Comparative Analysis of Haplotype Assembly Algorithms

Dr. Shuying Sun, Flora Cheng, Daphne Han, Sarah Wei, Alice Zhong

August 1st, 2020

Abstract

Haplotype information is useful for studying genetic variation, which can then be used to determine trait or disease association in a population. Haplotype assembly is a useful way that enables us to interpret genetic variations and its impacts. In this paper, we will be comparing various haplotype assembly algorithms to assess the advantages and disadvantages of each. We will do this by running multiple algorithms across a multitude of datasets to compare their accuracy and efficiency, as well as other algorithmic components. This comparison analysis is important because each algorithm has its own benefits and drawbacks, and it is necessary to compare all of them in order to develop more accurate algorithms in the future.

1 Introduction

There are numerous haplotype assembly software packages that have been created, and each of them have their own advantages and disadvantages. In this paper, we will be discussing 6 software packages: HapCUT2 [1], MixSIH [2], PEATH [3], WhatsHap [4], SDhaP [5], and MAtCHap [6]. HapCUT2, MixSIH, PEATH, SDhaP, and MAtCHap are known as single individual haplotype assembly algorithms (SIHA), whereas WhatsHap is a multiple individual haplotype assembly algorithm (MIHA). In the next few sections, we will discuss the datasets we ran these algorithms over, as well as our findings and any complications.

2 Background Information

Human somatic cells are diploid, meaning pairs of chromosomes are inherited from each parent. In these chromosomes, there are often variations at a specific site on a chromosome, such as SNVs (single nucleotide variants), which are sites at which a single DNA base differs between the paternal and maternal chromosomes [7]. One type of SNV is an SNP (single nucleotide polymorphisms), which can be observed at the population level. Haplotypes are ordered sequences of alleles on a chromosome, with one of each allele being inherited from each parent, see Figure 1 [1].

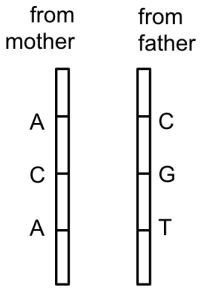

Figure 1: A child's haplotype alleles, inherited from each parent.

It is important to infer haplotypes because they are useful for studying genetic variants. Haplotype information can be used to study a population and generate markers and maps as a means of understanding how genetic variation evolves and contributes to phenotypes. Additionally, certain haplotypes may be associated with diseases or traits in a population

[1]. The reconstruction of haplotypes from DNA sequencing reads is known as haplotype assembly [4]. Haplotype assembly is important and necessary for us to understand and interpret variants in humans, as well as how these variants will impact effects of diseases on certain groups of individuals [4]. However, it is difficult to conduct haplotype assembly, as DNA sequencing datasets are often large and contain errors. There have been numerous methods devised to improve accuracy; however, many of these lack confidence and efficiency in the extraction of assembled regions [2].

3 New Results

3.1 826T Dataset

We familiarized ourselves with the aforementioned haplotype assembly software packages (HapCUT2, MixSIH, PEATH, and WhatsHap). We ran these software packages on 826T single-end, pair-end, and whole pair-end datasets. We then used Perl to reformat the output for the 826T pair-end dataset. Using the reformatted outputs, we used R to obtain values, such as the number of SNVs (single-nucleotide variants), number of haplotype assembly blocks, mean number of SNVs per block, as well as the maximum, minimum, and quartiles for SNVs per block. We were able to compare these values across the different software packages, finding that HapCUT2, MixSIH, and PEATH yielded the same results.

3.2 NA12878 Dataset

For the NA12878 dataset, we used the four aforementioned haplotype assembly software packages, as well as SDhaP and MAtCHap. We utilized two versions of the NA12878 dataset: one version had the homozygous variants and indels (insertions or deletions in the DNA sequence) removed, and the other version was left unedited. Each package was run on each dataset twice, once using the sbatch command and once without. Note that the MAtCHap software was only able to run on the version of the NA12878 dataset with the homozygous variants and indels removed; we did not run the MAtCHap software package on the unedited dataset. Also note that for SDhaP, it can only run on an input file with

a maximum of 700,000 DNA fragments, so we split the dataset into two parts. The split location was determined by using R to find the locations with the largest position difference that would still satisfy the limiting condition of SDhaP.

We then replicated the same reformatting procedure as before, using Perl and R, and obtained the SNV and block summary values. The results between the various runs were then compared to find differences in the values. On average, HapCUT2 was able to generate outputs the fastest. Interestingly, the first part of the input file for SDhaP took the longest amount of time to run, taking over 16 hours, whereas the second part of the input file for SDhaP took only 3 to 4 minutes to run.

	Time	HapCUT2	MixSIH	PEATH	WhatsHap	MAtCHap
raw, with Sbatch	real	0m0.027s	213m56.187s	12m57.435s	22m41.267s	
	user	0m0.021s	214m3.906s	12m56.604s	22m23.670s	
	system	0m0.004s	0m0.390s	0m1.130s	0m5.659s	
raw, without Sbatch	real	1m23.592s	217m28.723s	10m9.962s	24m15.866s	
	user	1m22.744s	217m36.602s	10m9.063s	24m1.699s	
	system	0m0.609s	0m0.462s	0m0.796s	0m7.548s	
no indels/homovars, with Sbatch	real	0m0.027s	166m47.585s	9m10.835s	22m3.683s	125m32.571s
	user	0m0.020s	166m53.561s	9m9.623s	21m47.595s	124m51.927s
	system	0m0.004s	0m0.425s	0m0.713s	0m4.690s	0m45.194s
no indels/homovars, without Sbatch	real	1m26.764s	188m55.662s	8m50.215s	24m40.068s	139m20.331s
	user	1m26.197s	189m1.981s	8m49.187s	24m36.676s	138m41.827s
	system	0m0.620s	0m0.888s	0m0.790s	0m7.024s	0m43.396s

Figure 2a: Run time for HapCUT2, MixSIH, PEATH, WhatsHap, and MAtCHap on the NA12878 dataset with and without using the sbatch command in Unix on the raw dataset and a revised version of the dataset that does not contain homozygous variants or indels. Note that MAtCHap does not have run times for the raw data, since the software was only able to run on the NA12878 dataset with the homozygous variants and indels removed.

	Split location	Time	SDhaP -1st half	Sdhap - 2nd half
raw, with Sbatch		real	1056m20.111s	3m10.001s
		user	1056m53.506s	N/A
	136135 split	system	0m7.245s	N/A
no indels/homovars, with Sbatch		real	1037m50.615s	4m1.958s
		user	1038m22.430s	3m57.365s
	100595 split	system	0m7.826s	0m4.707s
		real	1179m52.398s	2m52.346s
		user	1180m29.543s	2m47.180s
	123331 split - diff line	system	0m8.155s	0m5.180s
		real	450m10.006s	3m10.005s
		user	450m18.859s	3m.5.418s
	123331 split - same line	system	0m8.204s	0m4.673s

Figure 2b: Run time for SDhaP on the NA12878 dataset – which was split into two parts due to the limitations of the algorithm – on the raw dataset and a revised version of the dataset that does not contain homozygous variants or indels. The total number of fragments of the two parts of each run are 554827 and 412108 for the 136135 split on the raw data; 589226 and 377709 for the 100595 split on the no homozygous variants or indels data; and 680496 and 286439 for the 123331 split on the no homozygous variants or indels data.

For HapCUT2 and MixSIH, there were dashes ("-") in the outputs. We used Perl and R to remove the SNVs that contained dashes in the outputs. This eliminated some blocks from the output. Furthermore, for HapCUT2, the removal of dashes resulted in some blocks only containing 1 SNV. The outputs for HapCUT2 and WhatsHap both contained 1 SNV blocks, which needed to be removed. This led to the removal of 2 blocks from the HapCUT2 output and 1 block from the WhatsHap output.

Summary	# raw SNVs	# SNVs out of order	# SNVs with "-"	# SNVs no "-"	# raw blocks	# blocks no "-"
HapCUT2	115813	752	596	115217	32252	32152
HapCUT2 (length 1 block removed)	N/A	N/A	N/A	115215	N/A	32150
MixSIH	120555	752	4765	115790	32252	32252
PEATH	115813	0	0	115813	31355	31355
WhatsHap	178524	0	0	178524	10133	10133
WhatsHap (length 1 block removed)	178523	0	0	178523	10132	10132
SDhaP - 100595 split	115813	752	0	115813	32252	32252
SDhaP - 123331 split	115813	0	0	0	32252	32252
MAtChap	115813	0	0	115813	31355	31355

Summary	min # blocks	1st Q	median # blocks	mean # blocks	3rd Q	max # blocks
HapCUT2	1	2	2	3.583	4	770
HapCUT2 (length 1 block removed)	2	2	2	3.584	4	770
MixSIH	2	2	2	3.59	4	770
PEATH	2	2	2	3.694	4	770
WhatsHap	1	2	4	17.62	9	5194
WhatsHap (length 1 block removed)	2	2	4	17.62	9	5194
SDhaP - 100595 split	2	2	2	3.591	4	770
SDhaP - 123331 split	2	2	2	3.591	4	770
MAtChap	2	2	2	3.694	4	770

Figure 3: Summary of haplotype output for HapCUT2, MixSIH, PEATH, WhatsHap, SDhaP, and MAtCHap on the NA12878 dataset with homozygous variants and indels removed. Note that HapCUT2 and WhatsHap had blocks of length 1 SNV that were removed from the output. Also note that for HapCUT2, the blocks of length 1 SNV did not occur until after the dashes ("-") were removed.

3.3 Pairwise Comparison

After running HapCUT2, MixSIH, PEATH, WhatsHap, SDhaP, and MAtCHap on the NA12878 datasets, we performed a pairwise comparison between each ordered pair out of the six software packages. This included a self-comparison, which was performed to check the pairwise comparison method. We expected a result of 0% block disagreement and SNV

disagreement for every self-comparison.

We implemented R code that compared reformatted outputs of each software package when run on the NA12878 datasets; this yielded the SNV disagreement and block disagreement values. We identified the number and percentage of discrepancies between the outputs for each pair of haplotype assembly software packages. Additionally, we used Perl to run the same pairwise comparison. Since we were able to obtain the same results for both methods of pairwise comparison, this confirmed our disagreement percentages generated by the R code.

COMPARISON USING PERL							
Block Disagreement	HapCUT2	MixSIH	PEATH	WhatsHap	SDhaP - 100595 split	SDhaP - 123331 split	MAtCHap
HapCUT2 (32150 blocks)	0	459/32150 (1.43%)	187/32150 (0.58%)	1690/32150 (5.26%)	28024/32150 (87.17%)	27906/32150 (86.80%)	494/32150 (1.54%)
MixSIH (32252 blocks)	885/32252 (2.74%)	0	655/32252 (2.03%)	1924/32252 (5.97%)	28153/32252 (87.29%)	28044/32252 (86.95%)	663/32252 (2.06%)
PEATH (31355 blocks)	1005/31355 (3.21%)	976/31355 (3.11%)	0	2032/31355 (6.48%)	27503/31355 (87.71%)	27397/31355 (87.38%)	838/31355 (2.67%)
WhatsHap (10132 blocks)	6353/10132 (62.70%)	6344/10132 (62.61%)	6328/10132 (62.46%)	0		10128/10132 (99.96%)	6347/10132 (62.64%)
SDhaP - 100595 split (32252 blocks)	28121/32252 (87.19%)	26112/32252 (80.96%)	24882/32252 (77.15%)	19175/32252 (59.45%)	0	901/32252 (2.79%)	25640/32252 (78.94%)
SDhaP - 123331 split (32252 blocks)	28120/32252 (87.19%)	26102/32252 (80.93%)	24873/32252 (77.12%)	19148/32252 (59.37%)		0	25450/32252 (78.91%)
MAtCHap (31355 blocks)	1274/31355 (4.06%)	954/31355 (3.04%)	838/31355 (2.67%)	2105/31355 (6.71%)	27492/31355 (87.68%)	27388/31355 (87.35%)	0
SNV Disagreement	HapCUT2	MixSIH	PEATH	WhatsHap	SDhaP - 100595 split	SdhaP - 123331 split	MAtCHap
HapCUT2 (115215 SNV, 596 "-")	0	8881/115215 (7.71%)	6178/115215 (5.36%)	15960/115215 (13.85%)	102559/115215 (89.02%)	102174/115215 (88.68%)	9224/115215 (8.01%)
MixSIH (115792 SNV, 9717 "-")	11034/115780 (9.53%)	0	9953/115790 (8.60%)	17307/115790 (14.95%)	103217/115790 (89.14%)	10899/115790 (88.87%)	10240/115790 (8.84%)
PEATH (115813 SNV, 0 "-")	12872/115813 (11.11%)	13671/115813 (11.80%)	0	19263/115813 (16.63%)	103957/115813 (89.76%)	103605/115813 (89.46%)	12508/115813 (10.80%)
WhatsHap (178523 SNV, 0 "-")	165569/178523 (92.74%)	165603/178523 (92.76%)	165582/178523 (92.75%)	0		178514/178523 (99.99%)	165661/178523 (92.80%)
SDhaP - 100595 split (115813 SNVs, 0 "-")	103049/115813 (88.98%)	97212/115813 (83.94%)	92738/115813 (80.10%)	77462/115813 (66.89%)	0	10971/115813 (9.47%)	94315/115813 (81.44%)
SDhaP - 123331 split (115813 SNVs, 0 "-")	103070/115813 (89.00%)	97171/115813 (83.90%)	92656/115813 (80.00%)	77320/115813 (66.76%)		0	94293/115813 (81.42%)
MAtCHap (115813 SNVs, 0 "-")	15417/115813 (13.31%)	13563/115813 (11.71%)	12508/115813 (10.80%)	19750/115813 (17.05%)	103908/115813 (89.72%)	103579/115813 (89.44%)	0

Figure 4: Block disagreement and SNV disagreement for pairwise comparisons of outputs for HapCUT2, MixSIH, PEATH, WhatsHap, SDhaP and MAtCHap on the NA12878 dataset with homozygous variants and indels removed.

Overall, HapCUT2, MixSIH, PEATH, and MAtCHap gave relatively low disagreement percentages for both blocks and SNVs. For example, the pairwise comparison between HapCUT2 and PEATH gave only a 0.58% block disagreement and 5.36% SNV disagreement. However, both SDhaP and WhatsHap resulted in much higher disagreement percentages with the other algorithms. The largest difference was between WhatsHap and SDhaP split at position 123331 where the header line was split into two separate lines, which gave a 99.96% block disagreement and 99.99% SNV disagreement.

3.4 WhatsHap

As shown in the above figure, the WhatsHap output contained 178523 SNVs, around 50% more than the other HA algorithms, which made it an "improper" or "unbalanced" comparison. In order to have a "proper" comparison, we filtered the WhatsHap output to only contain chromosome positions found in the PEATH and MAtCHap output files. We chose these two algorithms because they had identical 115813 SNV positions, and HapCUT2 and MixSIH had a similar number and list of SNVs. We then performed pairwise comparisons on both the unfiltered and filtered WhatsHap output with the other HA algorithms. Doing so, we found that both the SNV and block disagreements were significantly lower for the filtered output than those for the unfiltered output (see Figure 5).

Regarding the WhatsHap-filtered results, 115745/178523 SNVs (after removing blocks with 1 SNV) were left. That is, WhatsHap and other HA algorithms had a large number of common SNVs after the filter/selection. The filtered output had about a 2362/8149 =28.99% block disagreement when compared with HapCUT2. These 2362 blocks had 94685 SNVs. This means that, on average, there were approximately 40 SNVs on each of these blocks. Thus, outputs generated by WhatsHap have disagreements with other HA algorithms on long/large blocks.

Block Disagreement	HapCUT2	MixSIH	PEATH	WhatsHap	MAtCHap
WhatsHap - unfiltered (10132 blocks)	6353/10132 (62.70%)	6344/10132 (62.61%)	6328/10132 (62.46%)	0	6347/10132 (62.64%)
WhatsHap - filtered (8149 blocks)	2362/8149 (28.99%)	2258/8149 (27.71%)	2227/8149 (27.33%)	0	2255/8149 (27.67%)

SNV Disagreement	HapCUT2	MixSIH	PEATH	WhatsHap	MAtCHap
WhatsHap - unfiltered (178523 SNVs)	165569/178523 (92.74%)	165603/178523 (92.76%)	165582/178523 (92.75%)	0	165661/178523 (92.80%)
WhatsHap - filtered (115745 SNVs)	94685/115745 (81.80%)	93576/115745 (80.85%)	93258/115745 (80.57%)	0	93491/115745 (80.77%)

Figure 5: Block disagreement and SNV disagreement for pairwise comparisons on unfiltered and filtered WhatsHap outputs for the NA12878 dataset with homozygous variants and indels removed. Note that both the unfiltered and filtered files already had all blocks with 1 SNV removed before the comparisons.

3.5 MixSIH

For MixSIH, we performed a pairwise comparison on multiple runs on both the raw dataset and the dataset with no indels or homozygous variants. Although the block and SNV disagreements were relatively low, we can see in the figure below that the number of SNVs per block that differed were relatively high (Figure 6). From this, we concluded

that the outputs generated by MixSIH disagree with each other on long/large blocks, and that these disagreements and differences arose due to the probabilistic nature of the MixSIH algorithm.

BLOCK DISAGREEMENT	Bertie -- raw	Bertie -- HI	Daphne -- Raw	Daphne -- no HI
Daphne -- raw (32252)	**378/32252 (1.17%)**			**366/32262 (1.13%)**
Daphne -- no HI (32252)		**348/32252 (1.08%)**	**363/32252 (1.13%)**	
without dashes				
SNV DISAGREEMENT	Bertie -- raw	Bertie -- HI	Daphne -- Raw	Daphne -- no HI
Daphne -- raw (115792)	**7601/115792 (6.56%)**			**7436/115792 (6.42%)**
Daphne -- no HI (115790)		**7258/115790 (6.27%)**	**7414/115790 (6.46%)**	

Figure 6: Pairwise comparisons of multiple runs of MixSIH on the NA12878 dataset based on whether or not the raw dataset was used, as well as different runs conducted on the same dataset but at separate times.

3.6 SDhaP

Another haplotype assembly software package considered was SDhaP. However, the algorithm has an upper limit for the number of lines in the input file. Thus, the NA12878 had to be split into two parts in order to use SDhaP. We did the following different SDhaP runs:

- Same input file, but different run (i.e., same input files run by different people)
- Dataset split at different locations
- Split the top line of the input file into two lines or kept them on the same line

We use the same fragment files. However, when we compared the SNVs, we found that only 93804/115813 = 81% of SNVs between the PEATH and SDhaP generated outputs were in common and 22009/115813 =19% were different. Because the SNV positions were not even the same, SDhaP had a large disagreement with others. The comparison results of different runs of SDhaP are shown in Figure 7. These comparison results show that there were approximately 900 blocks (880 - 938) that had disagreements for the dataset with homozygous variants and indels removed. In these 900 blocks, there were about 11,000 SNVs (10747 - 11267) that disagreed. This means that, on average, there were about 12

SNVs per block that disagreed. This comparison result shows that different runs of SDhaP had different results on long/large blocks.

BLOCK DISAGREEMENT	100595 split (no HI)	123331 split - separate lines (no HI)	123331 split - same lines (no HI)	136135 split (Raw)
100595 split (no HI)	0	938/32252 (2.91%)	907/32252 (2.81%)	12197/32252 (37.82%)
123331 split - separate lines (no HI)	938/32252 (2.91%)	0	880/32252 (2.73%)	12198/32252 (37.82%)
123331 split - same line (no HI)	907/32252 (2.81%)	880/32252 (2.73%)	0	12208/32252 (37.85%)
136135 split (Raw)	12197/32252 (37.82%)	12198/32252 (37.82%)	12208/32252 (37.85%)	0

SNV DISAGREEMENT	100595 split (no HI)	123331 split - separate lines (no HI)	123331 split - same lines (no HI)	136135 split (Raw)
100595 split (no HI)	0	11267/115813 (9.73%)	11065/115813 (9.55%)	952066/115813 (44.96%)
123331 split - separate lines (no HI)	11267/115813 (9.73%)	0	10747/115813 (9.28%)	52048/115813 (44.94%)
123331 split - same line (no HI)	11065/115813 (9.55%)	10747/115813 (9.28%)	0	52150/115813 (55.00%)
136135 split (Raw)	52066/115813 (44.96%)	52048/115813 (44.94%)	52150/115813 (45.00%)	0

Figure 7: Pairwise comparisons for multiple runs of SDhaP on the NA12878 dataset, depending on split location, whether or not the file was raw or had homozygous variants and indels removed, and the format of the top line of the input files.

For the runs that used the NA12878 dataset with no homozygous variants and indels split at position 123331 (one run with input file header line kept on one line and the other split into two lines), they had the same number (115813) of SNVs. For the inputs split at positions 100595 and 123331 on the no homozygous variants and indels run, they had the same number (115813) of SNVs. Additionally, when comparing the raw and no homozygous variants and indels files, they had 102239/115813 = 88% SNVs in common, while 13574/115813 =12% of them were different.

3.7 Ordering Outputs

When considering the output files of each HA algorithm, they will either be sorted numerically by SNV chromosome position or block number. Depending on how the file is sorted, this can affect the number of SNVs that are out of order. That is, the chromosome positions of SNVs in a given block are not necessarily smaller than those in the block after it. Using a simple example, say that block 1 has SNVs {1,2,5}, while block 2 has SNVs {3,7}. If the HA algorithm outputs files that are sorted by SNV position, the SNVs will read {1,2,3,5,7}, but the block IDs will be out of order, reading {1,1,2,1,2}. If the HA algorithm outputs files that are sorted by block ID and then SNV position, the block IDs will read {1,1,1,2,2}, but the SNVs will be out of order, reading {1,2,5,3,7}.

We ordered each of the outputs of HapCUT2, MixSIH, PEATH, WhatsHap unfiltered, WhatsHap filtered, SDhaP, and MAtCHap so that we had one file sorted by SNVs and another sorted by block ID. Then, we found that the pairwise comparisons for R and Perl

both gave the same results, regardless of how the input file was sorted. Additionally, we determined how each HA algorithm ordered its output by sorting and comparing with Unix commands. Doing so, we recognized that HapCUT2, MixSIH, and SDhaP sort by block ID, while WhatsHap sorts by SNV position. Finally, PEATH and MAtCHap give the cleanest outputs, for both the SNVs and blocks are ordered. Furthermore, we used an R function to find the number of SNVs that were out of order when sorted by block.

SNVs out of order	HapCUT2	MixSIH	PEATH	WhatsHap	WhatsHap - shared pos	SDhaP	MAtCHap
HA algorithm is sorted by (SNV or blk)	blk	blk	both	SNV	SNV	blk	both
Raw - original file	742	752	0	0	0	1593	not run
Raw - sorted by SNV	0	0	0	0	0	0	not run
Raw - sorted by blk	742	752	0	0	1142	1593	not run
noInoH - original file	742	752	0	0	0	752 *	0
noInoH - sorted by SNV	0	0	0	0	0	0 *	0
noInoH - sorted by blk	742	752	0	0	1142	752 *	0

NOTE: * = for both runs of SDhaP split at 100595 and 123331

Figure 8: Number of SNVs out of order for HapCUT2, MixSIH, PEATH, WhatsHap unfiltered, WhatsHap filtered, SDhaP, and MAtCHap. The table includes information for runs on the NA12878 dataset with indels and homozygous variants included or removed. For each HA algorithm, the SNVs out of order is recorded for the original file, SNV-sorted file, and block-sorted file.

4 Applications

By comparing the five haplotype assembly algorithms for efficiency and accuracy, we were able to better understand which methods are preferable when inferring haplotypes. This can minimize errors in the outputs of future algorithms. Having accurate haplotypes will allow researchers to study genetic variation across populations and generations. With a greater knowledge of haplotypes, the connections between traits and diseases on certain chromosomes can be discovered or confirmed.

5 Conclusion

In terms of efficiency, HapCUT2 is the fastest of the six haplotype assembly algorithms we tested, whereas SDhaP is the slowest by a significant margin (Figures 2a and 2b). We found in the pairwise comparison that four of the five SIHA algorithms (HapCUT2, MixSIH, MAtCHap, and PEATH) had relatively low disagreements between SNVs and blocks in their

outputs for the raw and revised datasets. Meanwhile, SDhaP had large SNV and block disagreements in their outputs for both the raw and revised datasets. Complications we ran into were the WhatsHap generated output had to be reformatted differently (i.e., without using the Perl code). Additionally, due to the large size of the dataset, SDhaP input files had to be split into two parts and run separately, and in general, it was difficult to compare many algorithms across large datasets.

5.1 Future Work

In the future, we would like to study more software packages. We will start by ironing out the issues we ran into with SDhaP, and we will use the output to complete more pairwise comparisons. Furthermore, we have only run one MIHA algorithm (WhatsHap), and we hope to run more MIHA algorithms so we can accurately compare them.

We also plan to read more relevant research papers in this field in order to familiarize ourselves with a greater variety of comparison metrics. This would allow us to provide a more well-rounded analysis of the strengths and weaknesses of each haplotype assembly algorithm.

References

[1] Edge, P., Bafna, V. Bansal, V., HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies, *Genome Res.* gr.213462.116 (2016). doi:10.1101/gr.213462.116

[2] Matsumoto, Hirotaka, and Hisanoria Kiryu, MixSIH: a mixture model for single individual haplotyping, *BMC Genomics*, (2012).

[3] Joong Chae Na, Jong-Chan Lee, Je-Keun Rhee, Soo-Yong Shin, PEATH: single-individual haplotyping by a probabilistic evolutionary algorithm with toggling, *Bioinformatics*, Volume 34, Issue 11, 01 June 2018, Pages 1801â1807, https://doi.org/10.1093/bioinformatics/bty012

[4] Bansal, Vikas, et al. An MCMC algorithm for haplotype assembly from whole-genome sequence data, *Genome Reseasrch*, Cold Spring Harbor Laboratory Press, (2008).

[5] Das, S., Vikalo, H. SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics* 16, 260 (2015). https://doi.org/10.1186/s12864-

015-1408-5

[6] Magi, Alberto, MAtCHap: an ultra fast algorithm for solving the single individual haplotype assembly problem, *bioRxiv*, (2019). http://doi.org/10.1101/860262

[7] Schwartz, Russel, Theory and Algorithms for the Haplotype Assembly Problem, *Communications in Information and Systems*, (2010).