Answer Set Programming based haplotype phasing of long reads for polyploid species - ROADEF 2022

Clara Delahaye[1], Jacques Nicolas[1]

Univ. Rennes, Inria, CNRS, IRISA, F-3500 Rennes, France
{clara.delahaye, jacques.nicolas}@irisa.fr

Mots-clés : *Constraint logic programming, Partitioning, Phasing, DNA sequencing data*

1 Biological context and motivations

Living organisms have their DNA organized into chromosomes, each complete set of chromosomes being present in two (for diploid organisms such as humans and most animals) or more copies (for polyploid organisms like many crops). We call *haplotype* each copy within a given set. Haplotypes are highly similar, but show biologically important differences called *variants*, that can be of high interest as they may be involved into biological processes or genetic diseases. However most of genome representations available today are made of a mix of all haplotypes, thus masking variants and leading to missing or erroneous information.

Our aim is to build reference sequences for each haplotype of a genome, taking as input genome subsequences called *reads* issued from a sequencing machine. Roughly, reads can either be short and accurate, or long and erroneous. In this paper, we consider long reads data, which allow to better reconstruct haplotypes, by observing series of correlated variants. Given a set of positions, and the variants at these positions for each read, the haplotype phasing task we consider consists in determining the true variant for each haplotype at each position.

2 Problem formalization

We now describe in mathematical terms the challenge of polyploid haplotype phasing. A chromosome can be represented by k haplotypes $H_1, H_2, ..., H_k$ and n positions of variation, with k and n fixed. Each haplotype H is represented by a vector of variants v. We note $H(p) = v$, where $p \in [1, n]$. The goal is to reconstruct them from an input set of reads. A read R is a subset of observed variants of an unknown haplotype. We note $R(p) = v$, where the set of p belongs to an interval $I(R)$ of $[1, n]$. Given two reads R_1 and R_2, two haplotypes H_1 and H_2 are said *locally different* with respect to R_1 and R_2 if $H_1(p) \neq H_2(p)$ for some p in $I(R_1) \cap I(R_2)$. A certain percent of variants may be erroneous, due to the sequencing technology to obtain reads. Finally, the problem is defined as:

Given:

- a global ploidy k, fixing a maximum number of different values per position;
- a set of N reads $\{R(p) = v\}$, where p belongs to an interval $I(R)$ of $[1, n]$;
- a graph on reads with overlapping intervals, with two types of edges, different and similar.

Find:

A k-partition of reads to form k haplotypes. The partition must minimize:

- the number of pairs (R, p) for reads R in haplotype H, for which $R(p) \neq H(p)$;
- the number of *different* read pairs assigned to a same haplotype;
- the number of *similar* read pairs assigned to locally different haplotypes.

3 State of the art

Most methods currently available are only designed for short reads and/or for diploid species. As said previously, long reads offer assets on haplotype phasing over short ones but such data are more recent and methods are lacking to realize their full potential. Concerning ploidy, haplotype phasing of diploid genomes is now quite well handled as it is reduced to a binary choice. Reasoning is more complex in the case of polyploid phasing. A common intuition is to assume that reads stemming from a same haplotype will be more similar than reads from different haplotypes. Thus, some common strategies rely on similarity between reads, using for example a similarity scoring function [3] or clique enumeration in an overlap graph of reads [1].

4 Method

Here we propose both a formalization and a combinatorial method for diploid and polyploid haplotype phasing of long read data. We address haplotype phasing as an optimization problem and use Answer Set Programming [2] (ASP), with clingo system to solve it. Rather than providing a unique and likely erroneous answer to this hard problem, the ASP framework allows to reason on the set of possible solutions. Moreover, ASP is a high-level declarative language that offers both efficiency (inspired on SAT-solver techniques) and expressiveness (more than ILP for example). One can easily express preferences and get a global view of confident and ambiguous positions in phased regions.

As the phasing problem is highly combinatorial, we need to pre-process the data to reduce the search space. A trivial pruning condition is to remove variant positions for which all values in reads are identical, as they do not provide any relevant information for phasing. Then we construct a graph of read overlaps and split it into connected components that will be phased independently, in order to reduce the size of each phasing problem.

For the phasing step, we use constraints and minimization criteria based on differences and similarities between reads (see definition of the similarity score bellow), taking into account potential unknown errors in reads. The phased fragments are then re-assembled to produce the final haplotypes. The overall (still ongoing) method will be designed in two complementary parts: a rather traditional one computing similarities between reads; and a combinatorial one that will interact with the user to explore the set of possible phasing solutions.

We define a similarity score between two reads R_1 and R_2 as the sum of similarity score of their variants on shared positions:

$$sim(R_1, R_2) = \sum_{p \in I(R_1) \cap I(R_2)} ind(R_1(p), R_2(p)) \quad \texttt{where } ind(v, w) = \begin{cases} +1 & \text{if } v = w \\ -1 & \text{if } v \neq w \end{cases}$$

We then produce the graph on reads by setting two thresholds $t_{high} > t_{likely}$ for labelling edges between two reads (R_i, R_j) sharing enough positions:

$$label(R_i, R_j) = \begin{cases} \texttt{highly similar (different)} & \text{if } sim(R_i, R_j) \geq t_{high} \ (\leq -t_{high}) \\ \texttt{likely similar (different)} & \text{else if } sim(R_i, R_j) \geq t_{likely} \ (\leq -t_{likely}) \end{cases}$$

References

[1] Jasmijn A. Baaijens et al. "De novo assembly of viral quasispecies using overlap graphs". In: *Genome Research* 27.5 (Apr. 2017), pp. 835–848. DOI: 10.1101/gr.215038.116. URL: https://doi.org/10.1101/gr.215038.116.

[2] Martin Gebser et al. "Answer Set Solving in Practice". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.3 (Dec. 2012), pp. 1–238. DOI: 10.2200/s00457ed1v01y201211aim019. URL: https://doi.org/10.2200/s00457ed1v01y201211aim019.

[3] Sven D. Schrinner et al. "Haplotype threading: accurate polyploid phasing from long reads". In: *Genome Biology* 21.1 (Sept. 2020). DOI: 10.1186/s13059-020-02158-1. URL: https://doi.org/10.1186/s13059-020-02158-1.