A NP-Hardness of MSMTP

Let $G = (V, E, w)$ be a connected, undirected, and weighted graph with weight function w. Call a partition $V_1, \ldots, V_k$ of the vertices into k disjoint sets valid if each $G(V_i)$ is connected. Then the min-sum max tree partition (MSMTP) problem is to find a valid partition of V into k sets that minimizes $SMTP_G^k(V_1, \ldots, V_k)$.

Theorem 3. *MSMTP is NP-Hard for $k \geq 3$.*

Proof. We take a reduction from graph coloring.

Let $G' = (V, E', w')$ be a complete graph where the vertices of G' are the same as G. Let the weight of any edge $e \in E'$ to be 2 if $e \in E$, the original graph, and let it be 1 otherwise.

Let $V_1, \ldots, V_k$ be a (valid) partition of V that solves $MSMTP$ for G'. Note that none of $V_1, \ldots, V_k$ are empty, otherwise we could move a single vertex to the empty set and it would have a lower $SMTP$ value for this graph. Therefore the total number of edges over all spanning trees of $G'(V_1), \ldots, G'(V_k)$ is $\sum_{i=1}^{k} |V_i| - 1 = |V| - k$. I claim that a k-coloring of G exists if and only if $SMTP_{G'}^k(V_1, \ldots, V_k) = |V| - k$.

If $SMTP_{G'}^k(V_1, \ldots, V_k) = |V| - k$, then the maximum spanning tree for each subset only contains edges with weight 1. In particular, this means that no subgraph $G'(V_i)$ has an edge with weight 2, so there are no edges between any vertices of V_i in G, otherwise the weight 2 edge would be included in the max spanning tree. Thus the partition $V_1, \ldots, V_k$ gives a $k-$coloring of G.

For the other implication, clearly if a $k-$coloring exists for G, then we can find a partition $V_1, \ldots, V_k$ such that $G'(V_i)$ only has edges of weight 1 between vertices. Then $SMTP_{G'}^k(V_1, \ldots, V_k) = |V| - k$ follows.

Therefore any algorithm which solves MSMTP also decides if G has a k-coloring, which is NP-Complete for $k \geq 3$.

B Iterative refinement algorithm

We present the pseudocode for the iterative UPEM maximization procedure that follows local clustering in Algorithm 2. In lines 4-15, we check how swapping vertices between partitions affects the overall UPEM score and take a fraction of the best swaps in line 15. We then execute the swaps and check if the UPEM score has increased. If it has not, we terminate the algorithm, otherwise we continue until n iterations has passed. We take $n = 10$ in practice, and note that almost always the algorithm terminates before 10 iterations pass.

C Correcting erroneous haplotype blocks by filling

After obtaining a set of partitions $P_1, \ldots, P_l$ of subsets of reads $B_1, \ldots, B_l$ according to the local haplotyping procedure described in Section 14, we correct poor clusters by the following procedure. After computing the UPEM scores for

Algorithm 2: Iterative refinement of UPEM score

Input : Partition $P = \{S_1, \ldots, S_k\}$, iterations n
Output: A modified optimized partition $P = \{S'_1, \ldots, S'_k\}$

1 **for** *i=1 to n* **do**
2 $OldScore \leftarrow UPEM(P)$
3 $L \leftarrow \emptyset$
4 $S \leftarrow \bigcup_{i=1}^{k} S_i$
5 **for** S_i *in* P **do**
6 **for** r *in* S_i **do**
7 **for** S_j *in* P, $S_j \neq S_i$ **do**
8 Compute change in $UPEM$, $\Delta(r, S_j)$ by moving r from S_i to S_j
9 **if** $\Delta(r, S_j) > 0$ **then**
10 $L \leftarrow L \cup \{(r, S_i, S_j, \Delta(r, S_j))\}$.
11 **end**
12 **end**
13 **end**
14 **end**
15 $L \leftarrow$ reverse sort L by $L[j] \rightarrow \Delta(r, S_j)$
16 **for** $k = 1$ *to* $\lceil \frac{|S|}{n} \rceil$ **do**
17 $(r, S_i, S_j, \Delta(r, S_j)) = L[k]$
18 $S_j \leftarrow S_j \cup \{r\}$
19 $S_i \leftarrow S_i \setminus \{r\}$
20 **end**
21 $NewScore \leftarrow UPEM(P)$
22 **if** $NewScore < OldScore$ **then**
23 Reverse the moves made in Lines 16-20
24 Return P
25 **end**
26 **end**
27 Return P

every partition, we use a simple 3.0 IQR (inter-quartile range) outlier detection method for the distribution of UPEM scores. For an outlying partition P_i of the read set B_i, if a partition P_{i-1} is not an outlier, we remove P_i and extend P_{i-1} to include B_i. To do this, we run a subroutine of Algorithm 1 where we skip the clique finding procedure and instead treat the partition P_{i-1} as the initial clusters. We then run lines 9-12 of Algorithm 1 where in line 9 we iterate over $v \in B_i \setminus B_{i-1}$ instead.

D Genotype polishing using VCF

We constrain the haplotypes using the VCF as follows. For every variant indexed over $1 \leq i \leq m$, let $c(i, j, a) \in \mathbb{R}$ be a value representing the confidence of

calling allele a at index i for the haplotype represented by $P[j]$. We produce $k-$haplotypes according to Algorithm 3.

Algorithm 3: Polishing output haplotypes using genotype information.

Input : Partition P, Genotyping information
Output: $k = |P|$ haplotypes $H_1, \ldots, H_k \in \{0, 1, 2, 3, -\}^m$

Initialize $H_1, \ldots, H_k$, $H_i[n] = -$ for all $1 \leq n \leq m$
for $i = 1$ *to* m **do**
  for $j = 1$ *to* k **do**
    for $a \in \{0, 1, 2, 3\}$ **do**
      $L \leftarrow c(i, j, a)$
    end
  end
  $L \leftarrow$ reverse-sort L
  for $c(i, j, a) \in L$ **do**
    if $|\{H_n : H_n[i] = a\}| <$ *# of a in VCF file and* $H_j[i] = -$ **then**
      $H_j[i] = a$
    end
  end
end

For the function $c(i, j, a)$ describing the confidence for calling allele a at position m for haplotype j, we choose the function

$$c(i, j, a) = \frac{|\{r \in P[j] : r[i] = a\}|}{|\{r \in P[j] : r[i] \neq a\}| + 1}.$$

Note that H-PoP uses the same method for polishing, but they choose a confidence function that is a difference instead of a ratio. A ratio does a better job because for a particular haplotype if there are 100 reads that have allele 1 at position 1, and 50 reads that have allele 2, we believe that this is a less confident call than a haplotype with 50 reads with allele 1 and 0 reads that have other alleles.

E MSMTP vs MEC

We ran flopp on four different simulated datasets (see Section 3.2) and calculated the SMTP score with the weight function $w(r_1, r_2) = d(r_1, r_2)$, see Equation 1, and the MEC score for each local partition.

We varied the coverage between 10x to 20x for a simulated 4x ploidy genome. We also varied the length of the local partition blocks by manually changing the parameter b mentioned at the end of Section 2.3 over three different values for each different data set to investigate how the size of the local clusters affects the SMTP and MEC relationship. The results are shown in Figure 6

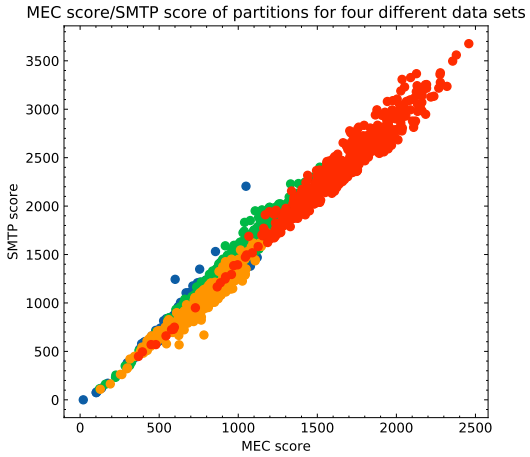

Fig. 6. A plot of the SMTP vs MEC score for every local cluster generated by flopp over four different simulated data sets across different coverages with each color representing a distinct data set.

F WHP Results on 3x ploidy data

We ran WhatsHap Polyphase (WHP) on the simulated genomes as described in Section 3.2. We ran WHP with default settings. In particular, the block-cut sensitivity setting which determines the contiguity of the blocks was set at the default value of 4. There were around 67000 variants in the contig of 3.02 Mb; The N50 over all iterations, coverages, and read types of WHP's output never exceeded 20. The results are shown in Figure 7.

We found that WHP takes a conservative approach as $> 15\%$ of the variants were not called by WHP, even on 20x coverage per haplotype data. This is included as part of the Hamming error rate in our tests. In [1], the authors also identify that WHP tends to be extremely conservative with respect to cutting haplotype blocks and calling variants. One of the implications is that the Hamming error rate for WHP on our data set is significantly higher than that reported in their study [33]. However, there are several differences in the settings for our respective studies, which we hypothesize may contribute to that disparity. For example, there are relatively high rates of heterozygosity in our test data set (45 bp between SNPs on average), which is less the case on the reported test results for WHP [33], as they test on artificial human polyploid chromosomes created by combining different human chromosomes; it is well known that human chromosomes have much lower rates of heterozygosity than potato genomes which have variants on average $<$ 50bp apart [35].

Another difference is in the simulator we used to generate long-read data for our benchmarks (PaSS [39], NanoSim[37]), which may differ in error profile than the data in their experiments.

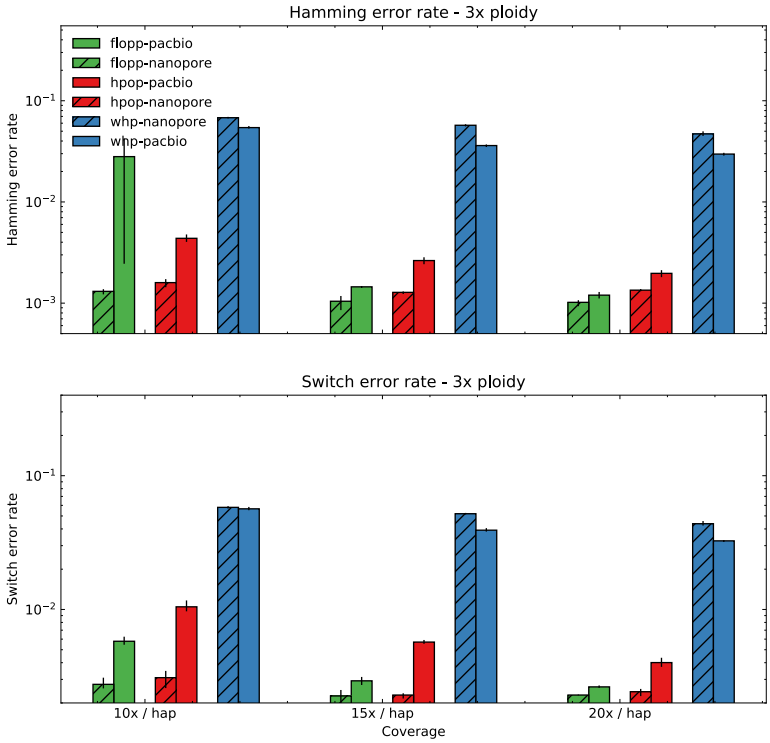

Fig. 7. SWER and Hamming error rates of flopp, H-PoPG, and WHP on 3x ploidy data set. The generation of the data set is described in Section 3.2. The error bars represent the lowest and highest values for the metric over three iterations. The Hamming and switch error rates are averaged over all broken blocks for WHP.

Finally, we noticed that on polyploid genomes that have very similar haplotypes, i.e. genomes which exhibit heavy amounts of collapsing, the run time of WHP is much faster than on our simulated data set. Regardless, we believe that more testing is needed to identify the regimes in which the various types of long-read based phasing algorithms perform optimally.

G Results on simulated data sets with different rates of heterozygosity

We include two additional tests based on the simulated data set described in the paper under the same testing conditions with the rates of heterozygosity changed from 45 bp to 90 bp and 135 bp between SNPs on average (mean). The results are shown in Figure 8 and Figure 9. At higher rates of heterozygosity, flopp has a higher likelihood of a major switch error occurring which may result

in a high hamming error rate. However, the switch error rate for flopp is still better than H-PoP, showing that at on a local scale, flopp still performs well.

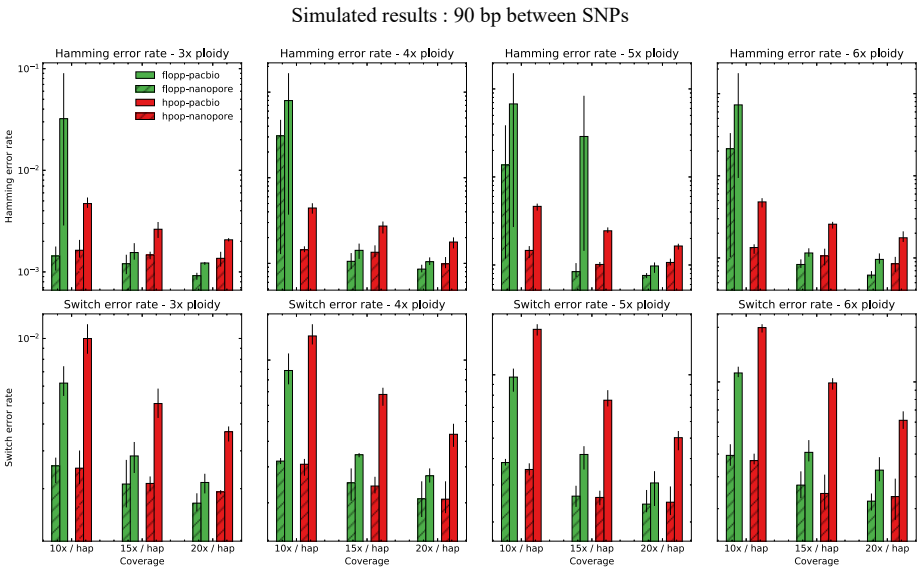

Fig. 8. The mean switch error rate and Hamming error rate from testing on simulated data sets as described in Section 3.2 except with 90 bp between SNPs on average (instead of 45bp). The error bars represent the lowest and highest values for the metric over three iterations.

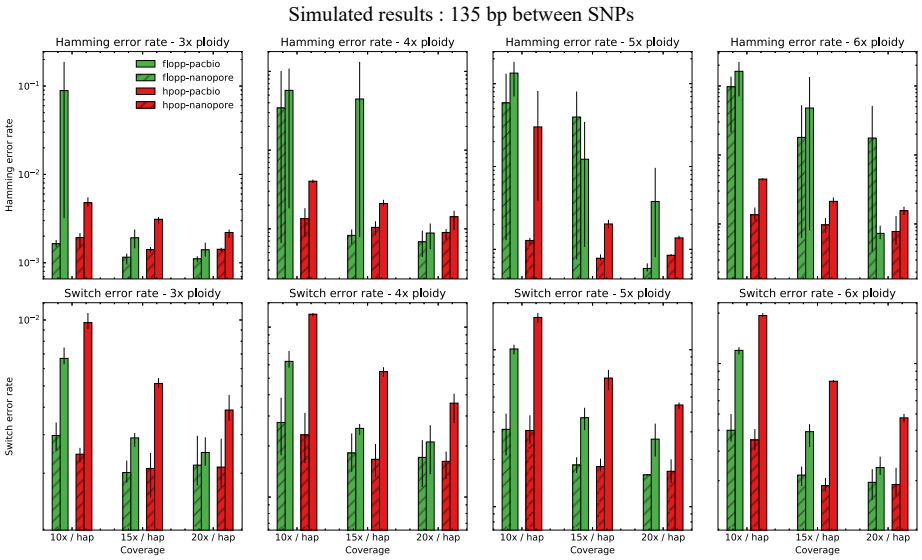

Fig. 9. The mean switch error rate and Hamming error rate from testing on simulated data sets as described in Section 3.2 except with 135 bp between SNPs on average (instead of 45bp). The error bars represent the lowest and highest values for the metric over three iterations.