the following definitions:

- Ω, a microbial community.
- O, a set containing each full genomic sequence, of each individual organism in environment Ω. O is *the* 'metagenome' of Ω: encompassing all possible genomes in the environment.

Arguably one could consider O as a bag, allowing multiple copies of the exact same genome in the community. For the purposes of haplotyping, we do not need to concern ourselves with recovering duplicate genomes. We also require some definitions to denote how the community is sampled:

- $\sigma \subseteq \Omega$, a sample is taken from microbial community Ω.
- M
  - A set containing the unique genomes $m \in M$, from the individual organisms captured in the sample σ.
  - M represents the metagenome that was captured in the sample σ, and is our insight into O.
  - As a subsample ($M \subseteq O$), M is not necessarily representative of the entire metagenome O.
- $R = \text{Sequencing}(\sigma)$
  - R is the set of **reads** obtained from the sequencing of isolated DNA from sample σ.
  - A read consists of a sequence of nucleotide bases $r_i[j] \in \{A, C, G, T, N\}, i \in 1..|R|, j \in 1..|r_i|$.
  - A read $m[u : v]$ describes a fragment of some genome $m \in M$, covering positions $u, v \in 1..|m|$, with some degree of error.
  - In addition, due to library preparation bias and sequencing errors, R is unlikely to provide uniform and non-zero coverage of the bases across all $m \in M$.

Formally, the goal of MIH appears to be the recovery of M from R. However, to construct a form of SNP matrix, we must identify the variants that comprise haplotypes, which in turn necessitates a common frame of reference for the reads. In the context of SIH, this would be a known reference sequence. Unfortunately for us, the ideal frame of reference for MIH is M—the very thing we are trying to recover. Thus, our naive definition of *de novo* MIH collapses to an exhaustive, special case of the *de novo* assembly problem. A solution to MIH is confounded by five problems: (i) DNA from every genome needs to be extracted and sequenced to a depth sufficient for recovery, (ii) genomes share homologous regions that require disambiguation, (iii) reads may be of an insufficient length to disambiguate repeats or resolve bridges between variants, (iv) sequencing error can be indistinguishable from rare haplotypes and (v) the presence of an unknown number of haplotypes complicates the already computationally difficult (NP-hard) (Cilibrasi *et al.*, 2005) problem of haplotyping. These issues all apply to both the metagenomic haplotyping and metagenomic assembly problems, which is why there is no metagenomic *de novo* assembler that attempts to exactly recover M from R. Metagenomic *de novo* assemblers such as flye and metaSPAdes still aim to recover consensus sequences. Recent experimental developments to flye allow a user to retain 'haplotigs', whereby bubbles in the graph structure are left uncollapsed, however, this still requires significant coverage for those alternative sequences to be assembled in the first place, and comes at the cost of assembly contiguity.

Although such *de novo* consensus sequences do not represent the entire metagenome of M, and typically collapse regions with shared homology into single broken contigs, we suggest that they provide a suitable approximation against which to align reads and construct a SNP matrix for the purpose of haplotyping. We, therefore, redefine the input to the MIH problem as a collection of sequencing reads R, their alignment to a set of *de novo* assembled contigs (and/or any existing references) and the variants determined by inspecting that alignment:

- $C = \text{Assemble}(R)$
  - Contig set C (assembly) constructed *de novo* from the reads R by some Assemble operation.
  - $c_k[j] \in \{A, C, G, T, N\}$ for $k \in 1..|C|, j \in 1..|c_i|$.
  - Assemble attempts to construct a set of consensus sequences representative of genomes in M, from the reads R. It should be noted that Assemble does not try to recover M, and typically fails to distinguish between similar sequences that should create distinct $c \in C$.
- $A = \text{Align}(R, C)$
  - Alignment A is generated by aligning read set R to contig set C with operation Align.
  - An alignment $a \in A$ defines a correspondence between some subsequence of read $r \in R$ and a subsequence of contig $c \in C$. Note that the length of the subsequences with correspondence between r and c may not necessarily be equal.
  - A_{c_k} is the set of all alignments to contig c_k.
  - $A_{c_k[i:j]}$ is the subset of alignments to contig c_k where any correspondence in R was found between positions i and j on contig c_k.
- $S_{c_k} = \text{Call}(A_{c_k})$
  - The list of positions on contig $c_k \in C$ determined to be heterogeneous by the operation Call.
  - We will henceforth refer to these positions as single nucleotide variants (SNVs) rather than SNPs in our descriptions to make it clear that there are no assumptions on variant frequency or validity.
  - Call may simply consider each 'column' $A_{c_k}[i]$ for $i \in 1..|c_k|$ and determine position i as a variant if there is a disagreement on the nucleotide at that position across the aligned reads. Call may also be a more complex variant prediction algorithm.

To enable computational tractability and reduce the reliance on the quality of the assembly, we can consider the local MIH problem. The input to the local MIH problem is constrained by selecting a contig from C and filtering for alignments in A between positions i and j:

- $c_k[i : j]$, a **region** of contig c_k, identified as biologically interesting
- $A_{c_k[i:j]}$, the **alignments** of reads R that map to the contig region $c_k[i : j]$
- $S_{c_k[i:j]}$, the list of positions determined to be **variants** over the region $c_k[i : j]$

As $c_k[i : j]$ was assembled from R, it is an approximation of one (or more) genomes in the sampled metagenome M. In the same way that an assembler can collapse multiple genomes with shared sequence into single contigs, or several broken contigs, an aligner may be unable to determine whether read r truly belongs to a single assembled contig in C. We cannot guarantee that reads aligned to $c_k[i : j]$ originate from the same genome m in the sampled metagenome M.

We use this to our advantage for local MIH. Consider first:

- f, a biologically interesting genomic **feature** such as (but not limited to) a gene or operon
- M_f, the subset of the genomes in the sampled metagenome M that have feature f

- m_k, some genome in M_f
- $m_k[i:j]$, a region $i..j$ on m_k that has concordance with feature f, with appropriate length $j - i \approx |f|$
- $\Gamma_f = set(\{m_k[i:j] \mid k \in 1..|M_f|,\ i,j \in 1..|m_k|\})$. Γ_f is the set of haplotypes of f in M.

Now let $c_k[i:j]$ have concordance to a feature f. The output of local MIH is the set of haplotypes Γ_f. Although the ambiguity in the alignments $A_{c_k[i:j]}$ enable the variation in Γ_f to be captured, the evidence they provide does not readily resolve to haplotypes. The Lancia *et al.* SNP matrix is not suitable for MIH, as the only information stored about a variant site on a read is whether it belongs to one of two haplotypes. In order to recover haplotypes that actually exist, we must define a new data structure that is capable of considering evidence from non-adjacent variant sites, and is agnostic to the number of potential haplotypes in Γ_f.

3 Data structure

3.1 The pairwise SNV co-occurrence matrix

We present a new form of SNP matrix: the pairwise SNV co-occurrence matrix, denoted H. Consider two positions i and j, on an assembled contig c_k. With a read r aligned to this contig, let α be the nucleotide on r at i, and β be the nucleotide at j. We say that the read supports an observation that symbol α_i co-occurs with symbol β_j. H is a rank 4 tensor (which we will refer to as a four-dimensional matrix for convenience) such that an individual element $H[\alpha,\ \beta,\ i,\ j]$ records the number of such observations supporting a co-occurring pair of symbols $(\alpha_i,\ \beta_j)$. Note that the reads are aligned with respect to the contig as a reference such that α, β are in the alphabet $\Sigma = \{A, C, G, T, N, -\}$, where – denotes a deletion in the read. We do not consider insertions.

This representation differs from the typical SNP matrix model (Lancia *et al.*, 2001) that forms the basis of many haplotyping approaches. Rather than a matrix of columns representing variants and rows representing reads, we discard the concept of a read entirely and aggregate the evidence seen across all reads with pairs of symbols and positions.

More importantly, H can be exploited to build other structures.

3.2 H as a simple graph

Consider $H[\alpha,\ \beta,\ 1,\ 2]$ for all symbol pairs $(\alpha,\ \beta)$. One may enumerate the available transitions from position 1 to position 2. Extending this to consider $H[\alpha,\ \beta,\ i,\ i+1]$ for all $(\alpha,\ \beta)$ and $i \in 1..n-1$ (where $n = |S_{c_k}|$ is the number of SNVs identified on contig c_k), one can construct a simple graph G of possible transitions between all symbols. G would represent a graph of transitions observed between SNVs, across all reads. A node in G represents a state, pairing a symbol to a position (e.g. an 'A' at position 1). An edge between a pair of nodes in G represents two adjacent SNVs $(i,\ i+1)$ and can be labelled with a weight using the number of corresponding co-occurrence observations in H. Figure 2 shows how H records information about SNV pairs, and how a simple graph can be derived from this information.

Formally, H can be considered as a graph $G = (V, E)$. Here, we define E and V as:

$$E = \cup i = 1..n-1\{(\alpha_i,\ \beta_{i+1}) \mid H[\alpha,\ \beta,\ i,\ i+1] > 0,\ \alpha,\beta \in \Sigma\} \quad (1)$$

$$V = \{v \mid (v,w) \in E\} \cup \{v \mid (w,v) \in E\}. \quad (2)$$

Intuitively, one may traverse a path through G by selecting the edges most highly supported by the corresponding evidence in H, in order to recover a series of symbols representing an ordered sequence of SNVs that potentially constitutes a haplotype. We can now formalize a haplotype $\hat{h}$ as a sequence of nodes ($v \in V$) encountered in this graph:

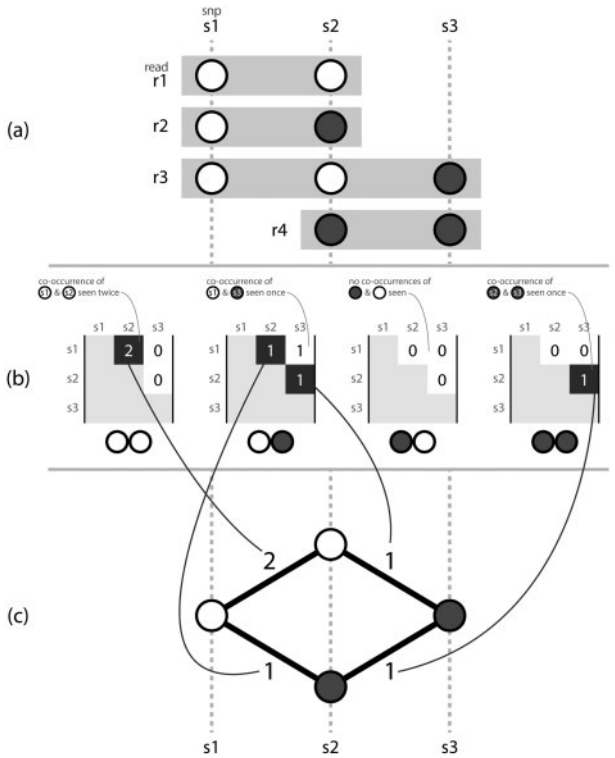

Fig. 2. Three corresponding representations, (**a**) a set of aligned reads $r1.r4$, with called variants $s1.s3$, (**b**) the pairwise SNV co-occurrence matrix H where each possible pair of symbols (00, 01, 10, 11) has a matrix storing counts of occurrences of that ordered symbol pair between two positions across the aligned reads, (c) a simple graph that can be constructed by considering the evidence provided by adjacent variants. H is represented by an upper triangular matrix, as it is unnecessary to store the same observations with reversed positions in the lower diagonal nor observations of transitions to self along the main diagonal. Note for simplicity this example uses an alphabet of only two symbols, but in practice we consider an alphabet $\Sigma = \{A, C, G, T, N, -\}$

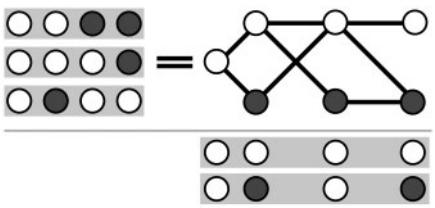

Fig. 3. Considering only adjacent SNVs, one may create paths for which there was no actual observed evidence. Here, the reads {0011, 0001, 0100} do not support either of the results {0000, 0101}, but both are valid paths through a graph that permits edges between pairs of adjacent SNVs

$$\hat{h} = v_1, v_2, \ldots, v_{n-1}, v_n. \quad (3)$$

Although the analogy to a graph helps us to consider paths, the available data in H cannot be fully represented with a graph such as that seen in Figure 2 because it includes data about co-occurrence of non-adjacent SNV positions. A graph representation constrains our representation as edges can only be drawn between adjacent SNVs $(i, i+1)$. Without considering information about non-adjacent SNVs, one can traverse G to create paths that do not exist in the observed dataset, as shown in Figure 3. To prevent construction of invalid paths and recover genuine paths more accurately, a SNV matrix for MIH must be able to consider evidence observed between non-adjacent symbols when determining which edge to traverse next.

3.3 H as a probabilistically weighted graph

We may take advantage of the additional information available in H and build upon the graph G. Rather than directly setting the edge

weights to values from H, we define a formula to weight edges based on the support of pairs between a path of visited nodes and any potential next node. That is, a decision to move to a symbol at position $i+1$ is informed not only by observations in H supporting $(i, i+1)$, but also the non-adjacent observations $(i-1, i+1)$, $(i-2, i+1)$, and so on.

Here, we formalize a Bayesian probabilistic framework to determine the edge weights in G, based on a path of already visited nodes. We define the probability of selecting v_{i+1}, conditioned on the path observed so far, as:

$$\begin{aligned}
&\mathbb{P}(v_{i+1} \mid v_1, v_2, \ldots, v_{i-1}, v_i) \\
&\propto \mathbb{P}(v_1, v_2, \ldots, v_i, v_{i+1}) \\
&= \mathbb{P}(v_1 \mid v_2 \ldots v_{i+1}) \times \mathbb{P}(v_2, \ldots v_{i+1}) \\
&= \mathbb{P}(v_1 \mid v_2 \ldots v_{i+1}) \times \mathbb{P}(v_2 \mid v_3 \ldots v_{i+1}) \times \mathbb{P}(v_3, \ldots v_{i+1}) \\
&= \mathbb{P}(v_1 \mid v_2 \ldots v_{i+1}) \times \mathbb{P}(v_2 \mid v_3 \ldots v_{i+1}) \times \cdots \times \mathbb{P}(v_{i-1} \mid v_i, v_{i+1}) \\
&\quad \times \mathbb{P}(v_i \mid v_{i+1}) \times \mathbb{P}(v_{i+1}).
\end{aligned} \tag{4}$$

Clearly, evaluating the equation becomes more computationally expensive and risks compounding estimation errors as the path length increases. To construct a path from $v_1 \ldots v_n$, the upper bound for the number of operations will be $|\Sigma| \times n$ with calculations becoming increasingly complex as i increases. We can reduce complexity with two assumptions: (i) conditional independence between variants and (ii) one can limit the number of elements in the prior path without loss of accuracy because the reads providing this co-occurrence support are limited in length. This simplifies our previous equation as we only need to consider the pairwise appearances of each v_i encountered thus far against v_{i+1}; and limit the number of variants to consider, from the current position in the path i, back some small and sensible number of steps L:

$$\begin{aligned}
&\log_{10}(\mathbb{P}(v_{i+1} \mid v_{i-L}, \ldots, v_{i-2}, v_{i-1}, v_i)) \\
&\quad \approx \log_{10}(\mathbb{P}(v_{i+1})) + \sum_{l=0}^{L-1} log_{10}(\mathbb{P}(v_{i-l} \mid v_{i+1})).
\end{aligned} \tag{5}$$

As a minor implementation detail, to overcome inaccuracies encountered through floating point error when performing mathematical operations on very small fractional values, we provide these definitions with log probabilities instead. We define L as the 'look-back' size, the number of variants of the current path to consider when selecting v_{i+1}.

We use H to estimate the marginal and conditional probabilities. Equation 6 provides an estimate for the marginal distribution of a symbol β appearing at position j. As a minor implementation detail, we expand our alphabet Σ to include a dummy symbol $\varnothing$. When constructing H, the last SNV γ_j on a read r, is automatically linked to $\varnothing_{j+1}$ (i.e. $H[\gamma, \varnothing, j, j+1]$ is incremented). This allows the span between j and $j+1$ to be calculated even in the case where j is the last observed position on a read.

$$\begin{aligned}
\hat{}(v_j = \beta) &= \frac{\text{Number of reads with symbol } \beta \text{at position } j}{\text{Number of reads spanning position } j} \\
&= \frac{\sum_{\delta \in \Sigma} H[\beta, \delta, j, j+1]}{\sum_{\gamma \in \Sigma} \sum_{\delta \in \Sigma} H[\gamma, \delta, j, j+1]}.
\end{aligned} \tag{6}$$

Equation 7 provides an estimate for the conditional distribution of symbol α appearing at position i given that β was observed at position j. To avoid the potential of dividing by 0 in cases where a suitable read spanning i and $v_j = \beta$ does not exist, we apply Laplace smoothing, adding one dummy read that will provide support for each possible pair of SNV symbols (including ones that were not observed).

$$\begin{aligned}
&\hat{}(v_i = \alpha \mid v_j = \beta) \\
&= \frac{1 + \text{Number of reads featuring} \alpha \text{ at } i \text{ and } \beta \text{ at } j}{\text{Variants at } i + \text{Number of reads spanning } i \text{ featuring symbol } \beta \text{ at } j} \\
&= \frac{1 + H[\alpha, \beta, i, j]}{|\{\gamma_i \mid H[\gamma, \sigma, i, i+1] > 0, \gamma \in \Sigma, \sigma \in \Sigma\}| + \sum_{\gamma \in \Sigma} H[\gamma, \beta, i, j]}.
\end{aligned} \tag{7}$$

4 Algorithm

4.1 A greedy graph traversal algorithm for local MIH

We have formulated the above ideas into an algorithm for finding solutions to local MIH. Here, we interpret the problem of MIH as a search problem rather than an optimization problem. Unlike Lancia and Lippert's algorithms where changing or removing evidence from the SNP matrix formed the solution to SIH, the method we introduce will use the pairwise SNV co-occurrence matrix (H) as a data structure to calculate the probabilities of edge traversals in the graph. Our algorithm traverses the haplotype space represented by the graph G in a greedy manner, selecting the edge with the highest likelihood (given the path traversed so far) in order to recover the highest likelihood haplotypes. The core of the algorithm is as follows:

1. Given $c_k[i:j]$, the location of a feature of interest (f) in the *de novo* assembly, parse the read alignments ($A_{c_k[i:j]}$) and construct the pairwise SNV co-occurrence matrix H
2. Initialise a haplotype $\hat{h} = [\,]$, $i = 0$ and assign L to be the mean number of SNVs per read
3. Representing H as a graph G;
   - Query for the available edges from current position i, to the next position $i+1$
   - Calculate the probabilities for each available edge using Equation 5, given $\hat{h}[i-L:i]$
   - Traverse the most likely edge and append the new node to $\hat{h}$, increment i; however, if there are no edges that can be traversed, terminate the algorithm
   - If $i < |S_{c_k}|$ repeat step 3; else proceed to step 4
4. Report path $\hat{h}$ as a haplotype, and modify H to reduce the evidence supporting $\hat{h}$.
5. Repeat (3–4) until encountering a node with no edges that can be traversed, or an additional stopping criterion has been reached.

4.2 Reweighting H to find multiple haplotypes

Given H, the algorithm will behave deterministically and return the same haplotype for any traversal. However, the goal of local MIH is to recover Γ_f: the set of haplotypes corresponding to feature f, rather than just one haplotype. To recover more than just one haplotype, our algorithm must modify H to remove evidence for that haplotype, in order to weight the calculation of edge probabilities in favour of alternative haplotypes on the next traversal. We achieve this by using the smallest marginal distribution from path $\hat{h}$ as a ratio (λ) to decrease the observations in H that directly support the path $\hat{h}$. The intuition is that the least supported node in $\hat{h}$ estimates the proportion of evidence supporting the entire haplotype.

$$\lambda = \min(\{\mathbb{P}(\hat{h}[i]) \mid i = 1..|S_{c_k}|\}). \tag{8}$$

This ratio is used to decrease the evidence for all adjacent entries in H that correspond to $\hat{h}$.

$$\begin{aligned}
H[\hat{h}[i], \hat{h}[i+1], i, i+1] = H[\hat{h}[i], \hat{h}[i+1], i, i+1] \\
- (\lambda \times H[\hat{h}[i], \hat{h}[i+1], i, i+1]).
\end{aligned} \tag{9}$$

After multiple iterations of path finding and subsequent reweighting, elements in H will begin to approach 0, causing edges in the graph to become unavailable for traversal. At this point the algorithm will terminate. Alternatively, if this criterion is not reached after some predefined number of iterations, the algorithm will terminate.

4.3 Using H to score and rank haplotypes

We can use the estimated probabilities to also score and rank the haplotypes recovered. For a completed haplotype, $\hat{h}$, we compute its

likelihood based upon the sum of the marginal log probabilities for each element of $\hat{h}$ given the state of H prior to reweighting.

$$\begin{aligned} L(\hat{h}) &= \mathbb{P}(H \mid \hat{h}) \\ &= \mathbb{P}(v_1 = \hat{h}[1], v_2 = \hat{h}[2], \ldots, v_{n-1} = \hat{h}[n-1], v_n = \hat{h}[n]) \\ &= \prod_{i=1}^{n} \hat{\mathbb{P}}(v_i = \hat{h}[i]) \\ &= \prod_{i=1}^{n} \frac{\sum_{\gamma \in \Sigma} H[\hat{h}[i],\ \gamma,\ i,\ i+1]}{\sum_{\gamma \in \Sigma} \sum_{\delta \in \Sigma} H[\gamma,\ \delta,\ i,\ i+1]}. \end{aligned} \tag{10}$$

To overcome the potential for floating point arithmetic error, we calculate and report the log likelihood.

$$\log_{10}(L(\hat{h})) = \sum_{i=1}^{n} \log_{10}\left(\frac{\sum_{\gamma \in \Sigma} H[\hat{h}[i],\ \gamma,\ i,\ i+1]}{\sum_{\gamma \in \Sigma} \sum_{\delta \in \Sigma} H[\gamma,\ \delta,\ i,\ i+1]}\right) \tag{11}$$

5 Discussion

Like most common bioinformatics problems, haplotyping (SIH diploid, polyploid and now MIH) falls into the hardest class of computational problems (NP-hard), necessitating the use of heuristic algorithms. Heuristics sacrifice an element of optimality in favour of speed or feasibility and are an inevitable compromise in order to find candidate solutions to a problem at all. Our graph traversal algorithm is a greedy heuristic; selecting the highest likelihood edge to a variant at position $i+1$ given some path leading up to i. A greedy traversal will pick the optimal option for the current fork in the path, regardless of whether that will cause the search to be steered away from paths with future branches that would lead to a better path overall. Although a heuristic cannot guarantee optimality, our approach of reweighting (Equation 9) before starting a new path mitigates the problem of finding haplotypes from a local minima—as changes in the evidence available will force selection of different paths. The resulting haplotypes can be then be ranked by their assigned likelihoods (Equation 10) in order to filter out less probable haplotypes.

Similarly for any algorithm, data quality will also impact the optimality of generated solutions. The construction of the pairwise SNV co-occurrence matrix H is based on alignment of sequenced reads to a contig set generated by *de novo* assembly. Clearly, the quality of the contigs is a factor for the read alignments, which in turn affects the quality of the data inserted into H, which is why we consider the local MIH problem. Restricting the problem to shorter regions rather than full-length contigs significantly reduces the reliance on assembly quality. Even so, misassemblies should be poorly supported by the read alignments, indicating that the region in question is not suitable for haplotype recovery because it does not actually exist. Of course, the alignments are critical to the quality of the observations stored in H, and like any algorithm based on read evidence, a lack of coverage is a lack of evidence and will constrain the generation of solutions. Our approach requires every identified variant position to be linked to the next by at least one read (although ideally more). The graph traversal algorithm will abort upon finding a 'hole' in the graph—a position i where there are no outgoing edges to reach a new variant at position $i+1$. In a case where variants are sparse and the reads are too short to span them, the algorithm will stop during the first traversal. In a case where the read coverage is very low for a region under investigation, the algorithm will likely stop after a small number of traversals, as successive rounds of reweighting will quickly exhaust the little evidence available. In addition, the transition probabilities should be robust to noisy or lower quality reads where the noise is distributed non-uniformly, as errors should be unsupported by the reads in aggregate, yielding small numbers of observations in H that will not be chosen over more well-supported transitions.

Our approach is a fair compromise, offering a sensible heuristic that can run in reasonable time with reasonable resources, that should find approximately optimal haplotype solutions for local MIH.

6 Conclusion

With recent technical advancements reducing the cost and complexity of sequencing environmental samples researchers are turning their attention to focus on the population-level variation within microbial communities. However, the field is hampered by a lack of consensus in terminology. Segata recently analysed the terminology used for metagenomic strains and for the diversity present in microbiomes (Segata, 2018). He discussed the difficulties and misunderstandings that arise from a lack of strict definitions and states that it is 'crucial for this line of research to at least define practical and operational definitions' to address the open problem of characterizing unknown genomes within the human microbiome. More recently, Van Rossum *et al.* (2020) highlighted the 'overwhelming number of methods and terms to describe infraspecific variation' in a review. Their work provides much needed clarification to the terminology for metagenomic analyses. Figure 2 in their article depicts a proposed hierarchy of terms from species, through to sub-species, strain and down to the individual genome level. Even though they have provided copious terminology definitions, a formalization of the haplotype in the context of a metagenome is still absent. Our work clarifies further by providing a formal definition for the 'metagenomic individual haplotype'.

We have defined the pairwise SNV co-occurrence matrix that packs sequencing reads into a structure compatible with metagenomic haplotyping. Our SNV matrix can be used to build a graph, reducing the problem of local MIH to that of recovering the highest weighted paths from a graph. We provide a probabilistic framework to weight the edges in this graph based on the path observed so far, while also considering evidence from non-adjacent SNVs in the underlying read data. We provide a free and open implementation of this SNV matrix (Hansel) at github.com/samstudio8/hansel.

In addition, we propose a greedy solution to the problem of local MIH, using a graph representation of H to traverse paths through a graph to output haplotypes. Our algorithm requires no configuration, has no user-facing parameters and requires no pre-processing of reads, other than alignment. The haplotypes recovered with our algorithm can be scored and ranked by their likelihood. We provide a free and open implementation of this algorithm (Gretel) at github.com/samstudio8/gretel.

Our work extends the SIH problem introduced by Lancia *et al.*, to define the MIH problem. We offer our data structure and algorithm in the hope that they will form a foundation for future approaches to haplotyping microbial communities.

Acknowledgements

S.N. wishes to acknowledge Nicholas Loman (University of Birmingham) for supporting the completion of the manuscript.

Author Contributions

All authors discussed and defined the theoretical problem. S.N. wrote the code and documentation. All authors contributed to the manuscript.

Funding

S.N. was funded via the Aberystwyth University Doctoral Career Development Scholarship and the IBERS Doctoral Programme. S.N. was funded to travel and work at Katholieke Universiteit Leuven by the Erasmus+ Programme. W.A. is funded through the Coleg Cymraeg Cenedlaethol Academic Staffing Scheme. C.J.C. was funded by the BBSRC Institute Strategic Programme Grant, Rumen Systems Biology (BB/E/W/10964A01); DAFM Ireland/DAERA Northern Ireland under the Meth-Abate project (R3192GFS) and the EC via Horizon 2020 (818368, MASTER).

Conflict of Interest: none declared.

Data availability

Our reference implementation of the described pairwise SNV matrix (Hansel) and greedy haplotype path traversal algorithm (Gretel) are open source and

distributed freely under the MIT license online via github.com/samstudio8/hansel and github.com/samstudio8/gretel.

References

Aguiar,D. and Istrail,S. (2013) Haplotype assembly in polyploid genomes and identical by descent shared tracts. *Bioinformatics*, **29**, i352–360.

Churchill,G.A. and Waterman,M.S. (1992) The accuracy of DNA sequences: estimating sequence quality. *Genomics*, **14**, 89–98.

Cilibrasi,R. *et al.* (2005) On the complexity of several haplotyping problems. In: Casadio,R. and Myers,G. (eds) *Algorithms in Bioinformatics*. Springer, Berlin, Heidelberg, pp. 128–139.

Ebler,J. *et al.* (2019) Haplotype-aware diplotyping from noisy long reads. *Genome Biol.*, **20**, 116.

Forster,S.C. *et al.* (2019) A human gut bacterial genome and culture collection for improved metagenomic analyses. *Nat. Biotechnol.*, **37**, 186–192.

He,D. *et al.* (2018) Efficient algorithms for polyploid haplotype phasing. *BMC Genomics*, **19**, 110.

Kolmogorov,M. *et al.* (2019) Assembly of long, error-prone reads using repeat graphs. *Nat. Biotechnol.*, **37**, 540–546.

Kuleshov,V. *et al.* (2016) Synthetic long-read sequencing reveals intraspecies diversity in the human microbiome. *Nat. Biotechnol.*, **34**, 64–69.

Lancia,G. (2016) Algorithmic approaches for the single individual haplotyping problem. *RAIRO Oper. Res.*, **50**, 331–340.

Lancia,G. *et al.* (2001) SNPs problems, complexity, and algorithms. In: auf der Heide,F.M. (eds) *Algorithms – ESA 2001*. Springer, Berlin, Heidelberg, pp. 182–193.

Moeinzadeh,M.-H. *et al.* (2020) Ranbow: a fast and accurate method for polyploid haplotype reconstruction. *PLoS Comput. Biol.*, **16**, e1007843.

Nicholls,S.M. (2018) Computational recovery of enzyme haplotypes from a metagenome. Ph.D. thesis, Aberystwyth University.

Nicholls,S.M. *et al.* (2019) Ultra-deep, long-read nanopore sequencing of mock microbial community standards. *Gigascience*, **8**, giz043.

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

Rubino,F. *et al.* (2017) Divergent functional isoforms drive niche specialisation for nutrient acquisition and use in rumen microbiome. *ISME J.*, **11**, 1510–1510.

Schrinner,S.D. *et al.* (2020) Haplotype threading: accurate polyploid phasing from long reads. *Genome Biol.*, **21**, 252.

Segata,N. (2018) On the road to strain-resolved comparative metagenomics. *mSystems*, **3**, e00190.

Seshadri,R. *et al.*; Hungate1000 Project Orators. (2018) Cultivation and sequencing of rumen microbiome members from the Hungate1000 Collection. *Nat. Biotechnol.*, **36**, 359–367.

Stewart,R.D. *et al.* (2019) Compendium of 4,941 rumen metagenome-assembled genomes for rumen microbiome biology and enzyme discovery. *Nat. Biotechnol.*, **37**, 953–961.

Van Rossum,T. *et al.* (2020) Diversity within species: interpreting strains in microbiomes. *Nature Reviews Microbiology*, **18**, 491–506.

Wenger,A. *et al.* (2019) Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat. Biotechnol.*, **37**, 1155–1162.

Zhang,C. and Kim,S.-K. (2010) Research and application of marine microbial enzymes: status and prospects. *Marine Drugs*, **8**, 1920–1934.