AUTHORIZATION TO LEND AND REPRODUCE THE THESIS

As the sole author of this thesis, I authorize Brown University to lend it to other institutions or individuals for the purpose of scholarly research.

Date _____ _____

Jong Wha Joanne Joo, Author

I further authorize Brown University to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Date _____ _____

Jong Wha Joanne Joo, Author

Haplotype Phasing Algorithms and Comparison

By

Jong Wha Joanne Joo

B.A., Seoul National University, 2005

Thesis

Submitted in partial fulfillment of the requirements for the

Degree of Masters of Science in the Department of Computer Science

at Brown University

PROVIDENCE, RHODE ISLAND

May 2007

This thesis by Jong Wha Joanne Joo is accepted in its present form
by the Department of Computer Science as satisfying the
thesis requirements for the degree of Master of Science.

Date _____ _____

Sorin Istrail, Advisor

Approved by the Graduate Council.

Date _____ _____

Sheila Bonde, Dean of the Graduate School

Preface and acknowledgements

I would like to thank my advisor, Professor Sorin Istrail, who truly made a difference in my life. It was under his tutelage that I developed a focus and became interested in Computational Biology. I appreciate his vast knowledge and kind guidance that always gave me answers and directions whenever I need them. It was his assistant and encouragement that leaded me to complete this thesis and my masters' degree.

Contents

List of Figures

Part I

PHASEM Algorithm

Chapter 1

Introduction

It is widely hoped that the study of sequence variance in the human genome will provide a means of elucidation the genetic component of complex diseases and variable drug responses. The majority of human sequence variation is due to substitutions that occurred once in the history of mankind at individual base pair, called single nucleotide polymorphisms(SNPs). The sequence of alleles in contiguous SNP positions along a chromosomal region is called a haplotype. This haplotype information is important for fine-scale molecular-genetics data, for example, in disease mapping, or inferring population histories. Most organisms of interest are diploid, and for diploid organisms, the genotype specifies for every SNP position the particular alleles that are present at this site in the two chromosomes. However, the genotype does not provide phase information, it does not provide information of association of each allele with one of the two chromosomes but only provides information of combination of alleles at a given site. It is possible to determine haplotypes by use of experimental techniques, but such approaches are considerably more expensive and time consuming than modern high throughput genotyping. We can also obtain phase information partially thorough genotyping of additional family members. Alternative to these methods, it is good to use a statistical method to infer phase at

linked loci from genotypes and reconstruct haplotypes, which is called Haplotype Inference(HI) problem. The main issue of the statistical method is that whether it could accurately and effectively estimate the haplotypes. Numerous approaches that try to resolve haplotypes from genotypes have been suggested such as Clark method, parsimony method, maximum likelihood method(EM method), Bayesian method, perfect-phylogeny-based method, and so on. Even though the basic idea of those algorithms are the same and many researchers used them to implement their own programs, the algorithms are flexible and the detailed part of those algorithms could be different. In addition,there could be numerous ways to implement them. I referenced many papers that introduced those algorithms and implemented them, however, the detail of the algorithms could be different with them and my own ideas are added to them. I also suggested a new algorithm named ,"Parsimony method", which showed improved performance for some cases, dependent on the data set. Most of the programs, now available on the web, used Java, C or C++ to implement those algorithm, but I used Mathematica. In partI of this paper, I introduced several representative methods, Clark Method and EM Method. Also I suggested new parsimony method, named "Parsimony Method". And at the end of partI, I analyzed and compare those methods. In partII, I implemented those methods, using Mathematica program and compared the performance.

Chapter 2

Clark method

In this chapter, I will introduce one of the popular phasing method, Clark method, which uses Clark's algorithm. For Clark Algorithm, the result of the phase could be different, dependent on the order of the resolving genotypes. So I suggest two different Clark algorithms, Clark algorithm1 and Clark Algorithm 2, which use different order to resolve the given genotypes. In addition, to compare the different results by the order of the resolving genotypes, I also show Clark Phasing Tree Graph, which shows all possible routes of phasing genotypes, dependent on the order of the phase. The basic idea and some parts of this chapter is from Clark's paper[1].

2.1 Definition and Notation

We define genotype as a multilocus sequence whose haplotype phase is unknown priory. A genotype is a sum of two sequence so that each locus takes value from {0,1,2}. Otherwise, haplotype is a homozygous sequence that takes value from {0,1}. What we want to solve is given a set of observed genotypes, $GenoT = \{g_1, g_2, ..., g_n\}$, solve corresponding set of haplotypes, $HapT = \{h_1, h_2, ..., h_m\}$, that

solves the genotype set. Here, n is the number of genotypes in a sample and m is the number of types of genotypes in a sample

- genotype: An observed multilocus sequence whose haplotype phase is unknown a priori
- GenoT: A genotype set $GenoT = \{g_1, g_2, ..., g_n\}$
- n: The number of genotypes in a genotype set
- haplotype: A homozygous sequence which is not observed
- HapT: A haplotype set $HapT = \{h_1, h_2, ..., h_m\}$
- m: The number of haplotypes in a haplotype set
- A sequence is expressed using values {0,1,2}. 0 and 1 indicates homozygous site and 2 indicates heterozygous site.
- When resolving a genotype with a combination of two haplotypes, $\oplus$ indicates the combination. When two locus have values i and j,

$$i \oplus j = \begin{cases} 0 & \text{if } i = j = 0, \\ 1 & \text{if } i = j = 1, \\ 2 & \text{if } i \neq j \end{cases} \tag{2.1}$$

  For example, 0001$\oplus$0111 = 0221
- Resolve a genotype with two haplotype: Find a combination of haplotypes that could infer a genotype. For example, for a genotype 0221, there are two possible combinations of haplotypes, 0001$\oplus$0111 and 0011$\oplus$0101 that could infer the genotype.

2.2 Clark Algorithm

2.2.1 Algorithm description

Clark's algorithm(1990) can be viewed as an attempt to minimize the total number of haplotypes observed in the sample and, hence, as a sort of parsimony approach. The algorithm begins by listing all haplotypes that must be present unambiguously in the sample. This list comes from those individuals whose haplotypes are unambiguous from their genotypes-that is, those individuals who are homozygous at every locus or are heterozygous are only one locus. Once this list of known haplotypes has been constructed, the haplotypes on this list are considered one ar a time, to see whether any of the unresolved genotypes can be resolved into a known haplotype plus a complementary haplotype. Such a genotype is considered resolved and the complementary haplotype is added to the list of known haplotypes. The algorithm continues cycling through the list until all genotypes are resolved or no further haplotypes can be resolved in this way.
In summary, the algorithm is as follows:

1. Identify all homozygotes and single-site heterozygoses and consider their haplotypes as "resolved.". If a homozygote is found, we have unambiguously identified a haplotype. If a single-site heterozygote is found, we have unambiguously identified two haplotypes. For example, if we found 0120, we identifies two haplotypes 0100 and 0110 as 0120=0100$\oplus$0110.

2. For each known haplotype, we then look at all the remaining unresolved sequences and ask whether the known haplotype can be made from some combination of the ambiguous sites. Each time such a haplotype is found, recover the complement of the haplotype as another potential haplotype.

3. Continue step2 until all haplotypes have been recovered, or until no more new

haplotypes can be found.

By performing these steps with different orderings of the data, the uniqueness of the solution can be determined. The solution that resolves the most haplotypes is almost always valid.

2.2.2 Clark Algorithm 1

There could be several ways of pairing haplotypes and genotypes. Paring which haplotypes to which genotypes may affect the result,some genotypes might not be resolved or wrongly resolved, dependent of the order of resolving. To solve this problem, as Clark suggested, we need to run with different order of genotypes or haplotypes and also run several times to get a better performance(resolve more genotypes). In addition, we can change the rule of pairing. When comparing haplotypes with genotypes to check whether a haplotype can infer a genotype, I tried two different rules of pairing genotypes and haplotypes.

I will introduce the first Clark algorithm using the first rule, "Clark Algorithm 1", in this subsection. We are given a genotype set GenoT and haplotype set HapT. We remove homozygous genotypes from GenoT and put them into HapT. Also we remove single-site heterozygous genotypes from GenoT and make complementary and put then into HapT. As a result, suppose we get GenoT = $\{g_1, g_2, ..., g_{gnum}\}$ and HapT = $\{h_1, h_2, ..., h_{hnum}\}$.

1. Pick the first haplotype h_1 and find a genotype, suppose it is g_2, which could be inferred from h_1. Remove g_2 from GenoT and make complementary of g_2, which is h_1', put it into HapT.

   →Find a genotype, suppose it is g_1, which could be inferred from h_1' . Remove g_1 from GenoT and make complementary of g_1, which is h_1'', put it into HapT.

   →Find a genotype, suppose it is g_3, which could be inferred from h_1'' . Remove g_3 from GenoT and make complementary of g_3, which is h_1''', put it into HapT.

2. Iterate #1 until $h_1^{'...'}$ could not find a genotype to infer.

3. Pick next element of HapT, and do the same process as #1 and #2.

4. Repeat #3, until the last element of HapT, h_{hnum}, is picked and compared with genotypes.

This procedure is described in Figure 2.1

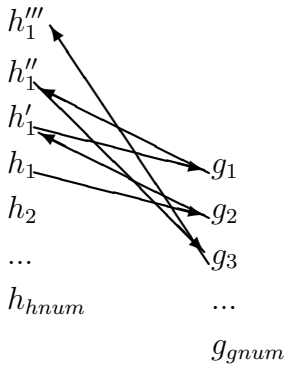


Figure 2.1: Resolving order of Clark Algorithm 1. Start with a haplotype to find a genotype which could be resolved with the haplotype. If there exist such a genotype, make a complementary haplotype and continue resolving with the haplotype until there is no genotype found which could be resolved with the complementary haplotype. Repeat the same process until all genotypes are resolved or there is no more genotypes which could be resolved with the haplotypes made so far. Starting with h_1 to find g_2, which could be resolved with h_1 and make complementary h_1'. Then, continue resolving with h_1' to find g_1 and make complementary h_1'', and so on

2.2.3 Clark Algorithm 2

The second rule of pairing is as follows.

1. Pick the first haplotype h_1 and find all the genotypes, suppose they are g_2 and g_{num}, from GenoT which could be inferred from h_1.

2. Make corresponding complementary haplotypes, h'_1 and h''_1, for each of the found genotypes.

3. Remove resolved genotype from GenoT and insert new haplotypes to HapT

4. Pick the next haplotype from HapT, h_2, and do the same process as #1, #2 and #3.

5. Repeat #4 until the last haplotype, h_{hnum} is picked and compared with genotypes.

This rule works since true haplotype and observed genotypes are not one-to-one. In other words, one haplotype may infer many genotypes. This procedure is described in Figure 2.2

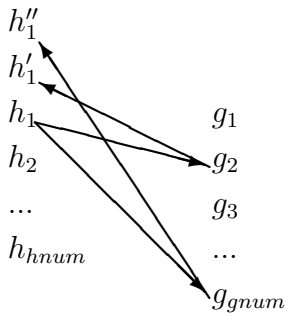


Figure 2.2: Resolving order of Clark Algorithm 2. Start with a haplotype to find all the genotypes which could be resolved with the haplotype. For all the found genotypes, make complementary haplotypes. Repeat the same process with another haplotype until all genotypes are resolved or there is no more genotypes which could be resolved with the haplotypes made so far. Starting with h_1 to find g_2 and g_gnum to make complementary h'_1 and h''_1. Repeat with another haplotype until all genotypes are resolved or there is no more genotypes which could be resolved with the haplotypes made so far

2.2.4 Problems of Clark Algorithm

While the algorithm should work in principle, there are three problems that could be arisen. First, if there is no homozygous of single-site heterozygous at the beginning, the chain of inference could not be started. Second, there could be some genotypes that is not resolved at the end of the inference. There is some "global" structure to the set of true haplotypes that underlying the observed genotypes, so that if some early choices in the method incorrectly resolve some of the genotypes, then the method will later becomes stuck, unable to resolve the remaining genotypes. Of course, if we run all the possible ways of choices, we can get the set of haplotypes that resolves all the genotypes, but it will take too much time, which seems impossible. Only a tiny fraction of all the possible data ordering can be tried. Clark suggested to run the algorithm numerous times with different ordering, and then the execution that resolved the most genotypes should be the one most trusted. The last problem that could be occur is that some haplotypes might be erroneously inferred if a combination of two haplotypes is identical to another true combination of two haplotypes. In summary,

1. One may fail to recover any homozygotes or single-site heterozygotes and may never get the cascade started

2. There any be unresolved haplotypes left at the end

3. Haplotypes might be erroneously inferred if a crossover product of two actual haplotypes is identical to another true haplotype

Now I will discuss the probability of the problems to occur and their solutions.

Problem1. Probability that the algorithm cannot get started

If there is no homozygous or single-site heterozygous genotype in the given genotypes set, we cannot start the algorithm to resolve the genotypes. However, this probability of no homozygous or single-site heterozygous is very small that it rarely happens in the real world.

As problem1 occur when none of the given genotype is homozygous-does not have ambiguous site- or single-site heterozygous-has only one ambiguous site-, we can get the probability by examining how many ambiguous sites there will be in the real world. As ambiguous sites are gained when there are mismatching sites between a pair of genes, by estimating the number of mismatching sites of diploids we can estimate the probability of ambiguous sites in the real world.

To examine this, we need some assumptions

- Infinite site model: A model of infinitely many alleles.
- Each mutation generates a novel allele which would provide an upper bound: As a gene consists of a large number of nucleotides in the real world(infinite site model), a mutation occurring at one nucleotide site will likely not result in a type already present in the population, but rather in a novel allele. However, if the mutation rate is homogeneous across a gene, doubling the size of a gene should double the total mutation rate, but because of the mutation-drift process generates correlations of heterozygosity across sites, this yields an upper bound for the value of θ. So, supposing that each mutation would generate a novel allele would then provide an upper bound for situations in which mutations could also result in alleles that are already present in the population
- Population evolves according to a one-locus, nuetral Wright-Fisher model

- Generations are not overlapping: In each generation, the entire population undergoes random mating
- Genes are drawn at random
- Population is randomly mating and in hardy-Weinberg equilibrium
- Diploid from natural population in steady state between mutation(gain of alleles) and drift(loss of alleles) so that population size is constant

Also I will introduce new notations and formulas

- L: Number of nucleotides in a DNA sequence
- μ: Neutral mutation rate per nucleotide site per generation
- N: Effective population size
- θ_t: θ_t=$4N\mu$
- θ: Expected number of mismatching sites for the DNA sequence of L nucleotides. Under the infinite-site model, $\theta = L\theta_t$
- The distribution of the number of mismatching sites expected when two genes are drawn from a population

$$\Pr(\text{2 sequences have } m \text{ mismatches}) = (\frac{1}{\theta+1})(\frac{\theta}{\theta+1})^m \tag{2.1}$$

- F: The probability that two genes will be identical. From (2.1), we get

$$\text{F=Pr(2 sequences have 0 mismatches)} = (\frac{1}{\theta+1}) \tag{2.2}$$

- Ewens sampling formula:

- Equilibrium properties of samples taken from a population that evolves according to the infinite allele model
- a_i: The number of alleles present exactly i times in a sample
- $(a_1, a_2, ...)$: The allelic partion
- n: The sample size which is the number of gametes($\frac{n}{2}$ diploids)
  $n = \sum_{i=1}^{n} i a_i$
- K_n: The number of different alleles(The number of types of alleles) in a sample size n
  $K_n = \sum_{i=1}^{n} a_i$
- $P_\theta(a_1, a_2, ..., a_n)$: The distribution of the allelic partition of a sample in equilibrium in the diffusion limit

$$P_\theta(a_1, a_2, ..., a_n) = \frac{n!}{\theta_{t(n)}} \prod_{j=1}^{n} (\frac{\theta_t}{j})^{a_j} \frac{1}{a_j!} \tag{2.3}$$

  where, $\theta_{t(n)} = \theta_t(\theta_t + 1)...(\theta_t + n - 1)$

Under the the assumptions and using the notations and formulas, first we will see the case of a sample of two diploids, then we will see the case of n haploids($\frac{n}{2}$ diploids). As we are drawing samples without replacement, subsequent samples are not independent of one another, so that calculating probability of drawing n diploids and getting no homozygotes is complicated. To calculate the probability, we must exhaustively enumerate all possible configurations of alleles in a sample and determine their probabilities with Ewens sampling formula.

In the case of a sample of two diploids, the probability that the algorithm cannot start, in other words, the probability that there will be no homozygous or single-site heterozygous is as follows

The configuration of two diploids will be

$\{A_1/A_2, A_3/A_4\}, \{A_1/A_2, A_2/A_3\}, \{A_1/A_2, A_1/A_2\}, \{A_1/A_2, A_3/A_3\},$

$\{A_1/A_1, A_2/A_2\}, \{A_1/A_2, A_3/A_4\}$, and $\{A_1/A_2, A_3/A_4\}$

Only the first three of these configurations lack homozygotes, and we can get their expected probabilities using (2.3),

$$
\begin{aligned}
\{A_1/A_2, A_3/A_4\} : a_1 = 4, a_2 = a_3 = a_4 = 0 \\
& k_4 = \textstyle\sum_{i=1}^{4} a_1 + a_2 + a_3 + a_4 = 4 + 0 + 0 + 0 = 4 \\
& n = \textstyle\sum_{i=1}^{4} 1 \cdot a_i = 1 \cdot a_1 + 2 \cdot a_2 + 3 \cdot a_3 + 4 \cdot a_4 \\
& = 1 \times 4 + 0 + 0 + 0 = 4 \\
& P_\theta(a_1, a_2, a_3, a_4) = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \textstyle\prod_{j=1}^{4} (\tfrac{\theta_t}{j})^{a_j} \tfrac{1}{a_j!} \\
& = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \times (\tfrac{\theta_t}{1})^{a_1} \tfrac{1}{a_1!} \times (\tfrac{\theta_t}{2})^{a_2} \tfrac{1}{a_2!} \times (\tfrac{\theta_t}{3})^{a_3} \tfrac{1}{a_3!} \times (\tfrac{\theta_t}{4})^{a_4} \tfrac{1}{a_4!} \\
& = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \times (\tfrac{\theta_t}{1})^{4} \tfrac{1}{4!} \times (\tfrac{\theta_t}{2})^{0} \tfrac{1}{0!} \times (\tfrac{\theta_t}{3})^{0} \tfrac{1}{0!} \times (\tfrac{\theta_t}{4})^{0} \tfrac{1}{0!} \\
& = \tfrac{\theta_t^3}{(\theta_t+1)(\theta_t+2)(\theta_t+3)} \\
\{A_1/A_2, A_2/A_3\} : a_1 = 2, a_2 = 1, a_3 = a_4 = 0 \\
& k_4 = \textstyle\sum_{i=1}^{4} a_1 + a_2 + a_3 + a_4 = 2 + 1 + 0 + 0 = 3 \\
& n = \textstyle\sum_{i=1}^{4} 1 \cdot a_i = 1 \cdot a_1 + 2 \cdot a_2 + 3 \cdot a_3 + 4 \cdot a_4 \\
& = 1 \times 2 + 2 \times 1 + 0 + 0 = 4 \\
& P_\theta(a_1, a_2, a_3, a_4) = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \textstyle\prod_{j=1}^{4} (\tfrac{\theta_t}{j})^{a_j} \tfrac{1}{a_j!} \\
& = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \times (\tfrac{\theta_t}{1})^{a_1} \tfrac{1}{a_1!} \times (\tfrac{\theta_t}{2})^{a_2} \tfrac{1}{a_2!} \times (\tfrac{\theta_t}{3})^{a_3} \tfrac{1}{a_3!} \times (\tfrac{\theta_t}{4})^{a_4} \tfrac{1}{a_4!} \\
& = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \times (\tfrac{\theta_t}{1})^{2} \tfrac{1}{2!} \times (\tfrac{\theta_t}{2})^{1} \tfrac{1}{1!} \times (\tfrac{\theta_t}{3})^{0} \tfrac{1}{0!} \times (\tfrac{\theta_t}{4})^{0} \tfrac{1}{0!} \\
& = \tfrac{6\theta_t^2}{(\theta_t+1)(\theta_t+2)(\theta_t+3)} \\
\{A_1/A_2, A_1/A_2\} : a_2 = 2, a_1 = a_3 = a_4 = 0 \\
& k_4 = \textstyle\sum_{i=1}^{4} a_1 + a_2 + a_3 + a_4 = 0 + 2 + 0 + 0 = 2 \\
& n = \textstyle\sum_{i=1}^{4} 1 \cdot a_i = 1 \cdot a_1 + 2 \cdot a_2 + 3 \cdot a_3 + 4 \cdot a_4 \\
& = 0 + 2 \times 2 + 0 + 0 = 4 \\
& P_\theta(a_1, a_2, a_3, a_4) = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \textstyle\prod_{j=1}^{4} (\tfrac{\theta_t}{j})^{a_j} \tfrac{1}{a_j!} \\
& = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \times (\tfrac{\theta_t}{1})^{a_1} \tfrac{1}{a_1!} \times (\tfrac{\theta_t}{2})^{a_2} \tfrac{1}{a_2!} \times (\tfrac{\theta_t}{3})^{a_3} \tfrac{1}{a_3!} \times (\tfrac{\theta_t}{4})^{a_4} \tfrac{1}{a_4!} \\
& = \tfrac{4!}{\theta_t(\theta_t+1)(\theta_t+2)(\theta_t+3)} \times (\tfrac{\theta_t}{1})^{0} \tfrac{1}{0!} \times (\tfrac{\theta_t}{2})^{2} \tfrac{1}{2!} \times (\tfrac{\theta_t}{3})^{0} \tfrac{1}{0!} \times (\tfrac{\theta_t}{4})^{0} \tfrac{1}{0!} \\
& = \tfrac{3\theta_t}{(\theta_t+1)(\theta_t+2)(\theta_t+3)}
\end{aligned}
$$

The probability of obtaining no homozygotes in a sample of two diploids is the sum of these probabilities

$$\Pr(\text{no homozygotes}) = \frac{\theta_t^3 + 6\theta_t^2 + 3\theta_t}{(1+\theta)(2+\theta)(3+\theta)} \tag{2.4}$$

For larger sample size, as we did for the two diploid case, we must determine the probability of all partitions of all configurations of alleles having no homozygotes, which is hard to do because there could be too many such configurations. However, for large population sizes(largeθ_t), successive samples from a population become nearly independent so that we can use formula (2.2) instead of listing all those configurations. As we have seen at (2.2), the chance of drawing a homozygote is $F = \frac{1}{1+\theta_t}$ and the chance of drawing a single-site heterozygote is $F = \frac{\theta}{(1+\theta_t)^2}$. As a result, the probability of having no homozygote or single-site heterozygote is Pr(probability to fail to start the algorithm)$\approx [a - \frac{1}{1/theta} - \frac{\theta}{(a+\theta)^2}]^n$. This probability is very small as even when θ is as large as 10, the chance of the failing to get the algorithm started is$< 1\%$[1]. To solve this problem, we need more sampling until we get a homozygote or a single-site heterozygote. Once, such a sample is gained, we can get started with the algorithm.

In summary,

- Probability of problem1: Pr(probability to fail to start the algorithm)$\approx [a - \frac{1}{1/theta} - \frac{\theta}{(a+\theta)^2}]^n$
- Result: As the sample size grow and θ gets smaller, the probability of the failing to get the algorithm started gets larger. Also when the number of mismatches is more than around 10, as the number of mismatches grows, the probability of the failing to get the algorithm started gets larger.
- Solution: Sample more individuals until we get a homozygote or a single-site

heterozygote. Once, such a sample is gained, we can get started with the algorithm.

Problem2. Probability of Orphaned Alleles

If a genotype $g_i \oplus g_j$ is found such that neither haplotype g_i nor g_j occurs in a homozygote which is observed or inferred by any other resolved heterozygote, then these haplotypes cannot be resolved and will be referred to as "orphans". Simulations were performed to explore the fraction of times that orphans will be encountered under a range of values of θ and a range of sample sizes. Draw samples of $2n$ gametes from the frequency distribution expected under the infinite-allele model, then combine these $2n$ gametes to form n diploid genotypes. Homozygotes were identified, and paths connecting alleles were constructed, in an attempt to connect all alleles to a homozygote. If orphaned alleles remain, then this sample is scored as an orphaned sample. As a result, we can see two noteworthy trends. First,larger values of θ result in a higher chance of obtaining orphans: Within the same number of samples and length, larger value of θ means larger values of mutation rate, larger number of expected mismatching sites, greater allelic diversity

Second, as the sample size increases, orphans are less likely to remain: Within the same length of genotypes and the same values of θ, large samples are more likely to contain paths connecting all alleles to homozygotes. Suppose that there are a orphan remain when run the algorithm with n number of samples. As we get extra samples, with the extra sample, we might resolve the orphan, in other words, the chance to get a haplotype which could resolve the orphan increases. So the solution for the problem2 is to sample more individuals until all genotypes are resolved. As we have seen at Result part, we can remove orphans by larger the values of θ or increasing the sample size. We cannot increase value of θ artificially but we can increase the sample size by getting more samples to resolve orphans.

In summary,

- Orphan: A genotype which could not be resolved by a haplotype which is observed of inferred by any other observed genotype.
  For example, when the given genotype is {0000, 0220, 1122}, then first we can identify a homozygote 0000, and with the homozygote, we can get the algorithm started. Then with 0000 we can resolve a genotype 0220 to get another haplotype 0110. However, 1122 cannot be resolved by any of the haplotypes, 0000 or 0110, and remain orphan.

- Test: Simulations were performed to explore the fraction of times that orphans will be encountered under a range of values of θ and a range of sample sizes.

  1. Draw samples of $2n$ gametes from the frequency distribution expected under the infinite-allele model
  2. combine these $2n$ gametes to form n diploid genotypes
  3. Identify homozygotes
  4. Start the algorithm with the identified homozygotes, construct paths connecting alleles
  5. If orphaned alleles remain, then this sample is scored as an orphaned sample

- Result:

  1. Larger values of θ result in a higher chance of obtaining orphans: Within the same number of samples and length, larger value of θ means larger values of mutation rate, larger number of expected mismatching sites, greater allelic diversity
     For example,

2. As the sample size increases, orphans are less likely to remain: Within the same length of genotypes and the same values of θ, large samples are more likely to contain paths connecting all alleles to homozygotes. Suppose that there are a orphan remain when run the algorithm with n number of samples. As we get extra samples, with the extra sample, we might resolve the orphan, in other words, the chance to get a haplotype which could resolve the orphan increases.

- Solution: Sample more individuals until all genotypes are resolved: As we have seen at Result part, we can remove orphans by larger the values of θ or increasing the sample size. We cannot increase value of θ artificially but we can increase the sample size by getting more samples to resolve orphans.

Problem3. Probability of Anomalous Matches

A genotype might be erroneously resolved. For example, suppose one observe a genotype 0220 which is composed of two haplotypes, 0000 and 0110, in nature which is not observed. If one observed or inferred a haplotype 0010, one might resolve 0220 with 0010 to get 0100, which is not true haplotype. Moreover, 0100 might result other genotypes to cascade errors.

- Anomalous Matches: A Genotype is erroneously resolved so that a wrong haplotype is inferred as a complementary of the genotype. And this wrong haplotype might result other getnotypes to cascade the errors.

- Test:

1. Get sample using tree-based algorithm which gives sample from a steady-state population, given values of θ, n, and recombination rate across the sequence

2. With the sample, simulate infinite-site model with the Clark algorithm to get the fraction of unresolved anomalous matches and frequency of anomalous haplotypes.
3. Change θ, n or recombination rate for each simulation.
4. Go back to step2

- Result:

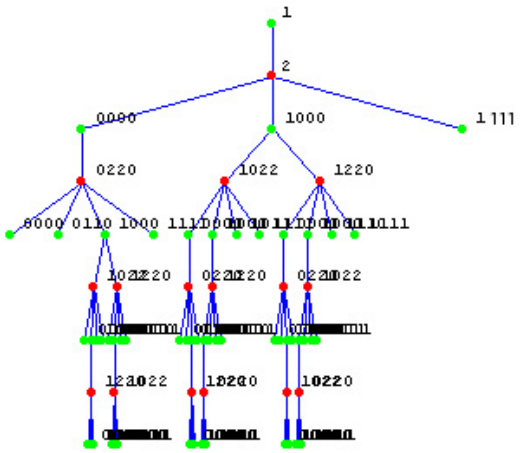
1. Larger sample size does not affect much of the result. Even though larger samples might encounter more genotypes that could anomalously match alleles, homozygotes that resolve the ambiguity are also more likely to be found in larger sample.
2. θ does not affect much of the result
3. Recombination rate does not affect much of the result. While recombination increases heterozygosity and the number of alleles, there is little effect of recombination on the expected heterozygosity. A higher heterozygosity assures that more of the common alleles are found in heterozygotes with rare alleles, so that, once the chain gets started, it continues to resolve more alleles
4. The solution with the fewest orphans resolves the greater number of true haplotypes(Parsimony Rule): In no case false complementary haplotypes found, and in every case in which anomalous matches were obtained, that solution had orphans. When anomalous matches occur, there left many orphans which demonstrates a parsimony rule and suggest that when a solution resolves all haplotypes it is likely to be a unique solution.

- Solution: Run the Clark algorithm several time with difference ordering of inference to get a better solution which leaves fewer orphans, in other words, resolves greater number of genotypes.

2.3 Clark Phasing Method Graph

As mentioned before, there could be many possible results according to the order of the resolving genotypes and haplotypes. Dependent on the order of resolved genotypes, the result haplotypes could be different. Also there could be multiple haplotypes which could resolves each genotypes so that dependent on the ordering of the haplotypes that possibly resolves each genotype, the resulted haplotype that resolves given genotypes could be different. To see all possible result that could be drawn dependent on the order of genotypes and haplotypes that resolves each genotype, I drew a tree graph that contains all possible pathes of resolving and report all the genotype routes that resolves the most number of genotypes. This could be helpful to understand and improve a phasing method. Figure 2.3 and Figure 2.4 shows examples of this tree graph. Figure 2.3 is an example of given genotypes 0000,0220,1000,1022,1111 and 1220. The tree graph shows all the possible routes of phasing. From the graph, we can see that order of resolving genotypes and pairing haplotypes for each genotypes are important to prevent the problems of Clark and get a true phasing. And Figure 2.4 shows another example of Clark Phasing Method Graph with genotypes 0000, 1000, 2200 and 1122. At the first step of Clark Method, homozygous and single heterozygous genotypes, 0000 and 1000 , are set to initial haplotypes so that there are two genotypes, 2200 and 1122 left to be resolved. From the Clark Phasing Method Graph above, we can see that the results are different dependent on the order of resolved genotypes and haplotypes. If first 1000 resolves 2200, we get new haplotype 0100. All three haplotype 1000, 0100 and 0000 cannot solve 1122 so this genotype cannot be resolved. This is reported as Problem2. However, if we change order so that first 0000 resolves 2200 we get new haplotype 1100, which can resolve 1122 to create 1111. Then all genotypes can be resolved. The implementation of the Clark phasing method graph is in SectionII part.

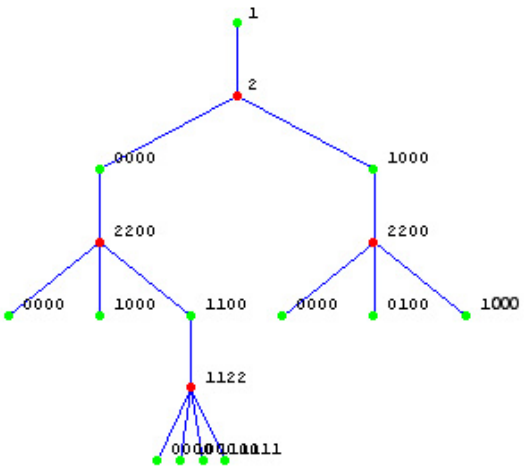
```
--------------------------------------------------------------------------------------------
Find all possible routes to resolve the genotypes
--------------------------------------------------------------------------------------------
Given genotypes= {{0, 0, 0, 0}, {0, 2, 2, 0}, {1, 0, 0, 0}, {1, 0, 2, 2}, {1, 1, 1, 1}, {1, 2, 2, 0}}
<Clark Phasing Graph>
Given GenoT= {{0, 0, 0, 0}, {0, 2, 2, 0}, {1, 0, 0, 0}, {1, 0, 2, 2}, {1, 1, 1, 1}, {1, 2, 2, 0}}
Initial HapTInit={{0, 0, 0, 0}, {1, 0, 0, 0}, {1, 1, 1, 1}}
Initial GenoT={{0, 2, 2, 0}, {1, 0, 2, 2}, {1, 2, 2, 0}}
< Draw all possible routes for Clark Phasing Method >
```




```
< Routes that resolve the most genotypes >
Route 1. {{0, 2, 2, 0}, {1, 0, 2, 2}, {1, 2, 2, 0}}
Route 2. {{0, 2, 2, 0}, {1, 2, 2, 0}, {1, 0, 2, 2}}
Route 3. {{1, 0, 2, 2}, {0, 2, 2, 0}, {1, 2, 2, 0}}
Route 4. {{1, 0, 2, 2}, {1, 2, 2, 0}, {0, 2, 2, 0}}
Route 5. {{1, 2, 2, 0}, {0, 2, 2, 0}, {1, 0, 2, 2}}
Route 6. {{1, 2, 2, 0}, {1, 0, 2, 2}, {0, 2, 2, 0}}
```

Figure 2.3: Clark Phasing Method Graph for genotypes 0000,0220, 1000, 1022, 1111 and 1220.

```
-------------------------------------------------------------------------------------------
Find all possible routes to resolve the genotypes
-------------------------------------------------------------------------------------------
Given genotypes= {{0, 0, 0, 0}, {1, 0, 0, 0}, {1, 1, 2, 2}, {2, 2, 0, 0}}
<Clark Phasing Graph>
Given GenoT= {{0, 0, 0, 0}, {1, 0, 0, 0}, {1, 1, 2, 2}, {2, 2, 0, 0}}
Initial HapTInit={{0, 0, 0, 0}, {1, 0, 0, 0}}
Initial GenoT={{1, 1, 2, 2}, {2, 2, 0, 0}}
< Draw all possible routes for Clark Phasing Method >
```




```
< Routes that resolve the most genotypes >
Route 1. {{2, 2, 0, 0}, {1, 1, 2, 2}}
```

Figure 2.4: Clark Phasing Method Graph for genotypes 0000, 1000, 2200 and 1122.

2.4 Discussion

Clark algorithm provides the first idea of haplotype phasing and noticeable to understand to develop other haplotype phasing algorithms. Also the running time is fast compared to other algorithms. However, Clark algorithm has three problems described above and as the problems have some probabilities to occur. Even though there are some solutions for those problems, also described above, the solutions may not possible for some cases and are not enough to provide a perfect phasing algorithm. In addition, as Clark suggested, several number of trials with different order of pairing the genotypes might help to prevent some problems, still problem 1 might happen, so that we need better algorithm to correct those problems and improve the performance of phasing.

Chapter 3

Parsimony method

In this chapter, I made algorithm which focus on the Pure parsimony haplotyping(PPH) approach, the goal of which is to find a minimum set of haplotypes that explains them. The Pure Parsimony criteria reflects the fact that in natural population, the number of distinct haplotypes observed in a population is vastly smaller than the number of combinatorially possible haplotypes, and this is also expected from population genetic theory, population bottleneck effects and genetic drift. For example, Patil et al. report that within short genomic regions, typically, some 70-90 percent of the haplotypes belong to very few(2-5) common haplotypes. As a result it is noticeable to try to find the minimum set of haplotypes that resolves a given set of genotypes. Here, I developed an algorithm which follows the pure parsimony criteria and for that I used the Clark Consistency Graph. Detail of the Clark Consistency Graph is in Section1. Both the time needed for a solution, and the accuracy of the solution, depend on the level of recombination in the input string. The speed of the solution improves with increasing recombination, but the accuracy of the solution decreases with increasing recombination.

3.1 Definition and Notation

- Pure Parsimony Haplotyping Approach: Find a solution to the Haplotype Inference(HI) problem that minimizes the total number of distinct haplotypes used

- Clark Consistency Graph: Each genotype consists a node and if two genotypes can be resolved using same sequence, there is an edge. In other words, if two genotypes has same values of 0 or 1 at all the sites where both genotypes does not have value 2, there is an edge between those genotypes. And the edge is labeled as follows. If one of the genotypes has 1, the label has 1 in that site. If one of the genotypes has 0, the label has 0 in that site. If both genotypes have 2, the label has 2 in that site.

  Nodes: Genotypes

  Edges: There is an edge between g_1 and g_2, if $\exists h$ that can explain both genotypes as it has other two haplotypes h', h'',such that$\{h, h'\}$ explain $g_1, \{h, h''\}$ explain g_2

```
-------------------------------------------------------------------------------------------
Clark consistency graph of the genotypes
-------------------------------------------------------------------------------------------
Given genotypes= {{1, 0, 1, 0}, {1, 1, 1, 1}, {1, 1, 2, 2}, {1, 2, 1, 2}}
<Clark Consistency Graph>
Given GenoT={{1, 0, 1, 0}, {1, 1, 1, 1}, {1, 1, 2, 2}, {1, 2, 1, 2}}
```

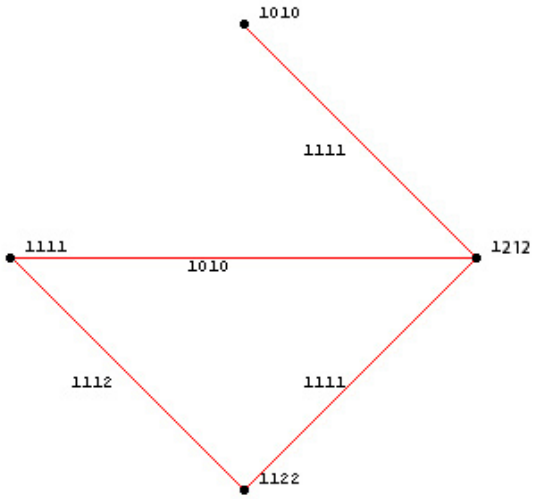


Figure 3.1: Clark Consistency Graph for genotypes 1111, 1122, 1010 and 1212

3.2 Algorithm

Idea: Phase each genotype with haplotypes which is shared the most between all the genotypes. We can get the number of genotypes for each haplotype using Clark Consistency Graph. Input: Genotype data set
Output: Haplotype data set
Algorithm:

1. Initialize haplotype set: Find homozygous genotypes and single-site heterozygous genotypes. For each single-site heterozygous genotype, generate a pair of haplotypes by replacing the single heterozygous site to 0 and 1. Put those sequences to haplotype set and remove from genotype set. Count the number of each kind of sequences which will be used in #2. Clark Consistency Graph set.

2. Generate Clark Consistency Graph set: This set will contain the shared haplotypes and haplotypes that is already picked. Also this set has count for each element that indicates the number of genotypes shares each haplotype(element). Initialize with haplotype set which is made at #1. Then compare each pair of genotypes, if there is an edge, put the edge to the Clark Consistency Graph set. - For a pair of genotypes, there is an edge if two genotypes has same values of 0 or 1, for sites which both are not 2 . Edge is generated as follows. If both site is 2 put 2, else if one site is 2 and the other is 0(1), put 0(1), else if both sites is 0(1), put 0(1) -When put each edges, if there are sites with 2, create all possible combinations by replacing site 2 to 0 and 1. One more thing to do, when putting edges(combinations) to the set is, if there are same sequence of edges(combinations), count the number of each kinds of edges(combinations). This is important since the count indicates the number of genotypes which shares the same kinds of edge(combination)

3. Generate a pair of haplotypes for each genotype: There are $2^{(k-1)}$ possible pairs of haplotypes for each genotype when the number of site with value 2 is k. Among them, we should pick a pair of haplotypes to resolve each genotype which satisfies the parsimony haplotyping(minimize the total number of haplotypes). To do this, for each genotype, generate all possible $2^(k-1)$ pairs of haplotypes and compare the counts from Clark Consistency Graph and pick a pair of haplotypes which has the highest score(count). The picked haplotypes will be added to the haplotype set. The way of scoring is,

   (a) If both haplotypes of a pair occurs in the Clark Consistency Graph set(includes edges of Clark Consistency graph and already picked haplotypes), pick the pair. This means both haplotypes are shared with other genotypes so that if other genotypes which share the haplotypes, pick these haplotypes later, it will not increase the number of haplotypes. This is parsimony haplotyping. But what if there is many pair of haplotypes that both haplotypes are occurred in Clark Consistency Graph set? The answer is, pick the one that has the highest sum of counts of two haplotypes. But what of there are multiple pairs of haplotypes with same highest sum of scores? Put off this genotype so that after resolving other genotypes, do this resolving process again. This works since, the element of Clark Consistency Graph set and the count of each element will be changes since the each haplotype's count decreases if they are not picked. What if after resolving all other genotypes it still has the same problem? Randomly pick one pair which has the highest score. This is also parsimony haplotyping since it increases the same number of haplotypes considering total number of haplotypes. For the rest of the haplotypes which occurred in the Clark Consistency Graph but not picked, decrease the count in the Clark Consistency Graph set. This is

because the genotype did not pick the haplotype which means one genotype no longer shares the haplotype but uses other haplotype to resolve itself. So if other genotype use this haplotype later, it may increase the number of haplotypes unless other third genotype shares the haplotype.

(b) If there is no pair which both haplotypes are occurred in the Clark Consistency Graph set, check whether there is pairs that one of the haplotype of the pair is occurred in the Clark Consistency Graph set. If there is one, pick the pair. This means one haplotype is shared with other genotypes so that if other genotypes which share the haplotype, pick this haplotypes later, it will not increase the number of haplotypes. Here, put the haplotype which is not occurred in the Clark Haplotype Graph Set to the Clark Haplotype Graph Set with count 1. This allows other genotypes to use this haplotype considering it is shared with a genotype which is resolved, and this will not increase the number of haplotypes. But what if there are many such pairs? Pick the one which has the highest score(count). What if there are multiple pairs of haplotypes which have the same highest score? Put off this genotype so that after resolving other genotypes, do this process again. What if after resolving all other genotypes it still has the same problem? Randomly pick one pair which has the highest score. For rest of the haplotypes which occurred in the Clark Consistency Graph but not picked, decrease the count in the Clark Consistency Graph set.

(c) If all the pairs are not occurred in the Clark Consistency Graph set, Put off this genotype so that after resolving other genotypes, do this process again. What if after resolving all other genotypes it still has the same problem? Randomly pick one pair which has the highest score. Here, put both haplotypes to the Clark Haplotype Graph Set with count

1. This allows other genotypes to use this haplotype considering it is shared with a genotype which is resolved, and this will not increase the number of haplotypes. For rest of the haplotypes which occurred in the Clark Consistency Graph but not picked, decrease the count in the Clark Consistency Graph set.

You can see examples with detailed description in Chapter 3 of PartII Section.

3.3 Discussion

The Parsimony Phasing method that I introduced in this Chapter improved Clark Phasing method in a sense that it solve the problem1 of Clark Algorithm so that it can start algorithm even if there is no homozygote or single-site heterozygote at the beginning. In addition, it picks haplotype pairs parsimoniously than Clark Algorithm. Clark picks randomly cascading from one resolved haplotype to another so that we cannot say that Clark Algorithm is truly parsimonious algorithm. However, the algorithm I introduced in this Chapter resolves a genotype with haplotype which could be most shared with other genotypes. Which makes it possible to satisfied the parsimony criteria. However, this algorithm's performance, number of haplotypes that resolves the given genotype and run time, still highly dependent on the pattern of the data set. This is because, time for the algorithm depends on the range of the region(number of SNPs), number of ambiguous sites and number of NN sites. For wider region, more number of ambiguous sites(number of sites with value 2) and less NN sites, it took more time and visa versa.

Chapter 4

EM method

In this chapter, I will introduce a different approach and place the problem of estimating haplotype frequencies in the general framework of the EM algorithm. I will introduce general EM algorithm first, then explain how to apply the algorithm to haplotype phasing.

4.1 Definition and Notation

To explain EM algorithm, we need to change the definition of genotype and we also need a new notation phenotype which is a particular combination of two multilocus haplotypes.

- haplotype: A homozygous sequence which is not observed
- genotype: A particular combination of two multilocus haplotypes which composes a phenotype
- phenotype: An observed multilocus genotype whose haplotype phase is unknown a priori
- n: The number of phenotypes

- m: The number of different types of phenotype
- n_j: The number of phenotypes per each different type of phenotype $j = 1, 2, ..., m$

$$n = \sum_{j=1}^{m} n_j \tag{4.1}$$

- h: The number of different types of haplotype
- h_k: Indicator of a specific haplotype k

  h_l: Indicator of a specific haplotype l
- s_j: The number of heterozygous loci of jth phenotype. $j = 1, 2, ..., m$
- c_j: The number of genotypes leading to $j$the phenotype. $j = 1, 2, ..., m$

$$c_j = \begin{cases} 2^{s_j - 1}, & \text{if } s_j > 0, \\ 1, & \text{if } s_j = 0 \end{cases} \tag{4.2}$$

- $p_t^{(g)}$: The probability of tth haplotype at time g . $t = 1, 2, ..., h$. As the probabilities are sum up to 1

$$p_1 + p_2 + ... + p_h = 1 \tag{4.3}$$

- $\widetilde{P}_{ji}(h_k h_l)^{(g)}$: The probability of ith genotype of jth phenotype, composed haplotype k and l, at time g

$$\widetilde{P}_{ji}(h_k h_l)^{(g)} = \begin{cases} (p_k^{(g)})^2, & \text{if } k = l, \\ 2p_k p_l, & \text{if } k \neq l \end{cases} \tag{4.4}$$

- $P_j^{(g)}$: The probability of jth phenotype at time g

$$P_j^{(g)} = \sum_{i=1}^{c_j} \widetilde{P}_{ji}(h_k h_l)^{(g)} \tag{4.5}$$

- $P_{ji}(h_k h_l)^{(g)}$: The normalized probability of ith genotype of jth phenotype, composed of haplotype k and l, at time g

$$P_{ji}(h_k h_l)^{(g)} = \frac{n_j}{n} \frac{\widetilde{P}_{ji}(h_k h_l)^{(g)}}{P_j^{(g)}} \tag{4.6}$$

- δ_{jit}: The number of times that haplotype t is present in ith genotype of phenotype j

$$p_t^{(g+1)} = \frac{1}{2} \sum_{j=1}^{m} \sum_{i=1}^{c_j} \delta_{jit} P_{ji}(h_k h_l)^{(g)} \tag{4.7}$$

- ε: Indicator of stop condition.

- f: The time when the iteration stops

4.2 Algorithm

4.2.1 General EM Algorithm

First, I will introduce a general EM Algorithm and show the application of EM Algorithm in next section.

An expectation-maximization(EM) algorithm is used in statistics for finding maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved hidden variables which is not directly observed but are rather inferred from other variables that are observed and directly measured. EM alternates between performing an expectation(E) step, which computes an expectation of the likelihood by including the hidden variables as if they were observed, and a maximization(M) step, which computes the maximum likelihood estimates of the parameters by maximizing the expected likelihood found on the E step. The parameters found on the M step are then used to begin another E step, and the process is repeated.

Let y denote incomplete data consisting of values of observable variables and z the missing data. Together, z and y form the complete data. z can either be actual missing measurements or a hidden variable that would make the problem easier if its value were known.

Let p be the joint probability distribution(continuous case) or probability mass function(discrete case) of the complete data with parameters given by the vector θ This function can also be considered as the complete data likelihood, that is, it can be thought of as a function of θ Further, note that the conditional distribution of the missing data given the observed can be expressed as

$$p(z|y,\theta) = \frac{p(y,z|\theta)}{p(y|\theta)} = \frac{p(y|z,\theta)p(z|\theta)}{\int p(y|\hat{z},\theta)p(\hat{z}|\theta)d\hat{z}} \tag{4.8}$$

This formulation only requires knowledge of the observation likelihood given the unobservable data $p(y|z,\theta)$ and the probability of the unobservable data $p(z|\theta)$.

An EM algorithm iteratively improves an initial estimate θ_0 by constructing new estimates θ_1, θ_2, and so on. An individual re-estimation step that derives θ_{n+1} from θ_n takes the following form:

$$\theta_{n+1} = \arg\max_{\theta} Q(\theta) \tag{4.9}$$

where $Q(\theta)$ is the expected value of the log-likelihood. In other words, we do not know the complete data, so we cannot say what is the exact value of the likelihood, but given the data that we do know(the y's), we can find a posteriori estimates of the probabilities for the various values of the unknown z's. For each set of z's there is a likelihood value for θ, and we can thus calculate an expected value of the likelihood with the given values of y's and which depends on the previously assumed value of θ because this influenced the probabilities of the z's.

Q is given by

$$Q(\theta) = \sum_{z} p(z|y,\theta_n) \log p(y,z|\theta) \tag{4.10}$$

where it is understood that this denotes the conditional expectation of being taken with the θ used in the conditional distribution of fixed at θ_n. The log of the likelihood is often used instead of true likelihood because it leads to easier formulas, but still attains its maximum at the same point as the likelihood. In other words, θ_{n+1} is the value that maximizes(M) the conditional expectation(E) of the complete data log-likelihood given the observed variables under the previous parameter value. Speaking of an expectation(E) step is a bit of a misnomer. What is calculated in the first step are the fixed, data-dependent parameters of the function Q. Once the parameters of Q are known, it is fully determined and is maximized in the second(M) step of an EM algorithm.

4.2.2 Applying EM Algorithm to Haplotype Phasing

Now let's apply EM Algorithm described above to haplotype phasing.
The probability of a sample of n individuals conditioned by the phenotype frequencies $P_1, P_2, ..., P_m$ is given by the multinomial probability

$$P(\text{sample}|P_1, P_2, ..., P_m) = \frac{n!}{n_1!n_2!...n_m!} \times P_1^{n_1} \times P_2^{n_2} \times ... \times P_m^{n_m} \tag{4.11}$$

where m and n_i follows the definitions in notation section.
Under the assumption of random mating, the probability P_j of the jth phenotype at time g is given by the sum of the probabilities of each of the possible c_j genotypes, following equation (4.5)

$$P_j^{(g)} = \sum_{i=1}^{c_j} \widetilde{P}_{ji}(h_k h_l)^{(g)}$$

Substituting equation (4.5) in equation (4.11), we obtain the probability of the sample as a function of the unknown haplotype frequencies. Therefore, the likelihood of the haplotype frequencies given phenotypic counts is

$$L(p_1, p_2, ..., p_h) = a_1 \prod_{j=1}^{m} (\sum_{i=1}^{c_j} \widetilde{P}_{ji}(h_k h_l))^{n_j} \tag{4.12}$$

where $p_h = 1 - p_1 - p_2 - ... - p_{h-1}$ as equation(1.3) and $a_1$is a constant incorporating the multinomial coefficient.

In principle, the maximum likelihood(ML) estimates of haplotype frequencies could be found analytically by solving a set of $h-1$ equations involving first partial derivatives of the logarithm of the likelihood, generally called scores. If U_t represents the score for the$t$th haplotype

$$U_t = \frac{\partial logL}{\partial p_t} = \sum_{j=1}^{m} \frac{n_j}{P_j} \frac{\partial P_j}{\partial p_t} \tag{4.13}$$

then, the setting the scores for the $h-1$ functionally independent haplotypes to zero and solving the resulting set of equations would lead to the ML estimates, but this procedure is tedious when h is large, and the number h is often unknown priori. As a result, we need alternative procedure involving numerical iterations. In next section I will show an EM algorithm extended to an arbitrary number of loci with an arbitrary number of alleles,allowing the treatment of DNA sequence and highly variable loci that estimates the haplotype frequencies which maximize the sample probability.

4.2.3 EM Algorithm on Haplotype Phasing

1. Initial conditions

   To avoid local maxima, we better perform the algorithm for a set of widely different initial values.

   There are several ways to construct initial conditions.

   - Make all haplotypes are equally likely

   $$p_t^{(0)} = \frac{1}{h}, t = 1, 2, ..., h \tag{4.14}$$

   - Make all possible genotypes for each phenotype are equally likely

   $$P_{ji}(h_k h_l)^{(0)} = \frac{1}{m}\frac{1}{c_j}, j = 1, 2, ..., m, i = 1, 2, ..., c_j \tag{4.15}$$

- Randomly choose haplotype frequencies that satisfies

$$\sum_{t=1}^{h} p_t^{(0)} = 1 \tag{4.16}$$

2. Expectation step(E-step) Using equation(4.4), for each phenotype and their genotypes, calculate the probability of ith genotype of jth phenotype.

$$\widetilde{P}_{ji}(h_k h_l)^{(g)} = \begin{cases} (p_k^{(g)})^2, & \text{if } k = l, \\ 2p_k p_l, & \text{if } k \neq l \end{cases}$$

3. Maximization step(M-step)

   (a) Using equation(4.5), calculate the probability of jth phenotype at time g

$$P_j^{(g)} = \sum_{i=1}^{c_j} \widetilde{P}_{ji}(h_k h_l)^{(g)}$$

   (b) Using equation(4.6), for each phenotype and its genotypes, calculate normalized probability of ith genotype of jth phenotype.

$$P_{ji}(h_k h_l)^{(g)} = \frac{n_j}{n} \frac{\widetilde{P}_{ji}(h_k h_l)^{(g)}}{P_j^{(g)}}$$

   As they are normalize, they sum up to 1

$$\sum_{j=1}^{m} \sum_{i=1}^{c_j} P_{ji}(h_k h_l)^{(g)} = \sum_{j=1}^{m} \sum_{i=1}^{c_j} \frac{n_j}{n} \frac{\widetilde{P}_{ji}(h_k h_l)^{(g)}}{P_j^{(g)}} = 1$$

   (c) For each haplotype, calculate the probability using equation(4.7).

$$p_t^{(g+1)} = \frac{1}{2} \sum_{j=1}^{m} \sum_{i=1}^{c_j} \delta_{jit} P_{ji}(h_k h_l)^{(g)}$$

   By equation(4.3), the probabilities of all haplotypes sum up to 1

$$\sum_{t=1}^{h} p_t^{(g+1)} = 1$$

4. Stop condition

   If the sum of the absolute value of differences of haplotype frequencies between consecutive runs, are less than ε, in other words converges small enough to the given threshhold, stop, otherwise go to ♯2

$$\begin{cases} \text{stop}, & \text{if } \sum_{t=1}^{h} |p_t^{(g)} - p_t^{(g+1)}| < \varepsilon \\ \text{go to } \#2, & \text{otherwise} \end{cases} \tag{4.17}$$

5. When updated frequencies satisfies stop condition(this time is f), we get the frequencies of every type of haplotypes, which are
   $p_t^f, t = 1, 2, ..., h$

6. For each phenotype, pick a genotype(a pair of haplotype)that has the maximum value of product of haplotype frequencies (haplotype that composes the genotype), from all possible genotypes that leads to the phenotype.

Examples with detailed description are in Chapter 5 of PartII Section.

4.3 Discussion

The EM algorithm is arguably the most popular statistical algorithm, because of its interpretability and stability. The output of the EM algorithm, if not trapped in a local mode, is the maximum-likelihood estimate (MLE), which possesses well-established statistical properties. The EM Method also solved some of the problems of Clark Method such as it always can start the algorithm and always can resolve all genotypes. Compared to the Clark method and parsimony method, the EM approach is a deterministic procedure, generally(not always)requires less computing time, and is easier for convergence check. Here, the running time is much dependent on the epsilon value so that by reducing the epsilon value, running time could be reduced a lot. This is because the number of iterations, which takes most of the time,

is dependent on the epsilon value. Also running time is dependent on the number of heterozygous site and size of the data set as other methods. If the sample size is small, EM algorithm showed better performance(small run time) than parsimony algorithm. However, as the sample size grows, EM algorithm took much more time than parsimony algorithm. EM algorithm and parsimony algorithm runs almost the same number of iteration, however, for EM algorithm, Mathematica takes a lot of time for calculating fraction numbers for frequencies. Moreover, EM algorithm needs a lot of memory space, which makes running time even worse. And from which we can see another problem for EM, which is that the capability of most EM-based approaches is restricted to approximately one dozen loci, because of the memory constraint. Another problem of EM Furthermore, it does not give the parsimony haplotype for every input genotypes. It concerns the maximum-likelihood, in other words the most frequent haplotypes, but not directly concerns the minimum number of total haplotypes. In other words, the maximum likelihood does not guarantee the total minimum number of haplotypes. For the parsimony algorithm, I increased the frequencies of picked haplotypes in each resolve of phenotypes. The EM algorithm also considers the frequencies when resolve each phenotype as parsimony algorithms does, however, this does not guarantee the parsimony. There are two reasons that EM does not guarantee the parsimony haplotyping.

1. First reason could be occur when the frequencies are the same at the last iteration. In this case, EM just pick one of the pairs(randomly choose or the first one), however, parsimony method does not pick random pair, but put off the genotype to resolve it later. Since the frequencies change as other genotypes are resolved(this is because picked haplotypes' frequencies are increased), after resolving other genotypes, we can pick a pair with higher frequency. Also, parsimony method picks a pair where both haplotypes has some extent of frequencies, but not the pair that one haplotype has high frequency and the other

haplotype has small frequency as EM picks. This makes the total number haplotypes smaller for parsimony method. This is described with an example in Figure 4.1. And the comparing result of each methods of the example is shown in Figure 4.2. EM algorithm results in the left hand side and parsimony algorithm results the right hand side. Which shows that for the given genotypes, parsimony method performs better in the sense of parsimony criteria. This is because when resolving a genotype, it considers whether pick a pair which contains 1111 or pick a pair which does not contains 1111, EM algorithm picks the one with 1111 and parsimony method picks the one without 1111. EM method picks the one with 1111 as 1111 has the most highest frequency(initially all haplotypes will start with the same frequency, but as iteration goes, 1111 will get the highest frequency) so that for each resolve of genotype, EM will pick the pair contains 1111. This is because the pair with 1111 will get the higher score(score:product of two haplotypes of the pair) than other pairs(in this sample, there are at most two pairs for each phenotype since there are at most 2 heterozygous sites). However, the parsimony method picks a pair where both haplotypes has some extent of frequencies, not a pair that one haplotype has high frequency and the other haplotype has small frequency. This minimizes the total number of haplotypes. For EM, even though the other pair, which does not have 1111, two haplotypes in the pair has some frequencies which is not 0(they could resolve other phenotypes), the product of those frequencies is small so that it does not overcomes the product of frequencies that 1111 and the complementary gives. So EM picks the pair which contains 1111. If the other haplotype, which makes pair with 1111, has frequency 0,than the product will be 0 so that it will be smaller than the product of frequencies of the pair which does not contains 1111. However, this is not so as the other haplotype, which makes pair with 1111, also has some frequency which is not

$$\left\{\begin{array}{l} 1111 \oplus 1111 \\ 1111 \oplus 0011 \\ 1111 \oplus 1001 \\ 1111 \oplus 1101 \\ 1111 \oplus 1010 \\ 1111 \oplus 0101 \\ 1111 \oplus 0110 \end{array}\right\} \leftarrow^{P_7} \left\{\begin{array}{l} 1111 \\ 2211 \\ 1221 \\ 1122 \\ 1212 \\ 2121 \\ 2112 \end{array}\right\} \rightarrow^{P_5} \left\{\begin{array}{l} 1111 \oplus 1111 \\ 0111 \oplus 1011 \\ 1011 \oplus 1101 \\ 1101 \oplus 1110 \\ 1011 \oplus 1110 \\ 0111 \oplus 1101 \\ 0111 \oplus 1110 \end{array}\right\}$$

Figure 4.1: Two different phasing result of given genotypes 1111, 2211, 1221, 1122, 1212, 2121 and 2112. Left hand side resolves genotypes with 7 haplotypes. Right hand side resolves genotypes with 5 haplotypes. EM algorithm results in the left hand side and parsimony algorithm results the right hand side.

0 since it resolves one genotype. And haplotypes in the other pair, which not contains 1111, has also small frequencies because at most it resolves two genotypes. Resolving one or two phenotypes does not give much difference of frequency so that the frequency of the haplotype which makes the pair with 1111 and other haplotype which makes pair with other haplotype(not 1111) has no big difference. What is worse, after products two frequencies of haplotypes that is not 1111 and could resolve a genotype, the value gets small so that it gives smaller difference score which is hard to over come the product of pair containing 1111, as 1111 has large frequency as it resolves 7 genotypes. Maybe we can solve this problem by making epsilon value very small(much smaller than 0.00001), which will take huge run time, seems not practical. The Clark algorithm, sometimes gives right hand side, sometimes left hand side, dependent on the order of resolving.

2. Second reason is that high frequency of a haplotype means the haplotype possibly could resolve many genotypes, however, this does not guarantee the

parsimony criteria. This is because, when resolving a genotype,g_i, when a haplotype, h_j,has high frequencies so that the product of frequencies for the pair that contains h_j is higher than the other pair's product, the pair with h_j is picked to resolve the genotype. However, h_j might not be picked for other genotypes, even though it could resolve those genotypes, as they have other haplotype pairs which have higher product of frequencies. As a result, when thinking of the total number of haplotypes, picking other haplotype pair, not including h_j, may minimize the number of haplotypes in total if the haplotypes in other pair is picked for other genotypes. The frequency means the possibility so that high frequency means it has high possibility to be picked for a genotype, but not real picked frequencies for genotypes.

However, the performance is still dependent on the shape of given genotypes. The EM algorithm does not give the parsimony haplotype for every case. It concerns the maximum-likelihood, in other words the most frequent haplotypes, but not directly concerns the minimum number of total haplotypes. The parsimony criteria reflects the fact that in natural populations, the number of observed distinct haplotypes is vastly smaller than the number of combinatorially possible haplotypes. However, in nature, maximum likelihood, not parsimony, might correctly infers haplotypes.

```
Given genotypes= {{1, 1, 1, 1}, {1, 1, 2, 2}, {1, 2, 1, 2}, {1, 2, 2, 1}, {2, 1, 1, 2}, {2, 1, 2, 1}, {2, 2, 1, 1}}

-------------------------------------------------------------------------------------------
Compare the performance of four algorithms
-------------------------------------------------------------------------------------------

<Haplotypes that resolved the genotypes>
ClarkHapT=          {{0, 0, 1, 1}, {0, 1, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 1}, {1, 0, 1, 0}, {1, 1, 0, 0}, {1, 1, 1, 1}}
Clark2HapT=         {{0, 0, 1, 1}, {0, 1, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 1}, {1, 0, 1, 0}, {1, 1, 0, 0}, {1, 1, 1, 1}}
ParsimonyHapT=      {{0, 1, 1, 1}, {1, 0, 1, 1}, {1, 1, 0, 1}, {1, 1, 1, 0}, {1, 1, 1, 1}}
EMHapT=             {{0, 0, 1, 1}, {0, 1, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 1}, {1, 0, 1, 0}, {1, 1, 0, 0}, {1, 1, 1, 1}}
BayesianHapT=       {{0, 0, 1, 1}, {0, 1, 0, 1}, {0, 1, 1, 0}, {1, 0, 0, 1}, {1, 0, 1, 0}, {1, 1, 0, 0}, {1, 1, 1, 1}}

<Number of haplotypes that resolved the genotypes>(Number of haplotypes/Number of resolved genotypes)
ClarkHapTNum=       7 / 7
Clark2HapTNum=      7 / 7
ParsimonyHapTNum=   5 / 7
EMHapTNum=          7 / 7
BayesianHapTNum=    7 / 7

<Runnung time>
ClarkTime=          0. Second
Clark2Time=         0. Second
ParsimonyTime=      0.0156250 Second
EMTime=             0.0312500 Second
BayesianTime=       0.0312500 Second

-------------------------------------------------------------------------------------------
Compare four PHASE methods
-------------------------------------------------------------------------------------------

<Number of haplotype that resolved genotypes>
```

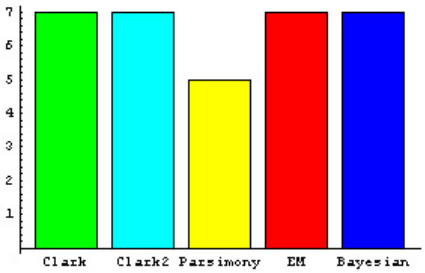


Figure 4.2: Result of methods for genotypes 1111, 2211, 1221, 1122, 1212, 2121 and 2112. The parsimony method results in the minimum number of haplotypes.

Chapter 5

Conclusion and Comparing phasing methods

Clark provided a basic algorithm for phasing genotypes and it has been as the frame work for haplotype phasing. The Clark algorithm fundamentally has three problems. As the detailed algorithm and way of implementation is flexible, we can apply and add some ideas to get an improved performance and solve some part of those problems. However, for some cases, the solutions are not impossible or impractical to applied. In addition, there are some problems, which seems unsolvable. The parsimony method solved some of those problems such as in any case, the parsimony method can get started and resolves all the genotypes. Moreover, the number of haplotypes are minimized so that it satisfies the parsimony criteria. However, for each haplotype, it needs to calculate the score which is based on the number of genotypes that could be resolved with the haplotype. Also it needs to upgrade the scores for each iteration of resolving each genotype. As a result, comparing to the Clark method, the running time gets worse and worse as the number of genotypes and the number of heterozygous loci grows. To compromise the trade off of running time and the parsimony criteria, the way of scoring each haplotypes are not perfect

so that for some genotypes, the scoring does not always guarantee the parsimony criteria. The Em Method also solved some problems of Clark Method and it possesses well-established statistical properties. However, it also has many problems such as not ensure the parsimony criteria, requires large memory space, and so on. All the methods has advantages and disadvantages, struggling with some trade off properties, and dependent on the shape of given genotypes such as recombination rate. However, all of them are worth while to be concerned and studied for the development of haplotype phasing. And their educational and practical contribution on the haplotype phasing is large enough to be praised. Also it has no doubt that those methods will be a foundation for the haplotype phasing methods in future work

Part II

PHASEM Program

Chapter 1

Datasets

For evaluation, I rely on more than 100 artificially made datasets and several publicly available datasets. For test, I provide 30 artificially made datasets named as s1.txt, s2.txt,...,s30.txt. i also provide eight publicly available datasets 140kbps.txt, 40kbps.txt, 10kbps.txt, 2kbps.txt, DTNB1CEU1.txt, DTNB1CEU2.txt, Chr9CEU.txt, and PLEM.txt

To compare the performance of methods dependent on the size of the genotypes, I got 140kbps.txt, 40kbps.txt, 10kbps.txt and 2kbps.txt from HAPMAP(www.hapmap.org), all of them are genotypes(not phased) of DTNBP1 from CEU population NM_032122 dystrobrevin binding protein 1 isoform a DTNBP1, only the difference is the size. For details of file format, see http://www.hapmap.org/genotypes/

To compare the performance for different population or chromosome, I also provide three datasets which is also from the HAPMAP(www.hapmap.org), DTNB1CEU1.txt, DTNB1CEU2.txt, Chr9CEU.txt, which genotypes are from different population or chromosome from genotypes above.

Lastly, To check the performance, I provide PLEM.txt. This is for comparing the result with PLEM program. The detailed contents of the file is as follows.

1. 140kbps.txt : SNPs genotyped in population CEU on ch6:15.63..15.77Mbp

(140.2kbp) SNPs genotyped in population CEU on chr6:15631019..15771249 rs# alleles chrom pos strand assembly# center protLSID assayLSID panelLSID QCcode NA06985 NA06991 NA06993 NA06994 NA07000 NA07019 NA07022 NA07029 NA07034 NA07048 NA07055 NA07056 NA07345 NA07348 NA07357 NA10830 NA10831 NA10835 NA10838 NA10839 NA10846 NA10847 NA10851 NA10854 NA10855 NA10856 NA10857 NA10859 NA10860 NA10861 NA10863 NA11829 NA11830 NA11831 NA11832 NA11839 NA11840 NA11881 NA11882 NA11992 NA11993 NA11994 NA11995 NA12003 NA12004 NA12005 NA12006 NA12043 NA12044 NA12056 NA12057 NA12144 NA12145 NA12146 NA12154 NA12155 NA12156 NA12234 NA12236 NA12239 NA12248 NA12249 NA12264 NA12707 NA12716 NA12717 NA12740 NA12750 NA12751 NA12752 NA12753 NA12760 NA12761 NA12762 NA12763 NA12801 NA12802 NA12812 NA12813 NA12814 NA12815 NA12864 NA12865 NA12872 NA12873 NA12874 NA12875 NA12878 NA12891 NA12892

2. 2kbps : SNPs genotyped in population CEU on chr6:15696134..15706133 rs# alleles chrom pos strand assembly# same to 1

3. 10kbps.txt : SNPs genotyped in population CEU on chr6:15700134..15702133 rs# alleles chrom pos strand assembly# same to 1

4. 40kbps.txt : SNPs genotyped in population CEU on chr6:15696134..15706133 rs# alleles chrom pos strand assembly# same to 1

5. DTNB1CEU1.txt : Genotypes(not phased) of DTNBP1 from CEU population. Size is 2kbps. NM_032122 dystrobrevin binding protein 1 isoform a DTNBP1. This data set has many ambiguous sites so that running time might be longer than other datasets.

6. DTNB1CEU2.txt : Genotypes from the same population but the size is 10kbps.

7. Chr9CEU.txt : SNPs genotyped in population CEU on Chr9:700000..719999. The size is 20kbps.

8. PLEM.txt : This genotypes are for checking the performance compared with PLEM program which is available on the web http://www.people.fas.harvard.edu/ junliu/plem/click.html. They provide the "input.txt" file and when run it we can get output file. As the input file format is different I made PLEM.txt file which contains the same genotypes so we can compare the results of programs by running the PLEM program and my program.

Chapter 2

Mathematica, Program description and Data type

2.1 Mathematica

Mathematica is one of the most popular computer programs that deals with symbolic mathematics. Far from being a form of artificial intelligence (it is not able to prove theorems), Mathematica is actually a very powerful tool in the hands of students and scholars for manipulating algebraic expressions. Being a computer language, the great advantage of Mathematica over other computer languages like Fortran, Pascal, C and so on, consists in its capability not only to enter algebraic expressions into its "mind", but also to process them as they are, that is formally, and to give the results in the form of algebraic expressions. Actually, the algebraic expressions are seen by Mathematica as objects, and one could say that Mathematica is a brilliant application of the pure computer concept of object oriented programming (http://shum.huji.ac.il/cc/mathintr.html). Also the Mathematica program is available at the site above.

2.2 Program description and How to run

The name of the program is PHASEM. The following is the contents and description of the program.

- PhasingH.nbd: This is the header file of the program
- HCreat: This file creates header file(PhasingH.nbd) of the program. You can add a new method to this program. If you want to add a new method to this program, add files for your method, and add the function name to the HCreat file and run it. It will create new PasingH.nbd file for the program.
- PHASEM.nb: This file contains the main function of the program. With only two files PhasingH.nbd and PHASEM.nb, you can run the program. When you run PHASEM.nb, a window pops up which ask for an input file. Figure 2.1 shows this window. When user types an input file name on the window

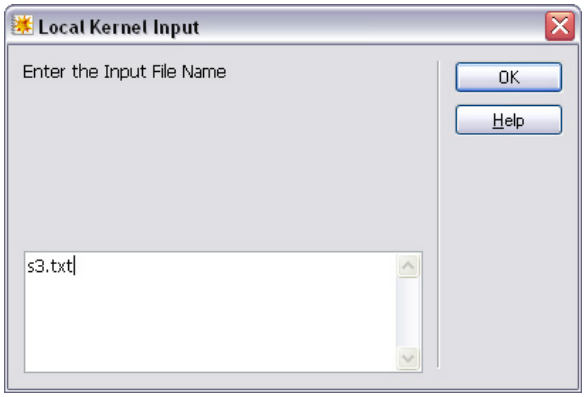


Figure 2.1: The first window when run the PHASEM program. It asks for the input file

and clicks the OK button, another window pops up. Figure 2.2 shows the second window.

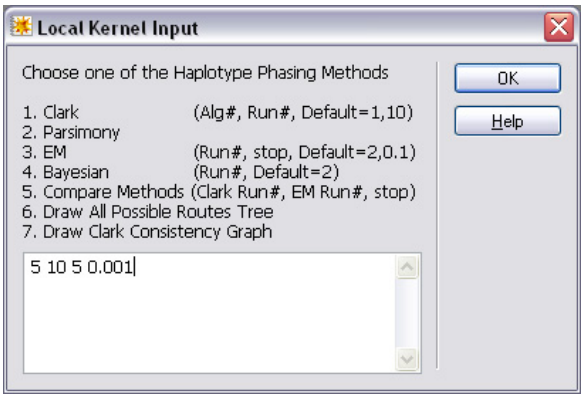


Figure 2.2: The second window when run the PHASEM program. It asks for the method to run and option for the method

- Clark.nb: This file contains the Clark Method function. Clark Method function implements Clark Method which is described in Chapter 2 of SectionI and Chapter 3 of SectionII.
- Clark2.nb: This file contains the Clark2 Method function. Clark2 Method implements the second rule of Clark Algorithm
- Parsimony.nb: This file contains the Parsimony Method
- EM.nb: This file contains the EM Method
- Functions.nb: This file contains functions for Clark, Clark2, Parsimony and EM Method
- ClarkFindAllRouts.nb: This file contains the function ClarkFindAllRouts function which implements the tree of all possible Routs or Clark Algorithm
- ClarkConsistencyGraph.nb: This file contains Clark Consistency Graph function which draws Clark Consistency Graph
- ShowGraphic.nb: This file contains ShowGraphic function which shows graphical view such as barchart and piechart.

- ComparePhasing.nb: This file contains ComparePhasing function that compares the result of Clark, Parsimony, EM method by running those methods and runs showgraphic function which show graphical view of comparison. This function also runs ClarkFindAllRouts and ClarkConsistencyGraph to show the Clark route tree and Clark Consistency Graph.

- ChooseMethod.nb: This file contains the function ChooseMethod which is called by PHASEM(main function). This function helps user to choose a method to run and choose options for the method.

2.3 Data type

- Data format is Presented with 0/1/2/3, Where 0/1 indicates homozygous site, 2 indicates a heterozygous site and 3 indicates NN site(missing data site).

- GenoT: Genotype set

- Datatype is double array. Each element indicates a SNP site, the inner array indicates a genotype, and the outer array indicates a Genotype set
  Each element has value $\in \{0, 1, 2\}$.
  For example,
  GenoT = {{0, 0, 2, 0, 0, 2}, {2, 2, 1, 1, 1, 1}, {2, 1, 2, 0, 0, 0}} Genotype set GenoT has three genotypes, 002002, 221111, and 212000.

- Input file format: Each column indicates a genotype and each row indicates a SNP site.
  For example, Figure 2.3 shows the format of s1.txt

- Output file format: There are four kinds of output data, ClarkOutput.txt, ClarkOutput2.txt, ParsimonyOutput.txt and EMOutput.txt

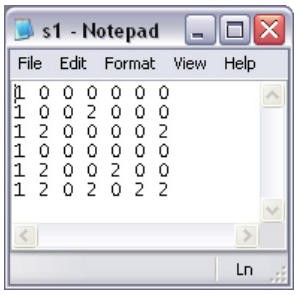


Figure 2.3: Input file format of s1.txt. Each column indicates a genotype. s1.txt has 7 genotypes 111111, 002022, 000000, 020002, 000020, 000002, and 002002

For each Output file genotypes and haplotypes has the same format as input file's genotype.

For example, Figure 2.4 shows the format of ParsimonyOutput.txt

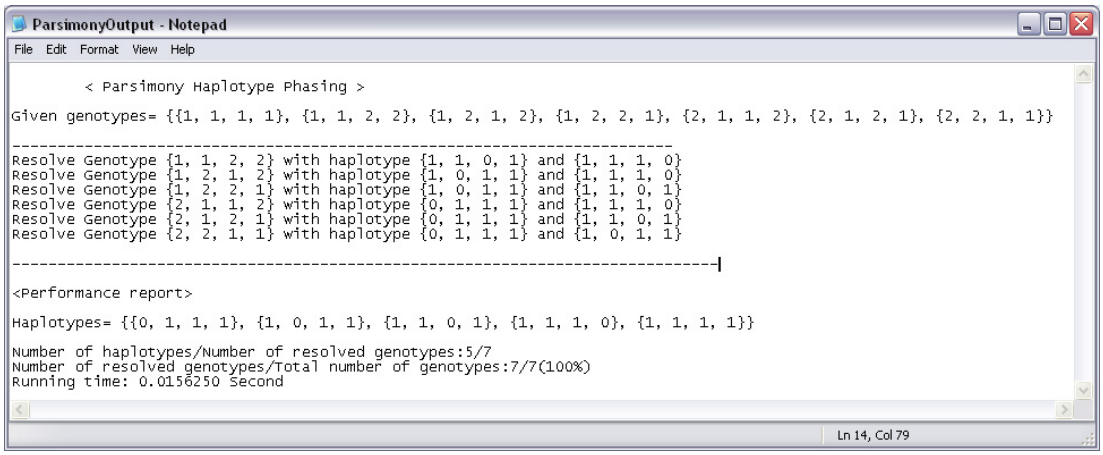


Figure 2.4: Output file format of ParsimonyOutput.txt. Given genotypes are 1111, 1122, 1212, 1221, 2112, 2121, and 2211. Genotype 1122 is resolved with haplotype pair 1101 and 1110. All of the 7 given genotypes are resolved with 5 number of haplotypes 0111, 1011, 1101, 1110, and 1111

Chapter 3

Clark method

I implemented Clark algorithm which is described in Chapter 2 of SectionI. As there are two Methods, Clark1 and Clark 2, I implemented both as Clark and Clark2. And user has option of the number of runs that each run with different order of genotypes and haplotypes. this option is given as the result might different with the order of pairing genotypes and haplotypes and Clark suggest to run several times to get a better result. To execute the program run PHASEM.nb file and when the second window pops up, choose #1 with two options, one of which is Clark algorithm number(1 or 2) and the other is the number of runs. Figure 3.1 and Figure 3.2 shows the execution. By default, if a user choose #1 and does not select options, the program runs Clark Method 1 and run 10 number of times.

3.1 Algorithm

1. Identify all homozygotes and single-site heterozygoses and consider their haplotypes as "resolved.". If a homozygote is found, we have unambiguously identified a haplotype. If a single-site heterozygote is found, we have unambiguously identified two haplotypes. For example, if we found 0120, we

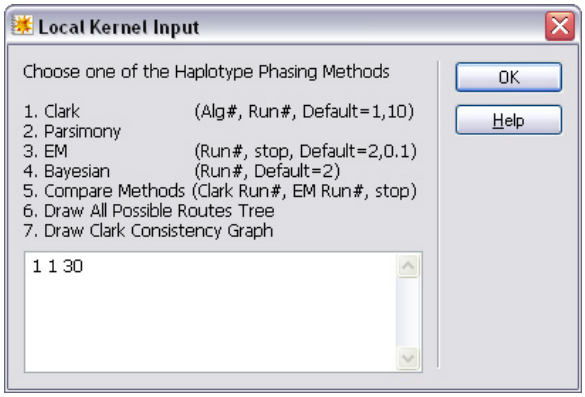


Figure 3.1: Run Clark1 Method with 30 number of runs

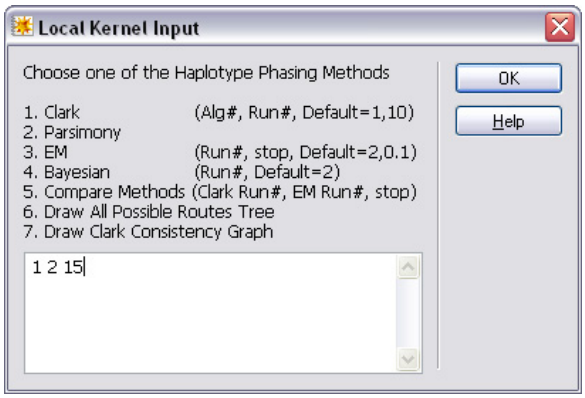


Figure 3.2: Run Clark2 Method with 15 number of runs

identifies two haplotypes 0100 and 0110 as 0120=0100⊕0110.

2. For each known haplotype, we then look at all the remaining unresolved sequences and ask whether the known haplotype can be made from some combination of the ambiguous sites. Each time such a haplotype is found, recover the complement of the haplotype as another potential haplotype.

3. Continue step2 until all haplotypes have been recovered, or until no more new haplotypes can be found.

There could be several ways of pairing haplotypes and genotypes. Paring which haplotypes to which genotypes may affect the result, in other words, problem2 or problem3 could happen dependent on the pairing. To solve this problem, as I mentioned at the PartI that Clark suggested, with different order of genotypes or haplotypes, run several times to get a better performance(resolve more genotypes). There could be many ways of ordering genotypes or haplotypes but change the order of genotypes in the genotype set or haplotypes in the haplotype set is the simplest way with small running time at the same randomizing performance(not described here). In addition, we can change the rule of pairing. When comparing haplotypes with genotypes to check whether a haplotype can infer a genotype, I tried two different rules of pairing genotypes and haplotypes.

3.1.1 Clark Algorithm 1

We are given a genotype set GenoT and haplotype set HapT. We remove homozygous genotypes from GenoT and put them into HapT. Also we remove single-site heterozygous genotypes from GenoT and make complementary and put then into HapT. As a result, suppose we get GenoT=$\{g_1, g_2, ..., g_{gnum}\}$ and HapT=$\{h_1, h_2, ..., h_{hnum}\}$.

1. Pick the first haplotype h_1 and find a genotype, suppose it is g_2, which could be inferred from h_1. Remove g_2 from GenoT and make complementary of g_2,

which is h'_1, put it into HapT.

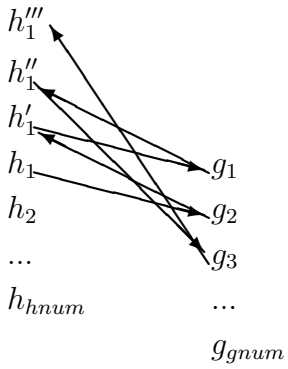
→Find a genotype, suppose it is g_1, which could be inferred from h'_1 . Remove g_1 from GenoT and make complementary of g_1, which is h''_1, put it into HapT.

→Find a genotype, suppose it is g_3, which could be inferred from h''_1 . Remove g_3 from GenoT and make complementary of g_3, which is h'''_1, put it into HapT.

2. Iterate #1 until $h_1^{',...'}$ could not find a genotype to infer.

3. Pick next element of HapT, and do the same process as #1 and #2.

4. Repeat #3, until the last element of HapT, h_{hnum}, is picked and compared with genotypes.



3.1.2 Clark Algorithm 2

I will call the second Clark method ans Clark2 method. The other rule of pairing is as follows.

1. Pick the first haplotype h_1 and find all the genotypes, suppose they are g_2 and g_{num}, from GenoT which could be inferred from h_1.

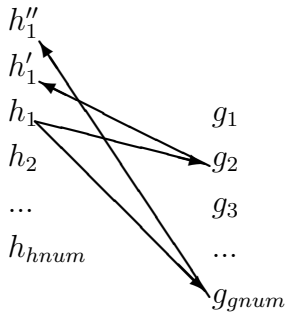
2. Make corresponding complementary haplotypes, h'_1 and h''_1, for each of the found genotypes.

3. Remove resolved genotype from GenoT and insert new haplotypes to HapT

4. Pick the next haplotype from HapT, h_2, and do the same process as #1, #2 and #3.

5. Repeat #4 until the last haplotype, h_{hnum} is picked and compared with genotypes.

This rule works since true haplotype and observed genotypes are not one-to-one. In other words, one haplotype may infer many genotypes.



3.2 How to run and the Result

When we run s3.txt with Clark2 Method, as given genotypes are 0000,1000,1122 and 2200, The algorithm solves the genotypes with 0000, 0100, 1000 but genotype 1122 is left unsolved to report Problem2. Figure 3.3 shows loading the input file s3.txt, Figure 3.4 shows how to run Clark2 Method with option(run 3 number of times with different order, if this option is not given, the program runs 10 number of times as the default) and Figure 3.5 shows the result of the program. After the execution, a file named Clark2Output.txt is created(if we execute Clark Algorithm 1 Method, it creates clarkOutput.txt), which has more detailed information of the execution and the result, such as at what order number(as we give 3 as the option of run number, there are Order1, Order2 and Order2), how many genotypes are resolved with what number of haplotypes and so on. Figure 3.6 shows the ClarkOutput2.txt.

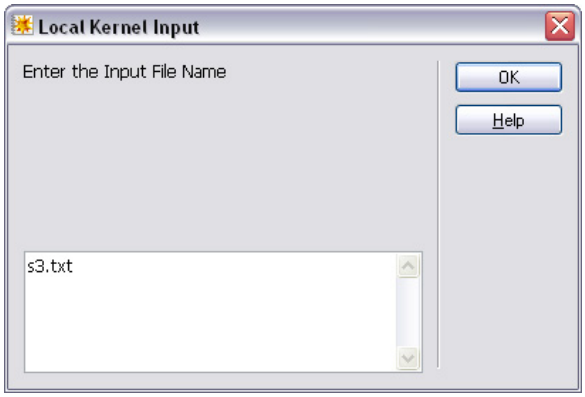


Figure 3.3: Load s3.txt file

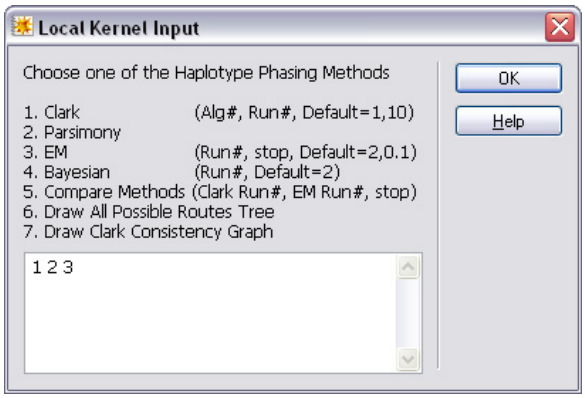


Figure 3.4: Execute the loaded file with Clark2 Method. Run 3 number of times

```
InputFile: s3.txt

--------------------------------------------------------------------------------
Clark haplotype phasing
--------------------------------------------------------------------------------
Given genotypes= {{0, 0, 0, 0}, {1, 0, 0, 0}, {1, 1, 2, 2}, {2, 2, 0, 0}}

Clark Algorithm 2

Problem2. There are unresolved genotypes left

<Haplotypes that resolved the genotypes>
Clark2HapT=         {{0, 0, 0, 0}, {0, 1, 0, 0}, {1, 0, 0, 0}}


<Number of haplotypes that resolved the genotypes>(Number of haplotypes/Number of resolved genotypes)
Clark2HapTNum=          3 / 3


<Runnung time>
Clark2Time=         0. Second
```

Figure 3.5: Result of the program when run s3.txt with Clark2 Method. Number of run is given 3. In this case, within 3 number of runs, there is no case that 0000 resolves 2200 to produce 1100 so that 1122 is not resolved. As there is unresolved genotype 1122 left, it reports Problem2.

3.3 Discussion

Clark algorithm provides the first idea of haplotype phasing and it is worth while understanding and implementing the Clark Method. Clark Method's running time is fast, needs not much memory space and not hard to implement as the algorithm is simple compare to other programs. Even it is easier for implement with Mathematica since there are many arithmetic functions that Mathematica packages such as DiscreteMath‘Combinatorica‘ provide. Also Mathematica is able to manipulate very large integer numbers which helps to implement the program. However, the correctness is not as good as other methods. We can improve the correctness by increase the number of runs, each with different runs, so that if we run all possible orders, we can get the minimum number of haplotypes but which is impractical. However, one good thing about Clark method is that as we increase the number of runs, even the running time increases, the memory space we need to execute does

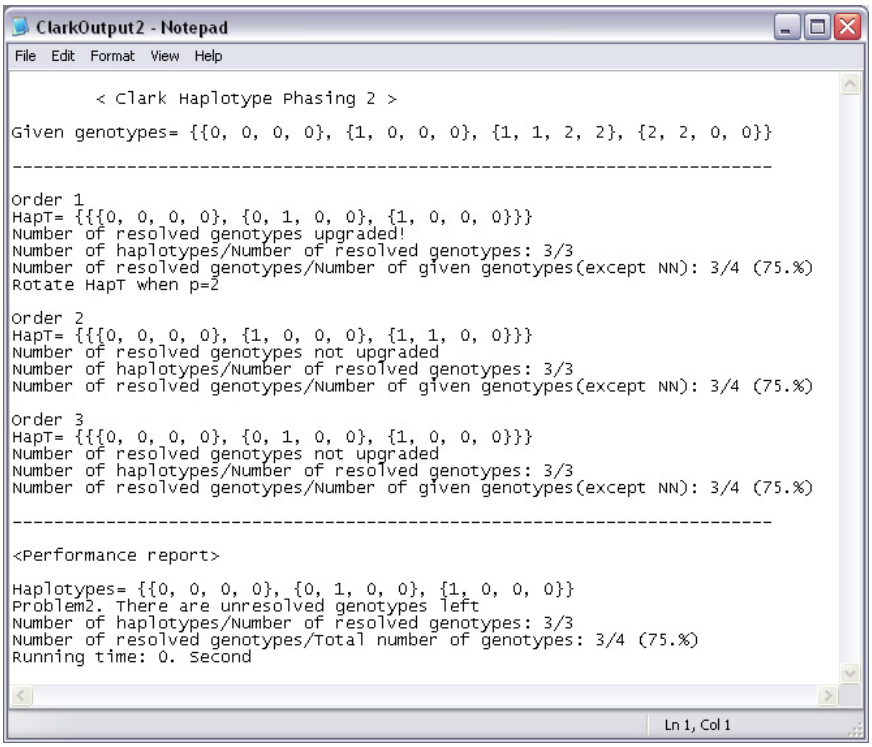


Figure 3.6: Clark2Output.txt after run s3.txt with Clark2 Method

not run as other methods. However, the running time could be huge as the number of genotypes and the number of ambiguous sites are large we need better algorithm than Clark.

Chapter 4

Parsimony method

In this chapter, I implemented a Parsimony method which I made applying Clark Consistency Graph.

4.1 Algorithm

1. Initialize haplotype set: Find homozygous genotypes and single-site heterozygous genotypes. For each single-site heterozygous genotype, generate a pair of haplotypes by replacing the single heterozygous site to 0 and 1. Put those sequences to haplotype set and remove from genotype set. Count the number of each kind of sequences which will be used in #2. Clark Consistency Graph set.

2. Generate Clark Consistency Graph set: This set will contain the shared haplotypes and haplotypes that is already picked. Also this set has count for each element that indicates the number of genotypes shares each haplotype(element). Initialize with haplotype set which is made at #1. Then compare each pair of genotypes, if there is an edge, put the edge to the Clark Consistency Graph set. - For a pair of genotypes, there is an edge if two genotypes has same values

of 0 or 1, for sites which both are not 2 . Edge is generated as follows. If both site is 2 put 2, else if one site is 2 and the other is 0(1), put 0(1), else if both sites is 0(1), put 0(1) -When put each edges, if there are sites with 2, create all possible combinations by replacing site 2 to 0 and 1. One more thing to do, when putting edges(combinations) to the set is, if there are same sequence of edges(combinations), count the number of each kinds of edges(combinations). This is important since the count indicates the number of genotypes which shares the same kinds of edge(combination)

3. Generate a pair of haplotypes for each genotype: There are $2^(k-1)$ possible pairs of haplotypes for each genotype when the number of site with value 2 is k. Among them, we should pick a pair of haplotypes to resolve each genotype which satisfies the parsimony haplotyping(minimize the total number of haplotypes). To do this, for each genotype, generate all possible $2^(k-1)$ pairs of haplotypes and compare the counts from Clark Consistency Graph and pick a pair of haplotypes which has the highest score(count). The picked haplotypes will be added to the haplotype set. The way of scoring is,

   (a) If both haplotypes of a pair occurs in the Clark Consistency Graph set(includes edges of Clark Consistency graph and already picked haplotypes), pick the pair. This means both haplotypes are shared with other genotypes so that if other genotypes which share the haplotypes, pick these haplotypes later, it will not increase the number of haplotypes. This is parsimony haplotyping. But what if there is many pair of haplotypes that both haplotypes are occurred in Clark Consistency Graph set? The answer is, pick the one that has the highest sum of counts of two haplotypes. But what of there are multiple pairs of haplotypes with same highest sum of scores? Put off this genotype so that after resolving other genotypes, do this resolving process again. This works since,

the element of Clark Consistency Graph set and the count of each element will be changes since the each haplotype's count decreases if they are not picked. What if after resolving all other genotypes it still has the same problem? Randomly pick one pair which has the highest score. This is also parsimony haplotyping since it increases the same number of haplotypes considering total number of haplotypes. For the rest of the haplotypes which occurred in the Clark Consistency Graph but not picked, decrease the count in the Clark Consistency Graph set. This is because the genotype did not pick the haplotype which means one genotype no longer shares the haplotype but uses other haplotype to resolve itself. So if other genotype use this haplotype later, it may increase the number of haplotypes unless other third genotype shares the haplotype.

(b) If there is no pair which both haplotypes are occurred in the Clark Consistency Graph set, check whether there is pairs that one of the haplotype of the pair is occurred in the Clark Consistency Graph set. If there is one, pick the pair. This means one haplotype is shared with other genotypes so that if other genotypes which share the haplotype, pick this haplotypes later, it will not increase the number of haplotypes. Here, put the haplotype which is not occurred in the Clark Haplotype Graph Set to the Clark Haplotype Graph Set with count 1. This allows other genotypes to use this haplotype considering it is shared with a genotype which is resolved, and this will not increase the number of haplotypes. But what if there are many such pairs? Pick the one which has the highest score(count). What if there are multiple pairs of haplotypes which have the same highest score? Put off this genotype so that after resolving other genotypes, do this process again. What if after resolving all other genotypes it still has the same problem? Randomly pick one pair which

has the highest score. For rest of the haplotypes which occurred in the Clark Consistency Graph but not picked, decrease the count in the Clark Consistency Graph set.

(c) If all the pairs are not occurred in the Clark Consistency Graph set, Put off this genotype so that after resolving other genotypes, do this process again. What if after resolving all other genotypes it still has the same problem? Randomly pick one pair which has the highest score. Here, put both haplotypes to the Clark Haplotype Graph Set with count 1. This allows other genotypes to use this haplotype considering it is shared with a genotype which is resolved, and this will not increase the number of haplotypes. For rest of the haplotypes which occurred in the Clark Consistency Graph but not picked, decrease the count in the Clark Consistency Graph set.

4.2 Description with examples

Example1

Given genotypes are $00020010220, 00002201202, and 00000222100$

Algorithm: Given genotype set.

$g1 = 00020010220$, $g2 = 00002201202$, $g3 = 00000222100$ (input file s13.txt contains these genotypes)

1. Initialize haplotype set: There is no homozygous genotypes or single-site heterozygous genotypes so that haplotype set is an empty set at the beginning.

2. Generate Clark Consistency Graph set: Initial Clark Consistency Graph set is same to haplotype set which is an empty set.
   Compare each genotypes to create Clark Consistency graph's edges
   $g1(00020010220)$ with $g2(00002201202) \rightarrow$ no edge

$g1(00020010220)$ with $g3(00000222100) \rightarrow 00000010100$

$g2(00002201202)$ with $g3(00000222100) \rightarrow 00000201100$

Make combinations of edges if there are sites with value 2

As a result, we get $00000010100, 00000001100, 00000101100$ with count 1,1,1

3. Generate a pair of haplotypes for each genotype: For each genotype generate possible pairs and compare scores

   Remember we got Clark Consistency Graph edges set and their count from #1 as $00000010100, 00000001100, 00000101100$ with count 1,1,1. The Clark Consistency Graph for $g1$, $g2$, and $g3$ is shown in Figure 4.1.

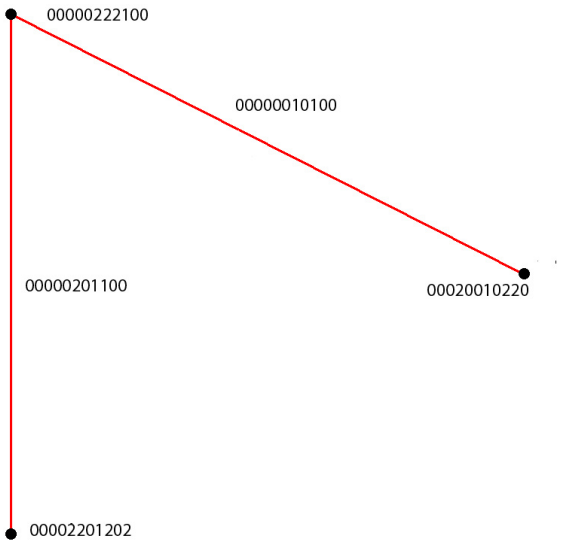


Figure 4.1: Clark Consistency Graph for Example1. Given genotypes are 00020010220, 00002201202, and 00000222100.

$g1(00020010220)$ could be resolved with $00000010100 \oplus 00010010010$ each has count 1,0 which sum up to 1

or $g1$ could be resolved with other haplotypes which all of them has count 0,0 which sum up to 0

Pick 00000010100 and 00010010010 as their counts sum up to the highest score

No other sequences are occurred in the Clark Consistency Graph set so that we don't have to do the decreasing process.

We need increasing process as a new haplotype 00010010010 is introduced. As a result we get $00000010100, 00000001100, 00000101100, 00010010010$ with count 1,1,1,1

Now let's resolve $g2$

$g2$(00002201202) could be resolved using$00000001100 \oplus 00001101001$ each has count 1, 0 which sum up to 1

or $g2$ could be resolved with $00000101100 \oplus 00001001001$ each has count 1, 0 which also sum up to 1

or $g2$ could be resolve with other haplotypes which all other pairs has count 0,0 which sum up to 0

We cannot pick a since two pairs has same highest score(sum of counts) 1 so that we put off this genotype and resolve this after resolving other genotypes.

Now let's resolve $g3$

$g3$(00000222100) could be resolved with $00000010100 \oplus 00000101100$ each has count 1,1

or $g3$ could be resolved with $00000001100 \oplus 00000110100$ each has count 1, 0 which sum up to 1

all other pairs has count 0,0 which sum up to 0

Pick 00000010100 and 00000101100 since both pair has occurred in the Clark Consistency Graph set and the count has sum up to the highest score 2. 00000001100 is occurred in the Clark Consistency Graph set but not picked so decrease 00000001100's count from 1 to 0

Now return to g2, which is not resolved and put off since two pairs had same highest score.

$g2$(00002201202) could be resolve with $00000001100 \oplus 00001101001$ each has

count 0, 0

or 00000101100 $\oplus$ 00001001001 each has count 1, 0

all other pairs has count 0,0 Since 00000001100's count is decreased to 0 in the process of resolving $g3$, for 00000001100 and 00001101001 pair, we get 0,0 which sum up to 0 in this time. As a result we pick 00000101100 and 00001001001 pair.

4. As a result, 4 haplotype resolve all genotypes.

   We got haplotype set 00000010100, 00010010010, 00000101100, 00001001001

$$\left\{\begin{array}{c} 00010010010\oplus \\ 00000010100, \\ 00000101100\oplus \\ 00001001001, \\ 00000101100\oplus \\ 00000010100 \end{array}\right\} \leftarrow^{P_4} \left\{\begin{array}{c} 00020010220 \\ 00002201202 \\ 00000222100 \end{array}\right\} \rightarrow^{P_6} \left\{\begin{array}{c} 00010010000\oplus \\ 00000010110, \\ 00001001000\oplus \\ 00000101101, \\ 00000100100\oplus \\ 00000011100 \end{array}\right\}$$

Left side is Parsimony which resolves with 4 number of haplotypes, whereas right side resolves with 5 number of haplotypes. The Parsimony algorithm introduced here gives the left side parsimony result.

Here is another example, Example2. Given genotypes are 1111, 2211, 1221, 1122, 1212, 2121, 2112 (input file s7.txt contains these genotypes)

$$\left\{\begin{array}{c} 1111 \oplus 1111 \\ 1111 \oplus 0011 \\ 1111 \oplus 1001 \\ 1111 \oplus 1101 \\ 1111 \oplus 1010 \\ 1111 \oplus 0101 \\ 1111 \oplus 0110 \end{array}\right\} \leftarrow^{P_7} \left\{\begin{array}{c} 1111 \\ 2211 \\ 1221 \\ 1122 \\ 1212 \\ 2121 \\ 2112 \end{array}\right\} \rightarrow^{P_5} \left\{\begin{array}{c} 1111 \oplus 1111 \\ 0111 \oplus 1011 \\ 1011 \oplus 1101 \\ 1101 \oplus 1110 \\ 1011 \oplus 1110 \\ 0111 \oplus 1101 \\ 0111 \oplus 1110 \end{array}\right\}$$

Right side is Parsimony which resolves with 5 number of haplotype whereas left side resolves with 7 number of haplotypes. The Parsimony algorithm introduced here gives parsimony result.

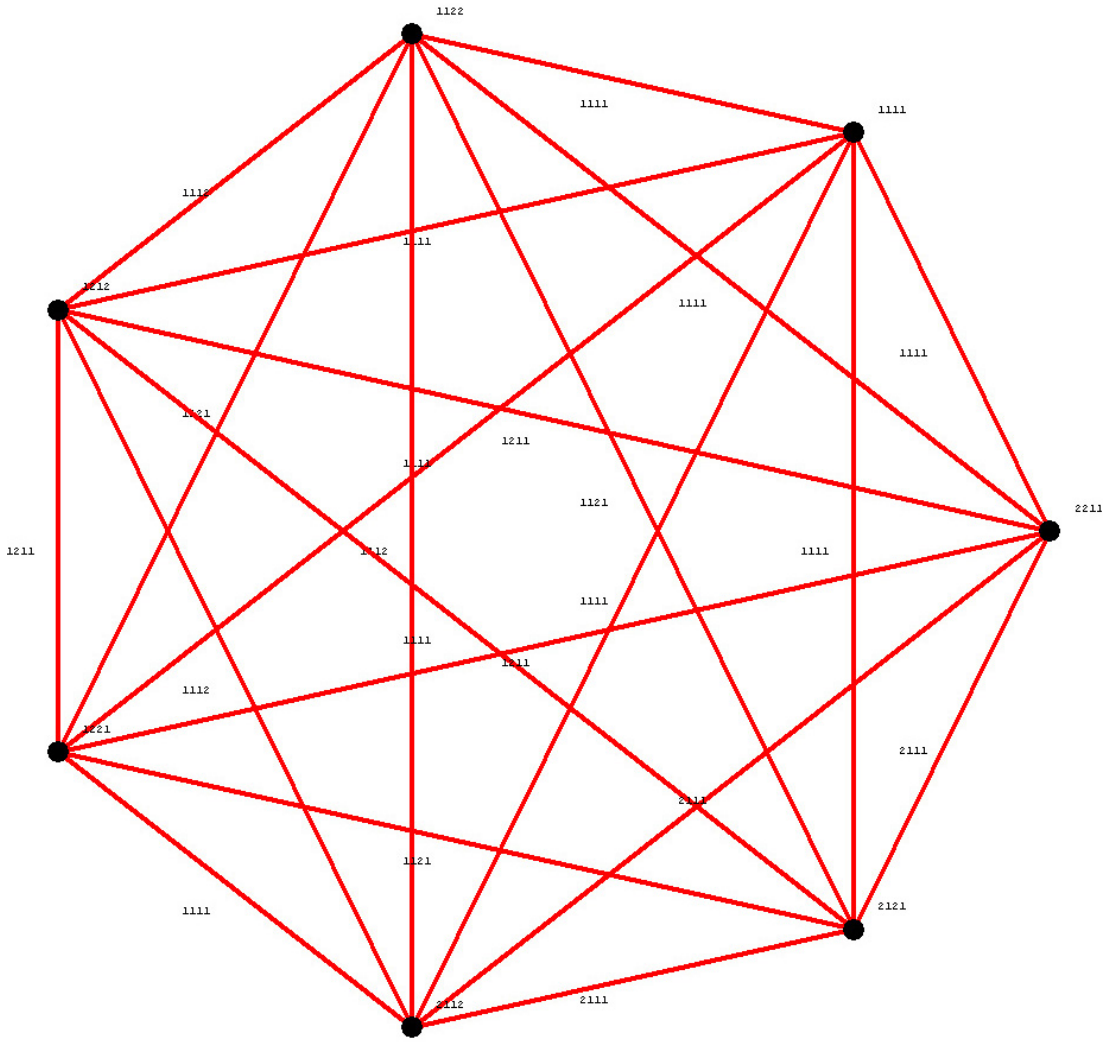


Figure 4.2: Clark Consistency Graph for Example2. Given genotypes are 1111, 2211, 1221, 1122, 1212, 2121, and 2112.

4.3 How to run and the Result

When we run s13.txt with Parsimony Method, it shows the result of example 1 of previous section. Figure 4.4 and Figure 4.5 shows how to run the program ,and Figure 4.6 shows the result of the program. After the execution, a file named ParismonyOutput.txt is created(if already exists, it overwrites the file), which has more detailed information of the execution and the result such as which genotype is resolved with which pair of haplotypes and so on. Figure 4.7 shows the Parsimony-Output.txt.

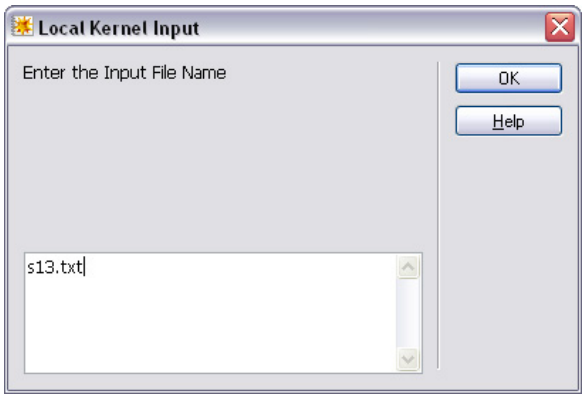


Figure 4.3: Load s13.txt file

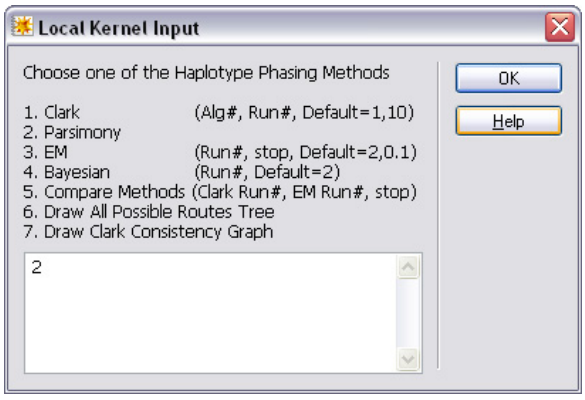


Figure 4.4: Execute the loaded file with Parsimony Method

```
InputFile: s13.txt

-------------------------------------------------------------------------------------
Parsimony haplotype phasing
-------------------------------------------------------------------------------------
Given genotypes= {{0, 0, 0, 0, 0, 2, 2, 2, 1, 0, 0}, {0, 0, 0, 0, 2, 2, 0, 1, 2, 0, 2}, {0, 0, 0, 2, 0, 0, 1, 0, 2, 2, 0}}

<Haplotypes that resolved the genotypes>
ParsimonyHapT=          {{0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0},
  {0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0}, {0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1}, {0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0}}


<Number of haplotypes that resolved the genotypes>(Number of haplotypes/Number of resolved genotypes)
ParsimonyHapTNum=          4 / 3


<Runnung time>
ParsimonyTime=          0. Second
```

Figure 4.5: Result of the program after run s13.txt with Parsimony Method

ParsimonyOutput - Notepad

File Edit Format View Help

```
        < Parsimony Haplotype Phasing >

Given genotypes= {{0, 0, 0, 0, 0, 2, 2, 2, 1, 0, 0}, {0, 0, 0, 0, 2, 2, 0, 1, 2, 0, 2},
                  {0, 0, 0, 2, 0, 0, 1, 0, 2, 2, 0}}

-------------------------------------------------------------------------------
Resolve Genotype {0, 0, 0, 0, 0, 2, 2, 2, 1, 0, 0}
        with haplotype {0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0} and {0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0}
Resolve Genotype {0, 0, 0, 0, 2, 2, 0, 1, 2, 0, 2}
        with haplotype {0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0} and {0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1}
Resolve Genotype {0, 0, 0, 2, 0, 0, 1, 0, 2, 2, 0}
        with haplotype {0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0} and {0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0}

-------------------------------------------------------------------------------

<Performance report>

Haplotypes= {{0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0},
             {0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1}, {0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0}}

Number of haplotypes/Number of resolved genotypes:4/3
Number of resolved genotypes/Total number of genotypes:3/3(100%)
Running time: 0. Second
```

Ln 24, Col 24

Figure 4.6: ParsimonyOutput.txt after run s13.txt with Parsimony Method

4.4 Discussion

As the parsimony method calculates the Clark Consistency Graph, maintains and upgrades the table(table contains the number of genotypes that possibly could be resolved for each haplotype which is gained from contains the Clark Consistency graph) and compares the scores for all the possible haplotypes, it takes more time and space than Clark Method. Running time and space get much larger as the number of ambiguous sites and the number of genotypes grows, however, it is acceptable for reasonable size of input. In addition, even it take more time and space, the number of haplotypes it phases satisfies the parsimony criteria so that the performance is much better than the Clark in sense of correctness. There is tradeoff between the running time, space, complexity of implementation and correctness. However, if the running time(space, complexity) is reasonable, correct phasing is more desirable. In the sense, even thought Parsimony method need more time and space, and more complex to implement than Clark method, it is more preferable as it gives more faithful answer for the phasing. As it is dependent on the input genotypes and what user wants, I made PHASEM program possible to choose the method.

Chapter 5

EM method

In this chapter, I will apply EM algorithm to haplotype phasing problem.

5.1 Notation and Equation

To explain EM algorithm, we need to change the definition of genotype and we also need a new notation phenotype which is a particular combination of two multilocus haplotypes. For example, for a genotype 0112201200, a combination of two multilocus haplotypes (0110001000 $\oplus$ 0111101100)is a phenotype, (0110101000 $\oplus$ 0111001100) is another phenotype and so on.

- haplotype: A homozygous sequence which is not observed
- genotype: A particular combination of two multilocus haplotypes which composes a phenotype
- phenotype: An observed multilocus genotype whose haplotype phase is unknown a priori
- n: The number of phenotypes
- m: The number of different types of phenotype

- n_j: The number of phenotypes per each different type of phenotype $j = 1, 2, ..., m$

$$n = \sum_{j=1}^{m} n_j \tag{5.1}$$

- h: The number of different types of haplotype

- h_k: Indicator of a specific haplotype k

  h_l: Indicator of a specific haplotype l

- s_j: The number of heterozygous loci of jth phenotype. $j = 1, 2, ..., m$

- c_j: The number of genotypes leading to $j$the phenotype. $j = 1, 2, ..., m$

$$c_j = \begin{cases} 2^{s_j - 1}, & \text{if } s_j > 0, \\ 1, & \text{if } s_j = 0 \end{cases} \tag{5.2}$$

- $p_t^{(g)}$: The probability of tth haplotype at time g . $t = 1, 2, ..., h$. As the probabilities are sum up to 1

$$p_1 + p_2 + ... + p_h = 1 \tag{5.3}$$

- $\widetilde{P}_{ji}(h_k h_l)^{(g)}$: The probability of ith genotype of jth phenotype, composed haplotype k and l, at time g

$$\widetilde{P}_{ji}(h_k h_l)^{(g)} = \begin{cases} (p_k^{(g)})^2, & \text{if } k = l, \\ 2p_k p_l, & \text{if } k \neq l \end{cases} \tag{5.4}$$

- $P_j^{(g)}$: The probability of jth phenotype at time g

$$P_j^{(g)} = \sum_{i=1}^{c_j} \widetilde{P}_{ji}(h_k h_l)^{(g)} \tag{5.5}$$

- $P_{ji}(h_k h_l)^{(g)}$: The normalized probability of ith genotype of jth phenotype, composed of haplotype k and l, at time g

$$P_{ji}(h_k h_l)^{(g)} = \frac{n_j}{n} \frac{\widetilde{P}_{ji}(h_k h_l)^{(g)}}{P_j^{(g)}} \tag{5.6}$$

- δ_{jit}: The number of times that haplotype t is present in ith genotype of phenotype j

$$p_t^{(g+1)} = \frac{1}{2} \sum_{j=1}^{m} \sum_{i=1}^{c_j} \delta_{jit} P_{ji}(h_k h_l)^{(g)} \tag{5.7}$$

- ε: Indicator of stop condition.

- f: The time when the iteration stops

5.2 Algorithm

1. Initial conditions

   To avoid local maxima, we better perform the algorithm for a set of widely different initial values.

   There are several ways to construct initial conditions.

   - Make all haplotypes are equally likely

$$p_t^{(0)} = \frac{1}{h}, t = 1, 2, ..., h \tag{5.8}$$

   - Make all possible genotypes for each phenotype are equally likely

$$P_{ji}(h_k h_l)^{(0)} = \frac{1}{m} \frac{1}{c_j}, j = 1, 2, ..., m, i = 1, 2, ..., c_j \tag{5.9}$$

   - Randomly choose haplotype frequencies that satisfies

$$\sum_{t=1}^{h} p_t^{(0)} = 1 \tag{5.10}$$

2. Expectation step(E-step) Using equation(5.4), for each phenotype and their genotypes, calculate the probability of ith genotype of jth phenotype.

$$\widetilde{P}_{ji}(h_k h_l)^{(g)} = \begin{cases} (p_k^{(g)})^2, & \text{if } k = l, \\ 2p_k p_l, & \text{if } k \neq l \end{cases}$$

3. Maximization step(M-step)

   (a) Using equation (5.5), calculate the probability of jth phenotype at time g

   $$P_j^{(g)} = \sum_{i=1}^{c_j} \widetilde{P}_{ji}(h_k h_l)^{(g)}$$

   (b) For each phenotype and its genotypes, calculate normalized probability of ith genotype of jth phenotype.

   $$P_{ji}(h_k h_l)^{(g)} = \frac{n_j}{n} \frac{\widetilde{P}_{ji}(h_k h_l)^{(g)}}{P_j^{(g)}} \tag{5.11}$$

   As they are normalize, they sum up to 1

   $$\sum_{j=1}^{m} \sum_{i=1}^{c_j} P_{ji}(h_k h_l)^{(g)} = \sum_{j=1}^{m} \sum_{i=1}^{c_j} \frac{n_j}{n} \frac{\widetilde{P}_{ji}(h_k h_l)^{(g)}}{P_j^{(g)}} = 1 \tag{5.12}$$

   (c) For each haplotype, calculate the probability using equation(5.7).

   $$p_t^{(g+1)} = \frac{1}{2} \sum_{j=1}^{m} \sum_{i=1}^{c_j} \delta_{jit} P_{ji}(h_k h_l)^{(g)}$$

   By equation(5.3), the probabilities of all haplotypes sum up to 1

   $$\sum_{t=1}^{h} p_t^{(g+1)} = 1$$

4. Stop condition

   If the sum of the absolute value of differences of haplotype frequencies between

consecutive runs, are less than ε, in other words converges small enough to the given threshhold, stop, otherwise go to ♯2

$$\begin{cases} \text{stop}, & \text{if } \sum_{t=1}^{h} |p_t^{(g)} - p_t^{(g+1)}| < \varepsilon \\ \text{go to } \#2, & \text{otherwise} \end{cases} \tag{5.13}$$

5. When updated frequencies satisfies stop condition(this time is f), we get the frequencies of every type of haplotypes, which are
   $p_t^f, t = 1, 2, ..., h$

6. For each phenotype, pick a genotype(a pair of haplotype)that has the maximum value of product of haplotype frequencies (haplotype that composes the genotype), from all possible genotypes that leads to the phenotype.

5.3 Description with examples

Input : Given phenotypes= 0220, 0000, 0000, 0000, 0000
Given stop condition $\varepsilon = 10^{-6}$

Output :Haplotype frequencies which satisfies the stop condition

- Given 5 number of phenotypes(0220, 0000, 0000, 0000, 0000)
  $n = 5$

- There are 2 different types of phenotype(0220, 0000)
  Let's consider 0220 as the 1st phenotype,0000 as the 2nd phenotype
  $m = 2$
  $n_1 = 1, n_2 = 4$
  By equation(1.), $n = \sum_{j=1}^{m} n_j; 5 = \sum_{j=1}^{2} n_j = n_1 + n_2 = 1 + 4 = 5$

- There are four possible types of haplotype that resolves the two phenotypes

$$0220 \begin{cases} (0000 \oplus 0110) \\ (0100 \oplus 0010) \end{cases}, 0000(0000 \oplus 0000)$$

Let's consider 0000 as h_1, 0110 as h_2, 0100 as h_3, and 0010 as h_4

$h = 4$

$s_1 = 2, s_2 = 0$

$c_1 = 2, c_2 = 1$

——————— 1st iteration ———————

1. Initial conditions

   - Make all haplotypes are equally likely

     $p_t^{(0)} = \frac{1}{h} = \frac{1}{4}, t = 1, 2, 3, 4$

     $p_1^{(0)} = p_2^{(0)} = p_3^{(0)} = p_4^{(0)} = \frac{1}{4}$

     $p_1^{(0)} + p_2^{(0)} + p_3^{(0)} + p_4^{(0)} = 1$

   - Make all possible genotypes for each phenotype are equally likely

     $P_{ji}(h_k h_l)^{(0)} = \frac{1}{m}\frac{1}{c_j} = \frac{1}{2}\frac{1}{c_j}, j = 1, 2, i = 1, ..., c_j$

     For 1st phenotype 0220

     $P_{11}(h_1 h_2)^{(0)} = \frac{1}{2}\frac{1}{2} = \frac{1}{4}$

     $P_{12}(h_3 h_4)^{(0)} = \frac{1}{2}\frac{1}{2} = \frac{1}{4}$

     For 2nd phenotype 0000

     $P_{21}(h_1 h_1)^{(0)} = \frac{1}{2}\frac{1}{1} = \frac{1}{2}$

     $P_{11}(h_1 h_2)^{(0)} + P_{12}(h_3 h_4)^{(0)} + P_{21}(h_1 h_1)^{(0)} = \frac{1}{4} + \frac{1}{4} + \frac{1}{2} = 1$

   - Randomly choose haplotype frequencies that satisfies

     $\sum_{t=1}^{h} p_t^{(0)} = 1$

     For example,

     $p_1^{(0)} = \frac{1}{8}, p_2^{(0)} = \frac{1}{4}, p_3^{(0)} = \frac{1}{12}, p_4^{(0)} = \frac{13}{24}$

$p_1^{(0)} + p_2^{(0)} + p_3^{(0)} + p_4^{(0)} = 1$

In this example, I will explain with first way(Make all haplotypes are equally likely.

2. E-step

   (a) Using equation (1.4)

$$\widetilde{P}_{ji}(h_k h_l)^{(g)} = \begin{cases} (p_k^{(g)})^2, & \text{if } k = l, \\ 2p_k p_l, & \text{if } k \neq l \end{cases}$$

   For 1st phenotype

   $\widetilde{P}_{11}(h_1 h_2)^{(0)} = 2p_1^{(0)} p_2^{(0)} = 2\frac{1}{4}\frac{1}{4} = \frac{1}{8}$

   $\widetilde{P}_{12}(h_3 h_4)^{(0)} = 2p_3^{(0)} p_4^{(0)} = 2\frac{1}{4}\frac{1}{4} = \frac{1}{8}$

   For 2nd phenotype

   $\widetilde{P}_{21}(h_1 h_l)^{(0)} = (p_1^{(0)})^2 = (\frac{1}{4})^2 = \frac{1}{16}$

3. M-step

   (a) $P_j^{(0)} = \sum_{i=1}^{c_j} \widetilde{P}_{ji}(h_k h_l)^{(0)}$

   For 1st phenotype

   $P_1^{(0)} = \sum_{i=1}^{2} \widetilde{P}_{1i}(h_k h_l)^{(0)} = \widetilde{P}_{11}(h_1 h_2)^{(0)} + \widetilde{P}_{12}(h_3 h_4)^{(0)}$

   $= \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$

   For 2nd phenotype

   $P_2^{(0)} = \sum_{i=1}^{1} \widetilde{P}_{2i}(h_k h_l)^{(0)} = \widetilde{P}_{21}(h_1 h_1)^{(0)} = \frac{1}{16}$

   (b) $P_{ji}(h_k h_l)^{(0)} = \frac{n_j}{n} \frac{\widetilde{P}_{ji}(h_k h_l)^{(0)}}{P_j^{(0)}}$

   For 1st phenotype

   $P_{11}(h_1 h_2)^{(0)} = \frac{n_1}{n} \frac{\widetilde{P}_{11}(h_1 h_2)^{(0)}}{P_1^{(0)}} = \frac{1}{5}\frac{1/8}{1/4} = \frac{1}{10}$

   $P_{12}(h_3 h_4)^{(0)} = \frac{n_1}{n} \frac{\widetilde{P}_{12}(h_3 h_4)^{(0)}}{P_1^{(0)}} = \frac{1}{5}\frac{1/8}{1/4} = \frac{1}{10}$

   For 2nd phenotype

$P_{21}(h_1h_1)^{(0)} = \frac{n_2}{n}\frac{\widetilde{P}_{21}(h_1h_1)^{(0)}}{P_2^{(0)}} = \frac{4}{5}\frac{1/16}{1/16} = \frac{4}{5}$

$P_{11}(h_1h_2)^{(0)} + P_{12}(h_3h_4)^{(0)} + P_{21}(h_1h_1)^{(0)} = \frac{1}{10} + \frac{1}{10} + \frac{4}{5} = 1$

(c) $p_t^{(1)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{jit}P_{ji}(h_kh_l)^{(0)}$

$p_1^{(1)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{ji1}P_{ji}(h_kh_l)^{(0)}$

$= \frac{1}{2}(\sum_{i=1}^{c_1}\delta_{1i1}P_{1i}(h_kh_l)^{(0)} + \sum_{i=1}^{c_2}\delta_{2i1}P_{2i}(h_kh_l)^{(0)})$

$= \frac{1}{2}(\sum_{i=1}^{2}\delta_{1i1}P_{1i}(h_kh_l)^{(0)} + \sum_{i=1}^{1}\delta_{2i1}P_{2i}(h_kh_l)^{(0)})$

$= \frac{1}{2}(\delta_{111}P_{11}(h_1h_2)^{(0)} + \delta_{121}P_{12}(h_3h_4)^{(0)} + \delta_{211}P_{21}(h_1h_1)^{(0)})$

$= \frac{1}{2}(1 \times \frac{1}{10} + 0 + 2 \times \frac{4}{5}) = \frac{17}{20}$

$p_2^{(1)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{ji2}P_{ji}(h_kh_l)^{(0)}$

$= \frac{1}{2}(\sum_{i=1}^{c_1}\delta_{1i2}P_{1i}(h_kh_l)^{(0)} + \sum_{i=1}^{c_2}\delta_{2i2}P_{2i}(h_kh_l)^{(0)})$

$= \frac{1}{2}(\sum_{i=1}^{2}\delta_{1i2}P_{1i}(h_kh_l)^{(0)} + \sum_{i=1}^{1}\delta_{2i2}P_{2i}(h_kh_l)^{(0)})$

$= \frac{1}{2}(\delta_{112}P_{11}(h_1h_2)^{(0)} + \delta_{122}P_{12}(h_3h_4)^{(0)} + \delta_{212}P_{21}(h_1h_1)^{(0)})$

$= \frac{1}{2}(1 \times \frac{1}{10} + 0 + 0) = \frac{1}{20}$

$p_3^{(1)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{ji3}P_{ji}(h_kh_l)^{(0)}$

$= \frac{1}{2}(\sum_{i=1}^{c_1}\delta_{1i3}P_{1i}(h_kh_l)^{(0)} + \sum_{i=1}^{c_2}\delta_{2i3}P_{2i}(h_kh_l)^{(0)})$

$= \frac{1}{2}(\sum_{i=1}^{2}\delta_{1i3}P_{1i}(h_kh_l)^{(0)} + \sum_{i=1}^{1}\delta_{2i3}P_{2i}(h_kh_l)^{(0)})$

$= \frac{1}{2}(\delta_{113}P_{11}(h_1h_2)^{(0)} + \delta_{123}P_{12}(h_3h_4)^{(0)} + \delta_{213}P_{21}(h_1h_1)^{(0)})$

$= \frac{1}{2}(0 + 1 \times \frac{1}{10} + 0) = \frac{1}{20}$

$p_4^{(1)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{ji4}P_{ji}(h_kh_l)^{(0)}$

$= \frac{1}{2}(\sum_{i=1}^{c_1}\delta_{1i4}P_{1i}(h_kh_l)^{(0)} + \sum_{i=1}^{c_2}\delta_{2i4}P_{2i}(h_kh_l)^{(0)})$

$= \frac{1}{2}(\sum_{i=1}^{2}\delta_{1i4}P_{1i}(h_kh_l)^{(0)} + \sum_{i=1}^{1}\delta_{2i4}P_{2i}(h_kh_l)^{(0)})$

$= \frac{1}{2}(\delta_{114}P_{11}(h_1h_2)^{(0)} + \delta_{124}P_{12}(h_3h_4)^{(0)} + \delta_{214}P_{21}(h_1h_1)^{(0)})$

$= \frac{1}{2}(0 + 1 \times \frac{1}{10} + 0) = \frac{1}{20}$

$p_1^{(1)} + p_2^{(1)} + p_3^{(1)} + p_4^{(1)} = \frac{17}{20} + \frac{1}{20} + \frac{1}{20} + \frac{1}{20} = 1$

4. Stop condition

$\sum_{t=1}^{4}|p_t^{(0)} - p_t^{(1)}| < \varepsilon = 10^{-6}$

$|p_1^{(0)} - p_1^{(1)}| + |p_2^{(0)} - p_2^{(1)}| + |p_3^{(0)} - p_3^{(1)}| + |p_4^{(0)} - p_4^{(1)}|$

$= |\frac{1}{4} - \frac{17}{20}| + |\frac{1}{4} - \frac{1}{20}| + |\frac{1}{4} - \frac{1}{20}| + |\frac{1}{4} - \frac{1}{20}| = 1.2 > 10^{-6}$

As 1.2 is greater than 10^{-6}, go to ♯2

——————— 2nd iteration ———————

2 . E-step

(a) Using equation (5.4)

$$\widetilde{P}_{ji}(h_k h_l)^{(g)} = \begin{cases} (p_k^{(g)})^2, & \text{if } k = l, \\ 2p_k p_l, & \text{if } k \neq l \end{cases}$$

For 1st phenotype,

$\widetilde{P}_{11}(h_1 h_2)^{(1)} = 2p_1^{(1)} p_2^{(1)} = 2\frac{17}{20}\frac{1}{20} = \frac{17}{200}$

$\widetilde{P}_{12}(h_3 h_4)^{(1)} = 2p_3^{(1)} p_4^{(1)} = 2\frac{1}{20}\frac{1}{20} = \frac{1}{200}$

For 2nd phenotype,

$\widetilde{P}_{21}(h_1 h_l)^{(1)} = (p_1^{(1)})^2 = (\frac{17}{20})^2 = \frac{289}{400}$

3 . M-step

(a) $P_j^{(1)} = \sum_{i=1}^{c_j} \widetilde{P}_{ji}(h_k h_l)^{(1)}$

For 1st phenotype,

$P_1^{(1)} = \sum_{i=1}^{2} \widetilde{P}_{1i}(h_k h_l)^{(1)} = \widetilde{P}_{11}(h_1 h_2)^{(1)} + \widetilde{P}_{12}(h_3 h_4)^{(1)}$

$= \frac{17}{200} + \frac{1}{200} = \frac{9}{100}$

For 2nd phenotype,

$P_2^{(1)} = \sum_{i=1}^{1} \widetilde{P}_{2i}(h_k h_l)^{(1)} = \widetilde{P}_{21}(h_1 h_1)^{(1)} = \frac{289}{400}$

(b) $P_{ji}(h_k h_l)^{(1)} = \frac{n_j}{n} \frac{\widetilde{P}_{ji}(h_k h_l)^{(1)}}{P_j^{(1)}}$

For 1st phenotype,

$P_{11}(h_1 h_2)^{(1)} = \frac{n_1}{n} \frac{\widetilde{P}_{11}(h_1 h_2)^{(1)}}{P_1^{(1)}} = \frac{1}{5} \frac{17/200}{18/200} = \frac{17}{90}$

$P_{12}(h_3 h_4)^{(1)} = \frac{n_1}{n} \frac{\widetilde{P}_{12}(h_3 h_4)^{(1)}}{P_1^{(1)}} = \frac{1}{5} \frac{1/200}{18/100} = \frac{1}{90}$

For 2nd phenotype,

$P_{21}(h_1h_1)^{(1)} = \frac{n_2}{n}\frac{\widetilde{P}_{21}(h_1h_1)^{(1)}}{P_2^{(1)}} = \frac{4}{5}\frac{1/16}{1/16} = \frac{4}{5}$

$P_{11}(h_1h_2)^{(1)} + P_{12}(h_3h_4)^{(1)} + P_{21}(h_1h_1)^{(1)} = \frac{17}{90} + \frac{1}{90} + \frac{4}{5} = 1$

(c) $p_t^{(1)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{jit}P_{ji}(h_kh_l)^{(1)}$

$p_1^{(2)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{ji1}P_{ji}(h_kh_l)^{(1)}$

$= \frac{1}{2}(\sum_{i=1}^{c_1}\delta_{1i1}P_{1i}(h_kh_l)^{(1)} + \sum_{i=1}^{c_2}\delta_{2i1}P_{2i}(h_kh_l)^{(1)})$

$= \frac{1}{2}(\sum_{i=1}^{2}\delta_{1i1}P_{1i}(h_kh_l)^{(1)} + \sum_{i=1}^{1}\delta_{2i1}P_{2i}(h_kh_l)^{(1)})$

$= \frac{1}{2}(\delta_{111}P_{11}(h_1h_2)^{(1)} + \delta_{121}P_{12}(h_3h_4)^{(1)} + \delta_{211}P_{21}(h_1h_1)^{(1)})$

$= \frac{1}{2}(1 \times \frac{17}{90} + 0 + 2 \times \frac{4}{5}) = \frac{161}{180}$

$p_2^{(2)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{ji2}P_{ji}(h_kh_l)^{(1)}$

$= \frac{1}{2}(\sum_{i=1}^{c_1}\delta_{1i2}P_{1i}(h_kh_l)^{(1)} + \sum_{i=1}^{c_2}\delta_{2i2}P_{2i}(h_kh_l)^{(1)})$

$= \frac{1}{2}(\sum_{i=1}^{2}\delta_{1i2}P_{1i}(h_kh_l)^{(1)} + \sum_{i=1}^{1}\delta_{2i2}P_{2i}(h_kh_l)^{(1)})$

$= \frac{1}{2}(\delta_{112}P_{11}(h_1h_2)^{(1)} + \delta_{122}P_{12}(h_3h_4)^{(1)} + \delta_{212}P_{21}(h_1h_1)^{(1)})$

$= \frac{1}{2}(1 \times \frac{17}{90} + 0 + 0) = \frac{17}{180}$

$p_3^{(2)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{ji3}P_{ji}(h_kh_l)^{(1)}$

$= \frac{1}{2}(\sum_{i=1}^{c_1}\delta_{1i3}P_{1i}(h_kh_l)^{(1)} + \sum_{i=1}^{c_2}\delta_{2i3}P_{2i}(h_kh_l)^{(1)})$

$= \frac{1}{2}(\sum_{i=1}^{2}\delta_{1i3}P_{1i}(h_kh_l)^{(1)} + \sum_{i=1}^{1}\delta_{2i3}P_{2i}(h_kh_l)^{(1)})$

$= \frac{1}{2}(\delta_{113}P_{11}(h_1h_2)^{(1)} + \delta_{123}P_{12}(h_3h_4)^{(1)} + \delta_{213}P_{21}(h_1h_1)^{(1)})$

$= \frac{1}{2}(0 + 1 \times \frac{1}{90} + 0) = \frac{1}{180}$

$p_4^{(2)} = \frac{1}{2}\sum_{j=1}^{2}\sum_{i=1}^{c_j}\delta_{ji4}P_{ji}(h_kh_l)^{(1)}$

$= \frac{1}{2}(\sum_{i=1}^{c_1}\delta_{1i4}P_{1i}(h_kh_l)^{(1)} + \sum_{i=1}^{c_2}\delta_{2i4}P_{2i}(h_kh_l)^{(1)})$

$= \frac{1}{2}(\sum_{i=1}^{2}\delta_{1i4}P_{1i}(h_kh_l)^{(1)} + \sum_{i=1}^{1}\delta_{2i4}P_{2i}(h_kh_l)^{(1)})$

$= \frac{1}{2}(\delta_{114}P_{11}(h_1h_2)^{(1)} + \delta_{124}P_{12}(h_3h_4)^{(1)} + \delta_{214}P_{21}(h_1h_1)^{(1)})$

$= \frac{1}{2}(0 + 1 \times \frac{1}{90} + 0) = \frac{1}{180}$

$p_1^{(1)} + p_2^{(1)} + p_3^{(1)} + p_4^{(1)} = \frac{161}{180} + \frac{17}{180} + \frac{1}{180} + \frac{1}{180} = 1$

4 . Stop condition

$\sum_{t=1}^{4}|p_t^{(1)} - p_t^{(2)}| < \varepsilon = 10^{-6}$

$|p_1^{(1)} - p_1^{(2)}| + |p_2^{(1)} - p_2^{(2)}| + |p_3^{(1)} - p_3^{(2)}| + |p_4^{(1)} - p_4^{(2)}|$

$= |\frac{17}{20} - \frac{161}{180}| + |\frac{1}{20} - \frac{17}{180}| + |\frac{1}{20} - \frac{1}{180}| + |\frac{1}{20} - \frac{1}{180}| \simeq 1.177777778 > 10^{-6}$

As 1.177777778 is greater than 10^{-6}, go to ♯2

——————— 3rd iteration ———————

after ♯2,3 and 4, we get sum of the differences $0.0220761308 > \varepsilon = 10^{-6}$

——————— 4th iteration ———————

after ♯2,3 and 4, we get sum of the differences $0.00001460861070 > \varepsilon = 10^{-6}$

——————— 5th iteration ———————

after ♯2,3 and 4, sum of the differences $5.930985422 \times 10^{-9} < \varepsilon = 10^{6}$

now we stop

$f = 5$

5. from ♯1 to ♯4, we get frequencies of every type of haplotypes, which are $p_t^{(5)}, t = 1, 2, ..., h$

   $p_1^{(5)} = 0.9$, $p_2^{(5)} = 0.1$, $p_3^{(5)} = p_4^{(5)} = 2.44282 \times 10^{-18}$

   For each iteration, frequencies sum up to 1 but when I expressed with decimal number(not fraction number) Mathematica did round for small numbers so that $p_1^{(5)} = 0.9, p_2^{(5)} = 0.1, p_3^{(5)} = p_4^{(5)} = 2.44282 \times 10^{-18}$ looks like they sum up to > 1 but this is just the matter of expression. For calculation, I used fraction numbers(not decimal numbers) of each calculation so that they sum up to 1.

   Decimal number is just for display(to show clear) not for calculation

6. For each phenotype, pick a genotype(a pair of haplotype) that has the maximum frequencies(product of frequencies of haplotypes that compose the genotype), from all possible genotypes that leads to the phenotype.

Pick genotype for phenotype j=

genotype(which is composed of haplotype k and l) with max $p_k^{(5)} p_l^{(5)}$ value among every genotype that leads to phenotype j, j=1,2

phenotype 1=pick a genotype which has the max value of max($p_1^{(5)} p_2^{(5)} = 0.09$, $p_3^{(5)} p_4^{(5)} = 5.96737 \times 10^{-36}$)

for 0220, $p_1^{(5)} p_2^{(5)} = 0.09$ is the max value so that we pick $h_1 = 0000$ and $h_2 = 0110$

phenotype 2= pick a genotype which has the max value of max($p_1^{(5)} p_1^{(5)} = 0.81$)

for 0000, $p_1^{(5)} p_1^{(5)} = 0.81$ is the max value so that we pick $h_1 = 0000$

As a result, we got two haplotypes 0000, 0110 to resolve all the given phenotypes.

<EM>
Given Phenotypes= {{0, 2, 2, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}

Haplotypes= {{0, 0, 0, 0}, {0, 0, 1, 0}, {0, 1, 0, 0}, {0, 1, 1, 0}}

Initial Haplotype frequencies at 1st run is $\left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right\}$

------------------------------ 0 th iteration ------------------------------

Sum of the frequency differences with previous iteration = 1.200000000

------------------------------ 1 th iteration ------------------------------

Sum of the frequency differences with previous iteration = 0.1777777778

------------------------------ 2 th iteration ------------------------------

Sum of the frequency differences with previous iteration = 0.02207613018

------------------------------ 3 th iteration ------------------------------

Sum of the frequency differences with previous iteration = 0.0001460861070

------------------------------ 4 th iteration ------------------------------

Sum of the frequency differences with previous iteration = $5.930985442 \times 10^{-9}$

Resolve phenotype {0, 0, 0, 0} with haplotype pair {0, 0, 0, 0} and {0, 0, 0, 0}

Resolve phenotype {0, 2, 2, 0} with haplotype pair {0, 1, 1, 0} and {0, 0, 0, 0}

Initial Haplotype frequencies at 2st run is $\left\{\frac{1}{4}, \frac{1}{16}, \frac{3}{8}, \frac{5}{16}\right\}$

------------------------------ 0 th iteration ------------------------------

Sum of the frequency differences with previous iteration = 1.200000000

------------------------------ 1 th iteration ------------------------------

Sum of the frequency differences with previous iteration = 0.1777777778

------------------------------ 2 th iteration ------------------------------

Sum of the frequency differences with previous iteration = 0.02207613018

------------------------------ 3 th iteration ------------------------------

Sum of the frequency differences with previous iteration = 0.0001460861070

------------------------------ 4 th iteration ------------------------------

Sum of the frequency differences with previous iteration = $5.930985442 \times 10^{-9}$

Resolve phenotype {0, 0, 0, 0} with haplotype pair {0, 0, 0, 0} and {0, 0, 0, 0}

Resolve phenotype {0, 2, 2, 0} with haplotype pair {0, 1, 1, 0} and {0, 0, 0, 0}

Performance report

At 1th run, all phenotypes are resolves with haplotypes={{0, 0, 0, 0}, {0, 1, 1, 0}}

All phenoypes are resolved with 2 number of haplotypes in time 0.0156250 Second

Figure 5.1: Result of EM Method

5.4 How to run and the Result

To avoid the local maxima the program runs with several initial conditions. At first, the initial condition is given as all the haplotypes has the same probability. Then, for RunNum-1 number of times, the program restarts with a new initial condition which is randomly pick haplotype frequencies which sum up to 1. Here, RunNum indicates the number of runs and it has difault values 2 but user can give it a values as big as he(she) wants. I will show execution of s3.txt with EM Method. After load s3.txt, Figure 5.2 shows how to run EM Method with option of the number runs(RunNum) and the stop condition(I call it epsilon value. When sum of the difference of frequencies become less than the epsilon value, the program stops executing the frequencies. It also has default values as 0.1). Figure 5.3 shows the result of the program. After the execution, a file named EMOutput.txt is created, which has more detailed information of the execution and the result such as which genotype is resolved with which pair of haplotypes and so on. Figure 5.4 shows the EMOutput.txt.

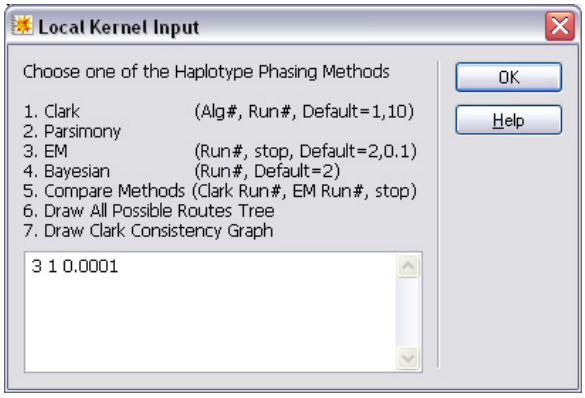


Figure 5.2: Execute the loaded file with EM Method. Run 1 number of times and stop condition(the sum of differences of frequencies is smaller than 0.0001, stops)

```
InputFile: s3.txt

--------------------------------------------------------------------------------
EM haplotype phasing
--------------------------------------------------------------------------------
Given genotypes= {{0, 0, 0, 0}, {1, 0, 0, 0}, {1, 1, 2, 2}, {2, 2, 0, 0}}

<Haplotypes that resolved the genotypes>
EMHapT=         {{0, 0, 0, 0}, {1, 0, 0, 0}, {1, 1, 0, 0}, {1, 1, 1, 1}}


<Number of haplotypes that resolved the genotypes>(Number of haplotypes/Number of resolved genotypes)
EMHapTNum=          4 / 4


<Runnung time>
EMTime=         0.0156250 Second
```

Figure 5.3: Result of the program when run s3.txt with EM Method

5.5 Discussion

As its interpretability and stability, the EM algorithm is very popular algorithm for statistical approach. The output of the EM algorithm, if not trapped in a local mode, is the maximum-likelihood estimate (MLE), which possesses well-established statistical properties. To prevent to be trapped in the local maximum user can give option of the number of run time. The EM Method is stable, always can start and always can resolve all genotypes. Compared to the Clark method and parsimony method, the EM approach is a deterministic procedure, generally(not always)requires less computing time, and is easier for convergence check. Here, the running time is much dependent on the epsilon value so that by reducing the epsilon value, running time could be reduced a lot. This is because the number of iterations, which takes most of the time, is dependent on the epsilon value. Also running time is dependent on the number of heterozygous site and size of the data set as other methods. If the sample size is small, EM algorithm showed better performance(small run time) than parsimony algorithm. However, as the sample size grows, EM algorithm took much more time than parsimony algorithm. EM algorithm and parsimony algorithm runs almost the same number of iteration, however, for EM algorithm, Mathematica

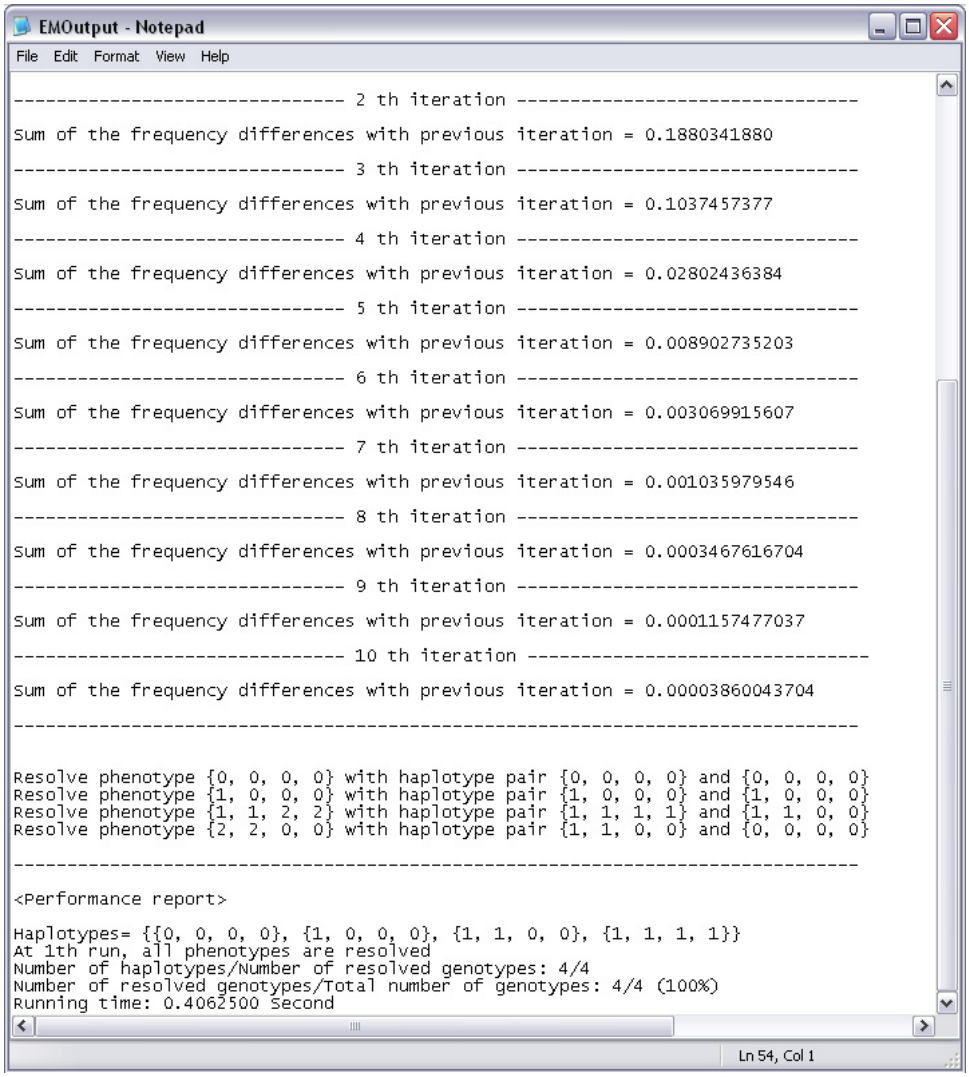


Figure 5.4: EMOutput.txt after run s3.txt with EM Method

takes a lot of time for calculating fraction numbers for frequencies. Moreover, EM algorithm needs a lot of memory space, which makes running time even worse. And from which we can see another problem for EM, which is that the capability of most EM-based approaches is restricted to approximately one dozen loci, because of the memory constraint. Another problem of EM Furthermore, it does not give the parsimony haplotype for every input genotypes. It concerns the maximum-likelihood, in other words the most frequent haplotypes, but not directly concerns the minimum number of total haplotypes. However, it is hard to say which method, the Clark, Parsimony, EM, or other methods such as Bayesian, is better and which method is worse, since the result is dependent much on the input data as no perfect and practical phasing method is developed yet, which we left for future work.

Chapter 6

Comparing methods

It is worth while to compare the performance of four methods so I made the program possible to run all the methods and compare the performance.

6.1 How to run

1. Open PHASEM directory. You will see PHASEM.nb, PhasingH.nbd file, some input files and a directory named "programFiles". You only need PHASEM.nb, PhasingH.nbd file and an input file to execute the program. "programFiles" directory contains program files which you don't need for execution, but if you want to change or add other methods and functions to the programs, you can use it.

2. Open the PHASEM.nb file which is the main function to run

3. If you are working on different directory with the directory that has the PhasingH.nbd file, when you run the PHASEM.nb file, it might report "Cannot find PhasingH.nbd". To set the current directory to the directory that contains PhasingH.bd file, change dirname="directory that contains PhainsgH.nbd file" which is in the 4th line of PHASEM.nb file.

4. Run PHASEM.nb file by putting the cursor on the file and pressing shift+Enter Then the first window will pop up asking the input file name.

5. Select an input file. Figure 6.1 shows selecting "PLEM.txt" as an input file. Then press the ok button.

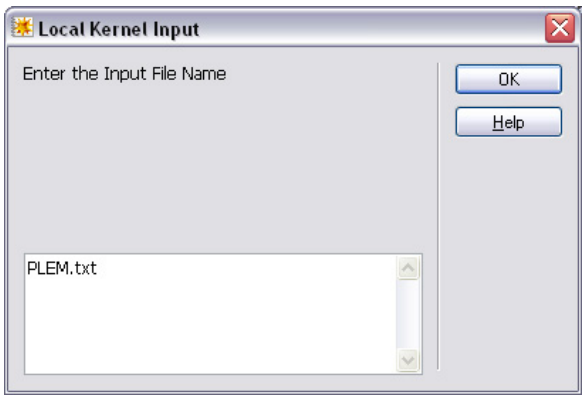


Figure 6.1: Select input file PLEM.txt

6. Select #5 and press the ok button to run all the methods and compare their results. Figure 6.2 shows this.

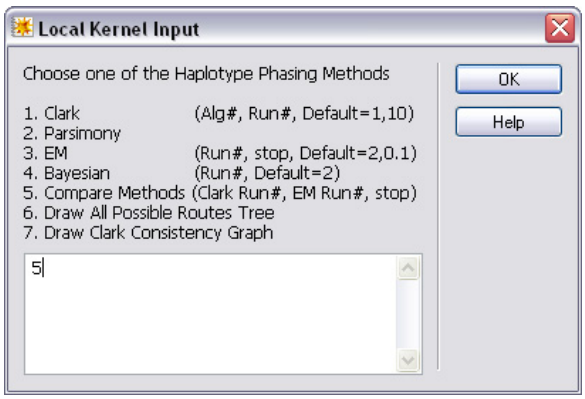


Figure 6.2: Select #5 to compare the results of methods of the input file PLEM.txt

6.2 Result of the program

When execute #5 of the PHASEM program, it initially shows the name of the input file and the genotypes that the input file contains as figure 6.3 shows.

```
InputFile: PLEM.txt

Given genotypes= {{0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}, {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2},
  {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
```

Figure 6.3: Genotypes of PLEM.txt. Name of the input file and the genotypes are shown as the first part of the result.

6.2.1 Compare the performance of methods

Figure 6.4 shows the first part of the result. It shows haplotypes for each methods(Clark, Clark2, Parsimony and EM methods) that resolved the given genotype set. It also shows the number of haplotypes that resolved the genotypes with the number of resolved genotypes and the running time for each method

```
-------------------------------------------------------------------------------------
Compare the performance of four algorithms
-------------------------------------------------------------------------------------
<Haplotypes that resolved the genotypes>
ClarkHapT=          {{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}}
Clark2HapT=         {{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1},
  {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}, {1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}}
ParsimonyHapT=      {{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}}
EMHapT=             {{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}}
BayesianHapT=       {{0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}}

<Number of haplotypes that resolved the
   genotypes>(Number of haplotypes/Number of resolved genotypes)
ClarkHapTNum=       4 / 5
Clark2HapTNum=      5 / 5
ParsimonyHapTNum=   4 / 5
EMHapTNum=          4 / 5
BayesianHapTNum=    4 / 5

<Runnung time>
ClarkTime=          0. Second
Clark2Time=         0. Second
ParsimonyTime=      0.9218750 Second
EMTime=             4.0625000 Second
BayesianTime=       4.0625000 Second
```

Figure 6.4: Result of comparing performance of methods of input file PLEM.txt.

6.2.2 Graphical comparison of the result

Comparison result also shows the graphical comparison of the number of haplotypes that resolved the genotypes using BarChart ,the percentage of resolved genotypes using the Piechart and the running time using the Barchart

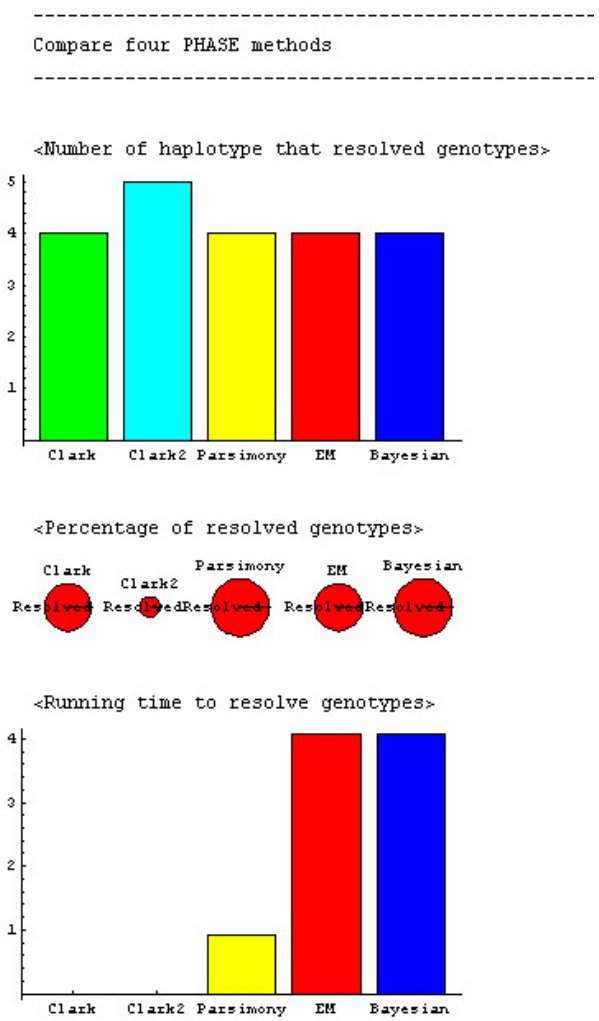


Figure 6.5: Graphical comparison of input file PLEM.txt. Compares the number of haplotypes that resolved the genotypes, Percentage of resolved genotypes and the running time

6.2.3 Clark Phasing Method Graph

Figure 6.6 shows the Clark phasing method graph of genotypes of input file "s26.txt". With the Clark phasing method graph, we can find all the routes that resolve all the given genotypes and which are shown in Figure 6.7. For the Clark phasing method graph and their routes, Chapter 7 of SectionII will explain the detailed algorithm and will describe the program.

```
-------------------------------------------------------------------------------------------
Find all possible routes to resolve the genotypes
-------------------------------------------------------------------------------------------
<Clark Phasing Graph>
Given GenoT= {{0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0},
  {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}, {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
Initial HapTInit={{0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}}
Initial GenoT=
 {{2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}, {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
< Draw all possible routes for Clark Phasing Method >
```

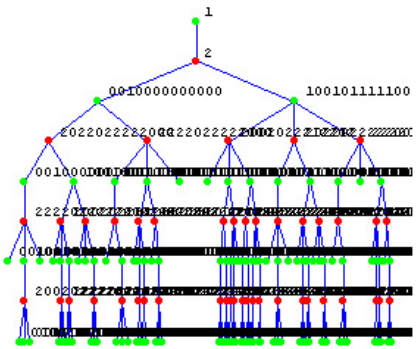


Figure 6.6: Clark phasing method graph that shows all possible routes of phase of input file PLEM.txt

```
< Routes that resolve the most genotypes >
Route 1. {{2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0},
  {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}}
Route 2. {{2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0},
  {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}, {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
Route 3. {{2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0},
  {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}, {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
Route 4. {{2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0},
  {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}}
Route 5. {{2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0},
  {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}}
Route 6. {{2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0},
  {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}, {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
Route 7. {{2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0},
  {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}, {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
Route 8. {{2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0},
  {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2}}
Route 9. {{2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2},
  {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
Route 10. {{2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2},
  {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
Route 11. {{2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2},
  {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
Route 12. {{2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2},
  {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
```

Figure 6.7: Routes that resolved all the genotypes. There are 24 possible routes that resolves all the genotypes. The program shows all the routes but as they are too many, I will show just first 12 routes here. The routes are from Clark Phasing Method Graph of input file PLEM.txt

6.2.4 Clark consistency graph

Figure 6.8 shows the Clark consistency graph of genotypes of input file s26.txt. For the Clark Consistency Graph, Chapter 8 of SectionII will explain the detailed algorithm and will describe the program.

```
-------------------------------------------------------------------------------------------
Clark consistency graph of the genotypes
-------------------------------------------------------------------------------------------

<Clark Consistency Graph>
Given GenoT={{0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0}, {2, 0, 0, 2, 0, 2, 2, 2, 1, 2, 2, 0, 2},
  {2, 0, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}, {2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0}}
```

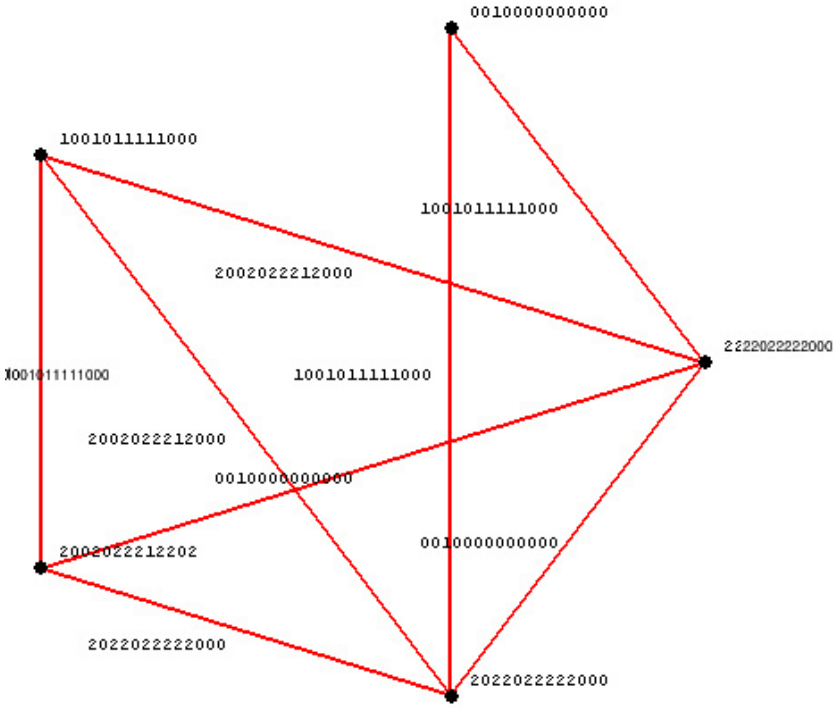


Figure 6.8: Clark consistency graph of the input file PLEM.txt

6.3 Output files

When execute the program, four output files, ClarkOutput.txt, Clark2Output.txt, ParsimonyOutput, and EMOutput.txt is created. If those files already exist, it overwrites. Each of which contains detailed description of the phasing

6.4 Discussion

The input file PLEM.txt that I used above is provided from the web http://www.people.fas.harvard.edu/ junliu/plem/click.html. I used this input to compare the result of my PHASEM program and their PLEM program. As the input file format is different I changed the format to fit in my program PHASEM, however, the contents of the genotypes are the same. Executing PLEM program with those genotypes resulted in the same number of haplotypes to all of my methods. The result of PLEM program is shown in Figure 6.9. As PLEM used EM algorithm,

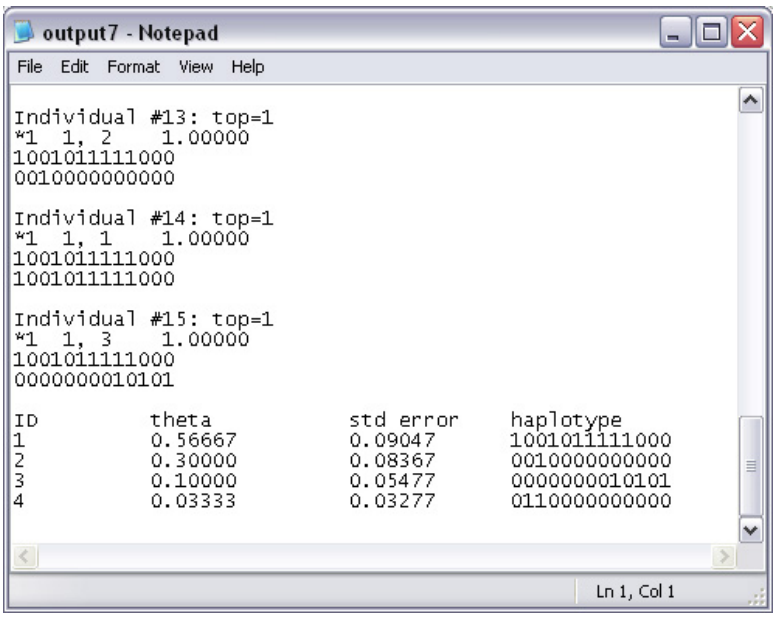


Figure 6.9: Output.txt of PLEM program

the haplotypes that resolved the genotypes are same with my EM method. And also Parsimony results in the same haplotypes. However, for the Clark1 method and the Clark2 method, even the number of haplotypes are the same, sometimes(dependent on the resolving order of genotypes and haplotypes) different haplotypes resolved the genotypes.

By comparing the results of the methods, we can compare the performance of the methods. Generally, Clark2 method showed the worst performance for the number of haplotypes. Most of the time Parsimony and EM showed similar performance(the number of haplotypes), however, dependent on the input file, sometimes EM need little more haplotypes, and sometimes Parsimony needs little more haplotypes. Figure 6.10 shows an example of the result which EM method required two more haplotypes than the Parsimony method. Controversially, Figure 6.11 shows an example of the result that the Parsimony method required one more haplotype than the EM method. Figure 6.11 and Figure 6.12 show the result of the input file, "10kbps.txt", which is the real genotypes from HAPMAP(For detail description of data sets, please see Chapter1 Data set of SectionII). From which we can see that EM method results in the smallest number of haplotypes, which is 8, so that EM method showed the best performance in the sense of the number of haplotypes. The Parsimony method results 9 number of haplotypes which is one more haplotype than the EM, however its performance seems acceptable as one difference is small. Clark method results in 10 number of haplotypes, and Clark2 method results in 15 number of haplotypes, so that we can see that the Clark algorithm is not so good as EM and Parsimony. As the result of the number of haplotypes and the running time are trade off, Clark method and Clark2 method required much smaller running time than EM. One noticeable thing is that Parsimony method is much faster than EM method. Even it is not as fast as Clark methods, it doesn't showed big difference. In conclusion, the Parsimony method seems to be a reasonable method for phasing since it sometimes

gives the minimum number of haplotypes, and even when it is not minimum, it gives acceptable number of haplotypes that are not much deviated from the number of haplotypes that EM method gives. Moreover, Parsimony method method's running time is reasonable compared to EM method which sometimes gives impractical running time. However, it is still dependent on the input file.

```
InputFile: 10kbps.txt
Given genotypes= {{0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 2, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 2, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 2, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 2, 1, 1, 1, 1, 1, 0, 2, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 2, 1, 1, 1, 1, 1, 2, 0, 1},
  {0, 2, 1, 0, 0, 2, 1, 0, 1, 2, 1, 2, 1, 2, 2, 0, 1}, {0, 2, 1, 0, 0, 2, 1, 0, 1, 2, 1, 2, 1, 2, 2, 2, 1},
  {0, 2, 1, 0, 0, 2, 1, 0, 2, 2, 1, 2, 1, 2, 2, 0, 1}, {0, 2, 1, 0, 2, 2, 1, 0, 2, 2, 1, 2, 1, 2, 2, 0, 1},
  {2, 1, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 0, 0, 2}, {2, 1, 2, 2, 2, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 0, 2, 2},
  {2, 1, 2, 2, 2, 1, 2, 2, 2, 2, 1, 1, 1, 2, 1, 0, 0, 2}, {2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 0, 2}}

--------------------------------------------------------------------------------------------
Compare the performance of four algorithms
--------------------------------------------------------------------------------------------

<Haplotypes that resolved the genotypes>
ClarkHapT=
 {{0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1}, {0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
  {0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1},
  {1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0}, {1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0}}
Clark2HapT=        {{0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1},
  {0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1},
  {1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0}, {1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0},
  {1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0}, {1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0}}
ParsimonyHapT=     {{0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1},
  {0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1},
  {0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1}, {1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0}}
EMHapT=            {{0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1},
  {0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1}, {1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0}}
BayesianHapT=      {{0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1},
  {0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1}, {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1},
  {0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1}, {1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0}}


<Number of haplotypes that resolved the genotypes>(Number of haplotypes/Number of resolved genotypes)
ClarkHapTNum=      10 / 17
Clark2HapTNum=     15 / 17
ParsimonyHapTNum=  9 / 17
EMHapTNum=         8 / 17
BayesianHapTNum=   8 / 17


<Runnung time>
ClarkTime=         0.0156250 Second
Clark2Time=        0.0156250 Second
ParsimonyTime=     33.1875000 Second
EMTime=            1501.7968750 Second
BayesianTime=      1501.7968750 Second
```

Figure 6.10: Result of input file 10kbps.txt

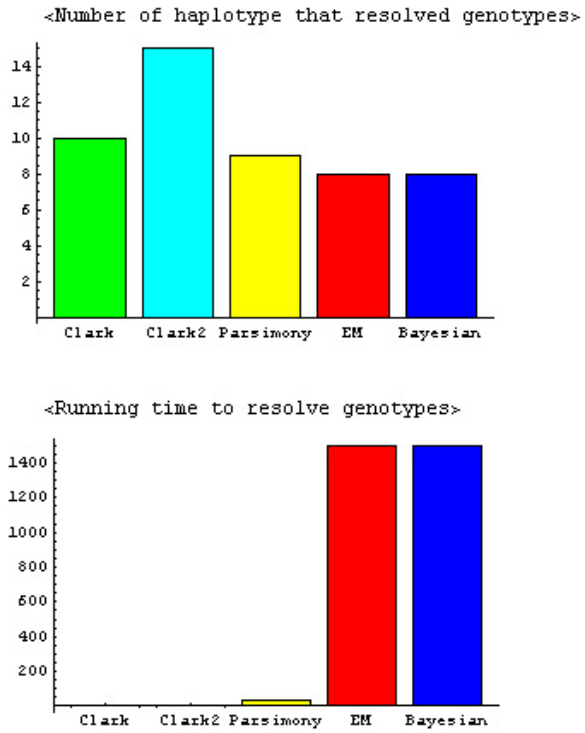


Figure 6.11: Graphical comparison of 10kbps.txt

Chapter 7

Clark Phasing Method Graph

There could be many possible results according to the order of the resolving genotypes and haplotypes. Dependent on the order of resolved genotypes, the result haplotypes could be different. Also there could be multiple haplotypes which could resolves each genotypes so that dependent on the ordering of the haplotypes that possibly resolves each genotype, the resulted haplotype that resolves given genotypes could be different. To see all possible result that could be drawn dependent on the order of genotypes and haplotypes that resolves each genotype, I drew a tree graph that contains all possible pathes of resolving and report all the genotype routes that resolves the most number of genotypes. This could be helpful to understand and improve a phasing method.

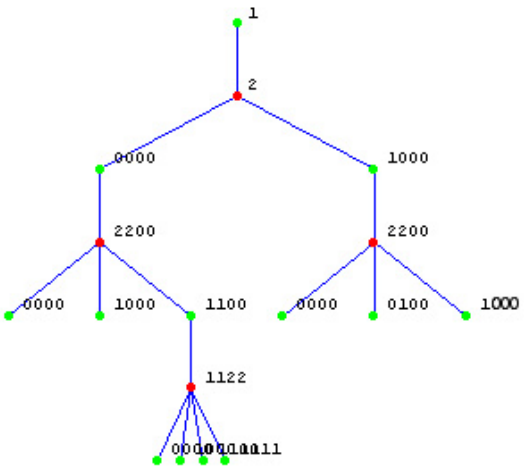
7.1 How to run

To get the Clark phasing method graph, when run PHASEM program, after give the input file name to the first pop up window, choose #6 to the second pop up window.

7.2 Description with an example

Clark Phasing Method Graph has tree structure that shows all the possible routes of phasing. Green node indicates each haplotypes and red node indicates each genotype(except the green node with label 1 at the first level of the tree and the red node with level 2 at the second level of the tree. They are initial nodes and they do not mean a haplotype or a genotype). There is an edge between a haplotype and genotype if the genotype could possibly be resolved with the haplotype. Figure 7.1 shows an example for input of genotypes 0000, 0220, 1000, 1022, and 1220. Initially, homozygous and single heterozygous genotypes, 0000 and 1000 , are set to initial haplotypes so that there are two genotypes, 2200 and 1122 are left to be resolved. As initially there is two haplotypes, 0000 and 1000, at the third level there are two green nodes, each with label 0000 and 1000(the first level of the tree is green node with label 1, and second level of the tree is the red node with label 2, and so on). Let's look at the left child of red node with label 2,which is green node with label 0000. Haplotype 0000 can resolve genotype 2200, which will create a new haplotype 1100. Now there are three haplotypes 0000, 1000 and 1100 which could be used for resolving the left genotypes. These haplotypes, 0000, 1000 and 1100 are shown as green node of fifth level of the tree. One genotype 1122 is left to be resolved and among 0000, 1000 and 1100, only 1100 can resolve the left genotype, 1122, creating a new haplotype 1111. As a result, all the genotypes are resolved and this is the only route which can resolve all the given genotypes. Let's see the right child of the red node with label 2. If 1000 resolves 2200 at the first resolving step, three haplotyes 0000, 0100 and 1000 will be the possible haplotypes that could resolve the left genotype, 1122. However, non of the haplotype could resolve 1122 so that this phase could not resolve all the given genotypes. From this, we can see that the results are different dependent on the order of the resolved genotypes and haplotypes.

```
-------------------------------------------------------------------------------------------
Find all possible routes to resolve the genotypes
-------------------------------------------------------------------------------------------
Given genotypes= {{0, 0, 0, 0}, {1, 0, 0, 0}, {1, 1, 2, 2}, {2, 2, 0, 0}}
<Clark Phasing Graph>
Given GenoT= {{0, 0, 0, 0}, {1, 0, 0, 0}, {1, 1, 2, 2}, {2, 2, 0, 0}}
Initial HapTInit={{0, 0, 0, 0}, {1, 0, 0, 0}}
Initial GenoT={{1, 1, 2, 2}, {2, 2, 0, 0}}
< Draw all possible routes for Clark Phasing Method >
```




```
< Routes that resolve the most genotypes >
Route 1. {{2, 2, 0, 0}, {1, 1, 2, 2}}
```

Figure 7.1: Clark Phasing Method Graph for genotypes 0000, 1000, 2200 and 1122.

Chapter 8

Clark Consistency Graph

Clark consistency graph is useful to see what haplotype could be shared between each pair of genotypes. Clark consistency graph is very useful for haplotype phasing and we can apply it in many ways. We can figure out which haplotype could resolve both genotypes of each pair, we can count the number of genotypes for each haplotypes, and so on. I applied Clark consistency graph to my Parsimony method.

8.1 How to run

To get the Clark phasing method graph, when run PHASEM program, after give the input file name to the first pop up window, choose #7 to the second pop up window.

8.2 Description with an example

Clark Consistency Graph: Each genotype consists a red node and if two genotypes can be resolved using the same sequence of haplotypes, there is an edge. In other words, if two genotypes has same values of 0 or 1 at all the sites where both genotypes does not have value 2, there is an edge between those genotypes. For each

edge, there is a label and the way of labeling the edge is as follows. If one of the genotypes has 1, the label has 1 in that site. If one of the genotypes has 0, the label has 0 in that site. If both genotypes have 2, the label has 2 in that site.

Nodes: Genotypes

Edges: There is an edge between g_1 and g_2, if $\exists h$ that can explain both genotypes as it has other two haplotypes h', h'',such that$\{h, h'\}$ explain $g_1, \{h, h''\}$ explain g_2 figure 8.1 shows an example of Clark consistency graph. Genotypes are given as 00000222100, 00002201202 and 00020010220. Between 00000222100 and 00002201202, an edge exist with label 00000201100 as 00000001100 and 00000101100 could be used to resolve both genotypes. Between 00000222100 and 00020010220, an edge exist with label 00000010100 which could be used to resolve both genotypes. However, between 00002201202 and 00020010220, there are no possible haplotypes that could be shared so that there is no edge between them.

```
InputFile: s13.txt


-----------------------------------------------------------------------------------------
Clark consistency graph of the genotypes
-----------------------------------------------------------------------------------------
Given genotypes=
 {{0, 0, 0, 0, 0, 2, 2, 2, 1, 0, 0}, {0, 0, 0, 0, 2, 2, 0, 1, 2, 0, 2}, {0, 0, 0, 2, 0, 0, 1, 0, 2, 2, 0}}
<Clark Consistency Graph>
Given GenoT=
 {{0, 0, 0, 0, 0, 2, 2, 2, 1, 0, 0}, {0, 0, 0, 0, 2, 2, 0, 1, 2, 0, 2}, {0, 0, 0, 2, 0, 0, 1, 0, 2, 2, 0}}
```

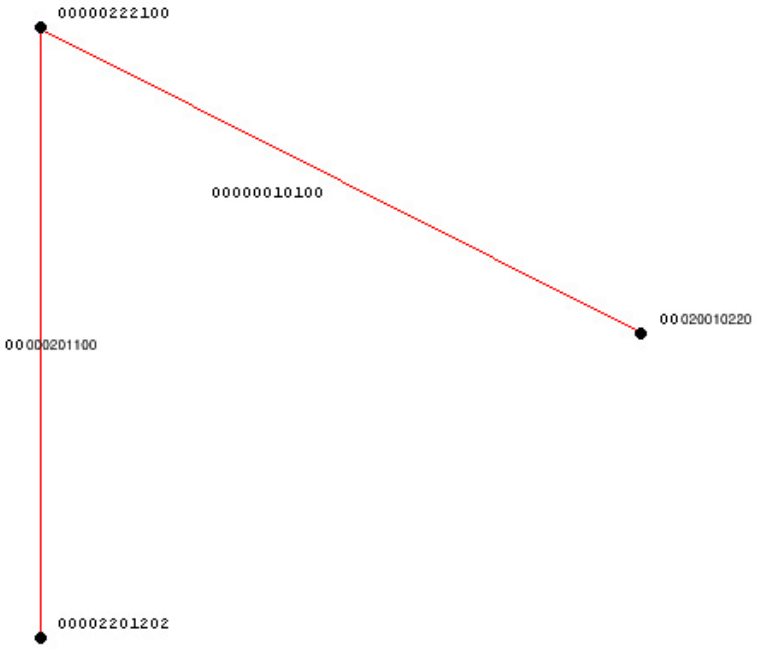


Figure 8.1: Clark Consistency Graph for input file s3.txt

Chapter 9

Conclusion and Future work

I implemented four kinds of haplotype phasing method. The first one and the second one are the Clark1 method and Clark 2 method, which implement Clark Algorithm in two different ways of comparing genotypes and haplotypes. The third one is Parsimony method, which is a method that I created, applying the Clark consistency graph to get the parsimony criteria. The fourth one is EM method, which applied EM algorithm. In addition to these methods, I implemented a comparison function which executes all those methods and compares the performance. For Clark1 method, Clark2 method and EM method, user can give options of the number of runs. For EM method, it also has option for stop condition. Also all the options has the default values, in case that a user does not give the values. I also implemented Clark phasing graph function which shows a tree with all possible routes of resolving the given genotypes using Clark algorithm. In addition, I implemented Clark consistency graph function which shows Clark consistency graph for a given genotypes. How to run the programs are explained in each corresponding chapter of SectionII.

There are two ways to compare the performance of methods. One of which is comparing the number of haplotypes that resolves the given genotypes and the other is

the running time of the methods. For the number of haplotypes, EM method and Parsimony method showed the best. Sometimes EM showed the best, and sometimes Parsimony showed the best, dependent on the input file. For the running time, Clark2 method showed the best and Clark1 method was also good, just a little bit more than Clark2 method. However, Clark2 has the worst performance for the number of haplotypes so that it is not preferable. The noticeable thing is that EM method showed poor running time compared to other methods, however, the Parsimony method showed not big difference in running time with Clark1 and Clark2 methods. In conclusion, I prefer to use Parsimony method as the number of haplotypes is small(sometimes the best and sometimes a little more than EM but still good) and also the running time is acceptable, which is much smaller compared to EM method. However, I still cannot guarantee which method is the best for haplotype phasing since the performance is dependent on the input genotypes. The performance and the best method are different, dependent on the number of genotypes, the number of ambiguous sites, recombination rates, and so on. I recommend to use this program, PHASEM, for phasing genotypes, for testing each algorithms, comparing algorithms. And I believe this program could be used helpfully for developing a good phasing method in future work. I left Bayesian method to be implemented as the Bayesian is also good to be used for haplotype phasing. User can implement their function such as Bayesian and add to the program(for detail see Chapter2 of SectionII).

There is no perfect haplotype phasing method yet and many researchers are trying to find a better method and also trying to improve the existing methods. I hope my work could be a part of such trials and could make a contribution towards the development.

Bibliography

[1] Andrew G. Clark (1990), *Inference of Haplotypes from PCR-amplified Samples of Diploid Populations*, Mol. Biol. Evol., 7(2):111–122.

[2] Laurent Exceffier and Montgomery Slatkin (1995), *Maximum-Likelihood Estimation of Molecular Haplotype Frequencies in a Diploid Population*, Mol. Biol. Evol., 12(5):921–927.

[3] Jdff A. Bilmes (1998), *A Gentle Tutorial of the EM Algorithm and iis Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*, International Computer Science Institution .

[4] Earl Hubbell (2000), *Finding a Parsimony Solution to Haplotype Phase Is NP-hard.*

[5] Earl Hubbell (2000), *Finding a Maximum Likelihood Solution to Haplotye Phase Is Difficult.*

[6] Dan Gusfield (2000), *A Practical Algorithm for Optimal Inference of Haplotypes from Diploid Populations*, Americam Association for Artificial Intelligence(www.aaai.org).

[7] C. Neuhauser (2000).*Mathematical Models in Population Genetics.*, In Handbook of Statistical Genetics, Wiley, 153-172.

[8] Matthew Stephens, Nicholas J. Smith, and Peter Donnelly (2001), *A New Statistical Method for Haplotype Reconstruction from Population Data*, Am J hum Genet, 68:978–989.

[9] Dan Gusrield (2001), *Inference of Haplotypes from Samples of Diploid Populations: Complexity and Algorithms*, J.Computational Biology, 8(3):305–324.

[10] Z.S. Qin, T. Niu and J.S. Liu (2002) "Partition-Ligation EM Algorithm for Haplotype Inference with Single Nucleotide Polymorphisms".

[11] Tianhua Niu, Zhaohui S. Qin, Xiping Xu, and Jun S. Liu (2002), *Bayesian Haplotype Inference for Multiple Linked Single-Nucleotide Polymorphisms*, Am J hum Genet, 70:157–169.

[12] Dan Gusfield (2003), *Haplotype Inference by Pure Parsimony*, UC Davis CST Report CSE-2003-2.

[13] Lusheng Wang and Ying Xu (2003), *The EM algorithm and its implementation for the estimation of frequenceis of SNP-Haplotypes*, Int. J. Appl. Math. Comput. Sci. 13(3):419–429

[14] Joanna POLANSKA (2003), *Haplotype inference by maximum parsimony*, Bioinformatics, Oxford University 19(14):1773–1780

[15] Bjarni V.Halldorsson, Vineet Bafna, Nathan Edwards, ross Lippert, Shibu Yooseph , and Sorin istrail (2004), *A Survey of Computational Methods for Determining Haplotypes*, S.Istrail et al.: SNPs and Haplotype Inference, LNBI 2983, pp.26-47. Springer Verlag.

[16] Dan Gusfield (2004), *An Overview of Combinatorial Methods for Haplotype Inference*, Lecture Notes in Computer Science, pp. 9–25. Springer Verlag.

[17] Danial G. Brown and Ian M. Harrower (2004), *A new Integer Programming Formulation for the Pure Parsimony Problem in Haplotye Analysis*, Lecture Notes in Computer Science, pp.254–265. Springer Verlag.

[18] Roded Sharan, Bjarni V. Hallorsson, and Sorin Istrail (2006), *Islands of Tractability for Parsimony Haplotyping*, IEEE/ACM, 3:303–311.

[19] Josh Akey *Lecture 1: Introduction to Statistical Algorithms for Haplotype Reconstruction and Linkage Disequilibrium*, http://noble.gs.washington.edu/ noble/genome541/lectures/Akey_Lecture1.pdf

[20] PLEM program. http://www.people.fas.harvard.edu/$\tilde{}$junliu/plem/click.html.

[21] Mathematica http://shum.huji.ac.il/cc/mathintr.html.

[22] Haplotype Estimation http://www.meb.ki.se/genestat/tl/genass_ldmap/ measuring_ld/estimation_haplo/haplotype_estimation.htm

[23] HAPMAP http://www.hapmap.org

[24] Wikipedia http://en.wikipedia.org