Haplotype Phasing: Minimum Parsimony Formulation

By: Vamsi Mokkapati

Motivating the Biological Problem: What is Haplotype Phasing?

- A **genotype** is defined to be a fraction of the genetic makeup of an individual, which defines its physical characteristics, or phenotype.
- On the other hand, a **haplotype** is only the group of genes the individual inherited from one parent.
- With the usage of recent technology such as high-throughput cost-effective sequencing, genotype data can be obtained, but *not* haplotype data.
- However, haplotype data is very useful to:
  - trace ancestry of individuals.
  - obtain the relationship between different genetic variations and various diseases, and form an association to arrive at a correlation.
  - perform **imputation**, which is when not all genotype data is collected and it is necessary to predict values of uncollected genotypes.
  - predict the points at which genetic recombination occurs.

The Computational Problem

- Given a string of values representing multiple individuals' genotype data, we have to obtain a set of haplotypes that successfully "phase" (or combine to form) all the genotype values.
- A given string of genotypes could look like the following:
  - A/T G/C G/G C/C A/A G/T C/T
- Our equivalent representation for the genotype string above would adhere to the following rules:
  - 0 = Homozygous Reference (major allele)
  - 1 = Heterozygous
  - 2 = Homozygous Alternate (minor allele)
- Therefore, given that Guanine (G) is the alternate, the genotype representation would look like the following:
  - 1120011
- An example of one of the many possible haplotype pairs that could phase this genotype, with a 0 representing the major allele and a 1 representing the minor allele:
  - 0110001
  - 1010010
- Since a 1 in the genotype stands for a heterozygous allele, the haplotypes for every corresponding 1 have to contain both a 0 and a 1. Meanwhile, a 2 in the genotype means that both corresponding haplotype positions have to contain a 1, while a 0 in the genotype means the corresponding positions have to be a 0.

Computational Problem cont.

- Based on the previous example, it is evident that the number of possible haplotypes increases exponentially depending on the number of 1's in the genotype
  - this is due to the fact that heterozygous alleles result in ambiguous genotypes
  - If there are N 1's in the genotype, there are 2^{N-1} possible haplotype pairs for it
- The problem at hand involves finding a set of haplotypes of the smallest size that covers all the genotypes in a given set – to find the minimum parsimony formulation
  - my baseline method to solve this problem involves finding all the 2^{N-1} haplotype pairs for each individual's SNPs; this solution is generally inaccurate and inefficient, but very straightforward
  - my method involves using a greedy method by modifying Clark's algorithm to arrive at a much more accurate solution that is closer to achieving minimum parsimony.
- My benchmarks for this experiment include computational speed and accuracy compared to the minimum parsimony solution

Baseline Method Description

- My baseline method involves finding all the possible haplotypes for each genotype in the given set
- The resulting haplotype set accurately phases the genotypes, but is a much larger set than the minimum parsimony solution, especially for a large number of SNPs.
- For each individual's genotype:
  - if there are N 1's are present in the genotype, output the 2^N possible haplotypes for that genotype
  - after obtaining all the haplotypes, remove any duplicates from the list, and output the result
- In addition to the set of haplotypes being much larger than the minimum parsimony solution, this method is also very slow, and operates at approximately $O(k \bullet 2^N)$, given k individuals and N SNPs

Baseline Method Example

- The example on the right shows the first step of the baseline method applied to 4 individuals, each with 6 SNPs
- If there are N 1's in a genotype, that corresponds to 2^N haplotypes, or 2^{N-1} haplotype pairings
- For example, the first genotype 010112 has 3 1's, and therefore has 8 corresponding haplotypes.
- In the second column, haplotypes that **occur more than once overall** are **highlighted in red**
- These duplicates are **removed** for the final output, which we will obtain in the next step

Genotypes:	Haplotypes
010112	000001
	000011
	000101
	000111
	010001
	010011
	010101
	010111
020002	010001
110020	000010
	010010
	100010
	110010
120222	010111
	110111

Baseline Method Example Cont.

- The final output is obtained by removing the duplicates from the second column (highlighted in red)
- It is clearly seen that we don't need that many haplotypes to efficiently phase the given four genotypes; therefore, the baseline method is inaccurate with respect to the minimum parsimony solution
- The fact that we need 2^N haplotypes for each genotype with N 1's shows that this solution will be very time-consuming for larger amounts of SNPs

Genotypes:	Haplotypes:	Final Output:
010112	000001	000001
	000011	000011
	000101	000101
	000111	000111
	010001	010001
	010011	010011
	010101	010101
	010111	010111
		000010
020002	010001	010010
		100010
110020	000010	110010
	010010	110111
	100010	
	110010	
120222	010111	
	110111	

My Method Description

- I initially followed the Clark's algorithm approach and found all my haplotypes for the unambiguous genotypes (the ones with no heterozygous alleles)
  - My code first goes through all the individuals' SNP data, solves all genotypes that have no 1's present, and stores the known haplotypes in an array.
- My code then uses an optimized version of Clark's algorithm, which maps all of the ambiguous genotypes that can be solved using the currently known haplotypes
  - Using this mapping technique increases access speed from an O(N) to an O(1) operation.
  - It finds which genotypes can be solved using the known haplotypes using the "match" function I made.
- I then find the occurrence frequencies of the known haplotypes, use the haplotype with the highest frequency to solve the ambiguous genotypes that mapped to it, and repeat the process till all the genotypes are solved.
  - ∴ My algorithm combines methods from both the EM and Clark's algorithms
- Removing any possible repeated genotypes before starting my algorithm also resulted in fewer accesses and increased my code's efficiency compared to Clark's method

Genotypes:	Haplotypes:	knownHaps:	Currently Mapped:
2000000022	1000000011	1000000011	2101100021 => 1000000011
	1000000011	1101110111	1201110121 => 1101110111
			2202210121 => 1101110111
1201100020	??????????		
	??????????		
2202220222	1101110111		
	1101110111		
2101100021	??????????		
	??????????		
1201110121	??????????		
	??????????		
2202210121	??????????		
	??????????		

My Method Example

Step 1: Any repetitions from individuals' genotypes will be removed
Step 2: Unambiguous genotypes will be solved; known haplotypes will be mapped

Genotypes:	Haplotypes:	knownHaps:	Currently Mapped:
2000000022	1000000011	1000000011	1201100020 => 1101100010
	1000000011	1101110111	
		1101100010	
1201100020	??????????	**0100000010**	
	??????????		
2202220222	1101110111		
	1101110111		
2101100021	1000000011		
	1101100010		
1201110121	1101110111		
	0100000010		
2202210121	1101110111		
	1101100010		

My Method Example

Step 3: Mapped haplotypes will be used to solve the previous step's ambiguous genotypes; the additional known haplotypes will be appended to the array

Genotypes:	Haplotypes:	knownHaps:	Currently Mapped:
2000000022	1000000011	1000000011	–
	1000000011	1101110111	
		1101100010	
1201100020	1101100010	**0100000010**	
	0100000010		
2202220222	1101110111		
	1101110111		
2101100021	1000000011		
	1101100010		
1201110121	1101110111		
	0100000010		
2202210121	1101110111		
	1101100010		

My Method Example

Step 4: The previous step's mapped haplotype will be used to solve for the last haplotype, which happens to already be in the knownHaps array.

Genotypes:	Haplotypes:	Final Output:
2000000022	1000000011	**1000000011**
	1000000011	**1101110111**
		1101100010
1201100020	1101100010	**0100000010**
	0100000010	
2202220222	1101110111	
	1101110111	
2101100021	1000000011	
	1101100010	
1201110121	1101110111	
	0100000010	
2202210121	1101110111	
	1101100010	

My Method Example

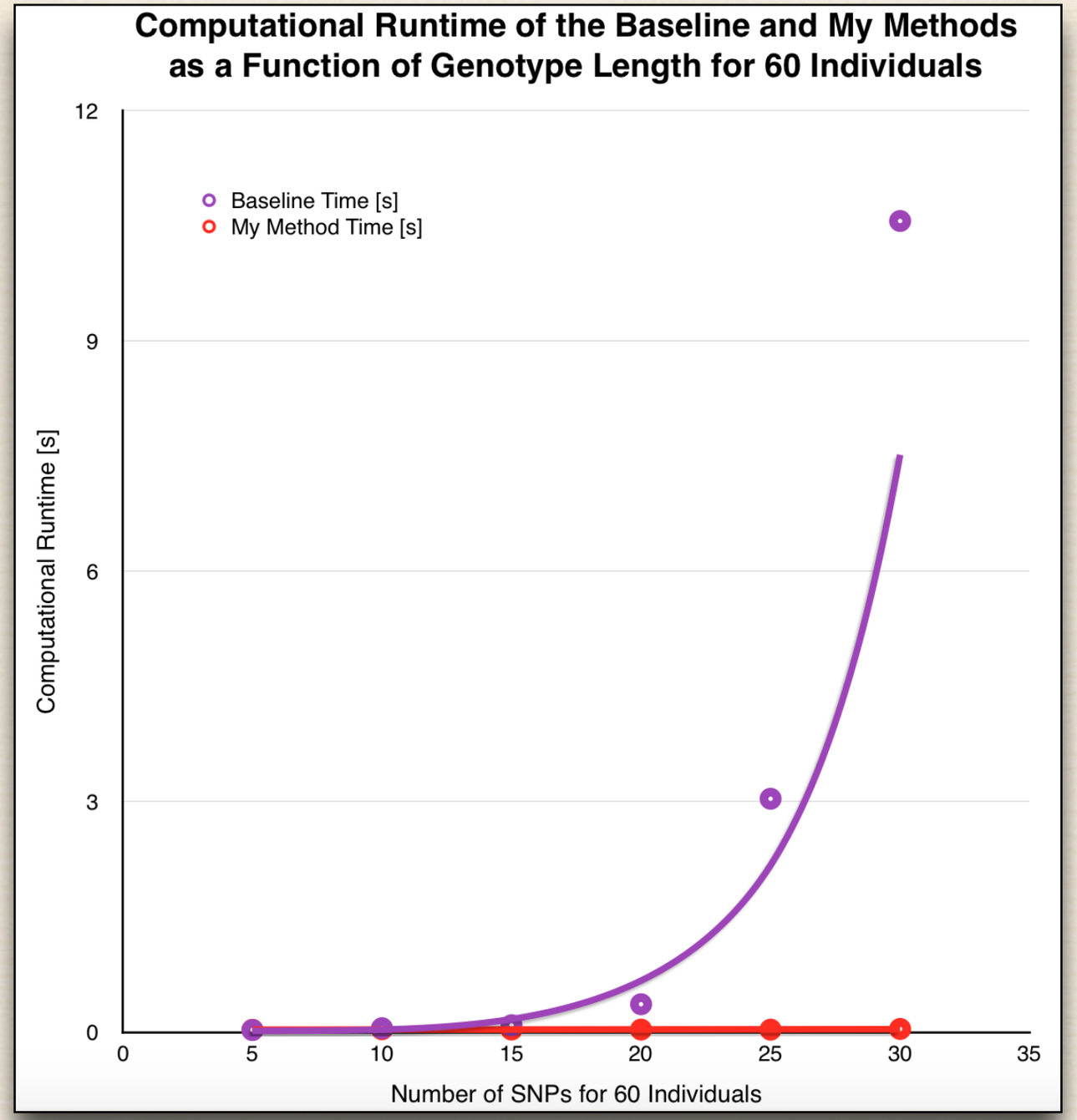
Step 5: Once the array of Haplotypes (second column) is completely full, the final output will always be that column with all duplicates removed.

Conditions Under Which My Method Works

- Since I used a greedy method to solve this problem, it is inherent that this does not always output a result that is 100% accurate compared to the minimum parsimony solution
  - I am trying to make the most optimum choice of haplotypes to solve at each round using my modification of Clark's algorithm; this may not always yield the minimum parsimony formulation
- Due to the limitations on Clark's algorithm, my method does not work if there are no unambiguous genotypes in the given read data.
- Even if there are a few unambiguous genotypes in the read data, there is a possibility of my algorithm getting stuck and producing an error if none of my known haplotypes at a given time are able to correlate with an ambiguous genotype to produce a complement haplotype.
  - Problem usually only present in small initial genotype sets; for instance, my code did NOT run into any problems for all the given test data
  - An example of this is found in my previous baseline method example, with genotypes 010112, 020002, 110020, and 120222
    - 020002 yields the haplotype 010001
    - 010001 can be used with 000111 to make the genotype 010112
    - However, neither 010001 nor 000111 can be used to make either of the other two genotypes => Error!

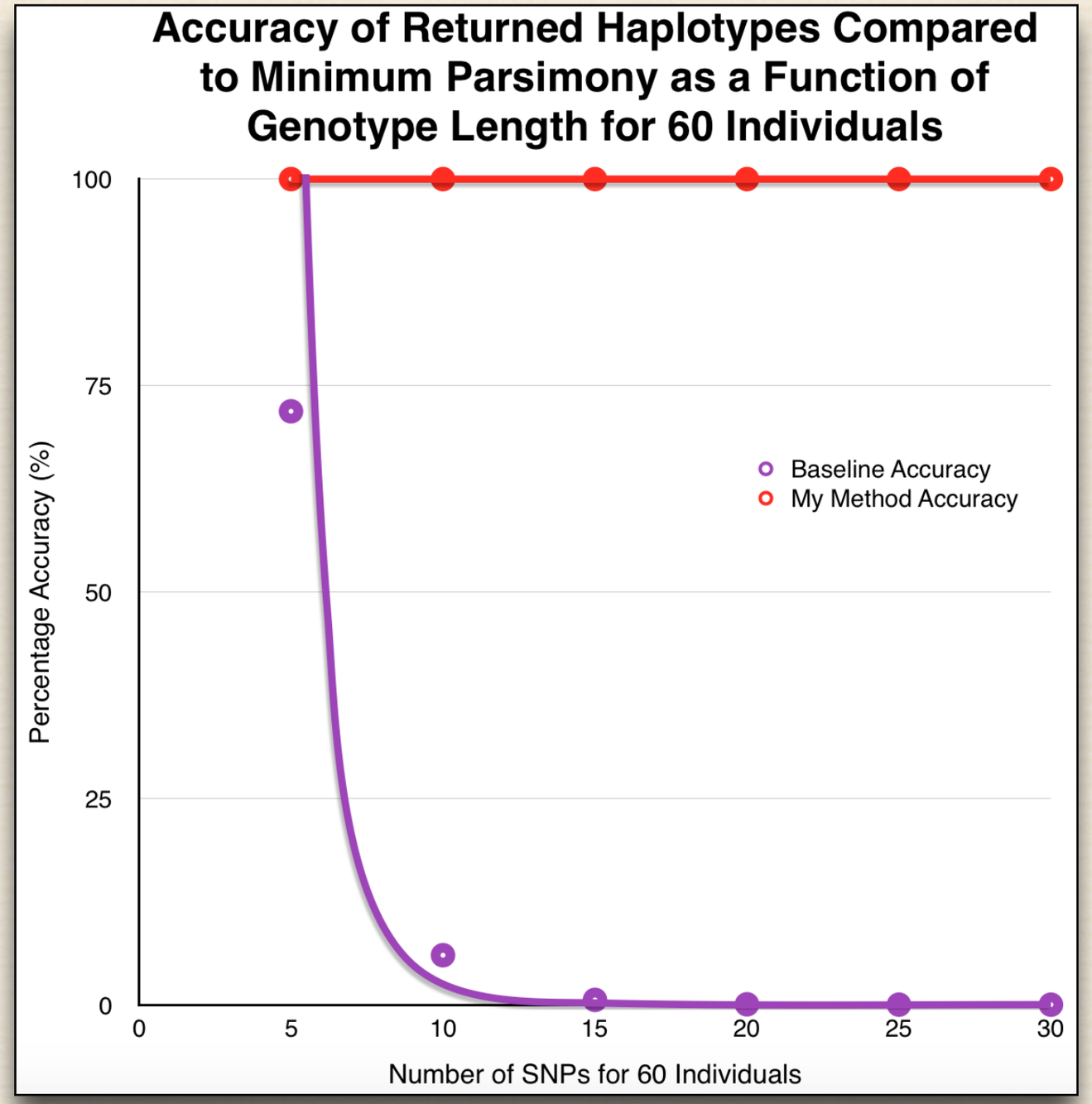
Analyzing Both Methods' Performances

- To obtain performance data for both my method and the baseline, I decided it would be most informative to test my code on test files with a constant number of individuals and varying amount of SNPs.
- I did this because after my analysis of the baseline method, it became apparent to me that as the SNP length became longer, the number of haplotypes to be analyzed increased exponentially due to the larger number of heterozygous alleles on average in the genotype
- For my test cases, I started out using given test data that had 60 individuals, and 30 SNPs per individual. Using this data as a starting point, I created test files containing 5, 10, 15, 20, and 25 SNPs for 60 individuals in order to test my method and the baseline method on.
- I decided not to test more than 30 SNPs per individual, since testing my baseline after that point took an unreasonable amount of time to run, as was expected due to its exponentially increasing running time



Performance of My Method Compared to the Baseline: Computational Speed

As the number of SNPs to be analyzed increases, the runtime of my baseline method gets exponentially longer



Performance of My Method Compared to the Baseline: Accuracy With Respect to Minimum Parsimony

As the number of SNPs to be analyzed increases, the accuracy of my baseline method with respect to the minimum parsimony solution gets exponentially smaller

Analysis: Observations and Future Implications

- For the specific training data given to me to test my algorithm, I noticed that my output always matched the minimum parsimony solution with almost 100% accuracy for the all the easy, medium, hard, and very hard input genotypes, and obtained an output in a reasonable amount of time (even for the very hard test case)
  - I still realize that since I used a greedy solution to develop my method, there are numerous cases where my solution would not yield the *most* optimal solution, just *an* optimal solution
  - This leaves the possibility that my haplotype results to the test data which I submitted might not match the minimum parsimony formulation with full accuracy
- A possible improvement upon my method to phase haplotypes is to try and find an efficient way to look at existing haplotypes and iteratively determine which haplotypes out of my resulting set are NOT needed to obtain the same input genotype set
  - eliminating these unneeded haplotypes will result in a more accurate solution
- It has to be noted that although I implemented various measures in my solution to improve upon the efficiency of Clark's algorithm (using a hash-map, and using some elements of the EM algorithm), my results did not illustrate a *change* in results compared to Clark's method.

Final Project Webpage

- Project Webpage Link:
  - http://haplophasingvamsi.weebly.com/
- The site above has information about me, the project, my goals for this project, my weekly schedule to complete this project, my code, and more. Check it out! :)