This thesis by Dong Wook Kim 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 _____ _____

Franco P. Preparata, Advisor

Approved by the Graduate Council.

DATE _____ _____

Sheila Bonde, Dean of the Graduate School

Haplotype Phasing using Pre-Resolved Table on the Ancestral Tree Structure

By
Dong Wook Kim
B.S., City University of New York Baccalaureate Program, 2006
B.A., Kyungwon University, 1999

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 2009`

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 _____ _____

Dong Wook Kim, 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 _____ _____

Dong Wook Kim, Author

Acknowledgements

I am grateful to my advisor, Professor Franco P. Preparata, who has always been very generous to me always with his time and advice. While I have studied in the field of Computational Biology which I found more fascinating problems in than others, I have derived inspiration from him.

I thank my parents, Kwang Je Kim and Myung Soon Oh, for their ceaseless love and support. I want them to know how much I love them. Without their sustaining love I could not complete anything.

Special thank to lovely Hee Chung and Grace and Samuel Kim, who are everything in my life. Everything was fulfilled by your sacrifices.

Contents

List of Figures and Tables

I. Introduction

1.1 Basic Genetic Background

In this work, we study and construct a haplotype phasing method on an ancestral tree structure which has a pedigree network shape based on phylogenetic information. In order to explain about haplotype inference, first we describe basic aspects of genetic structure related to haplotype phasing.

A diploid organism such as a human has two strands for each base pair. One strand can be represented its sequence by A, C, G and T. As is well known, the human genome of each individual has 23 pairs of chromosomes and totally about three billion base pairs across the whole human chromosomes.

A SNP (*Single Nucleotide Polymorphisms*) in a genome is to represent positions in a DNA (*Deoxyribonucleic Acid*) sequence variation which occurs at two or more different bases in the population for reasons such as recombination, mutation, insertion, deletion and so on. While every individual has almost the same sequence in more than 99% of genomes across the population, only a difference of less than 1% causes different characteristics, appearance in population. Moreover the susceptibility of family group to a certain disease is due to this small difference. The study of SNP basically will be helpful in understanding for this disease association condition.

For computational base study, we usually transform DNA's variance, which consist of alphabetical values {A, C, G, T}, into the numeric values {0,1}. For example, let's assume there is a part of the DNA strand in a genome like 'TAGA***T***A' in most of the population. If a certain

variation has been reported by some different individuals like 'TAGA***C***A', the position at which a SNP viewed can be called *allele*. While SNP shows a variation frequency, SNP can be denoted by *0* or *1*. Typically *0* is assigned the major allele which has a dominant frequency within a population while *1* denotes the minor allele. For example, if the nucleotide *T* is shown in 90% in the population while the nucleotide *C* is in 10% of them at the same position, *T* can be substituted for *0* while *C* is *1*. A *haplotype* (*hapl*oid geno*type*) is one DNA stand which consists of a set of allele in a chromosome. Thus, a haplotype with *n* loci can be represented as a length *n* string in $\{0,1\}^n$.

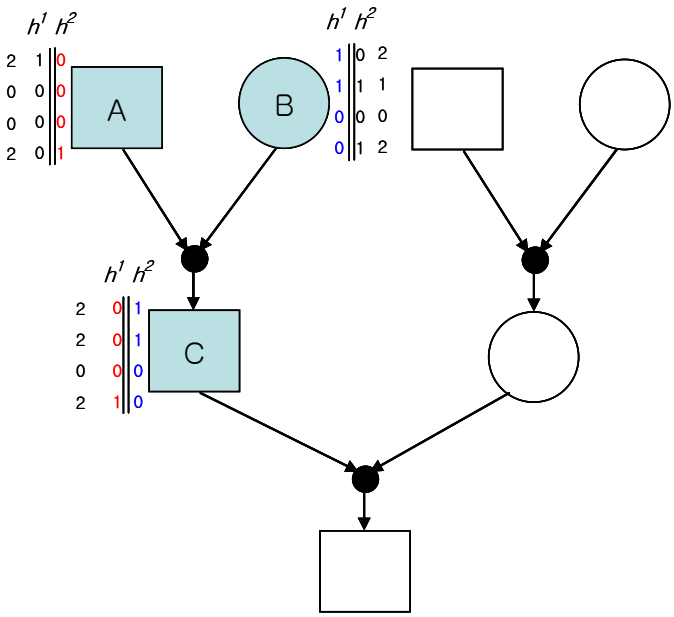
A pair of haplotypes $< h_1, h_2 >$ is a *genotype*. If the two alleles of h_1 and h_2 are the same at the same locus, it is called a *homozygous* locus. A homozygous locus is denoted by *0* or *1*. Otherwise if the two alleles of h_1 and h_2 are different at the same locus, it is called a *heterozygous* locus and the corresponding locus is *2*. Thus, a genotype with *n* loci can be represented as a length *n* string in $\{0,1,2\}^n$. For example, let's suppose the two haplotypes within the individual consist of $< h_1, h_2 >$ = < 110011, 101101 >. Both the first and the sixth loci are 1 which is a homozygous locus, and the second to the fifth loci are 0/1 or 1/0 which is a heterozygous locus. Hence, a genotype *g* is represented by < 122221 >.

1.2 Understanding of Genetic Information in a Pedigree

A *pedigree* is a formally connected graph of genetic information. It can be defined as a directed graph *G*.

$$G = (V, E)$$

, where $V = M \cup F \cup m$, with M denoting a male node, F denoting a female node and m the mating node, and $E = \{(u,v) \mid u \in M \cup F \text{ and } v \in m\}$.



<Figure 1> shows the pictorial representation of a pedigree graph with 7 individuals of a family, with a square representing a male node, a circle representing a female node, and a black dot the mating node. A child node is placed under ones parents. A–B–C combination can be called a *father–mother–child trio*, or a *parents–offspring trio*, or simply *trio*.

Diploid organisms, such as humans, have two identical copies of each chromosome. Already mentioned above in section 1.1, the one state copy of two is called allele which is regarded as a haplotype. The child must inherit one allele out of two from each parent under the *Mendelian law*[1]. For example, in the case of offspring C, the most left allele h_C^1 = '0001' is inherited from his father A , h_A^2 = '0001', while the right one h_C^1 = '1100' is from mother B , h_B^1 = '1100'. Moreovere a pair of < h_C^1 , h_C^2 > = < 0001, 1100 > is C's genotype, g_c = '2202'. (see in Figure 1)

Biologically, one individual must have only the one father-mother mating combination. A group of parents and child is called a *parents-offspring trio*, or simply just *trio*. According to whether an input data has this pedigree information or not, genotype data can be divided by pedigree data and population data. Haplotyping pedigree data is believed to be more reliable than haplotyping population data for unrelated individuals: the constraint provided by parents-offspring relationships in a pedigree could force one to settle on a unique haplotype configuration as being most probable. (Qiangfeng Zhang, 2005). In this paper, data on this pedigree graph concept will be used for the algorithm to solve the haplotype problem from genotype data.

[1] Mendelian law : This explains the theoretical understanding of the genetic inheritance. There is the hereditary determinant which is called gene. Each parent has a gene pair in each cell. One pair of the gene segregates into a gamete. Gametes are randomly accomplished with the involved gene pairs.
(*Wikipedia , http://en.wikipedia.org/wiki/Mendelian_inheritance*)

II. Haplotype Phase

2.1 The Concept of Haplotype Inference

Haplotypes are not directly observable though present experimental techniques, but only unphased genotype data can be obtained. Haplotyping is one of the major issues for genetic association studies because the cost of genotyping is still very high for whole-genome association studies. Furthermore, analyzing SNP provides to find genetic difference in certain diseases between a case and control population. These challenges lead to the development of the haplotype phasing method.

Input : A set of n genotypes

$G = \{g_1, g_2, \ldots, g_n\}$

Output : A set of n haplotype pairs

$< \overline{h_{i1}}, \overline{h_{i2}} >= (< h_{i1}, h_{i2} >| h_{i1} \oplus h_{i2} = g_i)$

<Figure 2> illustrates the basic concept of the haplotype phasing problem. While Input data is a set of unphased genotypes, the application for haplotypes phasing will resolve their corresponding haplotype pairs.

A genotype with n heterozygous loci has 2^{n-1} haplotype possible pairs technically. For example, g = < 2122 > can have four haplotype pairs: < h_1, h_2 > = { < 1111, 0100 > , < 1101, 0101> , < 1101, 0110> , < 1100, 0111> }. Hence, the algorithm for haplotype inference from unphased genotype

data is required for finding most probable haplotype pairs.

From the given problem, a haplotype pair h_{i1} and h_{i2} which seem most g_i explicable resolve the genotype. For example, there is a genotype input g = < 2122 >, then < h_1, h_2 > = < 1111, 0100 > are determined for its haplotype pair. However, the solution is not straightforward because there exists resolution ambiguity if < h_1, h_2 > = < 1101, 0101> are its real pair. To solve this ambiguity there are many methods.

Typically the methods are categorized based on four major principles : parsimony, phylogeny, maximum-likelihood and Bayesian inference (Eric Xing, 2006). Even though statistical methods such as maximum-likelihood or Bayesian methods usually infer the efficient ways to resolve the population genotype data from independent individuals, they are very time consuming, especially the case of large data or large number of marker loci. Besides, in the case of family based disease, such as Alzheimers or ischaemic heart disease, the genetic information on pedigree data is very important. In this study, the pedigree data and the new haplotype phasing methodology on ancestral pedigree data are most focused.

Through the next section 2.2, we will look over the numerous haplotype phasing approaches. For comparing performance and resolution of the statistical approach to the work in this paper, the EM algorithm will be explained and implemented.

2.2 The Well-Known-Methods of Hapotype Inferences

In order to understand better the haplotype inference, the representative algorithm of each distinguished categories will be simply viewed.

2.2.1 Parsimony-based methods.

The algorithm attempts to reduce the scope of possible haplotypes in observed genotype samples as a sort of parsimony approach. In this category, Clark's algorithm(1990) is the earliest and most famous haplotype phasing algorithm. It resolves haplotype pairs in the following way :

1) find all homozygotes and single-site heterozygosis that have unambiguous haplotypes.
2) in every remaining unresolved genotypes, attempt to find a haplotype pair which can be made of the ambiguous sites.
3) return to step 2) until all samples have been resolved.

Clark stated that all resolved haplotypes based on maximum parsimony have a unique and correct solution. Although the algorithm works in principle, there exist some limitations as following :

1) when there are no homozygotes or single-site heterozygotes in a genotype sample, it may never get the trigger to start.
2) There may exist the remaining unresolved haplotypes.
3) Depending on the order of sample of genotypes, the algorithm might bring a different solution instead of a unique one.

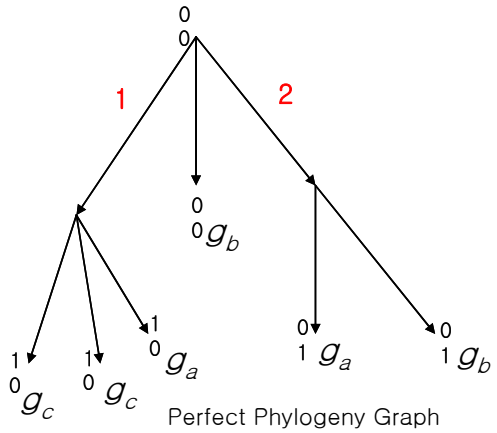
2.2.2 Phylogeny-based methods.

Basically, the method in this paper is the closest to phylogeny-based methods among the categories. A Phylogeny-based algorithm intends to deterministically deduce haplotype phases based on phylogenetic reconstruction (Gusfield, 2002). These approaches intend to find a set of haplotypes which resolve a genotype by following the coalescent model. In general, the coalescent model of haplotype evolution has two key assumptions: no recombination and the infinite-sites model. In a like manner, the work in this paper has the same assumption.

One of most famous methods is the Perfect Phylogeny Haplotyping (PPH) algorithm which can be solved in linear time, O(nmα(nm)) where α is the inverse Akerman function.(Gusfield, 2003) Basically the PPH is used to determine whether there exist unique resolved haplotypes where its inference can be derived on a perfect phylogeny tree.

	g_a	g_b	g_c
locus1	2	0	1
locus2	2	2	0

	h_{a1}	h_{a2}	h_{b1}	h_{b2}	h_{c1}	h_{c2}
locus1	1	0	0	0	1	1
locus2	0	1	0	1	0	0



< Figure 3> Example of a PPH problem

This represents an example of PPH (*Perfect Phylogeny Haplotyping*) approach. In this example, when $\begin{smallmatrix}0\\0\end{smallmatrix}$ is set as initial state, the first site of g_a and g_c are changed, then follow by 1. The second site of g_a and g_b are changed, then follow by 2. Finally perfect phylogeny can be drawn like right side tree and the pairs of haplotype will be resolved with this.

Formally the PPH algorithm is simply defined as that through the method of building a perfect phylogeny tree, if there is a given $s \times t$ genotype matrix M with $M[i,j] \in \{0, 1, 2\}$ where s is the number of individuals and t is the length of each genotype, the determining $2s \times t$ corresponding haplotype matrix M' with $M' \in \{0, 1\}$ will be its final goal. However, in some cases that perfect phylogeny is not possible, it cannot be resolved correctly.

2.2.3 Maximum-likelihood-based methods.

The EM (Expectation Maximization) algorithm is one of the popular methods for this category. Based on MLE (Maximum Likelihood Estimation), the EM algorithm for haplotype phasing infers the most reliable population haplotype probabilities. This is a more explicable HWE (Hardy–Weinberg equilibrium)[2] assumption than other methods because this assumption is based on solid statistical theory. However, it has strong limitation on genotype length. Moreover its performance is sensitive to the initial value. If there exist local maxima, the iteration may lead to locally optimal MLEs, which become most serious when there are many distinct haplotypes (Tianhua Niu, 2004). In the last section of Chapter II, the EM algorithm will be viewed in detail.

2.2.4 Bayesian inference-based methods.

Like the maximum-likelihood methods, Bayesian inference takes a statistical approach. However, while the maximum-likelihood methods focus on finding a haplotype pair that maximizes the probability of genotype from a given model, Bayesian inference aims to find the posterior

[2] HWE (Hardy–Weinberg equilibrium) : The Hardy–Weinberg principle states that both allele and genotype frequencies in a population remain constant unless evolutionary change occurred. (*Wikipedia , http://en.wikipedia.org/wiki/Hardy-Weinberg_principle*)

distribution of the model parameters given the genotype data. In other words, while the maximum likelihood method focuses on solving *argmax* $P(G|\theta)$, Baysian inference tries to find the posterior probability $P(H|G)$, where G denotes genotypes and H corresponding haplotypes.

2.3 The Implementation of the EM algorithm for Haplotype Phase

2.3.1 General Information about the EM algorithm

The Expectation Maximization (EM) algorithm is a general method of finding the maximum-likelihood estimate of the parameters of a distribution from a given data set. Typically the implementation of the EM algorithm alternates between *E-step* (expectation step), which computes an expectation of the likelihood from the observed data, and *M-step* (maximization step), which computes the MLE of the parameters by maximizing the results from the E-step. The computed parameters of the M-step are used as the parameters for the consequent E-step, and the process is repeated.

E-step $q^{(t+1)} = \arg\max_{q} Q(q \mid \theta^{(t)})$

,where

$$q^{(t+1)}(z \mid x) = P(z \mid x, \theta^t) = \frac{P(z, x \mid \theta^t)}{P(x \mid \theta^t)} = \frac{P(x \mid z, \theta^t)P(z \mid \theta^t)}{\sum_z P(x \mid z, \theta^t)P(z \mid \theta^t)}$$

M-step $\theta^{(t+1)} = \arg\max_{\theta} Q(q^{(t+1)}, \theta)$

,where

$$\begin{aligned} Q(q, \theta^t) &= \sum_z q(z \mid x) \log P(x, z \mid \theta^t) - \sum_z q(z \mid x) \log q(z \mid x) \\ &= \sum_z P(z \mid x, \theta^t) \log P(x, z \mid \theta^t) - \sum_z P(z \mid x, \theta^t) \log P(z \mid x, \theta^t) \\ &= \log P(x \mid \theta^t) \end{aligned}$$

<Figure 4> shows the simply defined formula of the EM algorithm. E-step(Expectation Step) and M-step(Maximization Step) will be iterated until they satisfy $l(\theta^{t+1}) \le l(\theta^t)$. Therefore, $\hat{\theta}_{ML}$ will be obtained.

Consider x for observed variables and z for latent variables. We want to model P(x, z | θ) by finding the most likely parameter set, $\hat{\theta}_{ML} = \arg\max_{\theta} P(x \mid \theta) = \arg\max_{\theta} \sum_{z} P(x, z \mid \theta)$. The EM algorithm iteratively improves an initial estimate $\theta^{(0)}$ by constructing new estimates $\theta^{(t)}$ until it converges to certain point, which means it satisfies $l(\theta^{t+1}) \le l(\theta^{t})$. After that $\hat{\theta}_{ML}$ will be obtained.

2.3.2 The EM algorithm for Haplotype Phasing

The EM method estimates the population for haplotype frequencies $P(h_i)$ based on their likelihood, L, given the genotype data G:

$$L(H) = P(G \mid H) \approx \prod_{i=1}^{N} P(g_i)^{f_i} = \prod_{i=1}^{N} \sum_{<h_j, h_k>: h_j \oplus h_k = g_i} P(h_j, h_k)^{f_i}$$

since f_i are the genotype sample frequencies.

When G denotes the observed unphased genotype for n individuals, g_i denotes the observed unphased genotype data for the ith individual, H denotes the overall haplotype frequencies, h_j and h_k denote the respective haplotype frequenceis for genotype g_i, such that the haplotype pair (h_j, h_k) is compatible with the i^{th} observed genotype g_i. If we assume m be the length of data, the maximum number of possible haplotypes will be 2^m, such that $P(h_j)$ is the population frequency of haplotype h_j, where j = 1, … , 2^m.

The EM procedure for the haplotype phasing problem is defined as follows :

All equal values are assigned to $P(h)^{(0)}$ initially. In the Expectation step, the haplotype frequencies are used to estimate the expected genotype frequency $P(h_j, h_k)^{(t)}$ where t denotes the t^{th} iteration. In the Maximization step, the expected genotype frequencies $P(h_j, h_k)^{(t)}$ are used to re-estimate the haplotype frequencies $P(h)^{(t)}$. Then expectation and maximization steps are iterated until $P(h)^{(t)}$ converges to some point.

2.3.3 Implementation

For estimating $P(h_k , h_l)$, let $F = f_{i[k\ l]}(p)$ be an m x n matrix in parameter $(h_1,\ldots,h_k,\ldots,h_l,\ldots,h_d)$ since d is the number of haplotypes which can explain all observed genotypes. Then, matrix F will denote the probabilities that the observed genotypes are generated by specific pairs of haplotypes. Each row represents an observed genotype g_i, and each column represents a pair of haplotypes (h_k, h_l).

$$f_{i[k,l]}(p) = \begin{cases} p_k p_l = p_k^2 & \text{If } k = l \text{ and } h_k \oplus h_l = g_i \\ 2 p_k p_l & \text{If } k \neq l \text{ and } h_k \oplus h_l = g_i \end{cases}$$

Matrix U will denote the distribution of f_{ij} with observed frequencies of genotypes. Then we can infer the parameter p, which is $P(h)$, to maximize the probability of observing the data.

$$\max L(P) = \arg\max \prod_{i=1}^{N} P(h)^u$$

Then $argmax\ L(P)$ will be used to re-estimate through iteration.

$$P_i^0 = \frac{1}{D}$$

While {

$$u^t_{i[k,l]} = u_i \frac{f_{i[k,l]}(p^t)}{\sum_{k,l} f_{i[k.l]}(p^t)}$$

E-step

$1 \leq \quad i \leq m, \quad$ where m = number of genotypes

$1 \leq \quad k < l \quad \leq \quad d \quad$ where d = number of haplotypes.

$P^{t+1} = \text{argmax } L(P^t)$ M-step

if $\quad |P^{t+1} \quad - \quad P^t| \quad < \varepsilon$

break;

}

$\overline{P} = P^{t+1}$

< Figure 5> The EM algorithm implementation

III. Haplotype Inference using the Pre-Resolved Table on the Ancestral Tree Structure

3.1 Haplotypes Rearrangement in the Parents-Child Trio

3.1.1 Deduction of Possible Genotype Trio Pattern

As mentioned above in Chapter I, the child in trio receives one allele from its paternal node and the other from its maternal node. The basic idea of the Pre-Resolved Table algorithm is based on the fact that the position order of haplotypes in a gene across all nodes can be rearranged without losing the pattern of genotype of each node, while we assume there is no recombination or mutation at any loci.

Theorem 1) The combination of genotypes in parents-child trio shows the pattern of haplotypes which the child node inherits from its parental node.

Let g_f be the genotype of father and g_m be of mother, g_c child. Then, the genotype g_f has a pair of haplotypes $<h_f^1, h_f^2>$ and g_m has $< h_m^1, h_m^2>$, g_c has $< h_c^1, h_c^2>$. For instance, if g_f is 0, then we can solve its pair of haplotypes into $< h_f^1, h_f^2> = < 0, 0 >$. If $< g_f, g_{m>}$ is $< 0, 1 >$, g_c should always be 2, that is $< h_c^1, h_c^2> = < 0, 1 >$ or $< 1, 0 >$. Furthermore, while $< g_f, g_{m>}$ is $< 0, 1 >$, then

we cannot observe the $g_c = 0$ or 1, that is $< h_c^1, h_c^2 > = < 1, 1 >$ or $<0, 0>$. In other examples, let us consider the case of $< g_f, g_m, g_c > = < 2, 0, 1 >$. This is an impossible combination. g_f can have either $< h_f^1, h_f^2 > = < 0, 1 >$ or $< 1, 0 >$, and g_m should be $< h_m^1, h_m^2 > = < 0, 0 >$. So, the only possible g_c is 2 or 0 that has one out of three cases among $< h_c^1, h_c^2 > = < 0, 1 >$, $< 1, 0 >$, or $< 0, 0 >$. In other words, $< g_f, g_m, g_c > = < 2, 0, 2 >$ or $< 2, 0, 0 >$ will be only possible cases when $< g_f, g_m > = < 2, 0 >$. Hence, there exist some possible genotype trio patterns of its combination such as $< g_f, g_m, g_c > = < 0, 0, 0 >$, $< 1, 1, 1 >$, $< 0, 1, 2 >$ and $< 1, 0, 2>$. Whereas, there are no such cases as $< g_f, g_m, g_c > = < 2, 0, 1 >$, $< 2, 1, 0 >$, $< 0, 2, 1 >$, $< 1, 2, 0 >$. Therefore, we can predict the pattern of the parents-offspring trio through examples out of all 27 ($\{0,1,2\}^3$) genotype trio combinations. We can observe only 15 possible cases as shown in Table 1.

< Table 1. All possible genotype trio >

g_f	g_m	g_c
0	0	0
1	1	1
0	1	2
1	0	2
0	2	0
0	2	2
2	0	0
2	0	2
1	2	1
1	2	2
2	1	1
2	1	2
2	2	0
2	2	1
2	2	2

3.1.2 Resolving Haplotypes

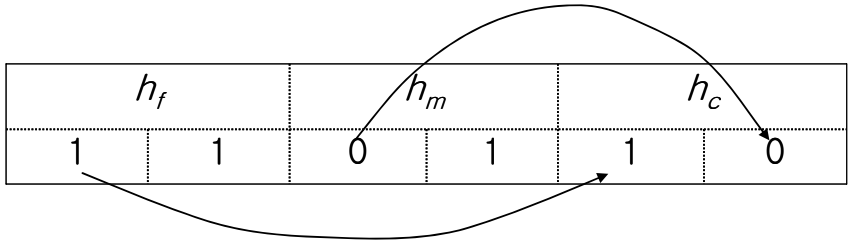
Theorem 2) At any node's genotype, g = 0 is always resolved to $< h^1, h^2 >= < 0, 0 >$, and g = 1 to $< h^1, h^2 > = < 1, 1 >$.

Theorem 3) There exists a unique haplotypes solution at any locus on *n*-marker loci in a trio, unless any locus has $< g_f, g_m, g_c > = < 2, 2, 2 >$ type trio.

In Figure 6, we rearrange haplotypes' order with the following rules:

Rule 1) the left most allele of paternal and maternal node will be inherited by the child node.

Rule 2) the left most allele of h_c is from paternal site, and the right most allele of h_c is from maternal site.



<Figure 6> illustrates the inheritance by haplotype re-arrangement with Rules: *1* which is the left most allele of paternal node, h_f, is inherited by the left-most position of child node, h_c. And *0* which is the left-most allele of maternal node, h_m, is to the most right site of h_c. By this arranged ordering, we can resolve haplotypes for all nodes.

In Figure 6, we illustrate how we can use Rules 1 and 2 to resolve haplotypes from given genotypes. Let us consider an example where the genotype trio is $< g_f, g_m, g_c > = < 1, 2, 2 >$. We

already know $< h_f^1, h_f^2 > = < 1, 1 >$. Thus, we can set $h_c^1=1$ by the Rule 2, and following $h_c^2 = 0$, that is $< h_c^1, h_c^2 >$ should be $< 1, 0 >$ out of two cases $< h_c^1, h_c^2 >$ either $< 0, 1 >$ or $< 1, 0 >$. So, $< h_m^1, h_m^2 >$ can be resolved to $< 0, 1 >$ by Rule 1 also out of two cases $< h_m^1, h_m^2 >$ either $< 1, 0 >$ or $< 0, 1 >$. With this pre-arranged information, we can make a table which shows the haplotypes' order in a trio.

g_f	g_m	g_c
0	0	0
1	1	1
0	1	2
1	0	2
0	2	0
0	2	2
2	0	0
2	0	2
1	2	1
1	2	2
2	1	1
2	1	2
2	2	0
2	2	1
2	2	2

All possible genotype trios

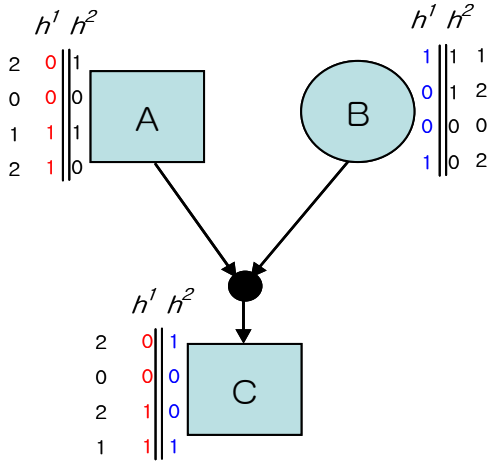
h_f		h_m		h_c	
0	0	0	0	0	0
1	1	1	1	1	1
0	0	1	1	0	1
1	1	0	0	1	0
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	0
1	0	0	0	1	0
1	1	1	0	1	1
1	1	0	1	0	1
1	0	1	1	1	1
0	1	1	1	0	1
0	1	0	1	0	0
1	0	1	0	1	1
0	1	1	0	0	1
1	0	0	1	1	0

Resolved Table

< Figure 7> shows a resolved table that has the rearranged haplotypes by Rules 1 and 2. However, the $< g_f, g_m, g_c > = < 2, 2, 2 >$ case has two possible results. To resolve this, we need more information and we will address this problem at the end of this section 3.1.

While these rules are applicable to all possible genotype trios, we can obtain the resolved haplotype table, called the resolved table, as shown in Figure 7. In Figure 7, the Rules 1 and 2 can be applied to find a unique haplotypes' combination. Unfortunately, $< g_f, g_m, g_c > = < 2, 2, 2 >$ case in trio cannot settle into one solution, ($< h_f^1, h_f^2 >$, $< h_m^1, h_m^2 >$, $< h_c^1, h_c^2 >$) can be either (< 0, 1 >, < 1, 0 >, < 0, 1 >) or (< 1, 0 >, <0, 1 >, < 1, 0 >). This problem will be treated in the next step.

These rules are even properly applied to *n*-marker genotypes in a trio. In other words, by rearranging the haplotypes' order, we can resolve haplotypes no matter the genotype marker's length.



A	B	C
g_f	g_m	g_c
2	1	2
0	2	0
1	0	2
2	2	1

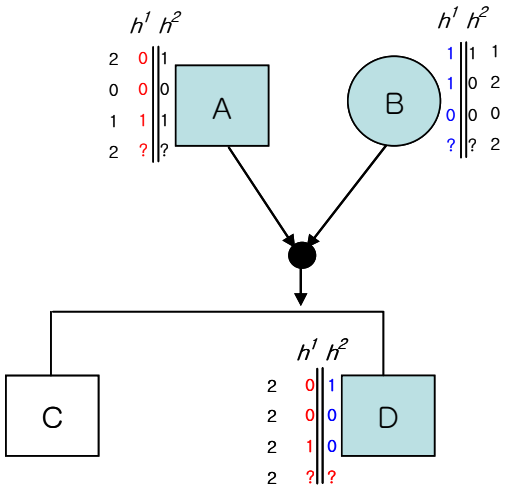
A		B		C	
h_f		h_m		h_c	
0	1	1	1	0	1
0	0	0	1	0	0
1	1	0	0	1	0
1	0	1	0	1	1

< Figure 8> shows that the resolved table is even properly working on *n*–marker genotypes with the Rules. In this example, $< g_f, g_m, g_c > = < 2012, 1202, 2021 >$ is resolved into ($< h_f^1, h_f^2 >$, $< h_m^1, h_m^2 >$, $< h_c^1, h_c^2 >$) = (<0011, 1010>, <1001, 1100>, <0011, 1001>).

Figure 8 illustrates an example input and the expected output. Consider a trio < g_f, g_m, g_c > = < '2012', '1202', '2021' >. By resolving the table by the Rules, we can determine a unique combination of haplotypes order like (< h_f^1, h_f^2> , < h_m^1, h_m^2> , < h_c^1, h_c^2>) = (< 0011, 1010 >, < 1001, 1100 >, < 0011, 1001 >).

Theorem 4) A trio, which has resolution ambiguity, can be treated by its following generation's trio or a precedent generation's trio.

Let's assume the mating between g_f and g_m in Figure 9 has another child D, genotype '2222',



A	B	D
g_f	g_m	g_c
2	1	2
0	2	2
1	0	2
2	2	2

A		B		D	
h_f		h_m		h_c	
0	1	1	1	0	1
0	0	1	0	0	1
1	1	0	0	1	0
?	?	?	?	?	?

< Figure 9 > illustrates an example with an ambiguous site. If there is no more information, < g_f, g_m, g_c > = < 2, 2, 2 > cannot be resolved into a unique solution.

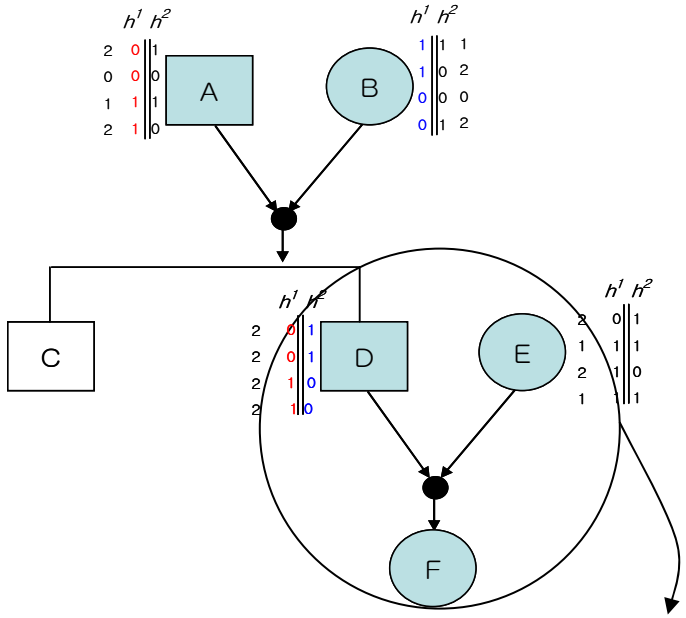
which means we can observe < g_f, g_m, g_c > = < ‘2012’, ‘1202’, ‘2222’ > trio. In this case, although they are rearranged by the resolved table, the rearranged trio still has ambiguity on the forth locus because the resolved table is sensitive to its order. In other words, both combination (< h_f^1, h_f^2> , < h_m^1, h_m^2> , < h_c^1, h_c^2>) = (< 0011, 1010 >, < 1100, 1001 >, < 0011, 1100>) or (< 0010, 1011 >, < 1101, 1000 >, < 0010, 1101>) might be possible. We cannot determine one out of two as a unique solution.

However, if g_c married and had a child like in Figure 10, we can be given another trio in which g_c becomes $g_f{}'$. Figure 10 illustrates their map. Through this following descendant trio, h_c can be derived and we will resolve a unique solution out of the above two possible combinations. i.e. < $g_f{}'$, $g_m{}'$, $g_c{}'$ > has been resolved < $h_f{}'$, $h_m{}'$, $h_c{}'$ > = (< 0011, 1100 >, < 0111, 1101 >, < 0011, 0111 >).

Hence $h_f{}'$= h_c = < 0011, 1100> should be obtained, and then a unique solution for < g_f, g_m, g_c > will be derived as (< h_f^1, h_f^2> , < h_m^1, h_m^2> , < h_c^1, h_c^2>) = (< 0011, 1010 >, < 1100, 1001 >, < 0011, 1100>). We can extend this idea to every related node of a pedigree into one table, and then solve for a unique solution. From this point on, the combined genotype table with pedigree information as in Figure 10 will be called the resolving table.

3.1.3 Define Pre-Resolved Table and Haplotype phasing

Rule 3) The resolved haplotypes have four order types < 0, 0 >, < 1, 1 >, < 0, 1 > and < 1, 0>. Here after, < 0, 0 > is denoted by 0, < 1, 1 > is 1, < 0, 1 > is 2, and < 1, 0 > is 3 as resolved haplotype’s order. Also it will be indicated r_f, r_m, and r_c instead of h_f, h_m, and h_c.



D	E	F
$g_c = g_f'$	g_m'	g_c'
2	2	0
2	1	2
2	2	1
2	1	1

D		E		F	
h_f'		h_m'		h_c'	
0	1	0	1	0	0
0	1	1	1	0	1
1	0	1	0	1	1
1	0	1	1	1	1

A	B	D	E	F
g	g	g	g	g
2	1	2	2	0
0	2	2	1	2
1	0	2	2	1
2	2	2	1	1

Resolving table

A		B		D		E		F	
h		h		h		h		h	
0	1	1	1	0	1	0	1	0	0
0	0	1	0	0	1	1	1	0	1
1	1	0	0	1	0	1	0	1	1
1	0	0	1	1	0	1	1	1	1

Resolved table

< Figure 10 > represents two generations's trios. If the resolved result of one generation still has ambiguous sites, then another result of following generation trio will help it to be resolved. < $g_c = g_f'$, g_m', g_c' > is resolved < h_f', h_m', h_c' > = (< 0011, 1100 >, < 0111, 1101 >, < 0011, 0111 >). From this result, we can resolve < g_f, g_m, g_c > which still has ambiguous sites as < h_f, h_m, h_c > = (< 0011, 1010 >, < 1100, 1001 >, < 0011, 1100>). For this, we can build the combined table which has the information of all generations in a table. Here after, we call it the resolving table.

Rule 4) There might be a site which we cannot resolve because of its ambiguity. Here after, character '**?**' denotes this ambiguous site which can be either r = 2 or 3.

With Rules 3 and 4, we can establish the Pre-Resolved Table as shown in Figure 11.

g_f	g_m	g_c	h_f		h_m		h_c	
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
0	1	2	0	0	1	1	0	1
1	0	2	1	1	0	0	1	0
0	2	0	0	0	0	1	0	0
0	2	2	0	0	1	0	1	0
2	0	0	0	1	0	0	0	0
2	0	2	1	0	0	0	1	0
1	2	1	1	1	1	0	1	1
1	2	2	1	1	0	1	0	1
2	1	1	1	0	1	1	1	1
2	1	2	0	1	1	1	0	1
2	2	0	0	1	0	1	0	0
2	2	1	1	0	1	0	1	1
2	2	2	0	1	1	0	0	1
			1	0	0	1	1	0

r_f	r_m	r_c
0	0	0
1	1	1
0	1	2
1	0	3
0	2	0
0	3	2
2	0	0
3	0	3
1	3	1
1	2	3
3	1	1
2	1	2
2	2	0
3	3	1
? (2 / 3)	? (3 / 2)	? (2 / 3)

The Pre-Resolved Table

<Figure 11> The Pre-Resolved Table (for the g-g-g case)

g-g-g case means the case that every node in trio is a genotype. We can transform genotypes in trio information to resolved types, r. For example, when $< g_f, g_m, g_c > = < 2, 2, 1>$, then $< h_f, h_m, h_c > = (<1,0>, <1,0>, <1,1>)$. Therefore, $< r_f, r_m, r_c >$ will be derived as $< 3, 3, 1>$. In another example, $< g_f, g_m, g_c > = < 2, 2, 2>$ can be either $< r_f, r_m, r_c > = < 2, 3, 2>$ or $< 3, 2, 3 >$. So, it is denoted as < ?, ?, ? >.

For example, when $< g_f, g_m, g_c > = < 1, 2, 2 >$, we can get $< h_f, h_m, h_c > = (< 1, 1 >, < 0, 1 >, < 1, 0 >)$ with the resolved table. Following Rule 3, $< r_f, r_m, r_c >$ will be represented by $< 1, 2, 3 >$. In another example, while $< g_f, g_m, g_c >$ is $< 2, 2, 1 >$, the trio $< r_f, r_m, r_c >$ will be resolved by $< 3, 3, 1 >$. However, when $< g_f, g_m, g_c >$ is $< 2, 2, 2 >$, $< r_f, r_m, r_c >$ can be either $< 2, 3, 2 >$ or $< 3, 2, 3 >$. As we have seen in similar cases before, we need more information. So, we denote it as $< r_f, r_m, r_c > = < ?, ?, ? >$ until more information is obtained to resolve this. Hence, once three nodes in a trio have passed through the Pre-Resolved Table, that trio's r_f, r_m, and r_c represent the order of resolved haplotypes.

Theorem 5) If more than one of the three nodes in a trio was resolved without any ambiguity, the others can also be resolved properly although they still have a type '2' ambiguious unphased genotype.

For instance, if $< g_f, g_m, r_c > = < 0, 2, 2 >$, then $< r_f, r_m, r_c > = < 0, 3, 2 >$. This even works properly on the case of $< g_f, g_m, r_c > = < 2, 2, 3 >$ or $< g_f, g_m, r_c > = < 2, 2, 2 >$ as proved on Figure 10. $< g_f, g_m, r_c > = < 2, 2, 3 >$ will be $< r_f, r_m, r_c > = < 3, 2, 3 >$ and $< g_f, g_m, r_c > = < 2, 2, 2 >$ be $< r_f, r_m, r_c > = < 2, 3, 2 >$. With this Theorem 5, the Pre-Resolved sub Tables are deduced such as Figures 12 to 18.

Figures 12 to 18 show the case that one or more out of 3 nodes in the trio were resolved already. While the shaded columns represent already resolved values, the remaining unresolved columns will be determined.

g_f	g_m	r_c
0	2	0
2	0	0
2	2	0
1	2	1
2	1	1
2	2	1
0	2	2
2	1	2
2	2	2
2	0	3
1	2	3
2	2	3

r_f	r_m
0	2
2	0
2	2
1	3
3	1
3	3
0	3
2	1
2	3
3	0
1	2
3	2

< Figure 12> Sub-Table 1 of Pre-Resolved Table (for the g-g-r case)

The case of child node, which is the shaded column in the left table, is resolved.

g_f	r_m	g_c
1	0	2
2	0	0
2	0	2
0	1	2
2	1	1
2	1	2
1	2	2
2	2	0
2	2	2
0	3	2
2	3	1
2	3	2

r_f	r_c
1	3
2	0
3	3
0	3
3	1
2	2
1	3
2	0
3	3
0	2
3	1
2	2

< Figure 13> Sub-Table 2 of Pre-Resolved Table (for the g-r-g case)

The case where one of the parents is resolved.

r_f	g_m	g_c
0	1	2
0	2	0
0	2	2
1	0	2
1	2	1
1	2	2
2	1	2
2	2	0
2	2	2
3	0	2
3	2	1
3	2	2

r_m	r_c
1	2
2	0
3	2
0	3
3	1
2	3
1	2
2	0
3	2
0	3
2	1
2	3

< Figure 14> Sub-Table 3 of Pre-Resolved Table (for the r-g-g case)

The case where one of the parents node is resolved.

g_f	r_m	r_c
2	0	0
2	0	3
2	1	1
2	1	2
2	2	0
2	2	3
2	3	0
2	3	2

r_f
2
3
3
2
2
3
3
2

< Figure 15> Sub-Table 4 of Pre-Resolved Table (for the g-r-r case)

The case where the child and one of the parents is resolved.

r_f	g_m	r_c
0	2	0
0	2	2
1	2	1
1	2	3
2	2	0
2	2	2
3	2	1
3	2	3

r_m
2
3
3
2
2
3
3
2

< Figure 16> Sub-Table 5 of Pre-Resolved Table (for the r-g-r case)

The case where the child and one of the parents is resolved.

g_f	r_m	r_c
2	0	0
2	0	3
2	1	1
2	1	2
2	2	0
2	2	3
2	3	1
2	3	2

r_f
2
3
3
2
2
3
3
2

< Figure 17> Sub-Table 6 of Pre-Resolved Table (for the g-r-r case)

The case where the child and one of the parents is resolved.

r_f	r_m	g_c	r_c
0	1	2	2
1	0	2	3
0	3	2	2
3	0	2	3
1	2	2	3
2	1	2	2
2	3	2	2
3	2	2	3

< Figure 18> Sub-Table 7 of Pre-Resolved Table (for the r-r-g case)

The case where both parents are resolved.

3.2 Pedigree Graph

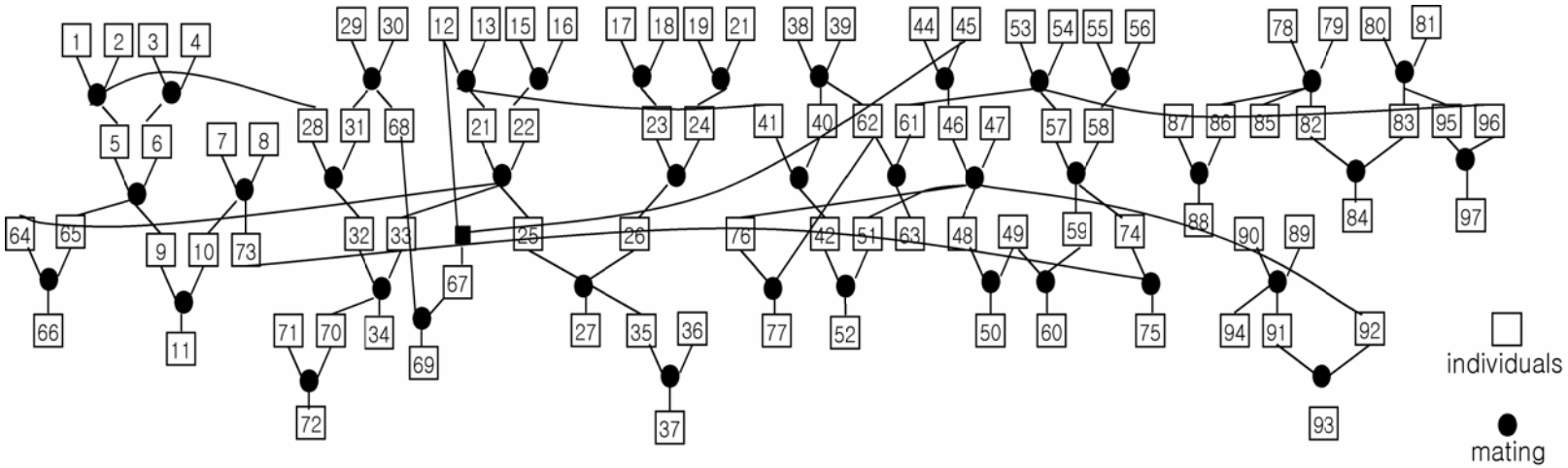
3.2.1 Tree Structure in Pedigree

As viewed in Chapter I, a pedigree can be drawn as a graph of family relationships. Although it appears to be a very complex network graph as shown in Figure 19, it is actually composed of several tree structures, especially binary tree structures.

Theorem 6) The pedigree can be separately drawn as several binary tree structures.

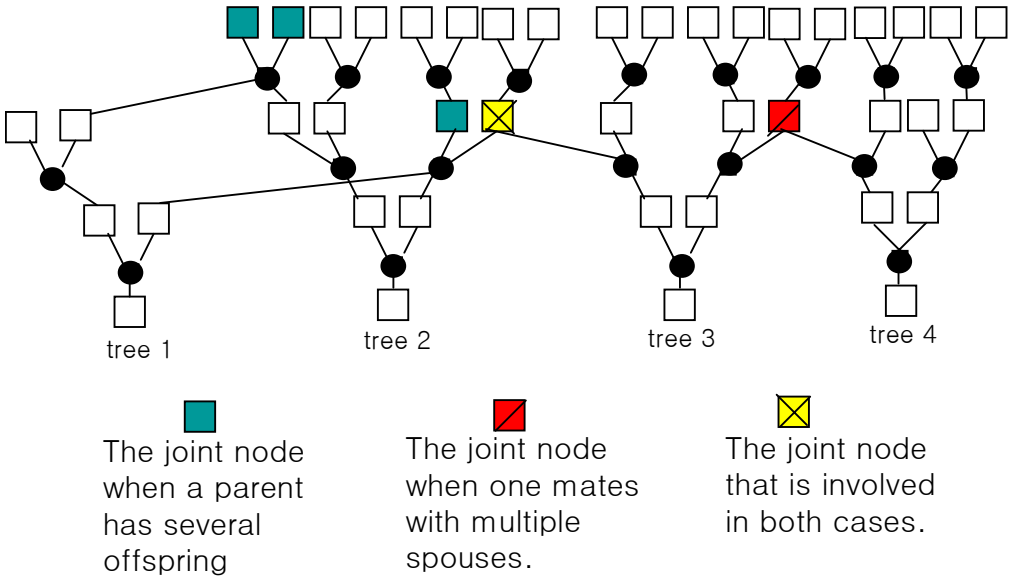
When we observe it from the standpoint of a descendant to ancestor, Theorem 6 is obvious because one can have only one father and mother pair biologically. Therefore we can divide a pedigree graph into several tree structures as shown in Figure 20. Each of the tree structures has a root descendant which does not have any more offspring. Therefore, the number of trees in a pedigree is equal to the number of root descendants. At this juncture, the earliest ancestor, which does not go up any further, is regarded as a leaf-node.

The reason we divide a pedigree into several trees is that only the nodes in the same tree have a genetic relationship. When we build a resolving table to solve haplotype phasing problems, the nodes across different trees have no shared information. Thus, a pedigree is divided into several separated tree structures and its shared information will be exploited in resolving tables to resolve its haplotypes.



<Figure 19> Schematic pictorial of pedigree

This schematic illustrates the relationship among nodes. A parents can have more than one offspring and an individual can mate with mutiple spouses. For instance, mating between node 1 and 2 produces two offspring, node 5 and 28. In another example, node 62 mates twice with different spouses, node 61 and 76.



<Figure 20> shows the separated tree structures from a pedigree. They are connected by the joint nodes.

3.2.2. Joint Node and Building Tree Structures

While the tree structures are built, there are nodes that connect multiple trees as shown in Figure 20. These are called Joint Nodes, and they occur under two circumstances. The first is when the same parents have more than one offspring. In this case each father and mother node becomes a joint node. Second, if an individual mates with multiple spouses and a different offspring comes from each mating, this individual becomes a joint node.

Theorem 7) Several joint nodes can be placed in a single tree and a joint node may appear in

its involved tree structures several times. However, the information of a joint node's ancestor can be used only once in a tree structure.

In other words, a joint node will appear in the different tree structures in which its different offspring are placed. Since a joint node is created because of different offspring rather than different ancestors, the ancestor's biological information of a joint node is always identical even though a joint node is involved in different tree structures. Therefore, its ancestral information can be used only once in one of the multiple trees connected by this joint node, reducing the computation of resolving haplotypes.

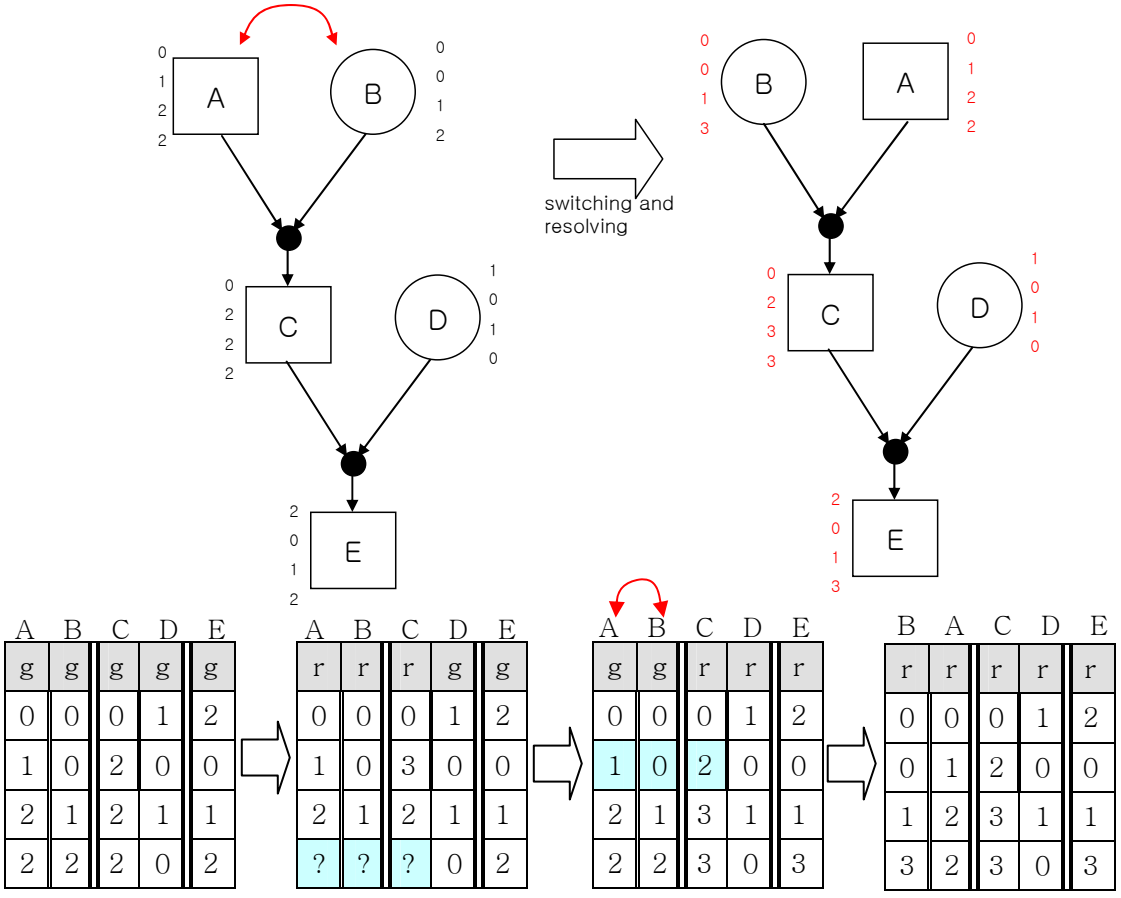
3.2.3. Possibility of Switching Position

Up to now, we have considered only $< g_f, g_m >$ order for mating. However, in some cases the mating order of $< g_f, g_m >$ cannot match to the resolved child node r_c when r_c is resolved from another trio (see Figure 21). In this case, we must consider switching g_f and g_m.

Theorem 8) Switching Scheme 1 (SS1): Once a child node is resolved, we can no longer say that the paternal node must be placed on the left side of mating.

In Figure 21, because the first trio (A-B-C) has an ambiguous locus at its fourth site; it cannot be resolved completely. To resolve the first trio, we first resolve the second trio and then we use

the Pre-Resolved sub-Table 1 for the g-g-r case as in Figure 12. However, we cannot determine the case of $< g_f, g_m, r_c > = < 1, 0, 2 >$ in this table because the most left allele of r_c must be placed to the left side of its parental node. To address this issue, we switch g_f and g_m such that $< g_m, g_f, r_c > = < 0, 1, 2>$, which leads to the properly resolved result as $< r_m, r_f, r_c > = <$ 0013, 0122, 0233 $>$



A	B	C	D	E
g	g	g	g	g
0	0	0	1	2
1	0	2	0	0
2	1	2	1	1
2	2	2	0	2

A	B	C	D	E
r	r	r	g	g
0	0	0	1	2
1	0	3	0	0
2	1	2	1	1
?	?	?	0	2

A	B	C	D	E
g	g	r	r	r
0	0	0	1	2
1	0	2	0	0
2	1	3	1	1
2	2	3	0	3

B	A	C	D	E
r	r	r	r	r
0	0	0	1	2
0	1	2	0	0
1	2	3	1	1
3	2	3	0	3

< Figure 21 > shows the switching case of the parental node. The left-most side table is the original resolving table. While the first trio (A-B-C) is resolved, there is an ambiguous site at the fourth locus as seen in the second table. So, the second trio (C-D-E) should be resolved first like in the third table. However, back on the first trio, we see an impossible case on its second locus, <g, g, r> = <1, 0, 2>. Thus, node A and B need to be switched, so that every node can be resolved properly like the right-most side table.

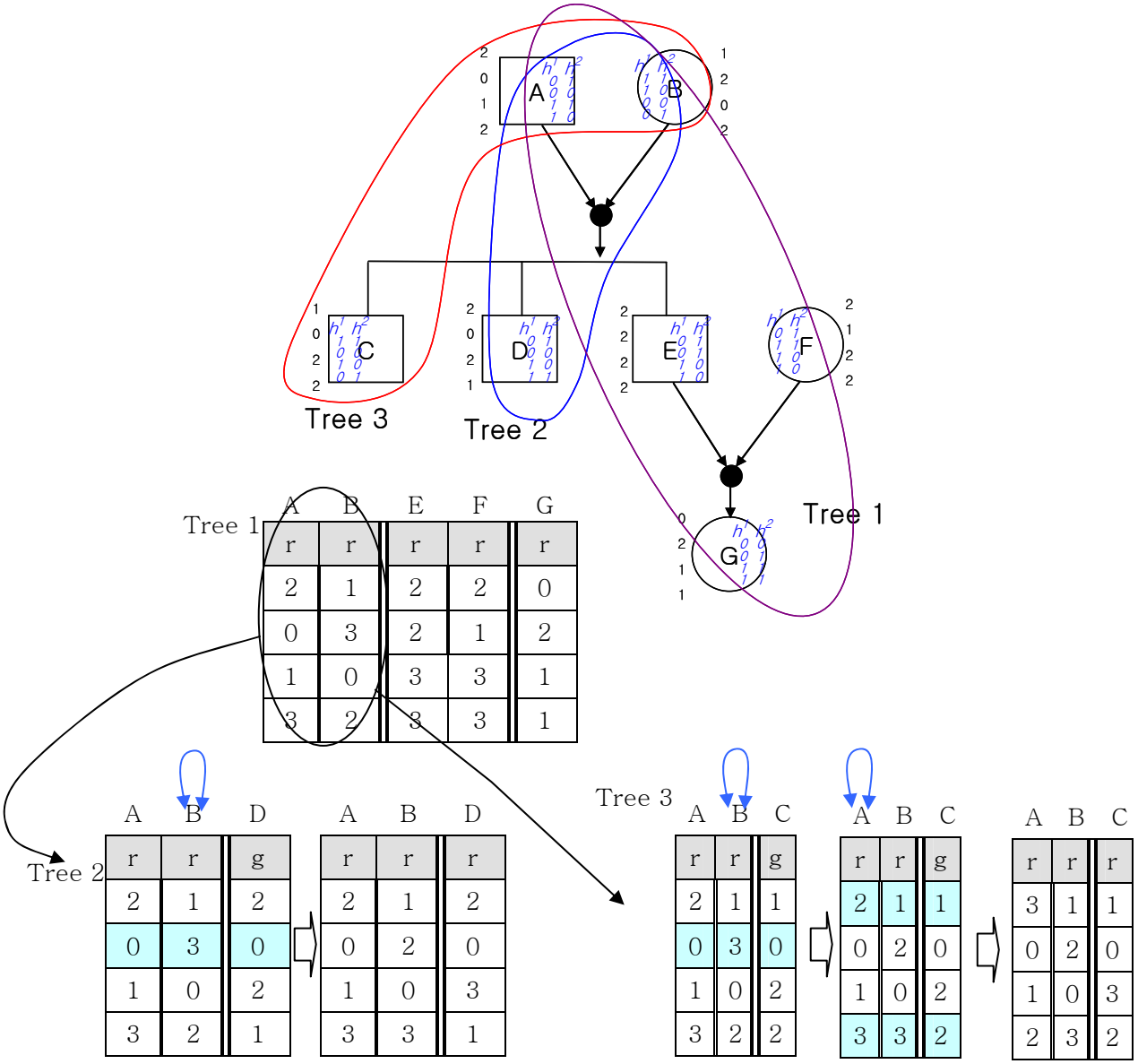
Theorem 9) Switching Scheme 2 (SS2) : Two haplotypes' position in a node can be switched, if it is a joint node.

Switching two haplotypes positions in a gene does not alter its original pattern. For instance, consider the r = < 3313 > case, where r is the resolved pattern of genotypes by Rule 3 in section 3.1.3. The combination of haplotypes of r is < h_1, h_2 > = < 1111, 0010 >. When we switch h_1 and h_2, < h_1, h_2 > = < h_2, h_1 > = < 0010, 1111 >, we do not effect the resolved genotype pattern, and r can be expressed as < 2212> instead of < 3313 >. In particular, the case of loci *r = 0* or *1* is independent of switching. Only *r = 2* or *3* must be changed into *r = 3* or *2*.

Based on this fact, when we resolve a joint node, we check whether its haplotype position needs to be switched or not. Let us consider when a joint node is resolved in one tree and we learn its resolved haplotypes. Then, when we employ this resolved pattern in another tree, the employed types and its offspring may not generate a trio pattern in the Pre-Resolved Table. For example, in Figure 22, the pedigree is constructed by three tree structures because it has three root descendants, C, D and G. From Tree 1, joint nodes A and B are resolved. Then their patterns are employed by Tree 2 and 3. However, the pattern of node B, r_m = < 1302 > cannot be directly used to resolve the trio (A-B-D) in Tree 2 because one of the loci yields orders (< r_f, r_m, g_c > = < 0, 3, 0>) which does not exist (See the Pre-Resolved sub-Table 2 for g-r-g case in Figure 13). It is caused by the fact that a different haplotype in B was inherited by its offspring D and E. Therefore, node B's haplotype pattern, r_m = < 1302 >, should be permutated in Tree 2. Using SS2 (Theorem 9) we can change r_m from < 1302 > to < 1203 >. Then we can obtain the resolved result for D, r_c = < 2031 >.

In the Tree 3 case, since B is a joint node, we switch the pattern of B to < 1203 > by using SS2.

However, the trio A-B-C still has some problems, < r_f, r_m, g_c > = < 2, 1, 1> and < 3, 3, 2>. In this case, A, r_f must switch its pattern, A = r_f = < 2013 > into < 3012 > (SS2). After the haplotype pattern of both A and B are changed, we can resolve for C = < 1032 >.



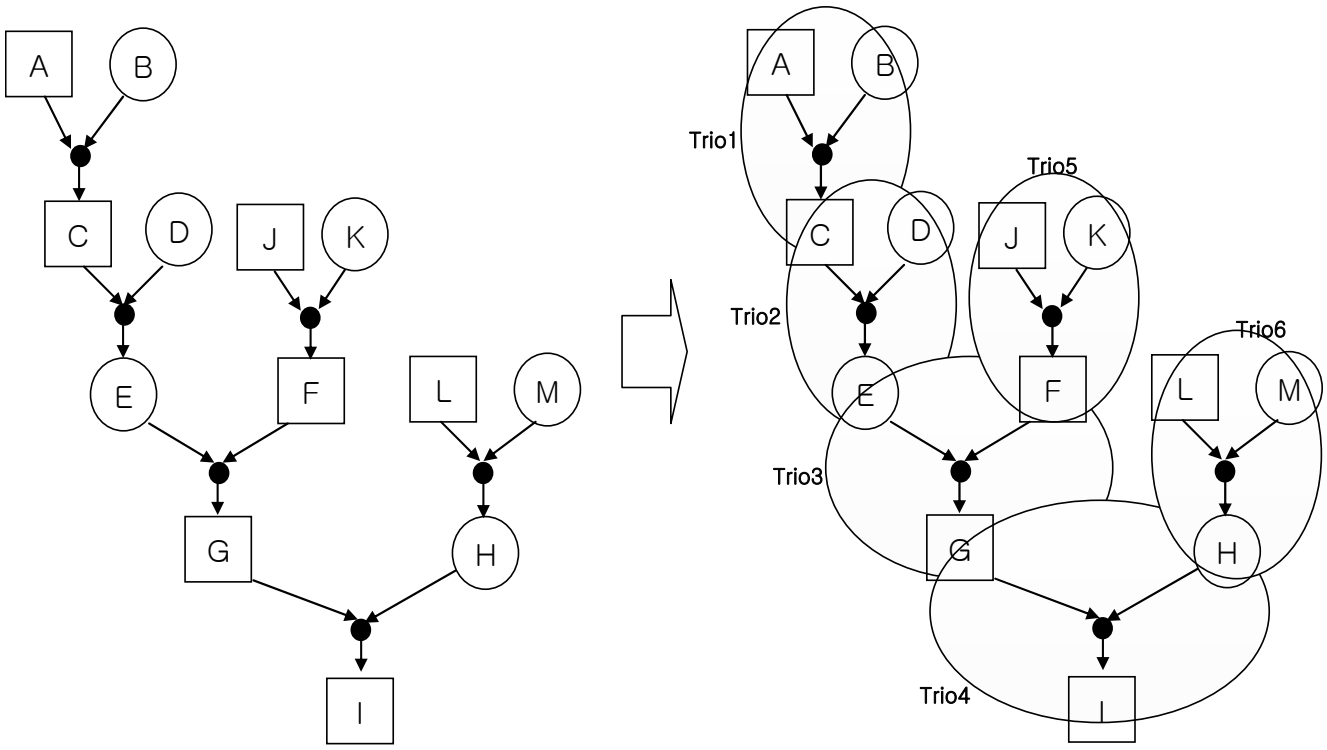
< Figure 22 > illustrates the switching case on a joint node. After Tree 1 resolves node A and B, then Tree 2 and 3 adopt their resolved patterns. However, they cannot be used directly. As shown in Figure 22, for Tree 2, node B has switched its pattern. For Tree 3, node A and B have switched both haplotype patterns. After all these changes, we can correctly resolve all genotypes in pedigree. (The shaded rows indicate where errors occur.)

3.3. Algorithm Construction

3.3.1. Tree Search List for Resolving Tables

Every node in a tree has a trio-relationship with its parental nodes or offspring node. Hence, when we construct resolving tables, we have to consider a trio as a single entity. This entity is regarded as a single node in a new tree, which is virtually constructed for building resolving tables. In Figure 23, the real tree has 13 nodes, but it is composed of 6 entities (i.e. A-B-C or C-D-E is regarded as one entity). So, in the virtual tree there are 6 corresponding nodes, and we search the virtual tree to build the list for the resolving table. In general, the virtual tree consists of $(t-1) / 2$ nodes at most, where t is the number of total nodes in the real tree.

Each virtual tree is scanned by the DFS (*Depth First Search*) algorithm in order to build the resolving table lists. When a virtual tree is scanned, the DFS algorithm generates a searching path. Then we segment the path by leaf node (the earliest ancestor node in this case). Each segmented path becomes a resolving table for the Pre-Resolved Table algorithm. Among the resolving table lists, the longest resolving table with the furthest ancestor from the root descendant (e.g. A, B in Figure 23) becomes the starting table used to resolve haplotypes. The longest resolving table is more efficient in general because it has more information than the shorter ones. For example in Figure 23, the segmented search paths are {trio1, trio2, trio3, trio4}, {trio5} and {trio6}. So, the table list has three elements of the resolving table. Therefore, each resolving table will be {A,B,C,D,E,F,G,H,I}, {J,K,F}, and {L,M,H}.



The real tree in a pedigree with 13 nodes

The virtual tree with 6 trio nodes

< Figure 23 > illustrates the pictorial representation of trios that are considered to be one entity in the virtual tree because each trio has one relationship in the left figure.

3.3.2. Construction of the Algorithm

At the initializing step, we always settle some trios such as $< g_f, g_m, g_c > = < 0, 0, 0 >, < 1, 1, 1 >, < 0, 2, 0 >, < 2, 0, 0 >, < 1, 2, 1 >, < 2, 1, 1 >, < 2, 2, 0 >$ and $< 2, 2, 1 >$ into $< r_f, r_m, r_c > = < 0, 0, 0 >, < 1, 1, 1 >, < 0, 2, 0 >, < 2, 0, 0 >, < 1, 3, 1 >, < 3, 1, 1 >, < 2, 2, 0 >$ and $< 3, 3, 1 >$.

At the resolving step, we determine whether each node is a joint node or not, and whether it

has already been resolved in previously resolved tables. If a node is a joint node or already resolved, its information is copied into the current resolving table. In this case, we must determine whether to use SS1 or SS2 (Theorem 8 and 9) to switch the joint node for each node that contains it. Then, the resolving table will be resolved through the Pre-Resolved Table.

```
for d = 1 : number( trees in input data )    %    same as number of the root descendant
    % build resolving tables in a tree by DFS algorithm
    resolving_talbles = build_resolving_table(tree(d));

    for i = 1 : number ( resolving_tables )
        init_table = initialize_table ( resolving_table(i) );

        % resolving step
        for j = number (trio in the resolving_table) :
            if is_joint_node(any node out of three trio node)
                check_switching_haplotypes_position (trio(j));
            end

            if is_already_resolved_in_other_table(any node out of three trio node)
                apply_to_table (i);
            end

            check_switching_parental_node_position (trio(j));
            if ( is_no_information_about_switing_node_position )
               && ( is_root_descendant (child) )
                set (no_switing);
            end

            for m = 1 : length ( markers )    % same as the length of a genotype
```

```
                    The_Pre_Resolved_Table (parent1, parent2, child);
                end
            end
        end
end
```

< Figure 24. the Pre-Resolved Table algorithm code >

The iteration i and j together will be the total sample genotype size. Hence, we see O(*nmd*) time complexity which is linear, where *n* is the size of pedigree (number of genotypes), *m* is the number of loci, and *d* is number of tree (number of root descendants).

IV. Experimental Results

For generating sample data, we use the MS (HUDSON) program which is a genetic sample generator under the neutral model. The sample data has 200 individuals' haplotypes without mutation and recombination.

4.1 Haplotype phasing using the EM algorithm

The 200 individuals' genotypes are randomly generated from the haplotypes which the MS (HUDSON) program generated. Input data has only the genotypes. After haplotype phasing using the EM algorithm, phased pairs are compared with original haplotype pairs.

```
most maximized haplotpye of each genotype : sample 200 individual

000000000, 100000011 -> genotype 200000022 -> phased 100000011, 000000000
100000001, 101000001 -> genotype 102000001 -> phased 101000001, 100000001
100000001, 000110000 -> genotype 200220002 -> phased 100110001, 000000000
100000001, 000000000 -> genotype 200000002 -> phased 100000001, 000000000
100000101, 000000000 -> genotype 200000202 -> phased 100000101, 000000000

.......
```

< Figure 25. Output of EM >

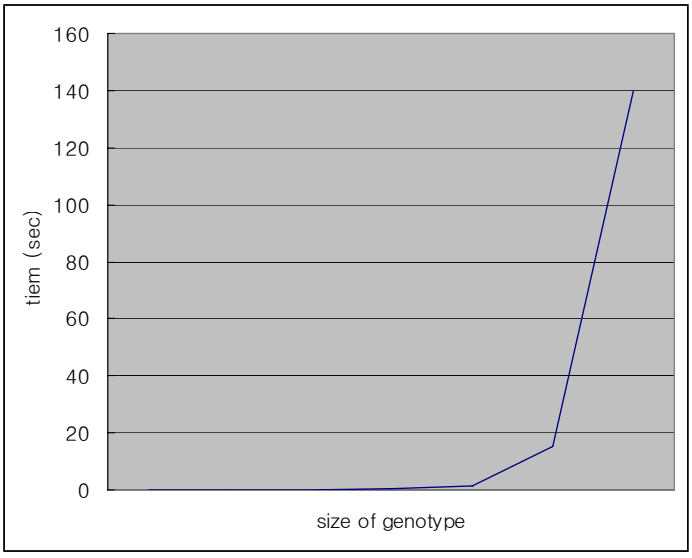
The left-most part is a generated haplotype by the MS-hudson program. The genotype is randomly generated based on haplotypes. The right part is phased haplotype pairs. The resolution accuracy is perfect.

We obtained the haplotype phasing results as described in Figure 25. They are 100% perfectly phased compared with the original pair. When we run this algorithm with various genotype lengths, it shows that the running time is sensitive to data marker's length. Table 2 shows the average time versus the genotype length when run 50 times.

<Table 2> Average running time variation.

With a different genotype length, computation time(second) is increased exponentially.

length	3	4	5	6	7	8	9
time(s)	0.083	0.095	0.197	0.359	1.618	15.277	139.743



< Figure 26 . Graph for average running time variation of EM>

The EM algorithm runs with exponential increase by size of genotypes in the sample.

4.2 Haplotype phasing using the Pre-Resolved Table on Ancestral Tree

A sample for the Pre-Resolved Table algorithm needs to have the pedigree information, although the EM phasing algorithm takes just genotype data. Hence, we built two pedigrees for this empirical study: one contains 100 individuals and the other has 200 individuals. Thus, the input data for the Pre-Resolved Table algorithm contains four entries such as node_id, parent 1, parents 2 and genotype of the node. The combination of node_id, parent 1 and parent 2 indicates the parents-child trio. The earliest ancestor node only has '0, 0' for parent 1 and parent 2. For example, if input data shows '*20, 0, 0, 1002002*', this node's id is *20* and this node is the earliest ancestor with genotype *1002002*. In the other example, if input data is '*100, 43, 80, 2110002*', this node's id is *100* which has *43* and *80* as its parent nodes, and the genotype is *2110002*. The earliest ancestors' genotypes are randomly generated from the haplotypes which the MS (HUDSON) program generated. The rest of the genotypes are generated based on the pedigree information. After haplotype phasing using Pre-Resolved Table algorithm, phased pairs will be compared with original haplotype pairs.

We obtained the haplotype phasing results as described in Figure 27. It is 100% perfectly phased compared with the original pair. When we run this algorithm with various genotype lengths (3 to 100) in the same pedigree information, we observe that the computation time does not exponentially increase with the genotype lengths (Table 3) but increases in a linear fashion. Even with the genotype length of 100, it took less than 6 seconds with 200 individuals' data. Table 3 shows the average running time versus the genotype length when run 50 times.

```
smple size : 200
site length : 8
resolved accuracy : 100.00 % correct phased
21 org 00000011 11100000 -> geno 22200022 -> ph 00000011 11100000
22 org 11100000 00000111 -> geno 22200222 -> ph 11100000 00000111
23 org 11100000 00000111 -> geno 22200222 -> ph 11100000 00000111
24 org 11100000 00000111 -> geno 22200222 -> ph 11100000 00000111
.......
```

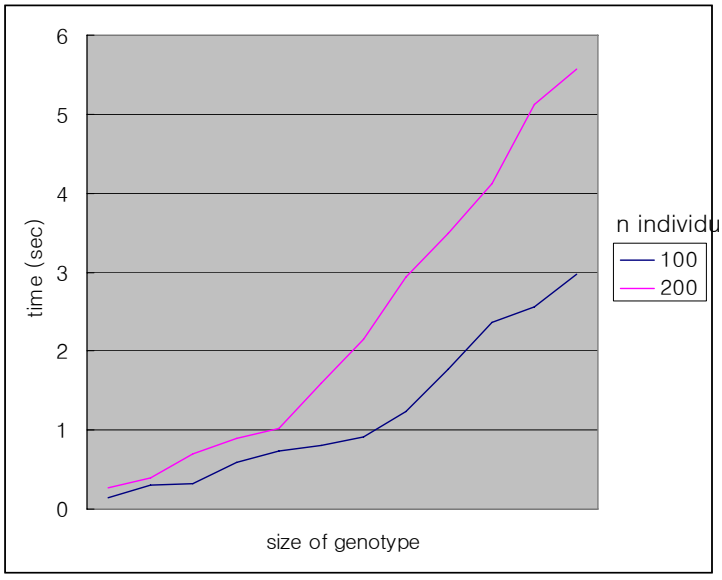
< Figure 27. Output of Pre-Resolved Table Algorithm >

The ouput sequence is node_id, original haplotype, genotype, phased haplotype. The original haplotypes are randomly generated with data from MS-hudson program.

< Table 3 > Average running time variation.

According to 100 and 200 individuals samples, computation time(second) increases in a linear fashion.

individuals	length	3	5	7	11	15	20	30	100
100	time(s)	0.137	0.312	0.327	0.599	0.735	0.798	0.921	2.976
200		0.274	0.391	0.699	0.891	1.012	1.592	2.145	5.876



< Figure 28 . Graph for average running time variation >

The computation time shows linear time variation instead of exponential one.

< Table 4 > Average resolution accuracy

The overall average resolution accuracy is 99.475% when PRT phasing was run 100 times : 50 times for the pedigree which has 100 individuals and 50 times for the pedigree which has 200 individuals. The earliest ancestors' genotypes are randomly generated from the haplotypes which MS (HUDSON) program generated. The rest of the genotypes are generated based on the pedigree information. (Note : for each run the sample data was randomly generated.)

length	3	5	7	11	15	20	30	100
Accuracy(%)	100	99.335	99.774	98.401	99.484	99.426	99.728	99.653

In Figure 27, we obtained 100% accuracy when there was no unresolved or incorrect haplotype pair. However, we found that accuracy slightly decreased with the presence of unresolved haplotype pairs (see Table 4). The haplotypes can not be resolved exactly in the Pre-Resolved Table algorithm when all of following conditions are met :

1) the node is the earliest ancestor node.

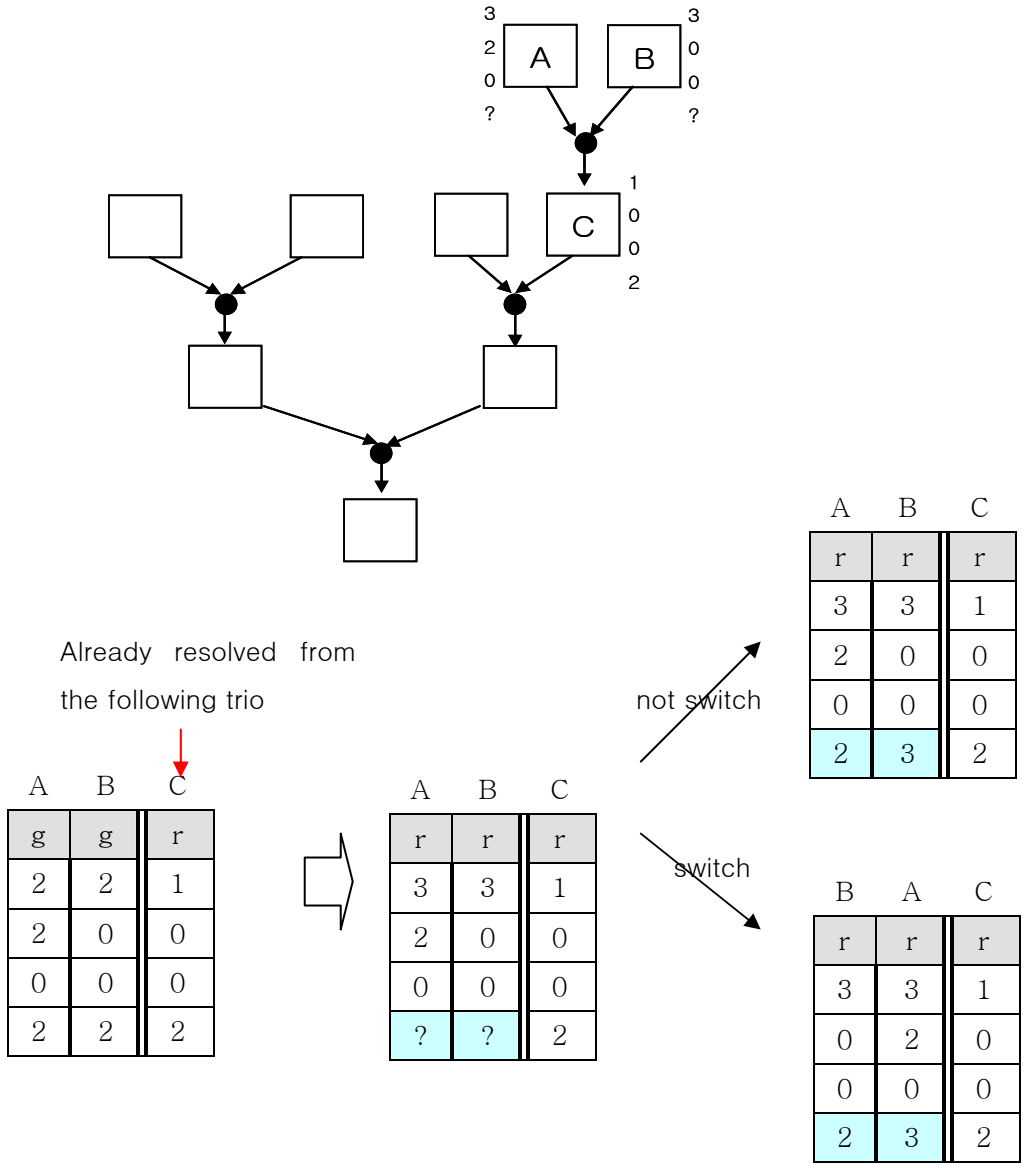
2) the node is not a joint node (has only one child).

3) the trio which includes this node contains <g, g, g/r > = < 2, 2, 2 >.

4) the order of parents nodes in the trio can not be determined.

For example in Figure 29, consider nodes A and B to be the earliest ancestor nodes, and that they have only child C in the pedigree. Since node A and B do not mate with any others, they are not a joint node. So, we only utilize information from a trio A-B-C for resolving the node A and B. After C is resolved as < 1002 > from the descendant trio, we have to figure out whether A and B need to be switched or not by SS1 (Theorem8). However, < A,B,C > = < g, g, r > = < 2202, 2002, 1002 > does not provide any further information about the decision to switch. Although the first three loci in trio are successfully resolved (see Figure 29), the fourth locus (< g, g, r > = < 2, 2, 2

>) yields an ambiguity. This is because the order of nodes A and B must be decided. In fact, the results from the order < A, B, C > and < B, A, C > are different on this ambiguous locus. While the order <A,B,C> produces < r, r, r > = < 3202, 3003, 1002>, the order < B, A, C > produces < r, r, r > = < 3002, 3203, 1002 >. Therefore, if a pedigree has this specific case, the resolved haplotypes may not be accurate. Nonetheless, since it rarely happens that the above four conditions are met, the resolution accuracy is still ~ 99%. In conclusion, the Pre-Resolved Table algorithm reduces the computation time significantly while it sustains its reliability in the haplotype phasing.



A	B	C
g	g	r
2	2	1
2	0	0
0	0	0
2	2	2

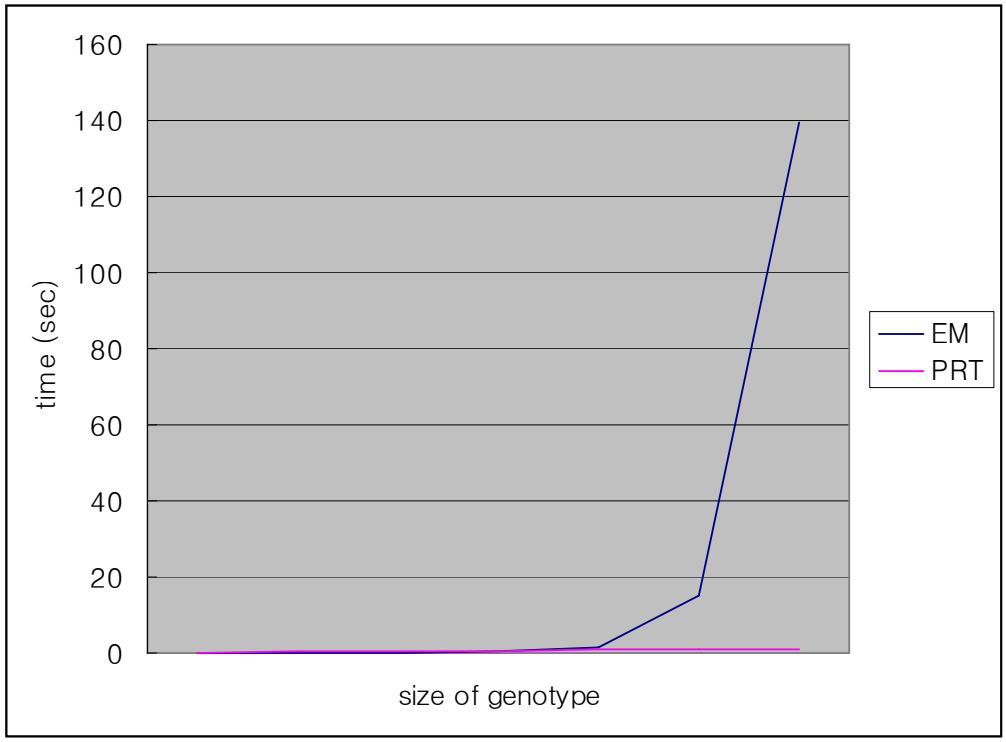
A	B	C
r	r	r
3	3	1
2	0	0
0	0	0
?	?	2

A	B	C
r	r	r
3	3	1
2	0	0
0	0	0
2	3	2

B	A	C
r	r	r
3	3	1
0	2	0
0	0	0
2	3	2

< Figure 29 > shows a case where unphased data remains. Whether A and B are switched or not varies the output. Although we need more information to resolve this case, there is no more information because A and B are not a joint node and do not have any more children, and also do not have any ancestor information. Therefore, we cannot resolve a unique solution in this specific case.

4.3 Comparison



< Figure 30 > illustrates the graph for comparison of average running time variation between the EM algorithm and the Pre-Resolved Table algorithm. The blue line indicates computation time of the EM algorithm, while the red line denotes the Pre-Resolved Table algorithm. (200 individual samples)

The EM algorithm is more efficient if the sample data indicates only the genotypes of its population. However, the time complexity for one iteration of the EM algorithm is $O(n2^k)$ where n is the number of genotypes, and k is the maximum number of heterozygous loci.

On the other hand, The Pre-Resolved Table algorithm has $O(nmd)$ for its time complexity, where n is the size of pedigree (the number of genotypes), m is the number of loci, and d is the number of

trees (number of root descendants).

Therefore, if the sample data contains the pedigree information, the Pre-Resolved Table algorithm will provide reliable computing with reduced running time.

V. Conclusion and Further Works

In this paper, we present the Pre-Resolved Table algorithm which only increases computing time linearly with various data sizes, while other statistical methods tend to increase the time complexity exponentially. The Pre-Resolved Table algorithm resolves haplotypes more accurately than other parsimony and phylogeny algorithms. The Pre-Resolved Table algorithm reduces comparing scope with pre-resolved order with every possible genotype combination for the parents-child trio. Because it has basically a binary tree structure, the running time is more reliable.

In this study we assume that there is no recombination and mutation. Even though the Pre-Resolved Table algorithm can resolve the pattern of haplotype with very reliable accuracy (~99%), it still exhibits several drawbacks. When there is recombination or mutation in the data, this can not produce exact results. Also, each genotype's specific locus is the ambiguity site, they can not be resolved properly because every <g,g,g> =<2,2,2> combination does not give any information. To address this problem, we are investigating the integration between the Pre-Resolved Table algorithm and the EM algorithm to resolve only these ambiguous sites, which will provide more accurate results.

Bibliography

1) Cornelis A. Albers, Tom Heskes, Hilbert J. Kappen, Haplotype Inference in General Pedigrees using the Cluster Variation Method, Genetics, Vol. 177, 1101-1116, October 2007
2) Dan Gusfield. Haplotype Inference by Pure Parsimony. CPM 2003: 144-155
3) Dana C. Crawford and Deborah A. Nickerson.(2005) Definition and Clinical Importance of Haplotypes, Annu.Rev.Med. 2005. 56. 303-320
4) Daniele Fallin, Nicholas j. Schork. Accuracy of Haplotype Frequency Estimation for Biallelic Loci, via the Expectation-Maximization Algorithm for Unphased Diploid Genotype Data, The American Journal of Human Genetics, Volume 67, Issue 4, 947-959, 1 October 2000
5) Eleazar Eskin, Eran Halperin, Richard M. Karp, Efficient reconstruction of haplotype structure via perfect phylogeny, EECS Department, University of California, Berkeley Technical Report No. UCB/CSD-02-1196 August 2002
6) Eran Halperin, Richard M. Karp, 2004, Perfect Phylogeny and Haplotype Assignment, RECOMB'04. March 27-31, 2004, San Diego, Califonia, USA
7) Eric P.Xing, Michael Jordan, Roded Sharna. 2007, Bayesian Haplotype Inference via the Dirichlet Process, Journal of Computational Biology, April 2007, 14(3):267-284.
8) Francis Y.L Chin, Qiangfeng Zhang, Hong Shen, 2005, k-Recombination Haplotype Inference in Pedigrees, V.S.Sunderam et al. (Eds): ICCS 2005, LNCS 3515, pp. 985-993
9) Frank Dellaert, 2002, The Expectation Maximization Algorithm, Colloege of Computing, Georgia Institute of Technology Technical Report number GIT-GVU-02-20
10) Jian Zhang, Nartin Vingron, Margret R. Hoehe, 2005, Haplotypes Reconstruction for Dipolid Populations, Human Heredity 59;144-156
11) Jing Li, Tao Jiang, PedPhase: Haplotype Inference for Pedigree Data, Publisher Unknown, 2003, Source www.cs.ucr.edu/~jili/PedPhase.pdf
12) Jonathan Marchini, etc. 2006, A Comparison of Phasing Algorithms for Trios and Unrelated Individuals, The American Journal of Human Genetics Vol. 78, March 2006
13) Tianhua Niu, 2004, Algorithms for Inferring haplotypes, Genetic Epidemiology 27:334-347
14) Qiangfeng Zhang, Francis Y.L Chin, Hong Shen, 2005, Minimum Parent-Offspring Recombination Haplotype Inference in Pedigrees, C. Priami, A Zelikovsky (Eds.): Trans. On Comput. Syst. Biol. II, LNBI 3680, pp. 100-112.

15) Vineet Bafna, Dan Gusfield, Giuseppe Lancia, Shibu Yooseph, 2003, Haplotype as Perfect Phylogeny:A Direct Approach, Journal of Computational Biology Vol.10, no 3-4, 2003 pp. 323-340
16) Zhihong Ding, Thomas Mailund, Yun S. Song, 2008, Efficient whole-genome Association Mapping using local phylogenies for unphased genotype data, Bioinformatic Vol. 24 no. 19 2008, pp 2215-2221
17) HAPMAP www.hapmap.org
18) Wikipedia www.en.wikipedia.org