Summary of the MIHA method comparison

**Performance of multiple individual haplotype assembly algorithms**

Shuying Sun, Sherwin Massoudian, Allison Bertie Johnson

Department of Mathematics, Texas State University, San Marcos, Texas 78666, USA

Department of Statistics, Texas A&M University, College Station, Texas 77843, USA

A haplotype is an ordered of alleles on part or all of a chromosome. Inferring haplotypes is useful for understanding haplotype block structures in a genome and finding genes associated with diseases. Haplotype assembly means to infer haplotypes from DNA sequencing data. With advanced sequencing technologies, many large datasets have been generated. The sequencing datasets of multiple individuals are often complex as far as the biological information and the sequencing technology are concerned. This complexity makes haplotype assembly a challenging task. A number of software packages have been developed to overcome this challenge. However, users often do not know how well each software performs and which one to choose for their own research problems. In order to figure out these questions, we have conducted a comprehensive comparative analysis for a few algorithms including DBM, HapSeq2, and WhatsHap. Our comparison is conducted using publicly available sequencing data. In this poster, we will show our detailed analysis results with a focus on pairwise comparison of different algorithms.

**Introduction/Background**

|  |
| --- |
| **Dr. Sun’s thoughts/ideas. Please keep them, that is, PLEASE DO NOT delete them!!!!**   1. Some genetic backgrounds, e.g., what is SNP, what is haplotype? What is haplotype assembly? Why we need to the haplotype assembly? 2. Briefly describes how each algorithm works using WORDS (not scripts, read the papers and summarize)   DBM: DBM takes sequencing read count data at each SNVs …. …  HapSeq2: HapSeq2 is developed based on …  Whatshap: Whatshap used a w-MEC method/algorithm …. |

**Comparative analysis**

|  |
| --- |
| **Dr. Sun’s thoughts/ideas. Please keep them, that is, PLEASE DO NOT delete them!!!!**   1. Description about the data, 826-850 datasets, alignment 2. Provide a workflow (chart, check Oct 2019 prepared by you/Sherwin prepared and before Bertie merged it with hers) 3. Summarize the input and output format **using WORDS** (plus/including some tables or perl/R code if you think that is helpful) 4. Explain how you prepare for the input files to run each software **using WORDS** 5. Explain how those 120 SNVs and those 628 SNVs are selected (**using Words,** may add script if you want)   **Show comparison results (e.g., tables and add interpretation to the tables).** |

**Comparative analysis**

**Introduction**

Human DNA contains sites in which a single DNA base differs from the reference genome. These positions can be observed at the population level and are call Single Nucleotide Polymorphism (SNP) or Single Nucleotide Variants (SNV). A haplotype is an ordered set of nucleotides or alleles inherited from either a mother or father. Important genetic variants are studied using haplotypes, because certain diseases and traits in a population are associated with specific haplotypes. Haplotype assembly is the reconstruction of haplotypes from DNA sequencing reads.

Haplotype assembly is an important task in the study of disease association and evolutionary genetics. Many of these haplotype reconstruction algorithms are developed using computational methods. However, only a few clearly indicate that it is developed for or support using multiple individual haplotype assembly. This analysis compares three multiple individual haplotype assembly algorithms: WhatsHap (2015.Patterson.Whatshap), Hapseq2 (2013.Zhang.Hapseq2), and DBM (2013.Zhang.DBM) on the agreement of their phased haplotypes.

Each software uses a unique algorithm. For example, WhatsHap solves the Minimum Error Correction problem in linear time by using an exact dynamic programming approach, assuming bounded coverage (*2015.whatshap, page4, left, 2nd paragraph*). Minimum Error Correction (MEC) aims at reconstructing two haplotypes by applying the minimum number of base corrections given a set of fragments is a prominent computational problem for haplotype assembly and is computationally hard to solve. WhatsHap claims to be the first haplotype assembly approach to allow for taking both increasing read length and sequencing error information into account. WhatsHap claims to handle datasets with coverage up to 20X, and that 15X coverage is generally enough for reliably phasing long reads, even at significantly elevated sequencing error rates (*2015.whatshap, abstract*). Furthermore, it claims that when comparing with state-of-the-art statistical phasers its switch and flip error rates of the haplotypes output are favorable. Lastly it claims that it is unique in its ability to further improve accuracy when multiple related samples are provided, allowing it to combine read-based with pedigree-based phasing.

HapSeq2 uses a Hidden Markov model that incorporates the joint distribution of two or more sites covered by a read over an arbitrary distance. It is a fully probabilistic model for joint genotype calling and haplotype phasing and its algorithm also incorporates a Markov Chain Monte Carlo (MCMC) model. HapSeq2 claims to integrate the elements of population-haplotype likelihood and read-haplotype likelihood, each capturing complementary haplotype information. It also claims that in each iteration, it uses a Hidden Markov model to jointly perform genotype calling and haplotype phasing, and then use the Metropolis–Hastings algorithm to sample haplotypes of each individual according to the likelihood based on sequencing reads and reference haplotypes (*2013.Zhang.Hapseq2, page 2, left bottom*).

The DBM algorithm is fully probabilistic and produces consistent inference of genotypes, haplotypes, and recombination probabilities by using a dynamic Bayesian Markov model (DBM) for simultaneous genotype calling and haplotype phasing in low-coverage NGS data of unrelated individuals. Its algorithm is also a variant of infinite-state Hidden Markov model and a non-parametric Bayes model equipped with Markov structures. One problem in Hidden Markov models is to determine the number of states. Although too many states may reduce inference efficiency and over fit the data, too few states may not be sufficient to capture all the information in the data and thus loose power. DBM allows the number of states to vary across SNPs and uses a non-parametric Bayesian process to dynamically infer the number of states across SNPs. It claims to have great flexibility to fit regions with either simple or complex structures while it avoids over fitting the data via Bayesian regularization (*2013.Zhang.Hapseq2.page2.right top*).

**Sept 5, 2020 notes:**

1. Need to add a bit more on HapSeq2 and DBM. Read their discussions.
2. About the logic of the first paragraph: Better to (1) first introduce the basic concepts, SNP, haplotype, then (2) haplotype assembly, and why it is important to have haplotype, (3) why is it important to do haplotype assembly? (4) What is the challenges of haplotype assembly? 🡪 record the 1st paragraph a bit. [A], [B], [3], make sure the idea flows
3. Why is it important to compare these MIHA algorithms?

*It is not clear how well each algorithm performs. Check if the whatshap paper shows the results of running it on multiple-individuals.*

1. See the example below for detailed reference/citation

*Disease has been at the center of human mortality long before it was a comprehendible concept. To date there is still much to be discovered in terms of disease processes, inheritance and mediation. DNA sequencing has played a pivotal role in human genomics and disease research. Rapid advancement(s) in sequencing technologies have aided in cost effective, high throughput sequencing data which can allow for the identification of human genetic variants that may affect health, disease, and an individual’s response to medications (****RefHap, Duitama et. al 2010, page 1, sec. intro, 3rd sentence****). One crucial computational task associated with reconstructing an individual’s genome from a set of reads is haplotype assembly, the procedure of inferring two DNA sequences from the set of reads (****FastHap, Mazrouee et. al 2014, page 1, sec.1 intro. last sentence****).*

**Aug 28, 2020 SS notes:**

1. The above two paragraphs are very brief or “dry”, you may expand it a bit to make it “juicy”. In order to have a “juicy” (rich/expanded) writing, please read the introduction of at least 3 other papers (e.g., DBM, HapSeq, HapSeq2, or Whatshap) to see how they write the Introduction and background section.
2. After you read each paper carefully, please also summarize them using a few sentences:

(1) How does this algorithm (e.g., DBM) work,

(2) What is the advantage of this algorithm as claimed in the paper,

(3) What is the disadvantage/limitation as stated in the paper (especially the *discussion* session) or mentioned/criticized by the other paper.

**3.** SS explanation notes on Aug 28,2020

**Haplotype inference (phasing)**: infer haplotypes from (or based on) genotypes (SNPs), no sequencing reads involved**.**

**Haplotype assembly (*some* paper or *some* algorithm, say phasing):**

1. Infer/reconstruct haplotypes from (or based on) sequencing reads. Sequencing reads 🡪 SNVs (including known SNPs + unknown/new SNVs) 🡪 haplotypes.
2. Some HA algorithms may not have the above SNVs step: sequencing reads 🡪 haplotypes

**Body**

DNA sequencing reads of five colon cancer samples (2012.Adams) are used to compare the pairwise agreement of the three haplotype assembly algorithms. The raw reads are aligned to the reference genome HG38 using Burrows-Wheeler Aligner version 0.6 (2009.Li.BWA). SNVs are called based on the aligned reads, and these SNVs are then used for haplotype assembly. Specifically, the SNVs are called using the library BCFTools 1.9 by setting an option of its function mpileup to only output the SNV positions. The input for BCFTools mpileup is a text file that lists the locations of each aligned sequence in BAM format, and mpileup produces a list of SNVs in variant call format or VCF. These SNVs were then used to assemble haplotypes with each haplotype assembly package requiring different levels of input preparation.

Among all three HA algorithms, WhatsHap is the simplest one in preparing input as it only requires the VCF file produced by BCFTools. It outputs a VCF file with haplotype blocks appended at the end of each phased row. HapSeq2 needs two input files *sites.txt* and *counts.txt*, the former includes the following columns: Position, Reference Allele, and Alternate Allele. The latter is prepared using an executable included in the HapSeq2 package called bam\_parser and generates a file with reference and alternate allele counts at each position for each sample. See examples of HapSeq2 input shown in Table 1 and Table 2. HapSeq2 phases two haplotypes for each sample. An example of an individuals’ haplotypes is shown in Table 3.

**Sites.txt**

47754 T C

47771 C T

47774 A C

1183069 G A

1183111 G T

1183505 C A

**Table 1: The hapseq2 input example file 1, sites.txt. The 3 columns are chromosome, reference allele, and alternative allele.**

**counts.txt**

**1 1 0 0 1** 1 1 39 19 0 24 0 42 14 15 0 23 0 138 0 111 0 121 22 49 0 0 1 4 88 40 0 30 0 22 3 5 0 54 0 20 5 0 170 0 18 31 158 78 4 14 5 2 80 63 0 2 73 0 51 0 61 0 16 16 1 1 1 1 0 56 0 6 0 3 17 30 0 1 0 0 0 0 0 0 39 0 0 186 0 8 0 17 148 50 3 67 8 2 0 78 10 29 13 3 0 180 0 15 177 23 0 6 0 0 0 12 0 12 1 230 0 52 2 153 0 192 0 0 0 228 0 3 0 3 0 13 0 6 0 0 1 150 0 123 1 0 1 0 0 4 0 6 0 4 0 0 0 0 0 2 0 4 0 2 0 2 31 14 0 7 0 10 0 11 0 18 2 0 0 1 0 2 0 19 54 211 0 2 0 12 131 36 0 139 0 2 5 0 15 1 0 0 0 13 0 17 20 55 0 159 0 94 0 6 0 3 0 0 4 1 3 0 71 19 87 19 0 119 77 26 137 46 10 1 20 9 0 0 0 2 0 2 0 0

**Table 2: The hapseq2 input example file 2, count.txt. The first five columns represent the family id, the individual id, the father id, the mother id, and the gender of that individual. Since HapSeq2 can only handles unrelated individuals at this moment, the father id and the mother id should be 0 in the count file. For example, ‘ 1 1 0 0 1” means the family ID is 1, the individual ID is 1, and the individual is female. For the subsequent columns, each pair of columns represents the read counts for allele 1 and allele 2 at that site. For example, “1 1 39 19” is a count of one reference allele and one alternate allele counted at the first SNV, with 39 reference allele and 19 alternate allele and the second SNV.**

**phased.chr10.hapseq.txt**

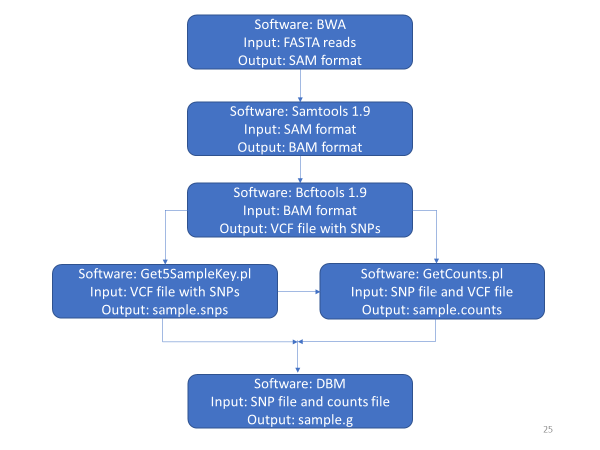
1->1 HAPLO1 tcaggctttgtgcatcggtttcgcgattcgccctcgcatcaccacccccctgtcctcgtcgaggtgccgcctgtgcggccagcgtcctcggccgctaggccacccctgggtccgacggcaagacaagtcagctcggccccatctccaggataggctgctggagtctccggaaactgttgctccgaatatcgcgtagggggcaatccgtccaacgccccggccgccccctgcatgcccgcgttacctacttacgctagttcccattccgtcaggcgccgtcagtccgcgtatagcataccctcgaggcccactacgggggtccacttaaaaccgagccaaaactgcccgtgttttccggcacatgagtccataatggccaatcgcgccgtaggccaggaggcgtcaaagtagttggggatacgacgaccgggcgtattacccctcacagccgcataagctcggggacaaaagagcccattggtgtcgcggccacgcgccctgcctgtcttcctctgcgaagggcccccggggggacgtcagacaccagcaccttactaatttgtggccccccttcatgagggccgtggggtacgagtcctacggccggctccaaaacgttcaggtatta

1->1 HAPLO2 tcaagccccgtgcatcggattgtgtggccgccctcacgcactcacccctttgcattggtcggggtgccgccggtgcggccggggtacctagtcgctaggccactgttgggtacgactgcccgacaagatcgctcgggcctgggatcgggatgcggcgctcgtctctccggaaactggggccccgaatatcacggggggcgcgattcgttaccacgaaatcctacgccccacacacctgcgttacctacttacgctagttccagttctgacagcggccgccagtcagtgtccggcataccctcgagcctcactacgggggtccatccccagccgaggcaaaaccgcccgtgttgtccggcgcatgaggccatcccggtcaatcgttccttgagccgggaggcttctaagtcgcggttgctacgagaaccgggcgtattacccatcgcagtcgcctaccctcggggcagccagcgcctactggctccgcgaccacgcgccgtgtttttcttcctctgcgacaggcacttgggggtccgtcagaaaccaacactctcctgcgttgtggccccctcccaggaaggccgtggtgtacgagtcgtacggccggcttcaaaccgcgtaggcttgg

**Table 3: HapSeq2 inferred pair of haplotypes for an individual. This file has the same format**

**as Thunder. The two columns are sample ID and haplotype.**

A workflow for phasing haplotypes using DBM is illustrated in figure 1. DBM needs two input files to phase haplotypes, these files have the extensions snps and counts with the same file name (e.g., chr10.snps and chr10.counts). The former has the columns ID, Chromosome, Position, Quality Score, Reference Allele, and Alternate Allele. The latter has a row for each SNV with the count of reference and alternate alleles in each sample, which can be found in the VCF file produced by BCFTools. A sample of the two DBM inputs are in table 4 and table 5. The output of DBM has columns ID, Chromosome, Position, and two columns of haplotype information for each sample provided. An example of haplotypes phased by DBM is found below in table 6.

****

**Figure 1: DBM workflow. This includes aligning sequences, calling their SNV, preparing both DBM input files (sample.snps and sample.counts), and phasing haplotypes.**

**chr10.5sample.snps**

1 chr10 47754 57.5615 T C

2 chr10 47771 49.6775 C T

3 chr10 47774 46.8073 A C

4 chr10 1183069 999 G A

5 chr10 1183111 17.1126 G T

6 chr10 1183505 3.02212 C A

**Table 4: DBM input example file 1. This input file has columns ID, Chromosome, Position, Quality Score, Reference Allele, and Alternate Allele.**

**chr10.5sample.counts**

I SID I1 I1 I2 I2 I3 I3 I4 I4 I5 I5

M 1 7 0 1 0 7 0 1 4 5 2

M 2 5 0 1 0 10 0 3 4 2 2

M 3 5 0 1 0 10 0 3 4 3 2

M 4 7 1 5 0 0 19 8 0 14 6

M 5 17 2 12 0 29 0 8 0 27 0

**Table 5: DBM input example file 2. This input file has a row for each SNV with the count of reference and alternate alleles in each sample. The first column “I” is the Family ID and M is selected as a family ID in this example. The second column “SID” means SNV ID which is assigned in the chr10.5sample.snps file.**

**I1 and I2 refer to individual 1 and individual 2, each individual has columns for reference and alternate allele counts.**

**chr10.5sample.g**

ID CHR POS I1 I1 I2 I2 I3 I3 I4 I4 I5 I5

1 chr10 47754 T T T T T T C T T C

2 chr10 47771 C C C C C C T C C T

3 chr10 47774 A A A A A A C A A C

4 chr10 1183069 A G G G A A G G G A

5 chr10 1183111 G G G G G G G G G G

**Table 6: DBM output example file. This file has phased haplotype and includes the following columns: SNV ID, Chromosome, Position, and two columns of haplotype information for each sample provided.**

The haplotypes generated for the five samples by each haplotype assembly library were compared on pairwise agreement using Perl and R. A Perl script called /home/s\_m774/Haplotype-Research/getGenotypeAgreement.pl was created to complete the pairwise comparison of genotypes. The script getGenotypeAgreement.pl takes two genotypes and calculates strict and non-strict agreements, the latter awarding half points to partial matches between the two genotypes. Strict agreement counts 1 when the genotype of two HA algorithms are the same (e.g. “AC” in both WhatsHap and HapSeq2). While non-strict agreement counts as 1 when both alleles are the same and counts 0.5 when at least 1 of the two alleles are the same (e.g. “AC” for WhatsHap and “AA” for hapseq2). For example, a 95 in table 7 column 1 means among the 120 SNVs, 95 genotypes are the same when comparing DBM with WhatsHap. An R script is used to calculate pairwise agreements between the haplotypes of each package similar to the genotype agreement but allowing for a switch in the blocks. Haplotype comparisons are shown in row 3 of table 7, for example 82.5 in Table 7 row 3 means an 82.5 agreement score allowing for switched blocks. An example usage of getGenotypeAgreement.pl is provided below:

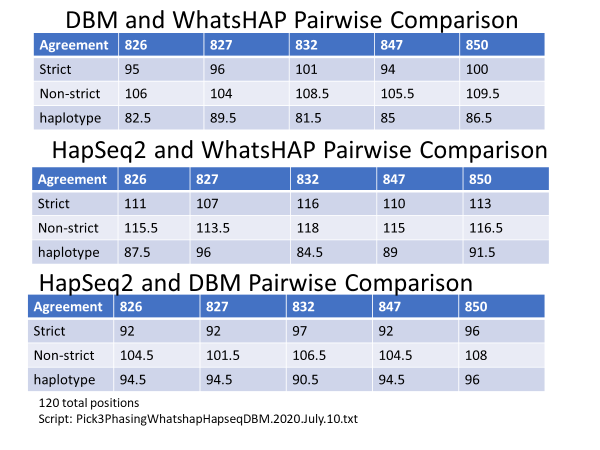
**getGenotypeAgreement.pl**

perl /home/s\_m774/Haplotype-Research/getGenotypeAgreement.pl dbm.genotype.826.chr10.txt hapseq2.genotype.826.chr10.txt

Strict genotype agreement: 92/120

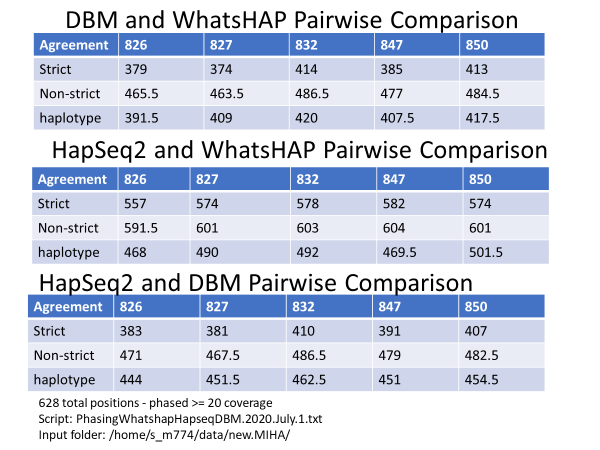
Non-Strict genotype agreement: 104.5/120

Two subsets of SNVs are selected from the five-sample dataset and phased to compare the agreements of the three packages. The first dataset createdis a set of 120 SNVs which are heterozygous sites and have at least 2X reads on three or more samples. The results of these pairwise agreements are shown in the table 7 below. This table shows that the WhatsHap and HapSeq2 algorithms have the greatest agreements in their haplotypes. While DBM produces the most conflicts when compared with either WhatsHap or HapSeq2. The second dataset are 628 SNVs that have a minimum of 20 depth across five samples. The results of these pairwise agreements are shown in table 8 below and are similar to table 7 in that the WhatsHap and HapSeq2 algorithms have the greatest agreements in their haplotypes. While DBM produced the most conflicts when compared with either WhatsHap or HapSeq2.



**Table 7: Pairwise comparison of 3 HA algorithms abased on 120 positions. [ Note, this table is from PPT slide 168] In the first column, a strict agreement counts 1 when the genotype of two HA algorithms are the same (e.g. “AC” in both WhatsHap and HapSeq2) and 0 when the genotype of two HA algorithms are not identical (e.g., one HA output AC, another is AA or AT). For example, the number 95 means out of the 120 SNVs, the genotype of 95 of two HA algorithms are the same. While non-strict agreement counts as 1 when both alleles are the same and counts 0.5 when at least 1 of the two alleles are the same (e.g. “AC” for WhatsHap and “AA” for hapseq2). For example, the number 106 means among the 240 alleles of the 120 SNVs, 212 (106 X2 ) of them are the same. The haplotype comparison is done by comparing the two haplotypes position by position and counting the total number of bases that they are the same. For example, 82.5 means 165 (82.5 X2) alleles are the same on a position by position basis. This is accomplished using R:**

(sum(sample826[,1]==sample826[,3])+ sum(sample826[,2]==sample826[,4]))/2



**Table 8: Pairwise comparison of haplotypes on 628 positions. [ Note, this table is from PPT slide 157] Strict agreement counts 1 when the genotype of two HA algorithms are the same (e.g. “AC” in both WhatsHap and HapSeq2). While non-strict agreement counts as 1 when both alleles are the same and counts 0.5 when at least 1 of the two alleles are the same (e.g. “AC” for WhatsHap and “AA” for hapseq2).**

The results in table 7 show for the 826 sample in the 120 SNV comparison between WhatsHap and DBM the strict genotype agreement is 95, the non-strict is 106, and the haplotype agreement is 82.5, the other samples have the similar pattern. We also see in the same table in the comparison between HapSeq2 and DBM produced similar results as WhatsHap and DBM. While we see a much better agreement when comparing WhatsHap and Hapseq2. For example, on sample 826 the strict genotype agreement is 111, the non-strict is 115.5, the haplotype agreement is 87.5, and the other samples have the similar pattern. We get similar agreements in Table 8 where we see a similar pattern, where the WhatsHap and HapSeq2 algorithms have the greatest agreements in their genotypes and haplotypes. The conclusion is that the WhatsHap and HapSeq2 algorithms have the greatest agreements in their haplotypes across each dataset tested. While DBM produced the most conflicts when compared with either WhatsHap or HapSeq2.

References

Adams, M. D., Veigl, M. L., Wang, Z., Molyneux, N., Sun, S., Guda, K., Yu, X., Markowitz, S. D., & Willis, J. (2012). Global mutational profiling of formalin-fixed human colon cancers from a pathology archive. Modern pathology : an official journal of the United States and Canadian Academy of Pathology, Inc, 25(12), 1599–1608. https://doi.org/10.1038/modpathol.2012.121

Bonizzoni, P., Dondi, R., Klau, G. W., Pirola, Y., Pisanti, N., & Zaccaria, S. (2016). On the Minimum Error Correction Problem for Haplotype Assembly in Diploid and Polyploid Genomes. Journal of computational biology : a journal of computational molecular cell biology, 23(9), 718–736. https://doi.org/10.1089/cmb.2015.0220

Li, H., & Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics (Oxford, England)*, *25*(14), 1754–1760. https://doi.org/10.1093/bioinformatics/btp324

Li H. (2011). A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics (Oxford, England)*, *27*(21), 2987–2993. https://doi.org/10.1093/bioinformatics/btr509

Patterson, M., Marschall, T., Pisanti, N., Van Iersel, L., Stougie, L., Klau, G. W., & Schönhuth, A. (2015). WhatsHap: weighted haplotype assembly for future-generation sequencing reads. *Journal of Computational Biology*, *22*(6), 498-509.

Zhang, K., & Zhi, D. (2013). Joint haplotype phasing and genotype calling of multiple individuals using haplotype informative reads. *Bioinformatics (Oxford, England)*, *29*(19), 2427–2434. https://doi.org/10.1093/bioinformatics/btt418

Zhang Y. (2013). A dynamic Bayesian Markov model for phasing and characterizing haplotypes in next-generation sequencing. *Bioinformatics (Oxford, England)*, *29*(7), 878–885. https://doi.org/10.1093/bioinformatics/btt065

**--------------------------------------------------------------------------------------------------------------------------------------------**

**Aug 12, 2020 SS notes: The following notes are about how to further compare 3 MIHA algorithms.**

Next, explore the switch distance/error for those regions that they are very different! Our pairwise comparison is a “rough” comparison, it is not in detail, we’d better to zoom in to compare them.

**The mixSIH and whatshap papers have detailed figures or explanation on switch distance.**

|  |
| --- |
| The following are copied from Flora Cheng’s whatshap presentation slide page 11 (check whatshap paper)  **PPT File: WhatsHap.Presentation.Flora.Cheng.pptx**  Phasing Completeness - variants are partitioned into haplotype blocks.   * Best case: the blocks correspond to the connected components of a connectivity graph   Phasing accuracy   * Cis - reference alleles of the connected variants are on the same haplotype * Trans - the reference alleles are on different haplotypes * **Switch errors:** cis/trans’ meaning is reversed from one point on forward within a block. Calculated by traversing the haplotype blocks and finding the number of times a jump from one haplotype to the other is needed to reconstruct the true haplotype. * **Flip error:** two switch errors in this model. Phasing error rate - sum of the switch errors over all blocks divided by the total number of phase connections |
| The following are copied from Daphne’s presentation on MixSIH page 16:  **File name: MixSH.PPT.by.Daphne.Han.pptm**   * The figure below shows the example of the case that the switch error rate isn’t suitable to evaluate the segment quality. * A single switch error in the middle of a reconstructed haplotype segment has a greater influence on downstream analyses such as detecting amplified haplotypes than a switch error located at the end of a segment |

|  |
| --- |
| **July 3, 2020 notes by SS.** Regarding the comparison of 3 MIHA algorithms. For the following notes on page 126 (page 157 is the copy of page 126), I have the following notes:  “red” number for haplotype comparison with NO “-”, which is incorrect,  the “green” one is the correct with the “-” for whatshap unphased positions.  This means that in “whatshap” output, if it did not infer haplotype at a specific position, its haplotype is “-“. In order to do the strict (genotype) and non-strict (allele) comparision, we change those “-“ to the correspodng allele, but they are NOT the haplotpe alleles, so when we do the haplotype comparison, we should not include them, that is why I have this note :” “red” number for haplotype comparison with NO “-”, which is incorrect”. |

|  |
| --- |
| July 3, 2020 notes by SS  Sherwin found that for the 3 MIHA comparison results obtained on July 3, 2020 (PPT slide page 156) is very different from the 3 MIHA comparison results obtained on April 24, 2020 (PPT slides 126 and 157, which are identical). It shows that the DBM performance or agreement with other two algorithms are improved more. This is likely due to the following reasons:   1. Both the April-2020 and July-2020 VCF filles, whathap used ONE VCF file consists of 5 samples, how the VCF file for DBM and HapSeq2 are different between the April-2020 and July-2020 runs, see (2) 2. For the April-2020 run of DBM and HapSeq2, there are 5 VCF files for 5 samples; for the July-2020 run of DBM and HapSeq2, the ONE VCF file used for whatshap is used. That is, the VCF files for the April-2020 and July-2020 are different.   **Shewin’s explanation (on July 3)**  “ I phased each package using the same VCF (with/of 5 samples) July, 2020. In April , 2020, each package phased a different VCF (for each of those 5 individual sample).  /home/s\_m774/data/miha/dbm.2020.mar20/  /home/s\_m774/data/new.MIHA.June2020/dbm  **Shewin’s explanation July 18, 2020 Saturday:**  In April 2020 run of 628 SNVs, the positions were phased (using the VCF of ALL 13558 SNVs) prior to filter.  In July 2020 run of 628 SNVs, the positions were filtered/selected first prior to phasing.  **Compare the 628 (or 120) count in the old way and new way, to see if they are different!** |

**# The following are from July 3, 2020 meeting notes**

[s\_s355@login1 hapseq2.2020.mar20]$ wc -l /home/s\_m774/data/miha/dbm.2020.mar20/chr10.5sample.counts  
**13558 /home/s\_m774/data/miha/dbm.2020.mar20/chr10.5sample.counts**[s\_s355@login1 hapseq2.2020.mar20]$ wc -l /home/s\_m774/data/new.MIHA.June2020/dbm/chr10.5sample.counts  
629 /home/s\_m774/data/new.MIHA.June2020/dbm/chr10.5sample.counts  
[s\_s355@login1 hapseq2.2020.mar20]$

[s\_s355@login1 ~]$ head /home/s\_m774/data/miha/dbm.2020.mar20/chr10.5sample.counts  
I SID I1 I1 I2 I2 I3 I3 I4 I4 I5 I5  
M 1 0 0 0 0 0 0 0 0 0 2  
M 2 0 0 0 0 0 0 0 0 0 2  
M 3 0 0 0 0 0 1 0 0 0 3  
M 4 7 0 1 0 7 0 1 4 5 2 \*\*   
M 5 5 0 1 0 10 0 3 4 2 2  
M 6 5 0 1 0 10 0 3 4 3 2 \*\*   
M 7 0 0 0 0 0 4 0 0 1 0  
M 8 0 0 0 0 0 0 0 0 0 4  
M 9 0 1 0 0 0 0 0 0 0 0  
[s\_s355@login1 ~]$ head /home/s\_m774/data/new.MIHA.June2020/dbm/chr10.5sample.counts  
I SID I1 I1 I2 I2 I3 I3 I4 I4 I5 I5  
M 1 7 0 1 0 7 0 1 4 5 2  
M 2 5 0 1 0 10 0 3 4 2 2  
M 3 5 0 1 0 10 0 3 4 3 2  
M 4 7 1 5 0 0 19 8 0 14 6  
M 5 17 2 12 0 29 0 8 0 27 0  
M 6 18 0 11 0 25 0 23 2 23 0  
M 7 9 2 10 0 39 0 6 0 21 0  
M 8 37 18 59 3 1 116 40 0 50 62  
M 9 7 1 13 0 0 18 2 0 8 5

# How to manipulate the VCF to get the 628 positions  
# For the July 2020 run, Sherwin issued the following,

To extract 628 positions: ./bcftools view -i 'DP>=20'

but the April 2020, he used "grep",

# July 3, 2020 notes, we checked, this does NOT cause the problem. That is the 628 positions of those 2 wrongs are these same.