HapCHAT: Adaptive haplotype assembly for efficiently leveraging high coverage in long reads

Stefano Beretta[1,†], **Murray Patterson**[1,†,*], **Tobias Marschall**[2,3], **Marcel Martin**[4], **Simone Zaccaria**[1,5], **Gianluca Della Vedova**[1], **and Paola Bonizzoni**[1,*]

[1] Department of Informatics, Systems, and Communication, University of Milano-Bicocca, Milano, Italy and
[2] Center for Bioinformatics, Saarland University, Saarbrücken, Germany and
[3] Department for Computational Biology and Applied Algorithmics, MPII, Saarbrücken, Germany and
[4] Science for Life Laboratory, Sweden and
[5] Department of Computer Science, Princeton University, Princeton, New Jersey, The United States of America
∗ **Contact**: murray.patterson@unimib.it, bonizzoni@disco.unimib.it

What is haplotype assembly?

In a diploid (polyploid) organism, the *haplotypes* are the different *copies* of its genome. Haplotype assembly is the process of reconstructing these from sequencing reads data.

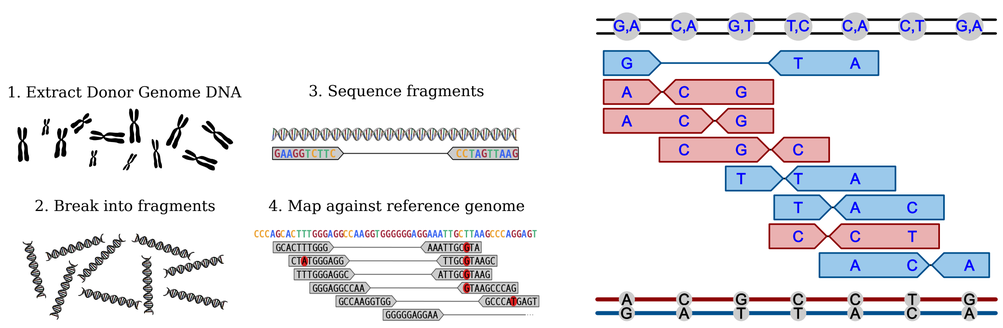

Figure 1: *Produce aligned reads (left) and use them to determine a pair of haplotypes (right).*

This work

Long reads, which are nowadays cheaper to produce and more widely available than ever before, have been used to reduce the fragmentation of assembled haplotypes since their ability to span several variants along the genome. Here, we propose a new method for assembling haplotypes which combines and extends the features of previous approaches to deal with *error-prone* long reads at *higher coverages*. In particular, our algorithm is able to dynamically adapt the estimated number of errors at each variant site, allowing a significant reduction in the required computational resources, allowing to consider datasets of higher coverages. The algorithm has been implemented in a freely available tool, **HapCHAT**: **Hap**lotype Assembly **C**overage **H**andling by **A**dapting **T**hresholds (`http://hapchat.algolab.eu` under the GPL). An experimental analysis on sequencing reads with up to 60× coverage reveals accuracy improvements achieved by considering a higher coverage with lower runtimes.

Background

The MEC. We model the haplotype assembly problem as the *minimum error correction* (MEC) problem [1]: given a *fragment matrix* M in $\{0,1,-\}^{n\times m}$, determine the minimum number of *corrections* (0 to 1, or 1 to 0) to M such that there exists a pair h_1, h_2 of *haplotypes* in $\{0,1\}^m$ where each row of M *agrees* with at least one of the haplotypes. A pair v, w of m-vectors *agrees*, $v \approx w$, when for every index $i \in [1,m]$, $v[i] = w[i]$ or one of $v[i], w[i] = -$.

The k-cMEC. More precisely, we model the problem as the *k-constrained MEC* (k-cMEC), a restricted version of the MEC where the number of corrections per column of M is bounded by k [2]. A *k-correction* C to a column j of M is then a correction to a particular subset of at most k rows in column j. In [2], the k-cMEC is solved by maintaining a table $D_j[C]$, which stores the minimum number of corrections needed to obtain a particular k-correction C to column j, in the *dynamic programming* recurrence,

$$D_j[C] = \min_{C' \in \delta_{j-1}, C \approx C'} \left\{ D_{j-1}[C'] + d(M_j, C) \right\},$$

where M_j in $\{0,1,-\}^n$ is the j-th column of M, and δ_j is the set of all k-corrections of M_j.

Methods

Here we propose a *preprocessing* step, and a way to cope with highly erroneous sites.

Merging. Given a pair r, r' of reads which agree on y sites and disagree on x sites, the probability of obtaining r, r' under the hypothesis that they originate from the *same* (resp., *different*) haplotype(s) is

$$p_s(r,r') = (1-p)^{2y} p^x (1-p/3)^x, \qquad p_d(r,r') = p^y (1-p/3)^x (1-p)^x.$$

We then build clusters of reads such that $p_s/p_d \geq 10^6$ for each pair of reads in a cluster, and then merge each cluster into a "super-read". Note that there is a merging heuristic in [3], where they determine when to merge a pair of reads, while we analyze all pairs of reads to determine which *sets* of reads to merge.

Adaptive k-cMEC. Here, we modified the above recurrence to allow the amount of corrections for a column to exceed k. Given C_k, a correction for a particular column j of M with *exactly* k corrections, we set $z_0 = k$, and then increment z (beyond k) as follows,

$$z_i = z_{i-1} + \lfloor \log_2(z_{i-1}) + 1 \rfloor, \qquad \text{then} \qquad k^* = \min_{i : D_j[C_{z_{i-1}}] \neq \infty} \{z_i\},$$

if $i > 0$, where $D[\cdot] \neq \infty$ means that it is a feasible correction. The new set of possible corrections of column j is then

$$\delta_j = \{C_k : 1 \leq k \leq k^*\},$$

which, by construction, guarantees the computation of a (feasible) solution.

Results

The *Genome in a Bottle* (GIAB) consortium has released publicly available high-quality sequencing data for seven individuals, using eleven different technlogies [4]. Here, we study the Ashkenazim individual NA24385 and the individual NA12878. As a phasing benchmark, we use the latest high confidence phased VCFs of the GIAB.

Real data. We use the sets of aligned *Pacific Biosciences* (PacBio) reads of the GIAB for chromosome 1 of NA24385, and chromosomes 1–22 of NA12878. Because chromosome 1 of NA24385 has average coverage of approximately 60×, we randomly downsample (with `DownsampleSam` of Picard) to average coverages 25×, 30×, etc. — no automosome of NA12878 has average coverage more than approximately 40×.

Simulated data. We first obtain a pair of "true" haplotypes by incorporating each of the pair of phases (of variants) of the benchmark into the reference genome (hg37), and then simulate reads off of these using PBSIM [5], and then align them using BWA-MEM (0.7.12-r1039) [6].

Experimental setup. We run HapCHAT with maximum coverage 30, comparing to two other dynamic programming approaches, WhatsHap [7] and HapCol [2] at maximum coverages 20 and 25, respectively. We also compared to HapCUT2 [8], Probhap [3], RefHap and FastHare.

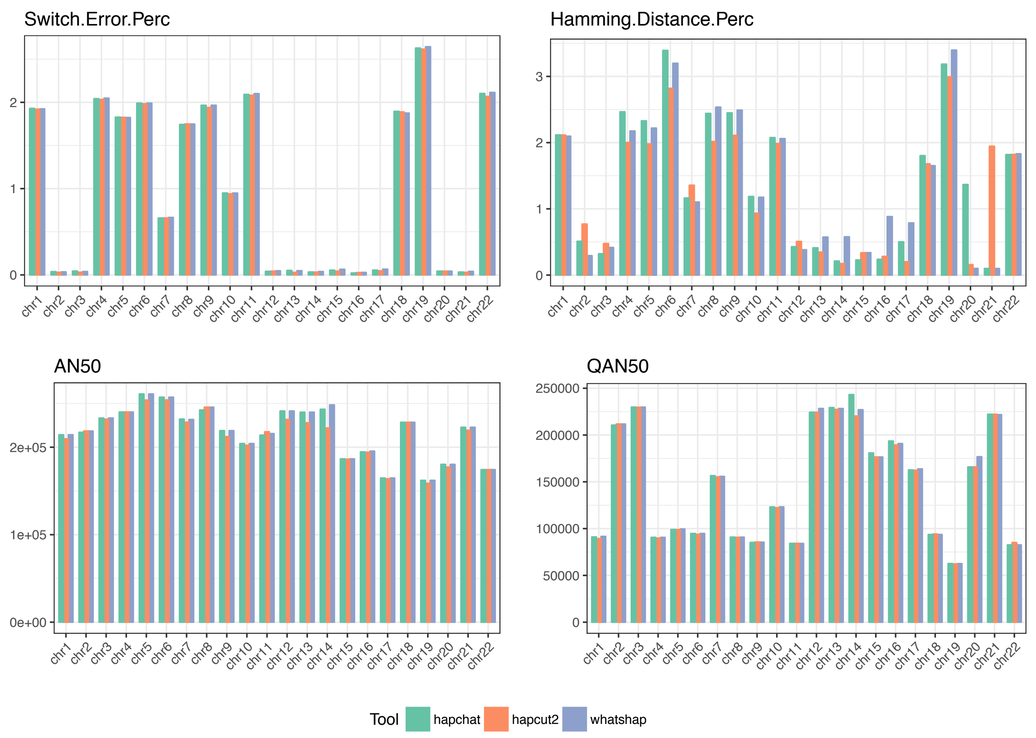

Figure 2: *Accuracy / recall of HapCHAT, HapCUT2 and WhatsHap on chromosomes 1–22 of the individual NA12878.*

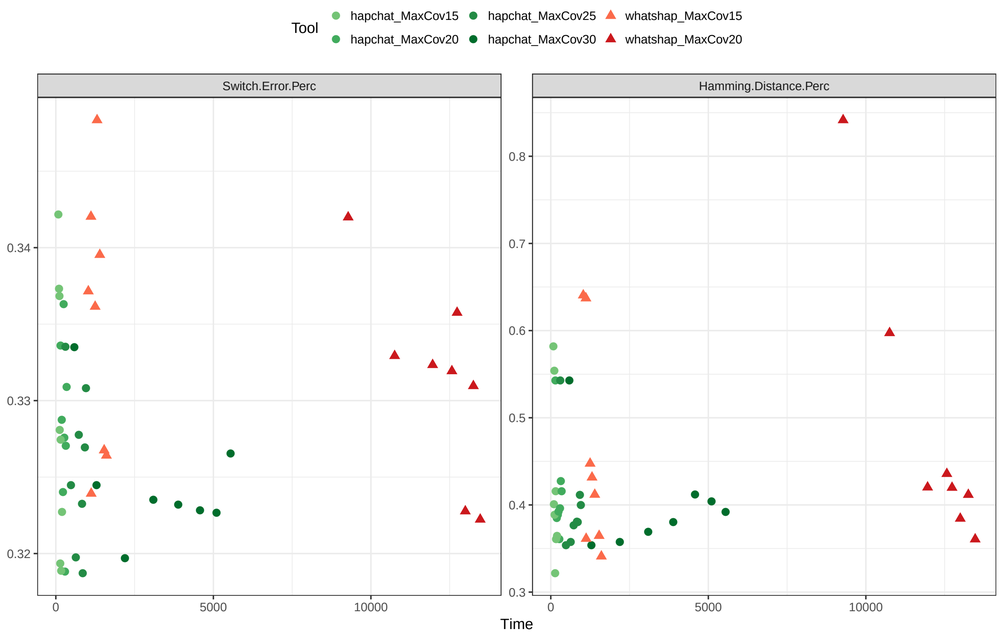

Figure 3: *Accuracy as a function of runtime for HapCHAT and WhatsHap at different maximum coverages on the Ashkenazim dataset. For each tool / maximum coverage, there is a point for each of the 8 possible values of the average coverage.*

References

[1] R. Lippert et al. **Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem.** 2002.

[2] Y. Pirola et al. **HapCol: accurate and memory-efficient haplotype assembly from long reads.** *Bioinformatics*, 2016.

[3] V. Kuleshov. **Probabilistic single-individual haplotyping.** *Bioinformatics*, 2014.

[4] J.M. Zook et al. **Extensive sequencing of seven human genomes to characterize benchmark reference materials.** 2016.

[5] Y. Ono et al. **PBSIM: Pacbio reads simulator–toward accurate genome assembly.** *Bioinformatics*, 2012.

[6] H. Li. **Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM.** *arXiv e-prints*, 2013.

[7] M. Martin et al. **WhatsHap: fast and accurate read-based phasing.** *bioRxiv* 085050, 2016.

[8] P. Edge et al. **HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies.** 2016.