Assembly of highly polymorphic diploid genomes (with polymorphism level 1-10%) is a computationally hard problem. Several approaches have already been proposed for assembling such datasets [1][2], but all known methods are based on overlap-layout-consensus and cannot be applied to NGS (Next Generation Sequencing) data. Existing NGS assemblers are inefficient too: they do not use information about genome diploidy which, as we show in this work, can help improve assembly. We present an algorithm that uses this information for efficient assembly of highly polymorphic diploid genomes sequenced with NGS. Implementation of the proposed algorithm is based on SPAdes genome assembler [3].

**PROBLEM STATEMENT**

**Definitions**

- Diploid genomes have two homologous copies of each chromosome.
- Collection of one copy from each homologous pair of chromosomes is called a haplome. Thus diploid genome $G$ consists of two haplotypes $G_1$ and $G_2$.
- Consensus genome $G$ is a haplome that corresponds to one of the possible haplotypes (alleles are chosen randomly from $G_1$ and $G_2$).

**Low and high polymorphism level**

- In case of low polymorphism our problem has a simple standard solution. Mask all polymorphisms: for each polymorphism choose one of the two alleles and correct all reads that map to this locus to match the chosen allele. Corrected reads represent a haploid consensus genome and can easily be assembled.
- Since polymorphism level is low reads can be aligned to consensus genome using standard alignment tools. Thus (2) reduces to a well known haplotype assembly problem.
- In case of high polymorphism level masking becomes more tricky while read alignment using standard tools becomes highly inaccurate. We use de Bruijn graphs to perform both tasks more accurately.

**ALGORITHM**

**Step 1. Computing contigs for both haplotomes $G_1$ and $G_2$**

We construct de Bruijn graph from reads. Paired reads are used to resolve repeats in the constructed de Bruijn graph. The resulting contigs represent an assembly of genome $G$. Thus alleles from both $G_1$ and $G_2$ are covered by these contigs.

Illustration of de Bruijn graph and contigs obtained by repeat resolving. Each black graph edge is covered by a single contig. Colored graph edges are covered by at least two contigs. Corresponding contig regions are highlighted with the same color.

**Step 2. Polymorphism masking in de Bruijn graph**

In de Bruijn graphs each polymorphism becomes a pair of paths that start from and end in vertices. We search for such paths and collapse them. We also keep information about how the paths were rerouted. Ideally this procedure allows us to mask all polymorphisms.

Each of the two pairs of paths on the figure in the de Bruijn graph represents a polymorphism. Green arrows show how we collapse each polymorphism. Information about path rerouting (presented with green arrows) is stored to be used in step 3.

**Step 3. Polymorphism masking in haplotype contigs**

We use information stored at step 2 to mask polymorphisms in contigs created at step 1.

Contigs are mapped to the graph using information stored at step 2. As edge colors show, each edge in the graph has at least two contigs mapped to it. Each contig is replaced with the path in the graph it is mapped to. Since polymorphisms are collapsed in the graph, in the resulting contig collection polymorphisms are also collapsed.

**Step 4. Construction of consensus contigs**

Masked contigs which were obtained at step 3 have exact overlaps and can be used as reads to construct overlap graph. This graph is then used to construct consensus contigs.

The figure shows overlap graph that is used to construct consensus contigs. Also we have enough information to map contigs obtained at step 1 to consensus contigs. This information can be further used to partially restore haplotypes of $G_1$ and $G_2$.

**REFERENCES**