

國立成功大學

資訊管理研究所

碩士論文

人類群體基因組單體型推論問題
之研究

**On solving population haplotype
inference problems**

指導教授：王逸琳 博士

研究生：楊惠娥

中華民國九十五年八月

國立成功大學
碩士論文

On solving population haplotype
inference problems

研究生：楊惠娥

本論文業經審查及口試合格特此證明

論文考試委員：

陳文智

李宇欣

許瑞麟

王逸琳

陳文智
李宇欣
許瑞麟
王逸琳

指導教授：王逸琳

系(所)主管：李賢得

中華民國九十五年五月二十七日

博碩士論文授權書

(國科會科學技術資料中心版本 93.2.6)

本授權書所授權之論文為本人在 國立成功 大學(學院) 資訊管理 系所
組 九十四 學年度第 二 學期取得 碩 士學位之論文。

論文名稱: On solving population haplotype inference problems

☒ 同意 ☐ 不同意

本人具有著作財產權之論文全文資料, 授予行政院國家科學委員會科學技術資料中心(或其改制後之機構)、國家圖書館及本人畢業學校圖書館, 得不限地域、時間與次數以微縮、光碟或數位化等各種方式重製後散布發行或上載網路。

本論文為本人向經濟部智慧財產局申請專利(未申請者本條款請不予理會)的附件之一, 申請文號為: _____, 註明文號者請將全文資料延後半年再公開。

☒ 同意 ☐ 不同意

本人具有著作財產權之論文全文資料, 授予教育部指定送繳之圖書館及本人畢業學校圖書館, 為學術研究之目的以各種方法重製, 或為上述目的再授權他人以各種方法重製, 不限地域與時間, 惟每人以一份為限。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。上述同意與不同意之欄位若未鈎選, 本人同意視同授權。

指導教授姓名: 王逸珉

研究生簽名: 楊惠娥

(親筆正楷)

學號: R76931038

(務必填寫)

日期: 民國 95 年 8 月 24 日

1. 本授權書 (得自 <http://sticnet.stic.gov.tw/sticweb/html/theses/authorize.html> 下載或至 <http://www.stic.gov.tw> 首頁右下方下載) 請以黑筆撰寫並影印裝訂於書名頁之次頁。
2. 授權第一項者, 請確認學校是否代收, 若無者, 請自行寄論文一本至台北市(106)和平東路二段 106 號 1702 室 國科會科學技術資料中心 黃善平小姐。(本授權書諮詢電話: 02-27377606 傳真: 02-27377689)

摘要

隨著人類基因體計劃的完成，我們對人類DNA的結構及序列已有初步認識，但仍無法確知其功能。DNA的功能可能是疾病研究的關鍵，為了確定其功能，科學家們會針對人類每條染色體上稱為基因組單體型(haplotype)的基因鹼基組合來做進一步的分析。

由於直接取得基因組單體型資料將耗費許多成本及時間，科學家們通常會使用在一對染色體上稱為基因型(genotype)的基因鹼基混合敘述資料，來代替基因組單體型資料以進行分析。然而，基因型資料所含的資訊並不足以正確地推論出每條染色體上基因的鹼基組合，因此我們必須求解人類群體基因組單體型推論(Population Haplotype Inference, PHI)問題，以從一群人的基因型資料來推論出其相對應的基因組單體型資料。在眾多的PHI問題相關文獻中，近年來以應用最大簡約原則(Pure Parsimony)的PHI問題(PHI problem based on pure parsimony criterion, HIPP)之相關研究最受矚目。HIPP問題旨在使用最少的不同基因組單體型，來解讀一個給定的基因型矩陣。

本論文特別針對HIPP問題加以探討，在探討並整理文獻中各類求解PHI問題之方法後，我們依照基因型資料間的相互包容或排斥的數學關係，提出兩個結合數學規劃與生物意義的啟發式演算法。其中，第一個演算法利用基因型資料間的容斥關係，將可行解空間大幅縮小，以整合的基因型資料來求解一個整數規劃問題；而第二個演算法賦予那些可解讀較多基因型的基因組單體型更高的權重，並使用貪婪(greedy)的方式來選取權重較大的基因組單體型以求解HIPP問題。

本論文並執行大規模的程式測試，來分析各類解法在求解 HIPP 問題時之效率及效果雙方面的表現。最後，我們提出一個可用來處理大規模 (large scale) HIPP 問題的技巧，將該問題切成較小規模的子問題，求解各子問題之後再將其解組合之；另外，我們亦針對文獻中一個極有效率的啟發式演算法加以改良，使之可以有效地求解出任一 HIPP 問題的所有最佳解。

關鍵字: 基因組單體型推論 (Haplotype inference)；基因型 (Genotype)；最大簡約 (Pure parsimony)；啟發式演算法 (Heuristic algorithm)；整數規劃 (Integer programming)

ABSTRACT

Due to the completion of Human Genome Project, we have known more about the structure and sequence, but not yet the function of human DNA. The function of DNA may be a key for human disease. To analyze the function of DNA, researchers have to obtain each haplotype, the genetic constitution of an individual chromosome, of an individual for analysis. Nevertheless, considering the significant efforts required in collecting haplotypes, usually the descriptions of one conflated pair of haplotypes called genotypes are collected.

Since the genotype data contains insufficient information to identify the combination of DNA sequence in each copy of a chromosome, one has to solve the population haplotype inference (PHI) problem which infers haplotype data from genotype data for a population. Previous researchers use mathematical programming methods and heuristic algorithms to solve the population haplotype inference problem. This thesis surveys these methods and conducts computational experiments on the efficiency and effectiveness for these methods in solving a population haplotype inference problem based on pure parsimony criterion (HIPP) which seeks the minimum number of distinct haplotypes to infer a given genotype matrix.

We propose two heuristic algorithms to solve the HIPP problem with promising performance. The first heuristic algorithm exploits the compatible relations among genotypes to solve a reduced integer linear programming problem in a smaller solution space. The second heuristic algorithm selects popular haplotypes that can resolve more genotypes in a greedy fashion. Extensive computational experiments have been

conducted for several PHI solution methods on both the biological and simulated data. The results show that our proposed algorithms are efficient and effective, especially for solving cases with larger recombination rates. Finally, we give a divide-and-concur technique to solve large-scale HIPP problems. We also improve a parsimonious tree growing heuristic to obtain all the multiple optimal solutions for an HIPP problem.

Key words : Haplotype inference, Genotype, Pure parsimony, Heuristic algorithm, Integer programming

ACKNOWLEDGEMENTS

本論文得以順利完成，首先以及最重要的必須要感謝指導教授王逸琳老師給予的細心指導，使我可以不斷挑戰自己的極限並培養踏實的作事態度，在此致上最誠摯的謝意。論文口試期間則承蒙陳文智老師、李宇欣老師及許瑞麟老師對本論文提供寶貴的建議，使本論文得以更加完善，學生在此一併致上由衷的謝意。

在這兩年研究所求學生涯中，我要感謝從大學時代就是同學的佳駿以及萬保，我們一起準備考試、熬夜以及分享心事的日子讓我成長許多，還有同研究室的修杰、正翰、姿君、群達以及橙坤，大家一起在研究室奮鬥，使得研究室充滿活力與歡笑，以及所有研究所的同學們，感謝你們使我的研究所生活多采多姿。另外，我還要特別感謝幫我整理實驗結果資料的學妹—家媛，有她的幫助，使我的論文能更完善。

最後，謹將完成這份論文以及碩士學位的榮耀與喜悅獻給我的父母及家人，感謝他們長期以來的支持及包容，使我能更有自信的迎接未來的挑戰。

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	i
LIST OF TABLES	iv
LIST OF FIGURES	vi
CHAPTER	
I. INTRODUCTION	1
1.1 Background and motivation	1
1.1.1 DNA, variations and mutations	2
1.1.2 SNP, haplotype and genotype	3
1.1.3 Haplotype assembly problem	6
1.1.4 Population haplotype inference problem	8
1.2 Objectives	9
1.3 Overview of thesis	10
II. SOLUTION METHODS FOR THE PHI PROBLEM	12
2.1 Statistical methods	12
2.1.1 EM algorithm	12
2.1.2 Bayesian methods	14
2.2 Combinatorial methods	15
2.2.1 Clark's inference rule	15
2.2.2 Perfect phylogeny	18
2.2.3 Pure parsimony	20
2.2.3.1 Integer linear programming	21
2.2.3.2 Quadratic integer programming approach	27
2.2.3.3 Branch-and-bound algorithm	29
2.2.3.4 Heuristic algorithm	30
2.2.3.5 Graph theory	34
2.3 Summary	36
III. TWO HEURISTIC ALGORITHMS FOR SOLVING THE PHI PROBLEM BASED ON PURE PARSIMONY	40

3.1	Compatible genotype pairs	40
3.2	The method of merged genotype pairs	42
3.3	The greedy heuristic inference method	48
3.4	Summary	52
IV. COMPUTATIONAL EXPERIMENTS AND ANALYSES . . .		54
4.1	Settings for our computational experiments	54
4.2	Experiments on β_2 AR gene	55
4.3	Experiments on simulated data	56
4.4	Experiments on GHI using other weight mechanisms	62
4.5	Summary	63
V. RELATED ISSUES IN SOLVING THE HIPPO PROBLEM . .		68
5.1	Techniques for solving large-scale HIPPO problems	68
5.2	Improving the PTG heuristic algorithm	70
5.3	Summary	71
VI. CONCLUSION		75
REFERENCES		79
BIOGRAPHY		85

LIST OF TABLES

Table

2.2	Summary of algorithms to the PHI problem	38
2.3	Summary of algorithm availabilities to the PHI problem	39
4.1	Ten haplotypes and eighteen genotypes of human β_2 AR gene	56
4.2	Optimality gap comparison (in %) for no recombination cases	58
4.3	Error rate comparison (in %) for no recombination cases	58
4.4	Computational time comparison (in seconds) for no recombination cases	59
4.5	Optimality gap comparison (in %) for recombination rate $r = 4.0$ cases	59
4.6	Error rate comparison (in %) for recombination rate $r = 4.0$ cases . .	59
4.7	Computational time comparison (in seconds) for recombination rate $r = 4.0$ cases	60
4.8	Optimality gap comparison (in %) for recombination rate $r = 16.0$ cases	60
4.9	Error rate comparison (in %) for recombination rate $r = 16.0$ cases . .	60
4.10	Computational time comparison (in seconds) for recombination rate $r = 16.0$ cases	61
4.11	Optimality gap comparison (in %) on our GHI using several different weight mechanisms for no recombination cases	63
4.12	Optimality gap comparison (in %) on our GHI using several different weight mechanisms for recombination rate $r = 4.0$ cases	64
4.13	Optimality gap comparison (in %) on our GHI using several different weight mechanisms for recombination rate $r = 16.0$ cases	64

4.14	Error rate comparison (in %) on our GHI using several different weight mechanisms for no recombination cases	64
4.15	Error rate comparison (in %) on our GHI using several different weight mechanisms for recombination rate $r = 4.0$ cases	65
4.16	Error rate comparison (in %) on our GHI using several different weight mechanisms for recombination rate $r = 16.0$ cases	65
4.17	Computational time comparison (in seconds) on our GHI using several different weight mechanisms for no recombination cases	65
4.18	Computational time comparison (in seconds) on our GHI using several different weight mechanisms for recombination rate $r = 4.0$ cases . . .	66
4.19	Computational time comparison (in seconds) on our GHI using several different weight mechanisms for recombination rate $r = 16.0$ cases . .	66

LIST OF FIGURES

Figure

1.1	Illustration of SNPs and haplotypes	4
1.2	Illustration of haplotype and genotype	5
1.3	Relationship of SNP and sickle cell anemia	6
1.4	An example of haplotype assembly problem	7
1.5	An example of genotype and genotype matrix element	9
2.1	The basic idea of Clark's inference rule	16
2.2	An example of PHI problem via perfect phylogeny	19
2.3	An example of PTG algorithm	31
3.1	An example of compatible and incompatible graphs	41
3.2	An example of compatible genotype pairs	45
3.3	Illustration of haplotype weight calculation using our GHI algorithm .	50
4.1	Illustration of recombination during the formation of gametes	57
5.1	An example of genotype pairwise comparisons	72
5.2	An example of ImprovedPTG - the first tree	72
5.3	An example of ImprovedPTG - the second tree	73
5.4	An example of ImprovedPTG - the third tree	73
5.5	An example of ImprovedPTG - the fourth tree	74

CHAPTER I

INTRODUCTION

1.1 Background and motivation

With the completion of the *Human Genome Project* in 2003, it is the beginning of *Post-Genomic Era*, which focuses on functional genome analysis. In the Post-Genomic Era, the development of a full *Haplotype Map* has high priority since the knowledge about the genetic constitution of an individual chromosome called as the *haplotypes* can be applied in various domains, such as linkage disequilibrium, inference of population evolutionary history, disease diagnosis, and customization of treatment for each individual (Helmuth, 2001; Nickerson et al., 1998).

There are several molecular methods for obtaining haplotype data, such as cloning, allele-specific polymerase chain reaction (AS-PCR) and single molecule dilution (Ruano and Kidd, 1989; Ruano, Kidd, and Stephens, 1990; Michalatos-Beloin, Tishkoff, Bentley, Kidd, and Ruano, 1996; Clark et al., 1998). Unfortunately, the time, labor, and expense involved in directly collecting the haplotype data force researchers to collect the descriptions of one conflated pair of haplotypes called as *genotype* data, rather than haplotype data (Futatsugawa et al., 2004). Therefore, efficient and effective algorithms to infer haplotype data from genotype data are required. There are two kinds of haplotyping problems in the literature: the haplotype assembly problem for one indi-

vidual called *Haplotype Assembly* (HA) problem and the haplotype inference problem for a population called *Population Haplotype Inference* (PHI) problem. Here we first introduce several terminologies of molecular biology and their applications related to the PHI problem.

1.1.1 DNA, variations and mutations

Nucleic acids, such as DNA and RNA, can be found in all living cells and viruses. There are four possible bases to form a DNA, which are *Adenine*(A), *Thymine*(T), *Guanine*(G) , and *Cytosine*(C). All DNA strands are folded into double helices, called double strand DNA, to maintain a stable structure. The most important feature of the double strand DNA structure is the *Watson-Crick base pairing*, which says that A and T, C and G are always in pairs, respectively. DNA replicates itself based on this base pairing rule. This pairing rule may be destroyed in some damaged DNA caused by either chemical, radiation, or ultraviolet rays, and lead to variations or mutations of DNA.

Variations are genetic differences between individuals. Since many phenotypic variations (for example, different eye colors between individuals) are caused by DNA variations, analysis of the human genome focuses primarily on DNA variations.

There are many different mutations defined by biologists, for instance, *point mutations*, *frame shift mutations*, *insertions*, and *deletions*. Some of these mutations may cause incorrect pairing for the paired bases A and T, C and G. Among these mutations, point mutation (a DNA base that is replaced by another DNA base) is mostly observed.

The effect of variations or mutations of DNA could be understood using an example of English sentence as follows: consider the genetic information in a DNA as the language that we use in daily life. In the sentence “We play ball with you.” if

the word “you” is replaced by “him”, the meaning of this sentence will be changed; if the word “you” is deleted, this sentence will be incomplete; and the meaning will be ambiguous if “her” is added after “you”. We may think the mutations to DNA as the changes of words in an English sentence, and some important proteins may not be produced or be produced correctly due to mutations just like some sentences may be incomplete or incorrect due to the changes of words.

1.1.2 SNP, haplotype and genotype

All diploid organisms, including human, have two copies of chromosomes, one copy is inherited from the father, and the other is from the mother. Identical DNA sequences never appear in two different persons. When we align more than or equal to two DNA sequences with each other, each different position (i.e. locus or site) are called *single nucleotide polymorphism* (SNP, pronounced *snip*). More formally speaking, SNP is a kind of variation, at which DNA bases are different among individuals, and this variation should be observed at least 1% in all populations. Roughly speaking, there are more than 1.8 million SNPs in all 3 billion DNA bases in humans. Thus on average, there exists a SNP among every 100 to 300 bases along the human genome. Since approximately 90% of the variations in humans are SNPs, there are many organizations and projects such as National Center for Biotechnology Information (NCBI), The SNP Consortium (TSC), and International HapMap Project trying to identify all haplotypes in human beings.

The haplotype is a sequence of closely linked SNPs in one copy of chromosome; hence there are two haplotypes in a pair of chromosomes in all diploid organisms. Using Figure 1.1 as an example, suppose we sequence four individuals’ DNA sequences from locus one to nine of chromosome 1b, we will have four DNA sequences from the same chromosome. As we can see in this figure, there are three SNPs and four

haplotypes. The three SNPs occur at locus 2, 5, and 9, and the four haplotypes are $\{ACC\}$, $\{ATC\}$, $\{TTC\}$, and $\{ATT\}$.

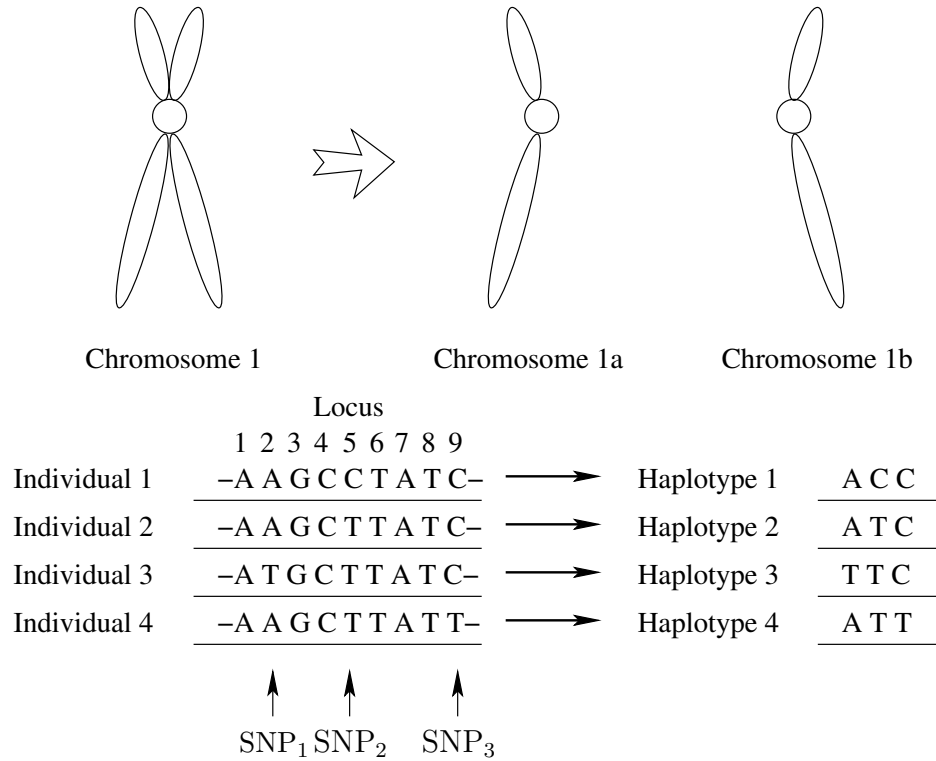


Figure 1.1: Illustration of SNPs and haplotypes

Due to the difference of genetic make-up between each individual, collected haplotypes can be used to customize some medical treatments and minimize their side effects. Since the collection of haplotypes data requires a huge amount of cost and time, the data for genotypes rather than haplotypes are collected. A genotype is the description of one conflated pair of haplotypes. That is, we sequence the *alleles* together when getting the genotype data. Using genotype data alone without other information can not help us to identify what DNA bases belong to which copy of chromosome. Referring to Figure 1.2 as an instance, there are three columns in this figure. The first column is the chromosome 1, the second and third ones are the chromosome 1a and 1b, respectively. If we sequence chromosome 1a and 1b separately, we shall get three SNPs

and two haplotypes (AGC and AGT), as shown in column 2 and column 3. However, considering the cost and time required, it is more convenient to collect genotype data instead of haplotype data. When sequencing chromosome 1a and 1b together, we can get three genotypes data, which are $\{A/A\}$, $\{G/G\}$, and $\{C/T\}$, but we are not sure whether the haplotype in chromosome 1a is $\{AGC\}$ or $\{AGT\}$.

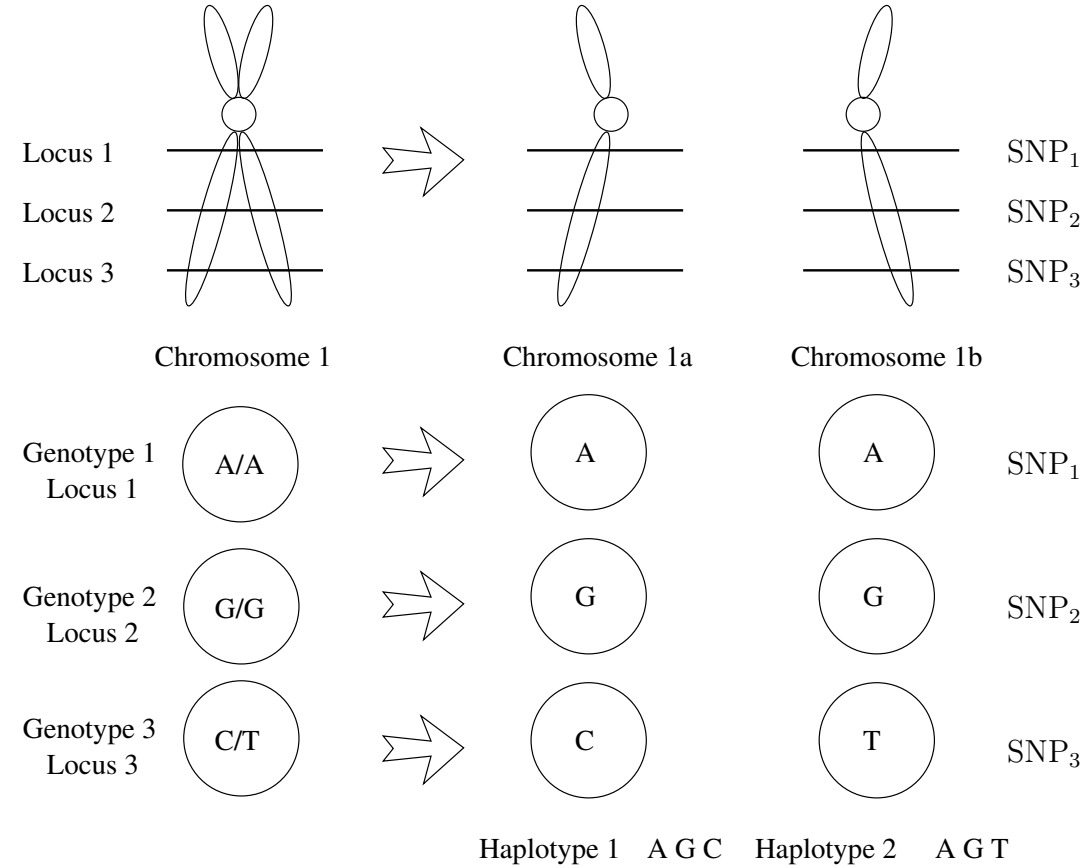


Figure 1.2: Illustration of haplotype and genotype

Because haplotypes are often inherited together from the parents, it serves as a useful tool to diagnose diseases. For example, to detect whether a person has *sickle cell anemia* (a disease more often seen in Blacks) or not, one may examine and analyze his SNP and haplotype. A person with sickle cell anemia will have the sixth amino acid (which should be *glutamic acid* for a normal person) in his red corpuscle be replaced by *valine*. Researchers have also investigated these genetic bases and confirmed that a

DNA base *A* is replaced by *T* in those sickle cell anemia patients (see Figure 1.3).

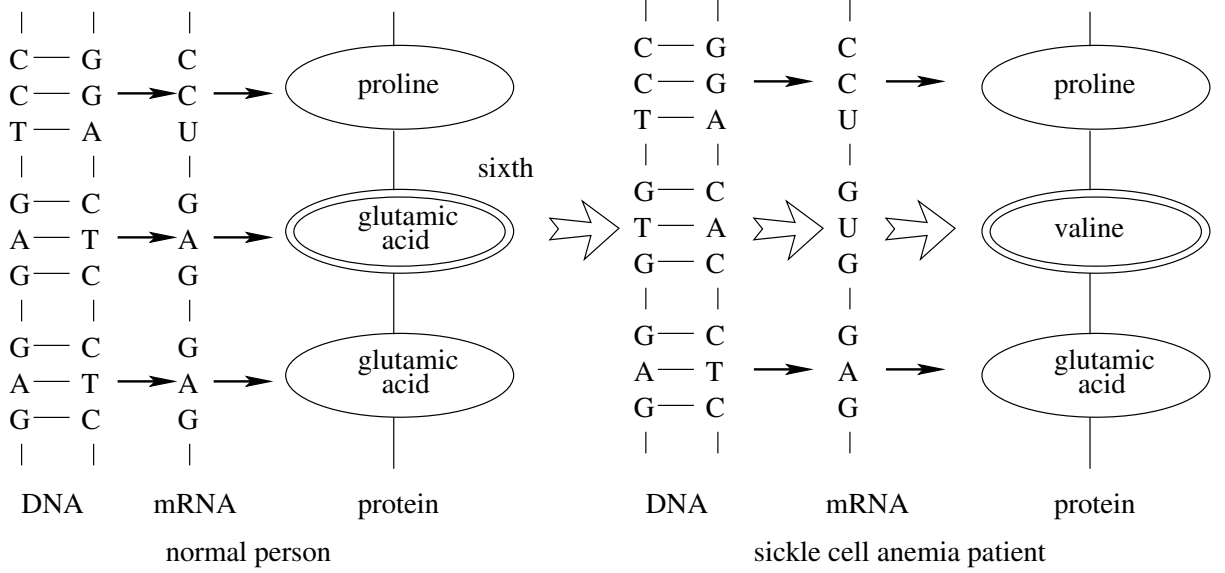


Figure 1.3: Relationship of SNP and sickle cell anemia

1.1.3 Haplotype assembly problem

HA problem infers an individual's haplotype from his short genome fragments. The input matrix for an HA problem is called a SNP matrix M which may contain some missing data with rows and columns corresponding to short genome fragments of one person and SNPs, respectively. Every element in M is 0, 1, or “-” where 0 denotes the *wild type*, 1 the *mutant type* and “-” the missing data. For example, Figure 1.4(a) illustrates five short genome fragments for some individual with the SNP loci marked in Figure 1.4(b). The HA problem tries to partition the fragments into two haplotype sets based on some optimization criteria according to SNP allele values, as shown in Figure 1.4(c).

Lancia et al. (2001) first bring up the HA problem and propose three optimization problems: *Minimum Fragment Removal* (MFR), *Minimum SNP Removal* (MSR), and *Longest Haplotype Reconstruction* (LHR). Given a SNP matrix M , the MFR and MSR

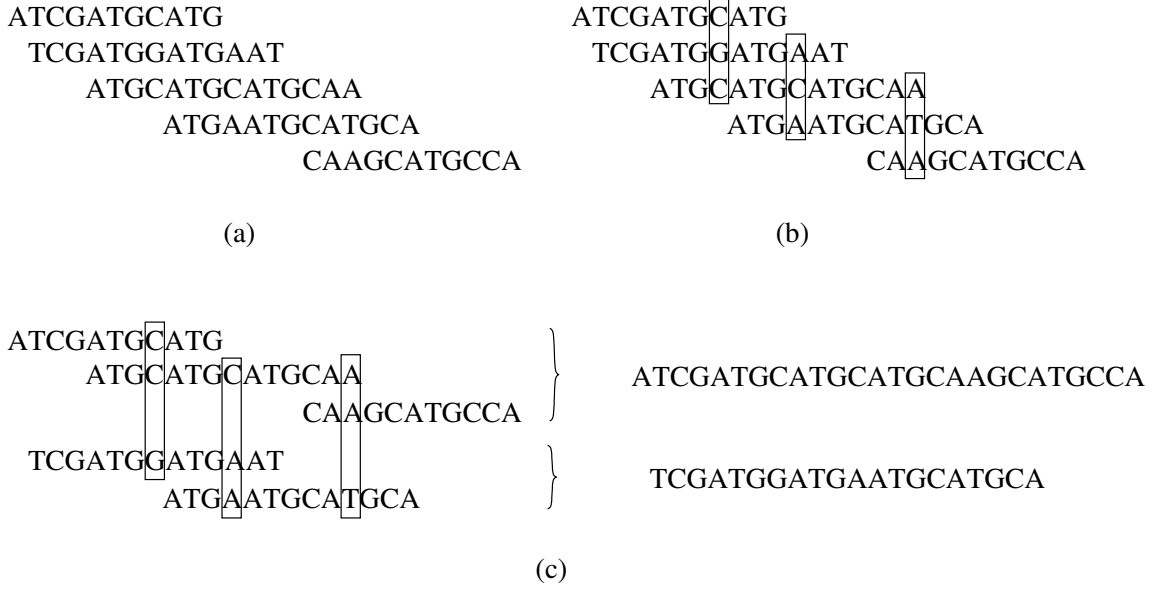


Figure 1.4: An example of haplotype assembly problem

problems try to remove the minimum number of fragments (rows) and SNPs (columns), respectively, such that the remaining fragments can be partitioned into two haplotype sets; the LHR problem removes a set of fragments (rows) such that the remaining fragments can be partitioned into two haplotype sets and the sum of lengths of the derived haplotypes is maximized. Based on these three optimization problems, Lippert (2002) proposes another optimization problem: *Minimum Error Correction* (MEC). Since it may have read-errors during sequencing, the objective in MEC is to correct (i.e. convert the elements with value 0 to 1 in the input SNP matrix or vice-versa) as few entries of the input matrix M as possible such that the fragments in the corrected matrix can be partitioned into two haplotype sets. These papers derive some possible research directions for the HA problem and examine the characteristics of optimal solutions without proposing any solution method. Rizzi (2002) proposes an $O(mn^2)$ and another $O(m^2n + m^3)$ dynamic programming algorithms to solve the MSR and MFR problems, respectively, where m is the number of input fragment and n is the number of SNPs. Recently, Wang et al. (2005) design a *genetic algorithm* (GA) to

solve the MEC problem.

1.1.4 Population haplotype inference problem

In this thesis, we focus on the problem of inferring haplotypes from genotypes for a group of individuals called population haplotype inference problem (henceforth PHI problem) defined as follows.

Suppose we are given m genotype vectors, each has length n , and denote this $m \times n$ matrix as a genotype matrix $G = [g_{i,j}]$. Each row in G corresponds to a genotype data for one individual, while each column stands for one SNP. The element $g_{i,j}$ is the genotype data for individual i at locus j . Let $G = \{g_1, g_2, \dots, g_m\}$ be the set of all rows in G and $g_i = \{g_{i,1}, g_{i,2}, \dots, g_{i,n}\}$ denote the genotype data for individual i . Every element $g_{i,j}$ in G is 0, 1, or 2 depending on whether the j th SNP data of the i th individual is *homozygous wild type*, *homozygous mutant*, or *heterozygous*, respectively. A single allele is said to be of wild type (for example, the base T in Figure 1.5) if it is a major observed DNA base of a species at that locus, or otherwise it is said to be of mutant type (for example, the base C in Figure 1.5). Suppose the chromosome for individual i is illustrated in Figure 1.5, since the DNA base T is of wild type at locus 1, $g_{i,1} = 0$; the DNA base C is of mutant type at locus 2, so $g_{i,2} = 1$, $g_{i,3} = 2$ since alleles A and G are different DNA bases at locus 3. Any element $g_{i,j}$ in G is said to be *resolved* if it is 0 or 1, and *ambiguous* if it is 2. Suppose h_a and h_b are two haplotypes vector with length n , and they constitute a pair which resolves a genotype g_i , denoted by $g_i = h_a \otimes h_b$, if and only if the following conditions hold:

1. $(h_{a,j}, h_{b,j}) = (0, 0)$ for every site j that $g_{i,j} = 0$,
2. $(h_{a,j}, h_{b,j}) = (1, 1)$ for every site j that $g_{i,j} = 1$,
3. $(h_{a,j}, h_{b,j}) = (0, 1)$ or $(1, 0)$ for every site j that $g_{i,j} = 2$.

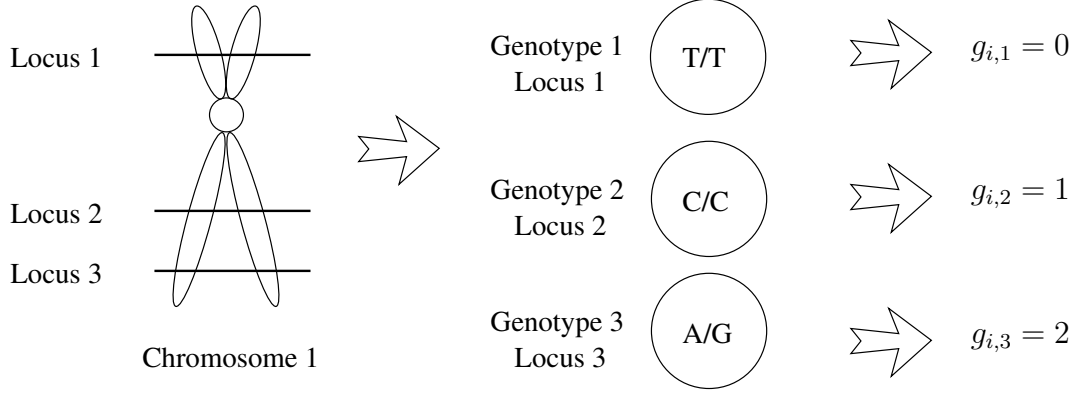


Figure 1.5: An example of genotype and genotype matrix element

When $g_i = h_a \otimes h_b$, we say h_a (or h_b) a *candidate* (or *explaining*) *haplotype* for g_i , (h_a, h_b) is a *candidate* (or *explaining*) *haplotype pair* for g_i , and h_a (or h_b) is the *conjugate* (or *complementary*) *haplotype* for h_b (or h_a) in the pair (h_a, h_b) . For example, if $g_1 = 012$, $h_1 = 010$ and $h_2 = 011$, we can say $g_1 = h_1 \otimes h_2$, (h_1, h_2) is a candidate haplotype pair for g_1 , and the conjugate haplotype for h_1 is h_2 for this candidate haplotype pair. Given a genotype matrix $G = \{g_1, g_2, \dots, g_m\}$, the PHI problem (some literature called it *Haplotype Reconstruction* problem) is to find a set of haplotypes $H = \{h_1, h_2, \dots, h_{2m}\}$ such that g_i is resolved by h_{2i-1} and h_{2i} . For instance, if the input genotype matrix $G = \{101, 012\}$, where $g_1 = 101$ and $g_2 = 012$, the PHI problem seeks four haplotypes (h_1, h_2, h_3 , and h_4) such that $g_1 = h_1 \otimes h_2$ and $g_2 = h_3 \otimes h_4$. In this example, $h_1 = h_2 = 101$, $h_3 = 010$, and $h_4 = 011$.

1.2 Objectives

In this thesis, we are interested in solving a variant of the PHI problem, called the PHI problem based on pure parsimony (henceforth HIPP). In particular, an HIPP problem not only seeks candidate haplotype pairs to resolve a given genotype matrix, but also require the total number of distinct candidate haplotypes in the solution to be minimized.

The solution methods in the literature for solving the HIPP problem are either based on mathematical programming techniques such as integer linear programming (see Gusfield, 2003; Brown and Harrower, 2004) and quadratic integer programming (see Huang, Chao, and Chen, 2005), or heuristic algorithm (see Li, Zhou, Zhang, and Chen, 2005). The mathematical programming based algorithms usually consume a lot of computational time and not suitable for solving large-scale HIPP problems. On the other hand, the heuristic algorithm is very fast but its effectiveness in terms of the optimality gap (i.e. the difference in the number of distinct candidate haplotypes used, compared with the optimal solution) remains to be evaluated. This thesis proposes two new HIPP heuristic algorithms which exploit both the mathematical properties and the biological insights of the HIPP problems.

To understand the practical efficiency and effectiveness of these HIPP solution methods, this thesis conducts more complete computational experiments for several HIPP solution methods in the literature, including our owns, on solving a real-world case and several simulated cases.

To solve large-scale HIPP problems using mathematically programming algorithms, this thesis also give a divide-and-concur technique and discuss its pros and cons. Finally we also give an improvement based on the mathematical properties of our heuristics for a parsimonious tree growing heuristic algorithm proposed by Li et al. (2005) to enumerate all the multiple optimal solutions as fast as possible.

1.3 Overview of thesis

The remainder of this thesis is organized as follows. Chapter 2 surveys literature for methods to solve the PHI problem and compares their pros and cons. In Chapter 3, we give two new heuristic algorithms to solve the HIPP based on the mathematical properties and biological insights. Several numerical experiments on both real-world

and simulated genotype datasets for different formulations and algorithms (including our heuristic algorithms) are conducted and summarized in Chapter 4. Chapter 5 illustrates a proposed *divide-and-conquer* algorithm for solving large-scale HIPP problems, as well as improvement over a heuristic algorithm called PTG to get a better solution. Finally, Chapter 6 concludes this thesis, lists our contribution and gives suggestions for future works.

CHAPTER II

SOLUTION METHODS FOR THE PHI PROBLEM

In order to solve the PHI problem and obtain a solution with biological meanings, we should design the solution methods based on some genetic or biological models. As suggested by Gusfield (2004), the solution methods for the PHI problem are either based on the models of statistics or the techniques of combinatorics. Previous survey papers in this topic can be found in Halldórsson et al. (2002) and Zhang et al. (2006). Here in this Chapter, we review these solution methods and summarize recent progress in solving the PHI problem.

2.1 Statistical methods

There are two major statistical methods to the PHI problem: the estimation-maximization (EM) algorithm and the Bayesian methods.

2.1.1 EM algorithm

Excoffier and Slatkin (1995), Long et al. (1995), and Hawley and Kidd (1995) firstly discuss the use of EM algorithm to the PHI problem under the assumption of *Hardy-Weinberg equilibrium* (HWE) which states that under the assumptions of random mating within a single population, no selection, no mutation, and after one

generation of random mating, the genotype frequencies at a single gene locus which can be represented as a simple function of the allele frequencies at that locus will be fixed at a particular equilibrium value. In particular, suppose in the simplest case of a single locus with two alleles A and a with allele frequencies of f_A and f_a , respectively, under the assumption of HWE, we will predict the homozygote AA with the genotypic frequencies f_A^2 , the Aa heterozygote with the genotypic frequencies $2f_Af_a$, and the homozygote aa with the genotypic frequencies f_a^2 .

In general, the EM algorithm includes two major steps: estimation and maximization. When applying the EM algorithm to the PHI problem, the first step (estimation) estimates the population haplotype probabilities based on maximum likelihood, and the second step (maximization) finds the haplotype probabilities optimizing the probability of the observed data. Let $G = \{g_1, g_2, \dots, g_m\}$ denote the given genotype data for m individuals, where g_i denotes the genotype data for the i th individual. Let $\Theta = (\theta_1, \dots, \theta_q)$ denote the unknown overall haplotype frequency (probability), where θ_k represents the frequency for haplotype h_k and there are totally q possible candidate haplotypes. Suppose that HWE holds, the population frequency of individuals with the haplotype pair (h_a, h_b) such that $g_i = h_a \otimes h_b$ is $\theta_a \theta_b$. Then, the likelihood function of the haplotype frequencies given the overall haplotype frequencies can be written as

$$L(\Theta) = P(G|\Theta) = \prod_{i=1}^m \sum_{(h_a, h_b): h_a \otimes h_b = g_i} \theta_a \theta_b. \quad (2.1)$$

The EM algorithm seeks an estimator $\hat{\Theta}^{EM}$ for the unknown population haplotype frequencies Θ . It contains iterative procedures which start from an initial guess of the haplotype frequencies $(\hat{\theta}_1^0, \dots, \hat{\theta}_M^0)$ and then update the haplotype frequencies that maximize the likelihood function until the changes of haplotype frequencies in consecutive iterations are less than a specified small threshold value.

After the estimator $\hat{\Theta}^{EM}$ is obtained, we can choose the most likely haplotype

assignment \hat{H}^{EM} , given the genotype data G and the estimated population haplotype frequencies $\hat{\Theta}^{EM}$. In other words, we choose the \hat{H}^{EM} , which maximizes the conditional probability $\Pr(H|\hat{\Theta}^{EM}, G)$, to solve the PHI problem.

2.1.2 Bayesian methods

The Bayesian method starts with a model formulation designed to describe the situation of interest (e.g. the relationship between the unknown overall haplotype frequencies and the observed genotypes). A *prior* distribution over the unknown parameters of the designed model formulation is then made to capture the hypotheses before the observed data are taken into account (e.g. setting equal probabilities for all candidate haplotypes). Likelihood functions of the observed data can be made as probability functions of the unknown parameters (e.g. the overall haplotype frequencies). Then, a *posterior distribution* which takes both the prior and the likelihood into account for the unknown parameters can be computed based on Bayes' Rule. Predictions for future observations (e.g. prediction for the possible probability that a genotype g_i is composed by one pair of haplotypes (h_1, h_2)) can be made from the posterior distribution.

The prior information to the PHI problem is the hypotheses about what patterns of haplotypes we would expect to observe in population samples, for example, giving equal probabilities for all possible haplotypes means we believe that we will observe every possible haplotype with equal chance. The likelihood and the posterior distribution to the PHI problem are the information in the observed data and the conditional distribution of the haplotype frequencies given the observed genotype data, respectively. The Bayesian PHI methods treat the unknown haplotypes as random variables and exploit the prior distribution and the likelihood to calculate the posterior distribution, which is the conditional distribution of the haplotype frequencies given the

observed genotype data. Then the haplotypes themselves can be estimated from this posterior distribution. For example, choosing the most likely explaining pair of haplotypes for each genotype can solve the PHI problem. Stephens et al. (2001) propose a *pseudo-Bayesian* (the method where the models used and priors can not be taken seriously as expressions of prior belief) algorithm, whereas Niu et al. (2002) propose another Bayesian algorithm to the PHI problem. Comparison on the differences between priors and the computational strategies of several Bayesian methods for the PHI problem is also discussed by Stephens and Donnelly (2003).

2.2 Combinatorial methods

When there are n heterozygous sites (i.e., sites with value 2) in a genotype, the number of possible pairs to resolve it is 2^{n-1} . For example, there are $2^{3-1} = 4$ possible pairs (000, 111), (001, 110), (010, 101), and (011, 100) to resolve a genotype $g_i = 222$. Therefore, it is a combinatorial problem to resolve all genotypes. We need algorithms based on biological models to select one haplotype pair from those potentially exponential number of candidate haplotype pairs. There are three major combinatorial methods to the PHI problem: *Clark's inference rule*, *perfect phylogeny*, and *pure parsimony*. In this section, we summarize these methods, give examples, and discuss their advantages and limitations.

2.2.1 Clark's inference rule

Clark (1990) first brings up the PHI problem and gives a solution method known as Clark's inference rule. The basic idea of Clark's inference rule is illustrated in Figure 2.1. Suppose we have 5 genotypes (g_1, g_2, g_3, g_4 , and g_5), and there is no heterozygous sites in g_1 . Suppose g_1 can only be resolved by the haplotype pair $h_1 \otimes h_1$. Thus we shall try to use h_1 for resolving other genotypes. For example, we obtain h_2 since

$g_2 = h_1 \otimes h_2$; h_3 and h_4 can be derived from $g_3 = h_2 \otimes h_3$ and $g_4 = h_2 \otimes h_4$, respectively. Suppose $g_5 = h_5 \otimes h_6$ but can not be resolved by h_1 , h_2 , h_3 , or h_4 , then Clark's algorithm stops and leaves g_5 as an *orphan* (the unresolved genotype).

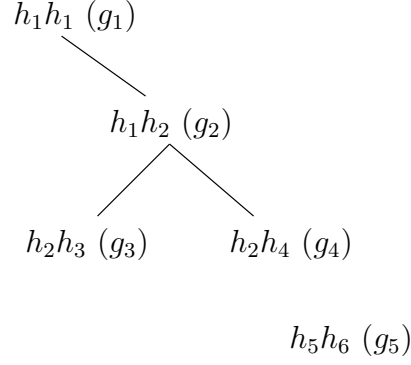


Figure 2.1: The basic idea of Clark's inference rule

In short, if a genotype has no or just one ambiguous site, it is called *homozygote*; otherwise, it is called *heterozygote*, and the algorithm of Clark's inference rule can be summarized as follows:

1. Identify each homozygote in a genotype matrix, treat it as resolved, and add it to the resolved set R .
2. Find one haplotype in the resolved set which can be applied to resolve one of the unresolved genotypes, if such a haplotype does not exist, then stop; otherwise, continue.
3. Add the haplotypes obtained in Step 2 into the resolved set R , then go to Step 2.

For example, given an input genotype matrix

$$G = \begin{bmatrix} 2 & 0 & 2 \\ 0 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} . \quad (2.2)$$

where $g_1 = 202$, $g_2 = 021$, and $g_3 = 212$, since there is only one homozygote (i.e. g_2) in G , we can start to resolve all genotypes using g_2 . Since $g_2 = 001 \otimes 011$, we add 001 and 011 into the resolved set. Using 001 to resolve g_1 , we will get a new haplotype 100. Likewise, 011 can be used to resolve g_3 . Suppose we first select 001 to resolve g_1 and get its complementary haplotype 100, the resolved set R becomes $\{001, 011, 100\}$. We may use the haplotype 011 in R to resolve g_3 , and obtain its complementary haplotype 110. Finally, $R = \{001, 011, 100, 110\}$ and there is no orphan.

The biological assumption behind the Clark's inference rule is that the homozygotes are the frequently observed haplotypes in the population. Therefore, it is preferred to use the homozygotes to resolve other genotypes. Clark's inference rule is very straightforward, with the following disadvantages and limitations:

1. This algorithm may never get started when there exists no homozygotes in the genotype matrix.
2. When this algorithm terminates, some genotypes may still left to be unresolved, such as g_5 in the Figure 2.1.
3. The resolving haplotypes may differ, given different order of the genotypes to be resolved.

Gusfield (2001) reformulates Clark's inference rule into a *maximum resolution* (MR) problem: given a set of genotypes, what is the maximum number of genotypes that can be resolved by successive application of Clark's inference rule? Gusfield also proves that the MR problem is NP-hard, which can be reduced to an integer linear programming with a graphical structure.

2.2.2 Perfect phylogeny

Hudson (1990) and Donnelly and Tavaré (1995) propose a powerful population-genetic concept model of the *infinite-site coalescence*, which states all haplotypes in a population will coalesce to a single common ancestor, only one mutation can occur at any particular site and there is a rooted tree that describes the evolutionary history of a set of haplotypes in sampled individuals. Under the infinite-site coalescence assumption, we find that the evolutionary history of $2m$ haplotypes, one from each of $2m$ individuals, can be described as a tree with $2m$ leaves, where each of the n sites labels exactly one edge of the tree to point out the time during evolution where a mutation occurred at that site. That is, in more biological terms, a group of $2m$ haplotypes can be explained by a *perfect phylogeny* under the assumptions of no-recombination and infinite-sites model. In particular, given a $2m \times n$ binary matrix H , a perfect phylogeny for H is a rooted tree T with exactly $2m$ leaves satisfying the following properties:

1. Each of the $2m$ rows in H labels exactly one leaf of T .
2. Each of the n columns in H labels exactly one edge of T .
3. Every internal edge, i.e., an edge which does not connect to a leaf, of T is labeled by at least one column.
4. For every row i in H , the columns that label the edges along the unique path from the root to leaf i specify the columns of H that have value 1. That is, the path is a compact representation of row i .

Under the infinite-site coalescence model of haplotype evolution, the PHI problem is now much more constrained. Thus, Gusfield (2002) proposes the *perfect phylogeny haplotype* (PPH) problem defined as below.

Given a genotype matrix $G = \{g_1, g_2, \dots, g_m\}$, for each genotype g_i , we duplicate itself to create a pair of genotypes (g_i, g'_i) and the resulting matrix is denoted by \hat{G} . Then for each such pair (g_i, g'_i) and every column c where $g_{i,c} = g'_{i,c} = 2$, we are asked to set exactly one of those two elements to have the value 0 and the other to have the value 1, so that the resulting matrix H has a perfect phylogeny $T(G)$. The setting of values in H together with the associated perfect phylogeny $T(G)$ is a solution to the PPH problem, and the matrix H is a solution to the PHI problem.

Take Figure 2.2 as an example. Given a genotype matrix $G = \{22, 02, 10\}$ where $g_1 = 22$, $g_2 = 02$ and $g_3 = 10$, we can first duplicate each row in G to create the matrix \hat{G} and then assign the values of 2 in \hat{G} to produce the haplotype matrix H such that G has a perfect phylogeny $T(G)$. Then $T(G)$ is a solution to the PPH and PHI problem. In this example, we know that $g_1 = 10 \otimes 01$, $g_2 = 01 \otimes 00$, and $g_3 = 10 \otimes 10$ solve the PHI problem.

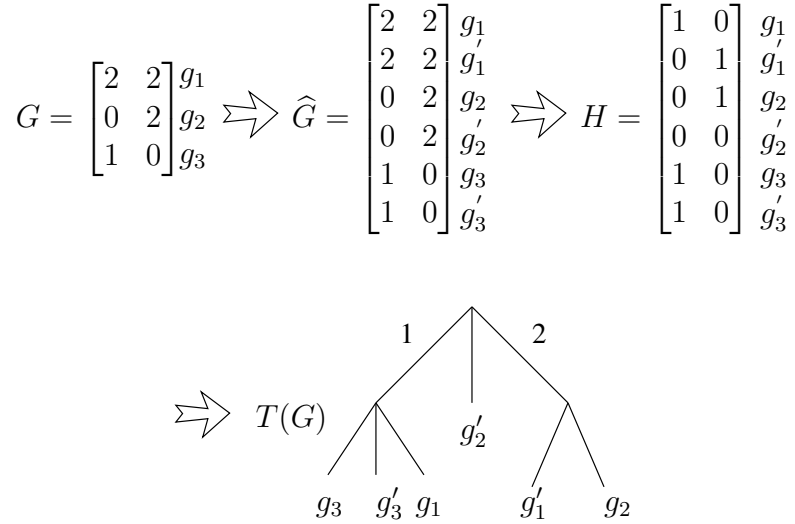


Figure 2.2: An example of PHI problem via perfect phylogeny

2.2.3 Pure parsimony

The possible pairs to resolve a genotype matrix are potentially exponential to the number of elements with value 2 that every genotype has. However, the number of distinct haplotypes that biologists find to exist in nature is very small, which leads to the pure parsimony criterion to the PHI problem. We call a PHI problem based on pure parsimony as an HIPP problem. One example to explain this phenomenon is the β_2 -Adrenergic receptors (β_2 ARs) gene. In particular, Drysdale et al. (2000) identify 13 SNPs in the human β_2 AR gene, which has $2^{13} = 8192$ possible combinations. However, they only find 10 haplotypes related to asthmatic cohort, far smaller than the number of all possible combinations.

The HIPP problem seeks the smallest number of distinct haplotypes to resolve a given genotype matrix. Take $G = \{202, 021, 212\}$ for example, where $g_1 = 202$, $g_2 = 021$, and $g_3 = 212$. Since there are two, one, and two 2s in g_1 , g_2 , and g_3 , respectively, we can resolve g_1 by either $p_1 = (000, 101)$ or $p_2 = (001, 100)$, resolve g_2 by $p_3 = (001, 011)$, and resolve g_3 by either $p_4 = (010, 111)$ or $p_5 = (011, 110)$. Thus we will have totally $C_1^2 C_1^1 C_1^2 = 4$ possible choices to resolve all the genotypes in G . There are 6, 5, 5, or 4 distinct haplotypes if we choose (p_1, p_3, p_4) , (p_1, p_3, p_5) , (p_2, p_3, p_4) , or (p_2, p_3, p_5) to resolve all the genotypes, respectively. Based on the pure parsimony criterion, we will choose p_2 , p_3 , and p_5 to resolve all genotypes since this combination has the smallest number of distinct haplotypes.

The HIPP problem is suggested by Hubbell, who also shows the problem is NP-hard. However, Hubbell's works have not been published. It is formally proven by Lancia et al. (2004) to be an APX-hard problem, which means, unless $P=NP$, there exists no δ -approximation algorithm for solving this problem, where δ is a constant greater than but close to 1. Lancia et al. (2004) also point out that determining the value of δ is

an interesting open problem. Huang et al. (2005) propose an $O(\log m)$ -approximation algorithm via integer quadratic programming and semidefinite programming relaxation to find an $O(\log m)$ -approximation solution of the optimal solution to a simplified HIPP problem in which they assume the total number of candidate haplotypes is bounded above by a polynomial function of the size of G .

Most of the solution methods to the HIPP problem exploit techniques of integer linear programming, integer quadratic programming, branch-and-bound, heuristics, and graph theory. We will survey these solution methods in the following sections.

2.2.3.1 Integer linear programming

Gusfield (2003) first enumerates all candidate haplotype pairs for each genotype, assigns binary variables for each candidate haplotype pair and for each distinct candidate haplotype, and then formulates an integer linear programming problem which seeks the minimum number of distinct candidate haplotypes to resolve all the genotypes.

In particular, let s_i denote the number of heterozygous sites in g_i . Thus there are totally 2^{s_i-1} possible distinct candidate haplotype pairs to resolve g_i . Let $(h_i^{\hat{k},a}, h_i^{\hat{k},b})$ denote the \hat{k} th pair possibly enumerated to resolve g_i where $\hat{k} \in [1, 2^{s_i-1}]$. Each possible explaining pair $(h_i^{\hat{k},a}, h_i^{\hat{k},b})$ will be enumerated and associated with a binary variable $y_{i,\hat{k}}$ to represent whether that pair appears (i.e. $y_{i,\hat{k}} = 1$) in the solution or not. Furthermore, another binary variable x_k is created for each distinct explaining haplotype h_k to denote whether that haplotype appears (i.e. $x_k = 1$) in the solution or not. Let Y_i and X denote the set of variables $y_{i,\hat{k}}$ and x , respectively. Using these variables, an integer linear programming formulation for the HIPP problem can be

described as follows:

$$\min \quad \sum_{x \in X} x \quad (2.3)$$

$$\text{s.t.} \quad y_{i,1} + y_{i,2} + \dots + y_{i,2^{s_i}-1} = 1, \quad \forall g_i \in G \quad (2.4)$$

$$y_{i,\widehat{k}} - x_a \leq 0, \quad y_{i,\widehat{k}} - x_b \leq 0, \quad \forall y_{i,\widehat{k}} \in Y_i, \text{ and } (h_i^{\widehat{k},a}, h_i^{\widehat{k},b}) = (h_a, h_b) \quad (2.5)$$

$$x_i, y_{i,\widehat{k}} \in \{0, 1\}, \quad \forall x_i \in X, i = 1, \dots, m, \widehat{k} = 1, \dots, s_i \quad (2.6)$$

Equation (2.4) ensures that g_i can be resolved by exactly one of its explaining pairs, and inequalities (2.5) force x_a and x_b to be 1, if $y_{i,\widehat{k}}$ is set to be 1. For instance, given a genotype matrix with $g_1 = 202$, $g_2 = 021$, and $g_3 = 212$, the formulation to this problem will be as follows:

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \\ \text{s.t.} \quad & y_{1,1} + y_{1,2} = 1 \\ & y_{2,1} = 1 \\ & y_{3,1} + y_{3,2} = 1 \\ & y_{1,1} - x_1 \leq 0, \quad y_{1,1} - x_2 \leq 0 \\ & y_{1,2} - x_3 \leq 0, \quad y_{1,2} - x_4 \leq 0 \\ & y_{2,1} - x_3 \leq 0, \quad y_{2,1} - x_5 \leq 0 \\ & y_{3,1} - x_6 \leq 0, \quad y_{3,1} - x_7 \leq 0 \\ & y_{3,2} - x_5 \leq 0, \quad y_{3,2} - x_8 \leq 0 \\ & x_i \in \{0, 1\}, \quad i = \{1, 2, \dots, 8\} \\ & y_{1,1}, y_{1,2}, y_{2,1}, y_{3,1}, y_{3,2} \in \{0, 1\} \end{aligned}$$

In this example, $x_1, x_2, x_3, x_4, x_5, x_6, x_7$ and x_8 represent whether the haplotypes 000, 101, 001, 100, 011, 010, 111 and 110 will appear in the optimal solution (i.e. $x = 1$)

or not (i.e. $x = 0$), respectively; and $y_{1,1}$, $y_{1,2}$, $y_{2,1}$, $y_{3,1}$, and $y_{3,2}$ represent whether the haplotype pairs (000, 101), (001, 100), (001, 011), (010, 111), and (011, 110) will be used in the optimal solution (i.e. $y = 1$) or not (i.e. $y = 0$).

Gusfield (2003) also proposes a problem size reduction mechanism to avoid expanding all possible haplotype pairs. The reduced ILP formulation is called RTIP. In particular, for each genotype that contains more than one candidate haplotype pair, RTIP only includes those pairs in which one of its constituted candidate haplotype is capable of resolving more than one genotype. In other words, a candidate haplotype pair whose constituted haplotypes only capable of resolving exactly one genotype will not be included in RTIP, unless necessary (e.g. when there is only one candidate haplotype pair). Such a candidate genotype pair can be ignored since it only increases the objective value.

Gusfield's approach leads to a potentially exponential-sized problem which may cause problems of memory shortage in practice. On the other hand, Brown and Harrower (2004) propose a polynomial-sized integer linear programming formulation to solve the same problem. They create variables to directly represent the two haplotypes for resolving each genotype without enumerating all the possible explaining pairs. In particular, for each site j at each genotype g_i , they assign two binary variables $\hat{y}_{2i-1,j}$ and $\hat{y}_{2i,j}$ to represent the value at site j for candidate haplotype pairs h_{2i-1} and h_{2i} . In particular, for each $g_{i,j}$ in a genotype matrix where $1 \leq i \leq m$ and $1 \leq j \leq n$, a constraint set of $O(mn)$ variables and constraints can be formulated as follows:

$$\hat{y}_{2i-1,j} = 0, \hat{y}_{2i,j} = 0, \text{ if } g_{i,j} = 0, \quad (2.7)$$

$$\hat{y}_{2i-1,j} = 1, \hat{y}_{2i,j} = 1, \text{ if } g_{i,j} = 1, \text{ and} \quad (2.8)$$

$$\hat{y}_{2i-1,j} + \hat{y}_{2i,j} = 1, \text{ if } g_{i,j} = 2. \quad (2.9)$$

Pairwise comparisons between haplotypes can define another set of constraints. In

particular, for each pair of different haplotypes, Brown and Harrower (2004) create a binary variable d_{i_1, i_2} , $1 \leq i_1 < i_2 \leq 2m$, and add constraints to force $d_{i_1, i_2} = 1$ if $h_{i_1} \neq h_{i_2}$, and $d_{i_1, i_2} = 0$ if $h_{i_1} = h_{i_2}$. Since $h_{i_1} \neq h_{i_2}$ whenever h_{i_1} differs from h_{i_2} in some site j , such a relation can be formulated as the following constraints for each candidate haplotype pair (i_1, i_2) and site j such that $1 \leq i_1 < i_2 \leq 2m$ and $1 \leq j \leq n$:

$$\hat{y}_{i_1, j} - \hat{y}_{i_2, j} \leq d_{i_1, i_2}, \quad (2.10)$$

$$\hat{y}_{i_2, j} - \hat{y}_{i_1, j} \leq d_{i_1, i_2}. \quad (2.11)$$

Therefore, these relations introduce $O(m^2)$ variables of d_{i_1, i_2} and $O(m^2n)$ constraints.

To count the number of distinct haplotypes which resolve the input genotypes, Brown and Harrower (2004) associate each haplotype h_i with a binary variable u_i to indicate the uniqueness of h_i in the haplotype sequence (h_1, \dots, h_i) where $1 \leq i \leq 2m$. In particular, $u_i = 1$ and $\sum_{\hat{i} < i} d_{\hat{i}, i} = i - 1$ if h_i is unique in (h_1, \dots, h_i) ; otherwise, $u_i = 0$ and $\sum_{\hat{i} < i} d_{\hat{i}, i} < i - 1$. Such a relationship further defines $O(m)$ variables of u_i and $O(m)$ constraints of the following format for $1 \leq i \leq 2m$:

$$u_i \geq 2 - i + \sum_{\hat{i}=1}^{i-1} d_{\hat{i}, i}. \quad (2.12)$$

In summary, the integer linear programming formulation proposed by Brown and

Harrower (2004) can be described as follows:

$$\min \quad \sum_{i=1}^{2m} u_i \quad (2.13)$$

$$\text{s.t.} \quad \widehat{y}_{2i-1,j} = 0 \text{ and } \widehat{y}_{2i,j} = 0, \quad \forall i, j \text{ such that } g_{i,j} = 0 \quad (2.14)$$

$$\widehat{y}_{2i-1,j} = 1 \text{ and } \widehat{y}_{2i,j} = 1, \quad \forall i, j \text{ such that } g_{i,j} = 1 \quad (2.15)$$

$$\widehat{y}_{2i-1,j} + \widehat{y}_{2i,j} = 1, \quad \forall i, j \text{ such that } g_{i,j} = 2 \quad (2.16)$$

$$d_{i_1,i_2} \geq \widehat{y}_{i_1,j} - \widehat{y}_{i_2,j}, \quad \forall 1 \leq i_1 < i_2 \leq 2m \text{ and } 1 \leq j \leq n \quad (2.17)$$

$$d_{i_1,i_2} \geq \widehat{y}_{i_2,j} - \widehat{y}_{i_1,j}, \quad \forall 1 \leq i_1 < i_2 \leq 2m \text{ and } 1 \leq j \leq n \quad (2.18)$$

$$u_i \geq 2 - i + \sum_{\widehat{i}=1}^{i-1} d_{\widehat{i},i}, \quad 1 \leq i \leq 2m \quad (2.19)$$

$$\text{all } u, y, \text{ and } d \in \{0, 1\}. \quad (2.20)$$

This ILP formulation, called PolyIP, has $\Theta(mn + m^2)$ variables and $\Theta(m^2n)$ constraints which are both polynomial in terms of size of G . Take $G = \{202, 021\}$ for example,

where $g_1 = 202$ and $g_2 = 021$, the formulation will be as follows:

$$\begin{aligned}
& \min \quad u_1 + u_2 + u_3 + u_4 \\
& \text{s.t.} \quad \widehat{y}_{1,1} + \widehat{y}_{2,1} = 1, \quad \widehat{y}_{1,2} = 0, \quad \widehat{y}_{2,2} = 0, \quad \widehat{y}_{1,3} + \widehat{y}_{2,3} = 1 \\
& \quad \widehat{y}_{3,1} = 0, \quad \widehat{y}_{4,1} = 0, \quad \widehat{y}_{3,2} + \widehat{y}_{4,2} = 1, \quad \widehat{y}_{3,3} = 1, \quad \widehat{y}_{4,3} = 1 \\
& \quad d_{1,2} \geq \widehat{y}_{1,1} - \widehat{y}_{2,1}, \quad d_{1,2} \geq \widehat{y}_{2,1} - \widehat{y}_{1,1} \\
& \quad d_{1,2} \geq \widehat{y}_{1,2} - \widehat{y}_{2,2}, \quad d_{1,2} \geq \widehat{y}_{2,2} - \widehat{y}_{1,2} \\
& \quad d_{1,2} \geq \widehat{y}_{1,3} - \widehat{y}_{2,3}, \quad d_{1,2} \geq \widehat{y}_{2,3} - \widehat{y}_{1,3} \\
& \quad d_{1,3} \geq \widehat{y}_{1,1} - \widehat{y}_{3,1}, \quad d_{1,3} \geq \widehat{y}_{3,1} - \widehat{y}_{1,1} \\
& \quad d_{1,3} \geq \widehat{y}_{1,2} - \widehat{y}_{3,2}, \quad d_{1,3} \geq \widehat{y}_{3,2} - \widehat{y}_{1,2} \\
& \quad d_{1,3} \geq \widehat{y}_{1,3} - \widehat{y}_{3,3}, \quad d_{1,3} \geq \widehat{y}_{3,3} - \widehat{y}_{1,3} \\
& \quad d_{1,4} \geq \widehat{y}_{1,1} - \widehat{y}_{4,1}, \quad d_{1,4} \geq \widehat{y}_{4,1} - \widehat{y}_{1,1} \\
& \quad d_{1,4} \geq \widehat{y}_{1,2} - \widehat{y}_{4,2}, \quad d_{1,4} \geq \widehat{y}_{4,2} - \widehat{y}_{1,2} \\
& \quad d_{1,4} \geq \widehat{y}_{1,3} - \widehat{y}_{4,3}, \quad d_{1,4} \geq \widehat{y}_{4,3} - \widehat{y}_{1,3} \\
& \quad d_{2,3} \geq \widehat{y}_{2,1} - \widehat{y}_{3,1}, \quad d_{2,3} \geq \widehat{y}_{3,1} - \widehat{y}_{2,1} \\
& \quad d_{2,3} \geq \widehat{y}_{2,2} - \widehat{y}_{3,2}, \quad d_{2,3} \geq \widehat{y}_{3,2} - \widehat{y}_{2,2} \\
& \quad d_{2,3} \geq \widehat{y}_{2,3} - \widehat{y}_{3,3}, \quad d_{2,3} \geq \widehat{y}_{3,3} - \widehat{y}_{2,3} \\
& \quad d_{2,4} \geq \widehat{y}_{2,1} - \widehat{y}_{4,1}, \quad d_{2,4} \geq \widehat{y}_{4,1} - \widehat{y}_{2,1} \\
& \quad d_{2,4} \geq \widehat{y}_{2,2} - \widehat{y}_{4,2}, \quad d_{2,4} \geq \widehat{y}_{4,2} - \widehat{y}_{2,2} \\
& \quad d_{2,4} \geq \widehat{y}_{2,3} - \widehat{y}_{4,3}, \quad d_{2,4} \geq \widehat{y}_{4,3} - \widehat{y}_{2,3} \\
& \quad u_1 \geq 1 \\
& \quad u_2 \geq d_{1,2} \\
& \quad u_3 \geq -1 + d_{1,3} + d_{2,3} \\
& \quad u_4 \geq -2 + d_{1,4} + d_{2,4} + d_{3,4}
\end{aligned}$$

Although PolyIP formulation has a size polynomial to the size of G , practically speaking it takes longer to solve the HIPP problem than the potentially exponential-sized formulation RTIP by Gusfield (2003). To improve the efficiency, they exploit a *branch-and-cut* strategy by constructing several valid constraints (cuts) to speed up finding the optimal solution. From their computational experiments, the PolyIP model usually requires more runtime than the RTIP model by Gusfield (2003), even when the branch-and-cut strategy is used. Nevertheless, their model can deal with some HIPP problems of larger size inapplicable for Gusfield’s RTIP model. Thus, there has to be a trade-off between the time and the memory for using these two ILP models. Recently, Brown and Harrower (2005) propose another hybrid formulation based on PolyIP and RTIP. Their new formulation first expands only a few candidate haplotype pairs, while other candidate haplotypes are to be expanded later, whenever necessary.

2.2.3.2 Quadratic integer programming approach

Huang et al. (2005) propose another formulation to the HIPP problem via a quadratic integer programming formulation (QIP). Similar to Gusfield (2003), they also first enumerate all the possible explaining haplotypes. To simplify the problem, they define the optimization haplotype inference (OHI) problem which is a special case of the general HIPP problem. In particular, OHI is assumed to have totally q candidate haplotypes where q is polynomial to the size of the genotype matrix. They show that OHI is NP-hard and can be formulated as a QIP.

In particular, Denote $H(G)$ to be the set of candidate haplotypes that can resolve G . Let $|H(G)| = q$. For each candidate haplotype h_k , $k = 1, \dots, q$, they associate it with an integer variable \hat{x}_k to indicate whether the k th candidate haplotype would appear in the solution (i.e. $\hat{x}_k = 1$, and thus $(1 + \hat{x}_k)^2 / 4 = 1$) or not (i.e. $\hat{x}_k = -1$, and thus $(1 + \hat{x}_k)^2 / 4 = 0$). For each genotype g_i , let $CHP(i) = \{(h_a, h_b) : g_i = h_a \otimes h_b,$

$\forall h_a, h_b \in H(G)\}$ denote the set of its candidate haplotype pairs. The OHI can be formulated as follows:

$$\min \quad \sum_{k=1}^q (1 + \hat{x}_k)^2 / 4 \quad (2.21)$$

$$\text{s.t.} \quad \sum_{(h_a, h_b) \in \text{CHP}(i)} (1 + \hat{x}_a) (1 + \hat{x}_b) / 4 \geq 1, \quad \forall 1 \leq i \leq m, \quad (2.22)$$

$$\hat{x}_k \in \{-1, 1\}, \quad 1 \leq k \leq q. \quad (2.23)$$

where the inequality (2.22) guarantees each genotype g_i to be resolved by at least one of its candidate haplotype pairs in $\text{CHP}(i)$, and the objective seeks the minimal number of candidate haplotypes that can resolve all the genotypes.

For instance, if the input genotype matrix is $G = \{202, 021, 212\}$, the formulation will be described as follows:

$$\begin{aligned} \min \quad & \frac{(1 + \hat{x}_1)^2}{4} + \frac{(1 + \hat{x}_2)^2}{4} + \frac{(1 + \hat{x}_3)^2}{4} + \frac{(1 + \hat{x}_4)^2}{4} + \frac{(1 + \hat{x}_5)^2}{4} + \frac{(1 + \hat{x}_6)^2}{4} \\ & + \frac{(1 + \hat{x}_7)^2}{4} + \frac{(1 + \hat{x}_8)^2}{4} \\ \text{s.t.} \quad & \frac{(1 + \hat{x}_1)(1 + \hat{x}_2)}{4} + \frac{(1 + \hat{x}_3)(1 + \hat{x}_4)}{4} \geq 1 \\ & \frac{(1 + \hat{x}_3)(1 + \hat{x}_5)}{4} \geq 1 \\ & \frac{(1 + \hat{x}_6)(1 + \hat{x}_7)}{4} + \frac{(1 + \hat{x}_5)(1 + \hat{x}_8)}{4} \geq 1 \\ & \hat{x}_k \in \{-1, 1\}, \quad 1 \leq k \leq 8. \end{aligned}$$

where $\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4, \hat{x}_5, \hat{x}_6, \hat{x}_7$ and \hat{x}_8 represent whether the haplotypes 000, 101, 001, 100, 011, 010, 111 and 110 appear in the solution (i.e. $\hat{x} = 1$) or not (i.e. $\hat{x} = -1$), respectively.

Solving a QIP is NP-complete, thus they solve its relaxation by an iterative semi-definite programming based approximation algorithm called SDPHapInfer to find an $O(\log m)$ -approximation solution, where m is the number of genotypes to be resolved.

They compare their algorithm with three other methods: HAPAR (see Wang and Xu, 2003), HAPLOTYPYER (see Niu et al., 2002), and PHASE (see Stephens et al., 2001) on both simulated data with size up to 24 genotypes and 10 SNPs generated from either 10 or 20 haplotypes. Their computational experiments show that SDPHapInfer has similar error rate performance with HAPLOTYPYER, which are both faster than HAPAR for inferring larger number of genotypes.

Another QIP solution method is proposed by Kalpakis and Namjoshi (2005). In particular, they define an *arithmetic genotype* δ_i for each genotype g_i , which is an n -dimensional vector with $\delta_{i,j} = 0, 1$ or 2 if $g_{i,j} = 0, 2$ or 1 , respectively, for $j = 1, 2, \dots, n$. The arithmetic genotype is beneficial since it can be easily obtained by $\delta_i = h_1 + h_2$ if $g_i = h_1 \otimes h_2$, where “+” is the standard vector addition.

For a given $m \times n$ genotype matrix $G = [g_{i,j}]$, let $\Delta = [\delta_{i,j}]$ be its associated $m \times n$ arithmetic genotype matrix, and $H = [h_i]$ be a $k \times n$ haplotype matrix with k haplotypes. If H resolves G , they show that $\Delta = S \cdot H$, where $S = [s_{i,j}]$ is an $m \times k$ *selector matrix* such that $\sum_{j=1}^n s_{i,j} = 2$ for each row $i = 1, \dots, m$, and element $s_{i,j} \in \{0, 1, 2\}$. Then, Kalpakis and Namjoshi (2005) define the k -HPP problem which seeks an $m \times k$ selector matrix S and an $k \times n$ haplotype matrix H such that $\Delta = S \cdot H$ and S has the smallest number of non-zero columns..

2.2.3.3 Branch-and-bound algorithm

Wang and Xu (2003) also use a *branch-and-bound* algorithm called HAPAR to find the optimal solution to the HIPP problem. Their branch-and-bound algorithm starts with an initial solution, searches all possible solutions and finds the best one. If the number of haplotypes of a partial solution is greater than or equal to the current minimum number of haplotypes (bound), their algorithm skips the bad subspace and moves to next possible good subspace. They compare their algorithm with four other

methods: HAPINFERX (see Clark, 1990), HAPLOTYPER (see Niu et al., 2002), Emdecoder (see Niu et al., 2002), and PHASE (see Stephens et al., 2001) on both biological and simulated data. They have tested two biological data: (1) the angiotensin converting enzyme (known as ACE) with size up to 11 genotypes and 52 SNPs generated from 13 haplotypes, and (2) the chromosome 5q31 studied by Daly et al. (2001) with size up to 20 genotypes and 12 SNPs generated from 9 haplotypes. They also tested five categories of simulated data: (1) random data sets with size up to 24 genotypes and 10 SNPs generated from 10 haplotypes, (2) random data sets with size up to 11 genotypes and 15 SNPs generated from 8 haplotypes, (3) the maize data set with size up to 10 genotypes and 17 SNPs generated from 4 haplotypes, (4) the coalescence-based model without recombination with size up to 40 genotypes and 10 SNPs generated from 80 haplotypes, and (5) the coalescence-based model with recombination with size up to 40 genotypes and 10 SNPs generated from 80 haplotypes. Their computational experiments show that under the coalescence model without recombination, PHASE (Stephens et al., 2001) has the best performance and HAPAR (Wang and Xu, 2003) performs the second; under the coalescence model with recombination, PHASE (Stephens et al., 2001) has the best performance when the number of genotypes m is small ($m < 25$), whereas HAPAR (Wang and Xu, 2003) is slightly better when the number of genotypes m is large ($m \geq 25$).

2.2.3.4 Heuristic algorithm

Since solving an ILP problem belongs to NP-hard class, the ILP and QIP based methods are usually time consuming. To obtain a good solution in short time, Li et al. (2005) propose a polynomial-time heuristic algorithm, called *Parsimonious Tree-Grow* (PTG), to solve the HIP problem. PTG resolves every column for a given genotype matrix G one by one consecutively. It tries to use fewer distinct haplotypes to resolve

a column, while keeping all genotype fragments resolved in growing a tree. That is, the i th iteration of PTG ensures the first $(i - 1)$ columns of G to be resolved. The parsimonious tree grows with the resolved genotype fragments until finally all columns are resolved. In particular, PTG first creates a root node $v_{0,1}$, adds indices $1, \dots, m$ to $v_{0,1}$, and sets $f(i) = false$ for all $i = 1, \dots, m$. The indices stored in a node v represents the genotypes remain to be resolved, and $f(i)$ records the first time when we observe 2 in g_i . After the i th iteration of PTG, let $G[1, i]$ represent the resolved submatrix composed by the first i columns in G , PTG then resolves the remaining submatrix column by column until finally $i = n$. The procedure of PTG is described in Algorithm 1.

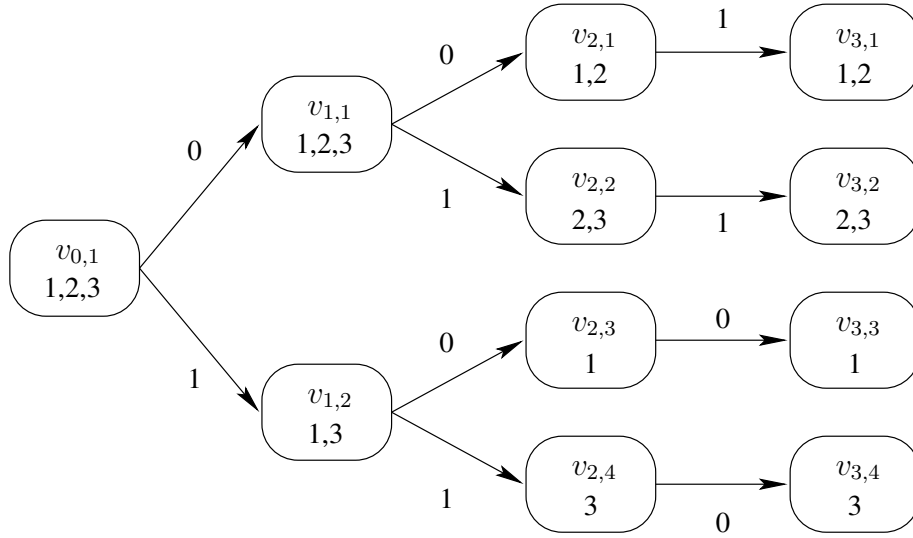


Figure 2.3: An example of PTG algorithm

Li et al. (2005) prove that PTG can solve the PHI problem in $O(m^2n)$ time for m given genotypes and n SNPs. Their computational experiments also show that PTG has excellent performance in both computational time and error rate. After PTG terminates, we can trace each path from $v_{0,1}$ to every leaf in the n th layer of the tree. The sequence of the branch type along each path represents the haplotype that resolves

Algorithm 1 Algorithm PTG

Initialization $j := 0$ **While** $(j + 1 < n)$ **do** **For** each node $v_{j,k}$ in the j th layer of the growing tree **For** each index i in the $v_{j,k}$ **If** $(g_{i,j+1} = 0)$ **then** **If** there is no branch of type 0 growing from $v_{j,k}$ **then** *Create branch of type 0* (i, j, k) **Else then** *Add index i to existing node* $0(i, j, k)$ **Else If** $(g_{i,j+1} = 1)$ **then** **If** there is no branch of type 1 growing from $v_{j,k}$ **then** *Create branch of type 1* (i, j, k) **Else then** *Add index i to existing node* $1(i, j, k)$ **Else If** $(g_{i,j+1} = 2 \text{ and } f(i) = \text{false})$ **then** **If** there is no branch growing from $v_{j,k}$ **then** *Create branch of type 0* (i, j, k) *Create branch of type 1* (i, j, k) **Else If** there are two branches growing from $v_{j,k}$ **then** *Add index i to existing node* $0(i, j, k)$ *Add index i to existing node* $1(i, j, k)$ **Else then** // there is just one branch growing from $v_{j,k}$ **If** there is a branch of type 0 growing from $v_{j,k}$ **then** *Add index i to existing node* $0(i, j, k)$ *Create branch of type 1* (i, j, k) **Else then** *Add index i to existing node* $1(i, j, k)$ *Create branch of type 0* (i, j, k) $f(i) = \text{true}$ **Else If** $(g_{i,j+1} = 2 \text{ and } f(i) = \text{true})$ **then** add index i into a list $I(j)$ add index k into a list $\hat{N}_{i,j}$ // $\hat{N}_{i,j}$ is a set of the j th layer nodes whose index set includes i **For** each index i in the list $I(j)$ *Decide branches* (i, j, k) $j := j + 1$ **End While**

Procedure: Initialization

create a root node $v_{0,1}$
add indices $1, \dots, m$ into $v_{0,1}$
For each $i = 1, \dots, m$
 $f(i) := false$

Procedure: Create branch of type 0(i,j,k)

$z :=$ the number of nodes in the j th layer $+1$
create a new node $v_{j+1,z}$
add a branch of type 0 growing from $v_{j,k}$
connect the new added branch of type 0 to $v_{j+1,z}$
add i to the index set of the new created node $v_{j+1,z}$

Procedure: Create branch of type 1(i,j,k)

$z :=$ the number of nodes in the j th layer $+1$
create a new node $v_{j+1,z}$
add a branch of type 1 growing from $v_{j,k}$
connect the new added branch of type 1 to $v_{j+1,z}$
add i to the index set of the new created node $v_{j+1,z}$

Procedure: Add index i to existing node 0(i,j,k)

$v_{j+1,z} :=$ the node connected to $v_{j,k}$ by existing type 0 branch
add i to the index set of $v_{j+1,z}$

Procedure: Add index i to existing node 1(i,j,k)

$v_{j+1,z} :=$ the node connected to $v_{j,k}$ by existing type 1 branch
add i to the index set of $v_{j+1,z}$

Procedure: Decide branches(i,j,k)

suppose the two nodes in $T_{i,j}$ are v_{jk_1} and v_{jk_2}

If there are no branches growing from node v_{jk_1} and v_{jk_2} **then**

add two different types of branches, one to node v_{jk_1} and the other to v_{jk_2}

Else If there are no two different types of branches growing separately from node v_{jk_1} and v_{jk_2} **then**

add a proper type of branch to node v_{jk_1} or v_{jk_2}

Else If there are two different types of branches growing separately from node v_{jk_1} and v_{jk_2} **then**

choose a pair of different type branches, one growing from v_{jk_1} , the other growing from v_{jk_2} randomly

add i into both node index sets of the $(j + 1)$ th layer connected to v_{jk_1} or v_{jk_2} by one of the chosen branches

the genotypes whose indices are stored in that leaf. Take Figure 2.3 for example which contains a parsimonious tree obtained by applying PTG to resolve $G = \{202, 021, 212\}$. The result shows that $g_1 = 001 \otimes 100$, $g_2 = 001 \otimes 011$, and $g_3 = 011 \otimes 110$.

Although PTG has good performance in practice, it can not guarantee to find the optimal solution to the HIPP problem. In this thesis, we shall propose some techniques in Chapter 5 to improve PTG for getting better solutions.

2.2.3.5 Graph theory

Davis and Holder (2003) and Blain et al. (2005) exploit certain technique from graph theory to solve the PHI problem. In particular, a special bipartite graph, called *diversity graph*, is used to solve the HIPP problem. A diversity graph can be defined as follows:

A bipartite graph $D = (V_H, V_G, E)$, where V_H and V_G are two vertex sets representing haplotypes and genotypes, respectively, and E is the edge set defined by the mating structure, is called a diversity graph if and only if the following conditions hold:

1. G is nonempty.
2. Each genotype in V_G is resolved by some haplotypes in V_H .

3. If $(h_1, g) \in E$, then there exists a $h_2 \in H$ such that $(h_2, g) \in E$ and $g = h_1 \otimes h_2$.

All the genotypes will be resolved, according to the second condition. The last condition ensures that if h_1 is used to resolve some genotype g , there must exist its conjugate haplotype h_2 that also resolves g . Davis and Holder (2003) prove that any bipartite graph can be extended and labeled to become a diversity graph by adding no more than a countable number of nodes. They also point out that if any haplotype is restricted to resolve at most two genotypes, the HIPP problem can be solved by decomposing that acyclic diversity graph into cycles and paths. Their algorithm for solving the restricted HIPP problems can be summarized below:

Algorithm 2 Decompose an acyclic diversity graph $D = (H, G, E)$ into paths

```

 $v := 0$ 
 $(H_v, G_v) = (H, G)$ 
Find the largest path  $P_v$  in  $(H_v, G_v)$ .
If no path exists then
     $P_v = \emptyset$ .
else
     $v := 1$ 
While  $(P_v \neq \emptyset)$  do
     $(H_{v+1}, G_{v+1}) = (H_v, G_v) \setminus P_v$ 
     $v := v + 1$ 
    Find the largest path  $P_v$  in  $(H_v, G_v)$ .
    If no path exists then
         $P_v = \emptyset$ .
End While

```

According to this algorithm, if the number of paths found by above algorithm is w , the fewest number of haplotypes needed to resolve G will be $m + w$, where m is the number of genotypes. However, this result will be invalid if every haplotype can be used to resolve more than two genotypes. They propose three interesting questions for future works. Let π denote the maximum number of genotypes that every haplotype can resolve and $\phi(\pi)$ be the function relating π to the number of haplotypes in a haplotype set which resolve G .

1. We could easily estimate the objective value of the HIPP problem if we have good lower bounds on $\phi(\pi) - \phi(\pi + 1)$. Hence, the first interesting open question is how fast $\phi(\pi)$ can grow.
2. Let π^* denote the minimum solution for $\phi(\pi)$. The second open question is to find bounds on π^* .
3. Finally, designing a randomized coloring algorithm to find the longest paths and cycles.

Since the algorithm can only be applied on the restricted HIPP problem, its practical efficiency and effectiveness remains to be evaluated.

2.3 Summary

Definitions, notations, methods, and mathematical formulations to the PHI problem are discussed in this Chapter. Two major categories of solution methods to the PHI problem: combinatorial type and statistical type, are surveyed. In the category of statistical methods, we review the EM algorithm and Bayesian method. There are three criteria to the combinatorial methods: Clark's inference rule, perfect phylogeny, and pure parsimony.

This thesis focus on solving the HIPP problem (i.e. the PHI problem based on pure parsimony criterion). Two ILP models: RTIP and PolyIP are summarized. RTIP has potentially exponential-sized formulation but is efficient in practice for resolving genotype matrix of small scale. On the other hand, although PolyIP has polynomial sized formulation, it usually consumes more time than RTIP.

Two QIP formulations are introduced, where techniques about approximation algorithms (e.g. SDPhapInfer) are required to give an approximated solution efficiently. HARPAR, a branch-and-bound method, is also introduced.

Heuristics are important in solving the PHI problems since the exact-solution methodologies based on ILP are not only theoretically difficult (APX-hard) but also time consuming in practice. PTG is an efficient tree-growing heuristic algorithm which always give a good solution in short time. Techniques based on graph theory provide another point of view to the PHI problem. However, the theoretically insights based on graph theory could not help to design an efficient algorithm for solving the general HIPP problem.

To help the reader have better ideas about these solution methods in the literature, their basic idea and computational experiments conducted are summarized in Table 2.2 as well as their availabilities in Table 2.3.

Algorithm	Source	Computational test case size	Methodology
Em-DeCODER	Niu et al., 2002	β_2 AR gene (18×12) ACE gene (11×52) CFTR gene (28×23) Simulated data (20×160 and 40×160) Compare with HAPINFERX (Clark, 1990) Haplotyper (Niu et al., 2002) PHASE (Stephens et al., 2001)	EM algorithm
HAPAR	Wang and Xu, 2003	ACE gene (11×52) Maize data (10×17) Simulated data (40×10) Compare with HAPINFERX (Clark, 1990) Haplotyper (Niu et al., 2002) PHASE (Stephens et al., 2001) Em-DeCODER (Niu et al., 2002)	Branch and bound
HAPINFERX	Clark, 1990	None	Clark's inference rule
Haplotyper	Niu et al., 2002	β_2 AR gene (18×12) ACE gene (11×52) CFTR gene (28×23) Simulated data (20×160 and 40×160) Compare with HAPINFERX (Clark, 1990) Em-DeCODER (Niu et al., 2002) PHASE (Stephens et al., 2001)	Bayesian method
PHASE	Stephens et al., 2001	Simulated data 50×100 Compare with HAPINFERX (Clark, 1990)	Pseudo-Bayesian method
PolyIP	Brown and Harrower, 2004	Simulated data 50×30 and 30×100 Compare with RTIP (Gusfield, 2003)	Integer linear programming
RTIP	Gusfield, 2003	Simulated data 50×30 and 30×100 Compare with PHASE (Stephens et al., 2001)	Integer linear programming
SDPHapInfer	Huang et al., 2005	β_2 AR gene (13×12) The cystic fibrosis data (13×23) Simulated data (300×10) Compare with HAPAR (Wang and Xu, 2003) Haplotyper (Niu et al., 2002) PHASE (Stephens et al., 2001)	Quadratic integer programming

Table 2.2: Summary of algorithms to the PHI problem

Algorithm	Algorithm availability
Em-DeCODER	http://www.people.fas.harvard.edu/~junliu/em/
HAPAR	http://theory.stanford.edu/~xuying/hapar/index.html
HAPINFERX	http://www.nslj-genetics.org/soft/hapinferx.f
Haplotyper	http://www.people.fas.harvard.edu/~junliu/Haplo/click.html
PHASE	http://depts.washington.edu/ventures/UW_Technology/Express.Licenses/PHASEv2.php
PolyIP	None
RTIP	None
SDPHapInfer	http://www.csie.ntu.edu.tw/~kmchao/tools/

Table 2.3: Summary of algorithm availabilities to the PHI problem

CHAPTER III

TWO HEURISTIC ALGORITHMS FOR SOLVING THE PHI PROBLEM BASED ON PURE PARSIMONY

In this Chapter, we first introduce the concept of compatible genotype pairs which helps to reduce the solution space of Gusfield's formulation (RTIP) and obtain good solutions faster. Then we propose two heuristic algorithms to solve the HIPP problem.

3.1 Compatible genotype pairs

Suppose ζ is a $1 \times n$ row vector whose elements are either 0, 1, or 2 (e.g. a genotype or a haplotype). We say two row vectors ζ_a and ζ_b are *compatible* to each other, denoted as $\zeta_a \sim \zeta_b$, if for each column $j = 1, \dots, n$ that $(\zeta_{a,j}, \zeta_{b,j}) \notin \{(0, 1), (1, 0)\}$. On the other hand, two row vectors ζ_a and ζ_b are *conflicting (incompatible)* to each other, if and only if there exists a column j such that $\zeta_{a,j} + \zeta_{b,j} = 1$. The compatible relation is reflexive and symmetric, but not transitive. For example, let $g_1 = 202$, $g_2 = 021$ and $g_3 = 212$. We know that $g_1 \sim g_1$, $g_2 \sim g_2$, $g_3 \sim g_3$, $g_1 \sim g_2$, $g_2 \sim g_1$, $g_2 \sim g_3$, and $g_3 \sim g_2$. However, $g_1 \sim g_3$ is not true even if $g_1 \sim g_2$ and $g_2 \sim g_3$ are both true.

A *compatible graph* G_c (see Figure 3.1) for a given genotype G can be constructed

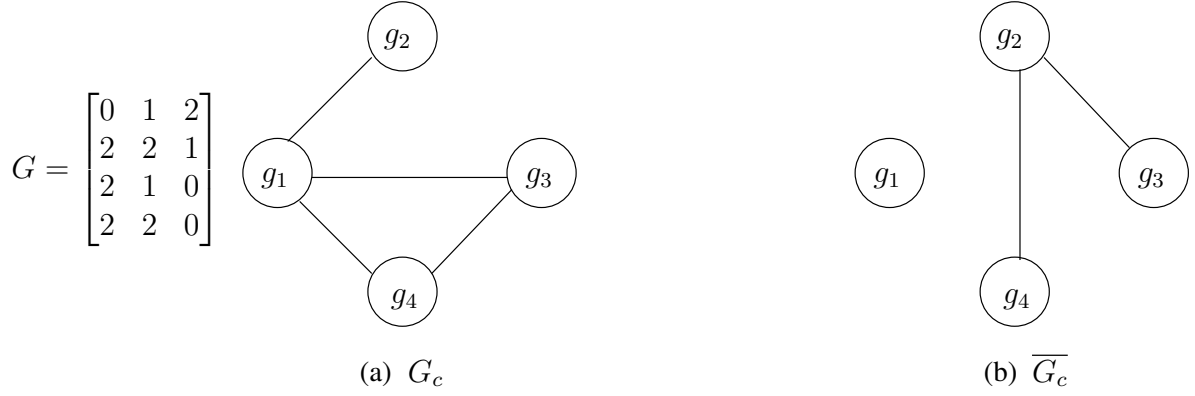


Figure 3.1: An example of compatible and incompatible graphs

by a set of m nodes and \tilde{q} arcs where each node corresponds to a genotype in G and each arc connects two compatible genotypes. It is clear that two incompatible genotype will not share common explaining haplotypes, which suggests an HIPP problem can be solved by solving several subproblems where each subproblem corresponds to a smaller HIPP problem whose genotypes form a component in G_c . Furthermore, compatible relations can help to reduce the solution space of the RTIP model proposed by Gusfield (2003). In particular, one only requires to enumerate those candidate haplotypes compatible with more than one genotype for the ILP formulation, since those incompatible candidate haplotypes will increase the objective value and thus will never appear in the optimal solution unless that genotype corresponds to an isolated node in G_c , in which case we can solve it separately as already described.

The compatible relations may provide a base for designing some intriguing heuristic algorithms that try to give good solutions for HIPP problems. For example, m and $2m$ are the trivial lower and upper bounds for Gusfield's formulation (RTIP), respectively. One may improve the upper bound by solving an arc covering problem on the compatible graph which seeks $A(G_c)$, the minimum number of arcs whose selection covers all the nodes. In particular, selecting an arc (a, b) can be thought as resolving

genotypes g_a and g_b using one of their common candidate haplotypes. Therefore, a selected arc will incur at most 3 haplotypes which provide a feasible solution that resolves all the genotypes with an estimated upper bound no more than $3A(G_c)$ on the optimal objective value. Similarly, one may construct a *conflicting graph* $\overline{G_c} = K_m \setminus G_c$ (see Figure 3.1) where K_m represents a complete graph of m nodes to estimate the lower bound in the objective value of RTIP. In particular, $\chi(\overline{G_c})$, the minimum number of node colors required for $\overline{G_c}$ such that adjacent nodes in $\overline{G_c}$ have different colors, provides an estimated lower bound $2\chi(\overline{G_c})$ in the objective value of RTIP, since nodes of the same colors may at least require 2 candidate haplotypes.

The estimated upper and lower bounds described in previous paragraph for the minimum number of candidate haplotypes for an HIPP problem may still be too loose. In next Section, we introduce a heuristic algorithm that seeks a good solution based on compatible relations.

3.2 The method of merged genotype pairs

Using the compatible genotype pair relations, we can identify common haplotypes for two genotypes to speed up enumerating all possible explaining pairs and reduce the solution space. First, we construct a *merged genotype pair*, denoted as $mg_{\tilde{k}}$, $\tilde{k} = 1, \dots, \tilde{q}$, for each pair of compatible genotypes g_a and g_b as follows:

1. the j th SNP site of $mg_{\tilde{k}}$, denoted as $mg_{\tilde{k},j}$, equals to $g_{a,j}$ or $g_{b,j}$, if $g_{a,j} = g_{b,j} \in \{0, 1, 2\}$, for $j = 1, \dots, n$.
2. $mg_{\tilde{k},j} = 1$ or 0 , if $(g_{a,j}, g_{b,j}) \in \{(1, 2), (2, 1)\}$ or $(g_{a,j}, g_{b,j}) \in \{(0, 2), (2, 0)\}$, respectively, for $j = 1, \dots, n$.

After conducting the pairwise comparisons, all the merged genotype pairs for a given genotype matrix G can be identified in $O(m^2n)$ time. Note that some merged

genotype pairs may be identical to each other. Without loss of generality, suppose there are totally \tilde{q} distinct merged genotype pairs, we denote the set of all merged genotype pairs as $MG(G) := \{mg_{\tilde{k}} : \tilde{k} = 1, \dots, \tilde{q}\}$, and define the set of merged genotypes covering each genotype g_i to be $IVMG(i) = \{mg_{\tilde{k}} : g_a \otimes g_i = mg_{\tilde{k}} \text{ or } g_i \otimes g_b = mg_{\tilde{k}}, \forall g_a \in G, g_b \in G\}$.

Here we give a heuristic algorithm, called as MGP, which is similar to the arc covering technique described in previous Section. In stead of selecting the minimum number of arcs (compatible relations) in a compatible graph G_c to cover all the nodes (genotypes), here we select the minimum number of merged pairs in a compatible graph G_c to cover all the nodes. Since a merged pair may contain more than one arc (compatible relation) in G_c , the objective value of MGP tends to be better than the objective value obtained by the arc covering heuristics proposed in previous Section. For each component \hat{c} in G_c , we may formulate an integer linear programming problem, denoted as $IP_{MGP1}(\hat{c})$, to select a set of minimal number of merged genotype pairs in G_c that covers all the nodes in component \hat{c} . Suppose these selected merged genotype pairs form a subgraph of G_c , denoted as $G_{mg\hat{c}}$. Then, we enumerate all the candidate haplotypes for each selected merged genotype pair, and formulate another integer linear programming problem, denoted as $IP_{MGP2}(\hat{c})$, which corresponds to Gusfield's formulation (RTIP) on the subgraph $G_{mg\hat{c}}$. Thus the optimal solution of $IP_{MGP2}(\hat{c})$ gives a set of minimum number of distinct haplotypes that resolves all the genotypes covered in complement \hat{c} from those candidate haplotypes that can resolve the selected merged genotype pairs defined in $G_{mg\hat{c}}$. Since $G_{mg\hat{c}}$ is a spanning subgraph of \hat{c} in G_c , in a sense MGP tries to compute a good feasible solution from a reduced solution space of the original problem. The final solution is obtained by the union of the solutions obtained for each complement in G_c .

The steps of MGP are described as follows:

- Step 1.** Preprocessing: conduct pairwise comparison to obtain $MG(G)$ and $IVMG(i)$ for each genotype g_i , $i = 1, \dots, m$.
- Step 2.** For each component \hat{c} in G_c , formulate $IP_{MGP1}(\hat{c})$, using its optimal solution to formulate another ILP $IP_{MGP2}(\hat{c})$
- Step 3.** Union all the solutions for each component to be the solution for the heuristic algorithm MGP.

Without loss of generality, suppose G_c only contains one complement \hat{c} . Assign a binary variable \tilde{x} for each merged genotype pair $mg_{\tilde{k}}$, $\tilde{k} = 1, \dots, \tilde{q}$, to represent whether $mg_{\tilde{k}}$ is selected (i.e. $\tilde{x}_{\tilde{k}} = 1$) to resolve a genotype that it covers or not (i.e. $\tilde{x}_{\tilde{k}} = 0$). The $IP_{MGP1}(\hat{c})$ can be formulated as follows:

$$\min \sum_{\tilde{k}=1}^{\tilde{q}} \tilde{x}_{\tilde{k}} \quad (3.1)$$


$$\text{s.t.} \quad \sum_{\tilde{k} \in \arg IVMG(i)} \tilde{x}_{\tilde{k}} \geq 1, \quad \forall i = 1, \dots, m, \quad (3.2)$$

$$\tilde{x}_{\tilde{k}} \in \{0, 1\}, \quad \forall \tilde{k} = 1, \dots, \tilde{q}. \quad (3.3)$$

The optimal solution for $IP_{MGP2}(\hat{c})$ can be used to form a solution space for ILP $IP_{MGP2}(\hat{c})$. In particular, we enumerate all the candidate haplotypes for the selected merged genotype pairs obtained from $IP_{MGP1}(\hat{c})$ which are used to solve $IP_{MGP2}(\hat{c})$. using Gusfield's RTIP formulation.

Take the genotype $G = \{012, 221, 210, 220\}$ in Figure 3.2 for example. After conducting $4(4 - 1)/2 = 6$ pairwise genotype comparisons, we obtain 4 compatible relations and 3 merged genotypes: $mg_1 = 011$, $mg_2 = 010$ and $mg_3 = 210$. The compatible graph for this example contains exactly one complement. Thus $IP_{MGP1}(\hat{c})$

 incompatible relation
  duplicated genotypes

 compatible relation with a merged genotype mg_i









	g_1	g_2	g_3	g_4	
g_1		mg_1	mg_2	mg_2	$mg_1 : 011$
g_2	mg_1				$mg_2 : 010$
g_3	mg_2			mg_3	$mg_3 : 210$
g_4	mg_2		mg_3		

Figure 3.2: An example of compatible genotype pairs

can be formulated as follows:

$$\begin{aligned}
 \min \quad & \tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3 \\
 \text{s.t.} \quad & \tilde{x}_1 + \tilde{x}_2 \geq 1 \\
 & \tilde{x}_1 \geq 1 \\
 & \tilde{x}_2 + \tilde{x}_3 \geq 1 \\
 & \tilde{x}_{\tilde{k}} \in \{0, 1\}, \quad \tilde{k} = \{1, 2, 3\}
 \end{aligned}$$

Note that the constraints for covering g_3 and g_4 are identical (i.e. $\tilde{x}_2 + \tilde{x}_3 \geq 1$) so that one of them can be further removed. There are two optimal solutions: $(\tilde{x}_1^*, \tilde{x}_2^*, \tilde{x}_3^*) \in \{(1, 1, 0), (1, 0, 1)\}$ with the optimal objective value equal to 2. The first optimal solution set of $IP_{MGP1}(\hat{c})$ shows that $\{mg_1, mg_2\} = \{011, 010\}$ can be directly used to derive five candidate haplotypes 010, 011, 101, 110, and 100 that resolve G . In particular, G can be uniquely resolved using $\{mg_1, mg_2\}$ since $g_1 = 010 \otimes 011$, $g_2 = 011 \otimes 101$, $g_3 = 010 \otimes 110$ and $g_4 = 010 \otimes 100$.

On the other hand, the second optimal solution set of $IP_{MGP1}(\widehat{c})$ shows that $\{mg_1, mg_3\} = \{011, 210\}$ can be further expanded to a set of three haplotypes $\{011, 010, 110\}$ which can be used to derive six candidate haplotypes 010, 011, 101, 110, 100, and 000 that resolve G . Let $\{p_1, p_2, p_3, p_4, p_5\}$ denote the set of candidate haplotype pairs $\{010 \otimes 011, 011 \otimes 101, 010 \otimes 110, 010 \otimes 100, 110 \otimes 000\}$ derived from those six candidate haplotypes. Note that in this case, g_1 , g_2 , and g_3 can be uniquely resolved by p_1 , p_2 , and p_3 , respectively, whereas g_4 can be resolved by either p_4 or p_5 . When some of the genotypes are resolvable by more than one haplotype pair derived from the optimal merged genotypes of $IP_{MGP1}(\widehat{c})$, we have to form the second ILP $IP_{MGP2}(\widehat{c})$ to decide the candidate genotype pairs that use fewer number of distinct haplotypes. The $IP_{MGP2}(\widehat{c})$ for this case is listed as follows:

$$\begin{aligned}
\min \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\
\text{s.t.} \quad & y_{1,1} = 1 \\
& y_{2,1} = 1 \\
& y_{3,1} = 1 \\
& y_{4,1} + y_{4,2} = 1 \\
& y_{1,1} - x_1 \leq 0, & y_{1,1} - x_2 \leq 0 \\
& y_{2,1} - x_2 \leq 0, & y_{2,1} - x_3 \leq 0 \\
& y_{3,1} - x_1 \leq 0, & y_{3,1} - x_4 \leq 0 \\
& y_{4,1} - x_1 \leq 0, & y_{4,1} - x_5 \leq 0 \\
& y_{4,2} - x_4 \leq 0, & y_{4,2} - x_6 \leq 0 \\
& x_i \in \{0, 1\}, & i = \{1, 2, \dots, 6\} \\
& y_{1,1}, y_{2,1}, y_{3,1}, y_{4,1}, y_{4,2} \in \{0, 1\}
\end{aligned}$$

where the binary variables x_1, x_2, x_3, x_4, x_5 , and x_6 represent whether the candidate haplotype 010, 011, 101, 110, 100, and 000 will be selected (i.e. $x = 1$) in the optimal solution or not (i.e. $x = 0$), and $y_{1,1}, y_{2,1}, y_{3,1}, y_{4,1}$, and $y_{4,2}$, represent whether the candidate haplotype pair p_1, p_2, p_3, p_4 , and p_5 is selected (i.e. $y = 1$) to resolve its corresponding genotype or not (i.e. $y = 0$).

The above example shows that our MGP algorithm does reduce the solution space which then solves a smaller RTIP (i.e. $IP_{MGP2}(\hat{c})$) to obtain a good solution. Although its solution guarantees to resolve all genotypes, it does not guarantee to give the optimal solution. Take $G = \{g_1, g_2, g_3, g_4\} = \{20, 10, 12, 22\}$ for example, there will be 6 compatible relations but only 3 merged genotypes $\{10, 20, 12\}$. The optimal solution of $IP_{MGP1}(\hat{c})$ will select the merged genotype 10 since it is compatible to all four genotypes. As a result, the optimal solution of $IP_{MGP1}(\hat{c})$ will select four candidate haplotypes $\{10, 00, 11, 01\}$. However, the optimal solution for the HIPP on G should contain only three candidate haplotypes $\{10, 00, 11\}$.

In summary, MGP is a heuristic algorithm that exploits the compatible relations of genotypes. A set of merged genotypes are produced by merging any two compatible genotypes. An ILP is formulated which seeks the minimum number of merged genotypes that can resolve all the genotypes. A second ILP based on Gusfield's RTIP formulation is then formulated using the candidate haplotypes for the selected merged genotypes to resolve all the genotypes. Since MGP can reduce the solution space of RITP, it is expected to be capable of solving a large-scale HIPP that would be difficult to solve if the original RITP is used. We have implemented MGP and evaluated its performance in Chapter 4.

3.3 The greedy heuristic inference method

Since the HIPP problem seeks the minimum number of haplotypes to resolve a given genotype matrix, we know that among all candidate haplotypes, one that can resolve more genotypes should be more likely to appear in the solution since its use reduces the required haplotypes. To select such haplotypes, we design our second greedy heuristic algorithm, called as GHI, by exploiting some weight mechanisms for all candidate haplotype and candidate haplotype pairs. Intuitively speaking, we give larger weights for a candidate haplotype compatible with more genotypes. In addition, based on the inference rule proposed by Clark (1990) which tries to infer heterozygotes (i.e. genotypes containing more than one ambiguous site) from homozygotes (i.e. genotypes with zero or one ambiguous site), we think those genotypes with fewer ambiguous sites should contain candidate haplotypes that are more likely to be used for resolving the genotype matrix. Therefore, the candidate haplotypes associated with a genotype with fewer ambiguous sites should be assigned with larger weights. Our algorithm GHI then tries to select candidate haplotype pairs having larger weights to resolve all the genotypes since such a choice tends to use fewer number of distinct candidate haplotypes.

Denote $H(G)$ and $HP(G)$ to be the set of candidate haplotypes and the set of candidate haplotype pairs that can resolve G , respectively. Let $|H(G)| = q$ and $|HP(G)| = \hat{q}$. For genotype g_i , we define $CH(i) = \{h_a : h_a \sim g_i, \forall h_a \in H(G)\}$ and $CHP(i) = \{(h_a, h_b) : g_i = h_a \otimes h_b, \forall h_a, h_b \in H(G)\}$ to be the set of candidate haplotypes and the set of candidate haplotype pairs that can resolve g_i , respectively. Therefore, $CH(i) \subset H(G)$, $CHP(i) \subset HP(G)$, $\bigcup_{i=1, \dots, m} CH(i) = H(G)$, and $\bigcup_{i=1, \dots, m} CHP(i) = HP(G)$. For each candidate haplotype h_k , $k = 1, \dots, q$, we define $IG(k) = \{g_i : h_k \sim g_i, \forall g_i \in G\}$ to be the set of genotypes compatible with h_k .

Let $ntwo(i)$ and $gw(i)$ denote the total number of ambiguous sites and the weights associated with genotype g_i for each $i = 1, \dots, m$. By setting $gw(i) = (\sum_{i=1}^m ntwo(i))/ntwo(i)$ for each $i = 1, \dots, m$, we assign a larger weight for a genotype with fewer sites with value 2. Then, we calculate the weight for each candidate haplotype by setting $hw(k) = \sum_{i \in \arg IG(k)} gw(i)$ for each $k = 1, \dots, q$. The weight for each candidate haplotype pair is set by the product of the weights of its constituted haplotypes. In particular, $hpw(\hat{k}) = hw(\hat{k}_a) \cdot hw(\hat{k}_b)$ for each $\hat{k} = 1, \dots, \hat{q}$, if the \hat{k} th candidate haplotype pair is composed by \hat{k}_a th and \hat{k}_b th candidate haplotypes.

This weight mechanism for each candidate haplotype pair are based on two intuitions: First, the $hw(\hat{k}_a)$ can be thought as the frequency used in the EM algorithm introduced in Section 2.1.1, where the idea of frequency is treated as a concept of probability. Therefore, the multiplication of the weights of the constituted haplotypes represents the probability for that candidate haplotype pair using those two specific constituted haplotypes. Similar to the EM algorithm which finds the haplotype probabilities optimizing the probability of the entire population, here we select the haplotype combinations with the largest estimated probability. Second, using the multiplication instead of addition to calculate the weight for each haplotype pair provides a better tie-breaking strategy for selecting haplotype pairs. For example, suppose a genotype g_i can be resolved by two haplotype pairs (h_a, h_b) and (h_c, h_d) , where $(hw(a), hw(b), hw(c), hw(d)) = (1, 9, 5, 5)$ using the universal genotype weight mechanism (i.e. $gw(i) = 1$ for $i = 1, \dots, m$). If the addition mechanism is used, these two pairs will be equally suitable since both $hpw(i) = 1 + 9 = 5 + 5$. On the other hand, the multiplication mechanism would prefer (h_c, h_d) rather than (h_a, h_b) since $5 \cdot 5 > 1 \cdot 9$. Although it is difficult to tell which mechanism is better, the weight mechanism used in this section has the best performance according to our computational experiments in Chapter 4.

Our GHI algorithm is straightforward and contains five major procedures:

Step 1. Construct $H(G)$, $HP(G)$ by $CH(i)$, $CHP(i)$ for each genotype $g_i \in G$; Construct $IG(k)$ for each candidate haplotype $h_k \in H(G)$.

Step 2. Calculate $ntwo(i)$ and $gw(i) = (\sum_{i=1}^m ntwo(i))/ntwo(i)$ for each $i = 1, \dots, m$.

Step 3. Calculate $hw(k) = \sum_{i \in \arg IG(k)} gw(i)$ for each $k = 1, \dots, q$.

Step 4. Calculate $hpw(\hat{k}) = hw(\hat{k}_a) \cdot hw(\hat{k}_b)$ where candidate haplotype pair $hp_{\hat{k}} = (h_{\hat{k}_a}, h_{\hat{k}_b})$ for each $\hat{k} = 1, \dots, \hat{q}$.

Step 5. Select the candidate haplotype $hp_{i^*} \in CHP(i)$ such that

$$i^* = \arg \max_{hp_{\hat{k}} \in CHP(i)} \{hpw(\hat{k})\} \text{ for each } i = 1, \dots, m.$$

$$\begin{aligned} g_1 = 202 & \Rightarrow p_1 = (000, 101), \text{ weight} = 2.5 \times 2.5 = 6.25 \\ & p_2 = (001, 100), \text{ weight} = 7.5 \times 2.5 = 18.75 \\ g_2 = 021 & \Rightarrow p_3 = (001, 011), \text{ weight} = 7.5 \times 7.5 = 56.25 \\ g_3 = 212 & \Rightarrow p_4 = (010, 111), \text{ weight} = 2.5 \times 2.5 = 6.25 \\ & p_5 = (011, 110), \text{ weight} = 7.5 \times 2.5 = 18.75 \end{aligned}$$

Figure 3.3: Illustration of haplotype weight calculation using our GHI algorithm

Take Figure 3.3 for example, if the input genotype matrix is $G = \{202, 021, 212\}$, three candidate haplotype pairs: one of p_1 and p_2 , p_3 itself, and one of p_4 and p_5 , have to be selected to resolve the genotype g_1 , g_2 , and g_3 , respectively. Since there are totally 5 elements with value 2 in G , $\{gw(1), gw(2), gw(3)\} = \{2.5, 5, 2.5\}$. The weights for haplotypes 000, 101, 001, 100, 011, 010, 111, and 110 are 2.5, 2.5, 7.5, 2.5, 7.5, 2.5, 2.5, and 2.5, respectively. The weights for candidate haplotype pairs p_1 , p_2 , p_3 , p_4 and p_5 become 6.25, 18.75, 56.25, 6.25, and 18.75, respectively, as illustrated in Figure 3.3. Then, GHI will select p_2 , p_3 , and p_5 to resolve g_1 , g_2 , and g_3 , respectively.

GHI is straightforward, easy to implement, and only requires a little more computational efforts after Step 1 which enumerates all the necessary candidate haplotype pairs as also required in algorithms by Gusfield (2003) and Huang et al. (2005). The techniques used in RTIP by Gusfield (2003) for enumerating the necessary rather than all candidate haplotype pairs can also be applied for Step 1.

Suppose there are totally q candidate haplotypes and $\hat{q} = \sum_{i=1}^m 2^{ntwo(i)-1}$ candidate haplotype pairs. The complexity analysis for GHI is discussed as follows:

1. The complexity of Step 1 depends on the data structure used for storing $H(G)$ and $HP(G)$. Suppose we store an haplotype using an array data structure where the information associated with haplotype h is stored in an array whose index equals to the decimal value of that haplotype. Then, it consumes $O(2^n)$ storage space. Using this data structure, Step 1 takes $O(\hat{q})$ time in $HP(G)$ and $O(q)$ time in $H(G)$. $IG(k)$ can be obtained in the process of constructing $H(G)$. Since $\hat{q} \leq q \leq 2\hat{q}$, we know Step 1 takes $O(\hat{q})$ (or, $O(q)$) time.

On the other hand, if we use linked lists to store $H(G)$ and $HP(G)$, an $O(\sum_{i=k}^q k) = O(q^2)$ time is required for each haplotype and an overall $O(2\hat{q} \sum_{i=1}^q i) = O(\hat{q}q^2) = O(\hat{q}^3)$ time is required for Step 1 with $O(q+\hat{q}) = O(\hat{q})$ storage space.

2. Calculating $ntwo(i)$ takes $O(n)$ time for each $i = 1, \dots, m$. $\sum_{i=1}^m ntwo(i)$ takes $O(m)$ time. $gw(i)$ takes $O(1)$ time for each $i = 1, \dots, m$. Therefore Step 2 takes $O(mn)$ time..
3. Calculating $hw(k)$ for each candidate haplotype $h_k, k = 1, \dots, q$, takes $O(|IG(k)|)$ time. Thus Step 3 takes $O(\sum_{k=1}^q |IG(k)|) = O(qm)$ time.
4. Calculate $hpw(\hat{k})$ for each candidate haplotype pair $hp_{\hat{k}}, \hat{k} = 1, \dots, \hat{q}$, takes $O(1)$ time. Thus Step 4 takes $O(\hat{q})$ time.

5. Selecting the candidate haplotype pair with the largest weight for each genotype g_i takes $O(|CHP(i)|)$ time, thus overall Step 5 takes $O(\sum_{i=1}^m |CHP(i)|) = O(\widehat{q})$ time.

In summary, GHI takes $O(\widehat{q} + mn + qm)$ time and $O(2^n)$ storage space using array to store $H(G)$ and $HP(G)$. Or, it takes $O(\widehat{q}^3 + mn + qm)$ time and $O(\widehat{q})$ storage space using linked list to store $H(G)$ and $HP(G)$. Since $\widehat{q} = \sum_{i=1}^m 2^{ntwo(i)-1}$, the efficiency of GHI depends very much in the total number of ambiguous sites that G contains. Note that the RTIP formulation by Gusfield (2003) and the SDPHapInfer algorithm by Huang et al. (2005) also require to enumerate all the necessary candidate haplotypes as the Step 1 of GHI, thus these three methods (i.e. GHI, RTIP, and SDPHapInfer) all consume similar time and storage space in the first step. Considering the computational efforts required after the first step, GHI only conducts simple calculation on weights whereas RTIP has to solve an ILP problem and SDPHapInfer has to solve a SDP (i.e. semi-definite programming) problem. Therefore GHI should consume less computational time than both RTIP and SDPHapInfer. Although GHI can not guarantee to compute an optimal solution for HIPP as does by RTIP and nor can GHI compute a solution with a guaranteed approximation like SDPHapInfer, it exploits ideas from Clark's rule and simple intuition to obtain good solutions efficiently. Its practical performance is shown to be good, especially for cases with recombination (see Chapter 4 for details).

3.4 Summary

The concept of compatible genotype pairs provides more insights about the property of HIPP problem which can be used to estimate the lower and upper bounds of the optimal objective value, as well as reducing the solution space for the HIPP problem

.Our first heuristic algorithm, MGP, exploits the idea of the compatible genotype pairs to construct the merged genotype pairs that guarantee to resolve all the genotypes. MGP first solves an ILP to select the minimum number of merged genotypes to resolve all the genotypes, and then formulate another reduced RTIP using the candidate haplotypes for the merged genotype pairs. MGP can cut down the solution space of the HIPP problem and obtain a good solution in a very short time. MGP has promising performance, according to our computational experiments in Chapter 4.

Our second heuristic algorithm, GHI, first enumerates all the necessary candidate haplotypes as also required in the RTIP formulation by Gusfield (2003) and the SD-PHapInfer algorithm by Huang et al. (2005). Then, GHI exploits the ideas from the inference rule proposed by Clark (1990) to give more opportunities for those candidate haplotypes that resolve the genotypes with fewer ambiguous sites to be selected in the solution. The weight assignment mechanisms of GHI are straightforward, intuitive, and easy to implement. The weight for each candidate haplotype pair in GHI are based on the ideas from the EM algorithm (see Excoffier and Slatkin (1995)). Different weight mechanisms have also been tested in our computational experiments and the results show that our GHI has good performance, especially for dealing with cases with recombination.

CHAPTER IV

COMPUTATIONAL EXPERIMENTS AND ANALYSES

This Chapter records the computational experiments for several PHI algorithms. After introducing the experimental settings, one biological genotype dataset and several simulated genotype datasets will be used for the experiments.

4.1 Settings for our computational experiments

In this Chapter, to evaluate the practical performance on several classes of the PHI algorithms, we conduct some computational experiments for those algorithms on a biological data and some simulated data. The *error rate*, defined as the proportion of genotypes whose inferred haplotype pairs are mismatched to their original ones, as well as the computational time are used to evaluate the effectiveness and efficiency for different algorithms. For those non-optimal algorithms, we also use the *optimality gap*, defined as the ratio of the difference in the number of selected inferred haplotypes obtained by an algorithm to the minimum number of inferred haplotypes (i.e. the optimal objective value of an HIPPP problem), as another evaluating parameter in our computational experiments.

Six algorithms are implemented and evaluated: (1) our merged genotype pair heuristic, denoted as “MGP”, is implemented in C++, compiled by Visual C++, and

linked with CPLEX 9.0 callable library; (2) our greedy heuristic, denoted as “GHI”, is implemented in C++ and compiled by g++ compiler; (3) the heuristic algorithm directly imported from Li et al. (2005), denoted as “PTG”, is implemented using Borland Delphi 5.0 in Pascal; (4) the semidefinite programming relaxation algorithm by Huang et al. (2005), denoted as “SDPHapInfer”, is implemented using Matlab; (5) the integer linear programming model by Gusfield (2003), denoted as “RTIP” is implemented in C++, compiled by Visual C++, and linked with CPLEX 9.0 callable library; and (6) the Clark inference rule algorithm directly imported from Clark (1990), denoted as “HAPINFERX”, is implemented using Fortran. All the computational experiments are conducted on a Pentium 4 PC with 3.2 GHz CPU, 1GB RAM and Windows XP operating system.

4.2 Experiments on β_2 AR gene

The human β_2 -adrenergic receptors (β_2 ARs) are G protein-coupled receptors that mediate the actions of catecholamines in multiple tissues. Drysdale et al. (2000) identify 18 different genotypes and 13 SNPs of a sample from 121 individuals in the human β_2 AR gene. They also find 10 haplotypes related to asthmatic cohort. The 18 haplotypes and 10 genotypes of β_2 AR gene are shown in Table 4.1 and a SNP site is omitted since it did not show any ambiguity in the sample included if we just consider the haplotypes whose frequencies are greater than or equal to 5% as described by Drysdale et al. (2000).

Among all algorithms we have tested, RTIP, MGP, GHI, and HAPINFERX all obtain 10 distinct haplotypes on human β_2 AR gene data and thus optimal with respect to the pure parsimony criterion. The error rate of MGP, GHI, and HAPINFERX are all 0. However, RTIP has the an error rate 6% due to the existence of multiple optimal solutions. Both PTG and SDPHapInfer conduct some mechanism of random

Table 4.1: Ten haplotypes and eighteen genotypes of human β_2 AR gene

	Haplotype	Genotype
h_1	100000010000	g_8, g_9
h_2	100111101000	$g_1, g_2, g_3, g_6, g_{10}, g_{11}, g_{12}, g_{13}$
h_3	011000010000	g_{11}, g_{14}
h_4	001000010000	$g_1, g_4, g_5, g_7, g_8, g_{14}, g_{15}, g_{16}, g_{17}$
h_5	001000000000	g_6, g_{15}
h_6	000000000101	g_3, g_5, g_9, g_{18}
h_7	000000000111	g_{12}, g_{16}, g_{18}
h_8	001000010101	g_{13}, g_{17}
h_9	000000000100	g_7
h_{10}	000000000000	g_{10}

selection which may result in different solutions even for the same input genotype matrix. Thus we apply SDPHapInfer and PTG to solve this genotype matrix for three times. SDPHapInfer obtains 12 haplotypes and has an error rate 11% in all the three experiments conducted. On the other hand, PTG obtains 10, 11, and 12 haplotypes, which correspond to the error rates of 0%, 17%, and 17%, respectively.

4.3 Experiments on simulated data

While the real genotype and haplotype data may be difficult to obtain, researchers have tried to simulate meaningful biological genotype and haplotypes for testing PHI algorithms. In particular, we use the program by Hudson (2002) to simulate a $2m \times n$ haplotype matrix, and then randomly pair two haplotypes from these $2m$ haplotypes to produce an $m \times n$ genotype matrix in a way that none of the $2m$ haplotypes will be repeatedly paired. Similar techniques to produce simulated genotypes can also be found in Stephens et al. (2001), Niu et al. (2002), Gusfield (2003), Wang and Xu (2003), Brown and Harrower (2004), and Huang et al. (2005).

Recombination is a process during the formation of gametes where portions from the paternal and maternal genome are exchanged (see Figure 4.1) to make one with a new combination of alleles. Recombination may cause the haplotypes in offspring

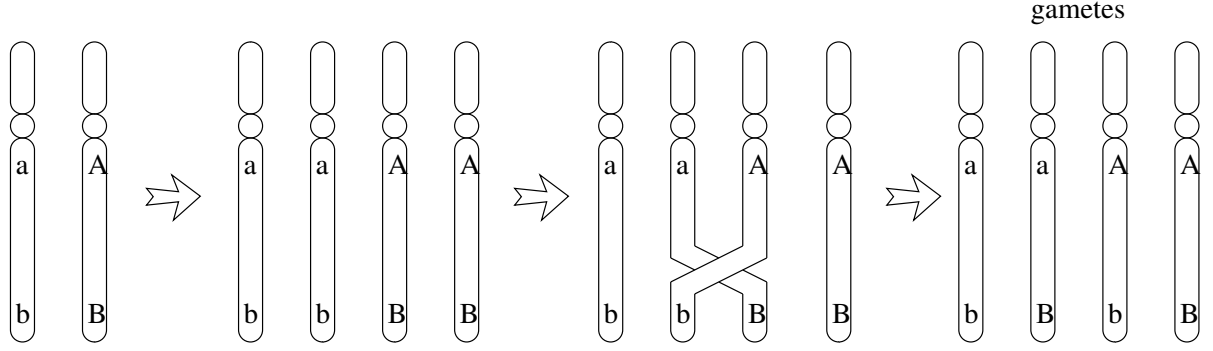


Figure 4.1: Illustration of recombination during the formation of gametes

to be different from those in their parents, and higher recombination rate cause more different haplotypes in offspring. We simulate the input genotypes and haplotypes with different recombination rate to evaluate the effectiveness for several PHI algorithms. In particular, we have tested three recombination rates: $r = 0$ (no recombination), $r = 4$ and $r = 16$.

For each different recombination rate (i.e. $r = 0, 4$, or 6), nine problem sets of different genotype matrix sizes (10×10 , 20×10 , 30×10 , 10×20 , 20×20 , 30×20 , 10×30 , 20×30 , and 30×30) are simulated, where 15 random test cases for each problem set have been generated. Three parameters: optimality gap, error rate, and computational time, for each algorithm in each test case are recorded for evaluation. For each parameter, we use their average on the 15 test cases of the same problem set to represent its overall performance. Note that we do not list the optimality gap for the algorithms RTIP and HAPINFERX. RTIP is already optimal and thus has zero optimality gap. HAPINFERX may not resolve all the genotypes which makes its optimality gap meaningless.

Table 4.2, Table 4.3, and Table 4.4 record the average performance in the optimality gap, error rate, and computational time, respectively, for different algorithms when there is no recombination (i.e. $r = 0$). The result of $r = 4$ and $r = 16$ are

Table 4.2: Optimality gap comparison (in %) for no recombination cases

Problem sets	Evaluated algorithms			
	MGP	GHI	PTG	SDPHapInfer
10×10	1	2	3	5
20×10	12	13	4	12
30×10	5	5	1	17
10×20	6	10	6	26
20×20	16	11	9	56
30×20	18	13	7	64
10×30	2	5	9	29
20×30	11	13	10	47
30×30	21	13	15	63

Table 4.3: Error rate comparison (in %) for no recombination cases

Problem sets	Evaluated algorithms					
	RTIP	MGP	GHI	PTG	HAPINFERX	SDPHapInfer
10×10	14	13	13	13	16	15
20×10	4	10	11	8	16	8
30×10	3	4	4	4	12	10
10×20	25	33	33	29	38	44
20×20	11	23	17	17	22	35
30×20	8	19	16	9	27	29
10×30	50	60	54	50	52	66
20×30	39	25	34	35	35	49
30×30	15	42	27	29	32	52

summarized in Table 4.5, Table 4.6, Table 4.7, Table 4.8, Table 4.9, and Table 4.10.

For the cases without recombination cases (i.e. $r = 0$), among those non-optimal algorithms, the optimality gap in Table 4.2 shows that PTG performs the best, GHI performs the second, MGP performs the third, whereas SDPHapInfer has smaller optimality gap for cases up to 10 SNPs but the gap increases dramatically for larger cases. In terms of error rate, RTIP has the smallest error rate than all other algorithms in all the cases without recombination (see Table 4.3), then PTG performs the second, GHI performs the third, MGP performs the fourth, HAPINFERX performs the fifth, and finally SDPHapInfer performs the sixth. Since most algorithms we have tested do not use the same programming language and compiler, the average runtime performance

Table 4.4: Computational time comparison (in seconds) for no recombination cases

Problem sets	Evaluated algorithms					
	RTIP	MGP	GHI	PTG	HAPINFERX	SDPHapInfer
10×10	0.07	0.06	0.02	0.01	0.03	0.87
20×10	0.07	0.05	0.04	0.01	0.03	10.14
30×10	0.09	0.09	0.06	0.01	0.04	6.81
10×20	0.88	0.07	0.57	0.01	0.03	5.32
20×20	72.66	0.08	62.32	0.02	0.04	3.74
30×20	6.00	0.26	5.06	0.02	0.06	9.68
10×30	6.08	0.12	2.67	0.02	0.05	10.28
20×30	11.05	0.08	3.00	0.03	0.04	2.00
30×30	7.78	0.12	3.05	0.04	0.04	3.99

Table 4.5: Optimality gap comparison (in %) for recombination rate $r = 4.0$ cases

Problem sets	Evaluated algorithms			
	MGP	GHI	PTG	SDPHapInfer
10×10	4	5	6	4
20×10	9	6	6	6
30×10	8	7	7	10
10×20	5	10	12	31
20×20	16	11	12	47
30×20	14	13	26	47
10×30	5	12	16	37
20×30	15	12	17	57
30×30	14	13	23	67

Table 4.6: Error rate comparison (in %) for recombination rate $r = 4.0$ cases

Problem sets	Evaluated algorithms					
	RTIP	MGP	GHI	PTG	HAPINFERX	SDPHapInfer
10×10	26	25	21	26	25	21
20×10	11	14	9	13	18	12
30×10	6	10	7	10	19	9
10×20	41	46	47	45	56	52
20×20	17	31	25	25	30	44
30×20	11	21	16	26	25	33
10×30	43	57	52	60	52	66
20×30	26	48	36	36	48	54
30×30	14	28	22	29	30	47

Table 4.7: Computational time comparison (in seconds) for recombination rate $r = 4.0$ cases

Problem sets	Evaluated algorithms					
	RTIP	MGP	GHI	PTG	HAPINFERX	SDPHapInfer
10×10	0.08	0.08	0.01	0.01	0.04	2.36
20×10	0.07	0.07	0.03	0.01	0.04	7.39
30×10	0.07	0.09	0.03	0.02	0.04	4.67
10×20	6.03	0.35	1.81	0.02	0.05	4.90
20×20	2.05	0.08	1.56	0.02	0.05	8.57
30×20	2.54	0.19	1.06	0.02	0.02	6.64
10×30	11.25	0.19	2.77	0.02	0.05	2.10
20×30	32.74	0.43	20.22	0.03	0.04	3.79
30×30	29.42	0.13	16.74	0.04	0.13	4.79

Table 4.8: Optimality gap comparison (in %) for recombination rate $r = 16.0$ cases

Problem sets	Evaluated algorithms			
	MGP	GHI	PTG	SDPHapInfer
10×10	7	6	15	10
20×10	18	9	22	12
30×10	19	13	24	19
10×20	4	14	25	29
20×20	8	15	24	45
30×20	12	12	23	40
10×30	3	13	20	30
20×30	8	14	26	48
30×30	14	14	27	57

Table 4.9: Error rate comparison (in %) for recombination rate $r = 16.0$ cases

Problem sets	Evaluated algorithms					
	RTIP	MGP	GHI	PTG	HAPINFERX	SDPHapInfer
10×10	23	30	25	35	29	33
20×10	23	33	26	37	33	26
30×10	18	24	18	28	28	24
10×20	45	52	52	71	66	64
20×20	24	33	30	42	34	50
30×20	21	32	26	43	32	41
10×30	57	56	57	63	58	71
20×30	33	47	39	52	45	59
30×30	27	46	34	42	39	59

Table 4.10: Computational time comparison (in seconds) for recombination rate $r = 16.0$ cases

Problem sets	Evaluated algorithms					
	RTIP	MGP	GHI	PTG	HAPINFERX	SDPHapInfer
10×10	0.08	0.09	0.01	0.01	0.03	3.54
20×10	0.07	0.05	0.02	0.01	0.05	7.59
30×10	0.10	0.05	0.05	0.02	0.05	27.69
10×20	0.19	0.11	0.08	0.01	0.03	4.43
20×20	0.82	0.10	0.35	0.03	0.04	6.69
30×20	1.77	0.27	1.69	0.04	0.04	9.25
10×30	49.62	4.34	4.14	0.02	0.04	3.34
20×30	16.60	0.14	15.05	0.04	0.03	8.73
30×30	19.06	0.25	7.72	0.06	0.03	16.35

listed in Table 4.4 can only be used for reference. However, in general we can still see that RTIP requires the longest runtime, and SDPHapInfer also takes more time than other algorithms.

For the cases with recombination cases (i.e. $r = 4$ or $r = 16$), among those non-optimal algorithms, the optimality gap in Table 4.5 and Table 4.8 shows that MGP performs the best, GHI performs slightly worse than MGP, PTG performs the third, whereas SDPHapInfer has smaller optimality gap for cases up to 10 SNPs but the gap increases dramatically for larger cases. In terms of error rate, RTIP again has the smallest error rate than all other algorithms in all the cases with recombination (see Table 4.6, and Table 4.9). For algorithm A and B, let $A \succ B$ represent A has smaller error rate than B, and $A \simeq B$ represent both A and B have similar error rates. In general, $GHI \succ MGP \simeq PTG \succ HAPINFERX \succ SDPHapInfer$ for the cases of $r = 4$, and $GHI \succ MGP \simeq HAPINFERX \succ PTG \succ SDPHapInfer$ for the cases of $r = 16$. From Table 4.3, Table 4.6, and Table 4.9, we find that the error rate for cases up to 10 genotypes (rows) are all large due to the lack of sufficient information to infer haplotypes (similar conclusion can also be found in Wang and Xu, 2003 and Huang et al., 2005).

4.4 Experiments on GHI using other weight mechanisms

Besides the original weight settings for each genotype, candidate haplotype, and candidate haplotype pair used in our GHI algorithm, we further design and test the following five weight mechanisms to see how different weight mechanism may affect the effectiveness of GHI algorithm.

1. GHI_1 : Similar to our default weight mechanism, except setting $gw(i) = 1 : i = 1, \dots, m$.
2. GHI_2 : Similar to the first weight mechanism, except setting $hpw(\hat{k}) = hw(\hat{k}_1) + hw(\hat{k}_2) : \hat{k} = 1, \dots, \hat{q}$, if the \hat{k} th candidate haplotype pair is composed by the \hat{k}_1 th and \hat{k}_2 th haplotypes, and there are totally \hat{q} distinct candidate haplotype pairs to resolve G .
3. GHI_3 : Similar to our default weight settings, except setting the weight for each genotype by equation (4.1). The idea of this mechanism is still to assign larger weight for those genotypes with fewer 2s in an incremental fashion. In particular, for the genotype containing the most 2s, it will have a weight equal to 1, whereas the genotype containing zero or one 2 will have a weight equal to m . All the other genotypes will have their weights in the range $[1, m]$, and $gw(i_1) \geq gw(i_2)$ if $ntwo(i_1) \leq ntwo(i_2)$.

$$gw(i) = \begin{cases} m - \frac{m-1}{\max_{i=1, \dots, m} ntwo(i) - 1} \times (ntwo(i) - 1), & \text{if } ntwo(i) \geq 1 \\ m, & \text{if } ntwo(i) = 0 \end{cases}, i = 1, \dots, m \quad (4.1)$$

4. GHI_4 : Similar to the third weight mechanism, except setting $hpw(\hat{k}) = hw(\hat{k}_1) + hw(\hat{k}_2) : \hat{k} = 1, \dots, \hat{q}$, if the \hat{k} th candidate haplotype pair is composed by

the \hat{k}_1 th and \hat{k}_2 th haplotypes, and there are totally \hat{q} distinct candidate haplotype pairs to resolve G .

5. GHI_5 : Similar to our default weight settings, except setting $hpw(\hat{k}) = hw(\hat{k}_1) + hw(\hat{k}_2) : \hat{k} = 1, \dots, \hat{q}$, if the \hat{k} th candidate haplotype pair is composed by the \hat{k}_1 th and \hat{k}_2 th haplotypes, and there are totally \hat{q} distinct candidate haplotype pairs to resolve G .

The comparisons on optimality gap, error rate, and computational time comparisons for these five weight mechanisms are listed in Table 4.11, Table 4.12, Table 4.13, Table 4.14, Table 4.15, Table 4.16, Table 4.17, Table 4.18, and Table 4.19, respectively. From these experimental results, we conclude our default weight settings give GHI better performance than other weight mechanisms. This indicates an integration of the biological knowledge does help to design a better heuristic algorithm in solving the HIPP problems.

Table 4.11: Optimality gap comparison (in %) on our GHI using several different weight mechanisms for no recombination cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	2	4	8	7	8	7
20×10	13	13	29	20	27	26
30×10	5	9	29	12	25	26
10×20	10	13	14	13	14	12
20×20	11	16	44	13	37	31
30×20	13	18	52	17	46	40
10×30	5	9	10	10	10	6
20×30	13	16	24	15	21	17
30×30	13	21	40	15	39	32

4.5 Summary

In this Chapter, we conduct several computational experiments to evaluate and compare the effectiveness (optimality gap and error rate) as well as the efficiency (com-

Table 4.12: Optimality gap comparison (in %) on our GHI using several different weight mechanisms for recombination rate $r = 4.0$ cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	5	5	9	10	9	8
20×10	6	10	21	8	20	20
30×10	7	8	30	13	27	28
10×20	10	11	13	10	12	11
20×20	11	16	31	11	28	28
30×20	13	13	33	14	31	29
10×30	12	14	13	14	13	12
20×30	12	18	28	14	24	22
30×30	13	17	37	13	32	29

Table 4.13: Optimality gap comparison (in %) on our GHI using several different weight mechanisms for recombination rate $r = 16.0$ cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	6	9	16	12	13	12
20×10	9	13	32	11	26	27
30×10	13	12	31	15	28	28
10×20	14	19	15	17	13	11
20×20	15	16	25	18	22	19
30×20	12	16	32	13	28	26
10×30	13	14	14	14	14	10
20×30	14	20	23	17	22	20
30×30	14	17	30	18	28	25

Table 4.14: Error rate comparison (in %) on our GHI using several different weight mechanisms for no recombination cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	13	15	19	16	18	17
20×10	11	10	22	12	19	18
30×10	4	6	17	7	15	15
10×20	33	39	49	37	46	42
20×20	17	23	46	18	40	33
30×20	16	19	38	16	35	32
10×30	54	61	72	59	68	63
20×30	34	41	57	37	51	41
30×30	27	35	59	28	53	46

Table 4.15: Error rate comparison (in %) on our GHI using several different weight mechanisms for recombination rate $r = 4.0$ cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	21	20	27	23	23	26
20×10	9	12	24	10	21	20
30×10	7	7	19	9	19	19
10×20	47	50	63	47	61	56
20×20	25	29	44	25	38	38
30×20	16	17	31	16	29	29
10×30	52	61	67	61	67	59
20×30	36	44	63	37	56	49
30×30	22	27	48	22	42	39

Table 4.16: Error rate comparison (in %) on our GHI using several different weight mechanisms for recombination rate $r = 16.0$ cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	25	30	40	27	33	32
20×10	26	27	45	26	39	40
30×10	18	17	27	16	26	28
10×20	52	61	67	58	63	59
20×20	30	32	46	31	42	39
30×20	26	28	51	27	47	44
10×30	57	65	70	65	70	62
20×30	39	46	58	42	53	51
30×30	34	34	55	35	52	48

Table 4.17: Computational time comparison (in seconds) on our GHI using several different weight mechanisms for no recombination cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	0.02	0.02	0.01	0.02	0.01	0.01
20×10	0.04	0.04	0.03	0.04	0.03	0.03
30×10	0.06	0.06	0.04	0.06	0.05	0.05
10×20	0.57	0.59	0.49	0.55	0.50	0.53
20×20	62.32	100.24	100.31	102.06	99.99	100.18
30×20	5.06	6.41	6.40	6.35	6.36	6.41
10×30	2.67	3.71	3.72	3.68	3.68	3.72
20×30	3.00	4.44	4.45	4.38	4.37	4.42
30×30	3.05	4.53	4.51	4.04	4.47	4.54

Table 4.18: Computational time comparison (in seconds) on our GHI using several different weight mechanisms for recombination rate $r = 4.0$ cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	0.01	0.01	0.01	0.02	0.01	0.02
20×10	0.03	0.03	0.03	0.04	0.03	0.04
30×10	0.03	0.03	0.03	0.04	0.04	0.04
10×20	1.81	1.79	1.79	3.78	3.74	1.80
20×20	1.56	1.57	1.55	1.55	1.55	1.56
30×20	1.06	1.05	1.01	1.05	1.04	1.05
10×30	2.77	2.76	2.73	2.77	2.77	2.76
20×30	20.22	21.95	22.22	20.11	20.12	21.94
30×30	16.74	16.65	16.60	16.63	16.61	16.63

Table 4.19: Computational time comparison (in seconds) on our GHI using several different weight mechanisms for recombination rate $r = 16.0$ cases

Problem sets	Evaluated algorithms					
	GHI	GHI_1	GHI_2	GHI_3	GHI_4	GHI_5
10×10	0.01	0.02	0.01	0.01	0.02	0.01
20×10	0.02	0.03	0.02	0.03	0.04	0.02
30×10	0.05	0.07	0.05	0.06	0.06	0.06
10×20	0.08	0.09	0.07	0.08	0.09	0.07
20×20	0.35	0.36	0.37	0.35	0.39	0.35
30×20	1.69	1.70	1.72	1.72	1.72	1.74
10×30	4.14	4.12	4.13	4.13	4.14	4.09
20×30	15.05	23.79	23.79	23.72	23.76	23.72
30×30	7.72	7.78	7.76	7.59	7.59	7.78

putational time) of six PHI algorithms on a real biology data set and nine simulated problem sets. We compare our two heuristic algorithms (MGP and GHI) with four other methodologies, including a parsimonious tree-growing heuristic (PTG), an integer quadratic programming approximation method (SDPHapInfer), a Clark inference rule method (HAPINFERX), and a linear integer programming model (RTIP).

For the real biological genotype case, β_2AR gene, MGP, GHI, HAPINFERX all get the optimal solution with zero error rate, whereas RTIP has positive error rate due to the existence of multiple optimal solution; PTG and SDPHapInfer may obtain the optimal solution by chance since their operations include randomized mechanisms.

For the simulated problem sets, our first heuristic algorithm (MGP) runs very fast and obtains promising performance in both optimality gap and error rate. It indicates that the intuition to use compatible genotypes does succeed in reducing the large solution space as required by RTIP. Our second heuristic algorithm (GHI) is straightforward, easy to implement, and capable of producing a solution with small error rate, small optimality gap in promising computational time. GHI outperforms all the other algorithms except RTIP in both the optimality gap and error rate for the cases with recombination (i.e. $r = 4$ and $r = 16$). On the other hand, for cases without recombination, PTG almost performs the best in all of the three aspects that we tested. These show that an intuitive implementation based on biological knowledge may in fact beat some complicated algorithms based on elegant mathematical proofs. SDPHapInfer is suitable for smaller cases with number of columns up to 10.

The computational experiments in inferring β_2AR gene also show that the existence of multiple optimal solutions may sometimes lead to larger error rates for some algorithms seeking the exact optimal solution. This also suggests the need for a better tie-breaking mechanism to improve the solution quality of many algorithms, including ours.

CHAPTER V

RELATED ISSUES IN SOLVING THE HIPPPROBLEM

In this Chapter, we first give a divide-and-concur technique which partitions a large-scale genotype matrix into several submatrices of the same number of rows, solves the HIPPPROBLEM for each submatrix and then combine the solution columnwisely. Then we improve the PTG a heuristic algorithm proposed by Li et al. (2005) to get a better solution.

5.1 Techniques for solving large-scale HIPPPROBLEMS

When the scale of a given genotype matrix G is large (e.g. $n \geq 100$), most of the proposed PHI algorithms such as RTIP, SDPHapInfer, and our GHI will encounter the problem of limited memory to store a huge number of candidate haplotypes. Taking the RTIP implemented using C/C++ as an example, since the largest value expressible in C/C++ is $2^{32} - 1$, which means we may not be able to store all the candidate haplotypes if G contains more than 32 columns (SNPs).

Such a large-scale HIPPPROBLEM can be solved using a divide-and-concur procedure. Let $G[i : j]$ represents the column collections from the i th through the j th columns in a given genotype matrix G . To deal with large scale input matrix G , first we can columnwisely partition G into several submatrices. In particular, we may

partition G into s segments such that $G = \{G_1, G_2, \dots, G_s\}$, where $G_k = G[i_k, j_k]$, $i_k = j_{k-1} + 1$, $j_0 = 0$, $k = 1, 2, \dots, s$. Then, we can use any haplotype inference algorithms to resolve each genotype submatrix separately, and finally concatenate all the segments of resolved haplotype pairs to obtain a solution. Note that such a solution may not be optimal for the HIPD problem, but it does guarantee to resolve all the genotypes.

For example, if the input genotype matrix is

$$G = \begin{bmatrix} 0211022012 \\ 2201212100 \\ 2211122012 \end{bmatrix},$$

We may partition G into two submatrices, G_1 and G_2 , as follows:

$$G_1 = \begin{bmatrix} 02110 \\ 22012 \\ 22111 \end{bmatrix} \text{ and } G_2 = \begin{bmatrix} 22012 \\ 12100 \\ 22012 \end{bmatrix}$$

Using the RTIP formulations by Gusfield (2003), we can obtain the resolved haplotype submatrices H_1 and H_2 , and thus H as follows:

$$H_1 = \begin{bmatrix} 00110 \\ 01110 \\ 00010 \\ 11011 \\ 00111 \\ 11111 \end{bmatrix}, H_2 = \begin{bmatrix} 00010 \\ 11011 \\ 10100 \\ 11100 \\ 00010 \\ 11011 \end{bmatrix}, \text{ and } H = \begin{bmatrix} 0011000010 \\ 0111011011 \\ 0001010100 \\ 1101111100 \\ 0011100010 \\ 1111111011 \end{bmatrix}$$

This divide-and-concur procedure is intuitive, but it does not guarantee to give an optimal solution for the original HIPD problem. Several reasons may influence the

solution quality towards the pure parsimony criterion, when such a divide-and-concur procedure is used. First, each submatrix may contain multiple optimal solutions and different optimal solutions may give a very different combination. Second, when we merge two submatrix, there are multiple qualified haplotype combinations for resolving the same genotype matrix. Using the $g_1 = 0211022012$ in the 3×10 genotype matrix introduced in this Section for example, we may have 2 different combinations of haplotype pairs: $0011000010 \otimes 0111011011$, or $0011011011 \otimes 0111000010$ by permuting each candidate haplotype pairs for each haplotype segments. In particular, if we have s partitions on m genotypes, we will have at most $(2^m)^{s-1}$ possible inferred haplotype matrices, since one merge produce 2 outcomes for each genotype and there are totally $s - 1$ times of merges. Third, different ways of partition may result in different solutions.

Solving large-scale HIPP problems is still a hard open problem. To the best of our knowledge, the PTG heuristic algorithm by Li et al. (2005) is the most suitable PHI algorithm in the literature for solving large-scale HIPP problems.

5.2 Improving the PTG heuristic algorithm

Though the PTG algorithm proposed by Li et al. (2005) performs well among all methodologies we have compared (see Section 4.2 for details), it grows the parsimonious tree in a column-by-column manner with minimal branching principle which means it always finds the local optima during tree-growing process. Since the local optima may not lead to the global optima, here we propose a method called "ImprovedPTG" by improving the PTG algorithm to find all the multiple global optimal solutions.

Our improved PTG algorithm contains four procedures:

Step 1. We first conduct the pairwise comparisons among all the m rows of a given

genotype matrix G and create a pairwise comparison matrix to record the compatible relations for any two genotypes. Figure 5.1 provides an example where g_1 has common haplotypes with g_2 , g_3 and g_4 and g_3 has common haplotypes with g_4 .

Step 2. Whenever we meet the site with value 2 during each tree-growing process, we will expand all possible combinations and constructing their corresponding trees. Suppose the given genotype matrix $G = \{012, 221, 210, 220\}$, then Figure 5.2, Figure 5.3, Figure 5.4 and Figure 5.5 illustrate the procedures of expanding and constructing new trees.

Step 3. For each expanded tree, we use the pairwise comparison matrix obtained in Step 1 to decide whether to cease its growth or not. Take Figure 5.2 and Figure 5.3 for example, since the only genotype compatible to g_2 is g_1 , but g_1 and g_2 are not in the same branch in these two trees, we will stop growing these trees.

Step 4. Repeat the Step 2 and 3 until all columns are resolved. The final global optimal solutions for $G = \{012, 221, 210, 220\}$ using our improved PTG algorithm contains 5 distinct haplotypes and are shown in Figure 5.4 and Figure 5.5.

Our improved PTG algorithm is more or less a method of exhaustive enumeration, whose complexity is difficult to analyze.

5.3 Summary

Among those issues related with PHI problem, we discuss techniques in solving large-scale PHI problems and improvement for the PTG heuristic algorithm. Large-scale PHI problems are considered to be very difficult since most of the PHI algorithms

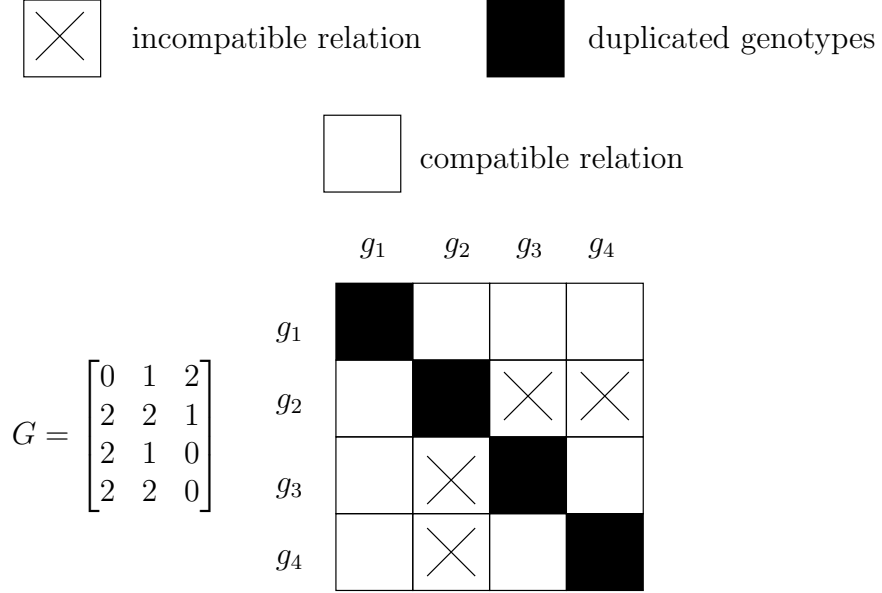


Figure 5.1: An example of genotype pairwise comparisons

Stop to grow this tree, since genotype 2 and genotype 1 does not have common haplotype

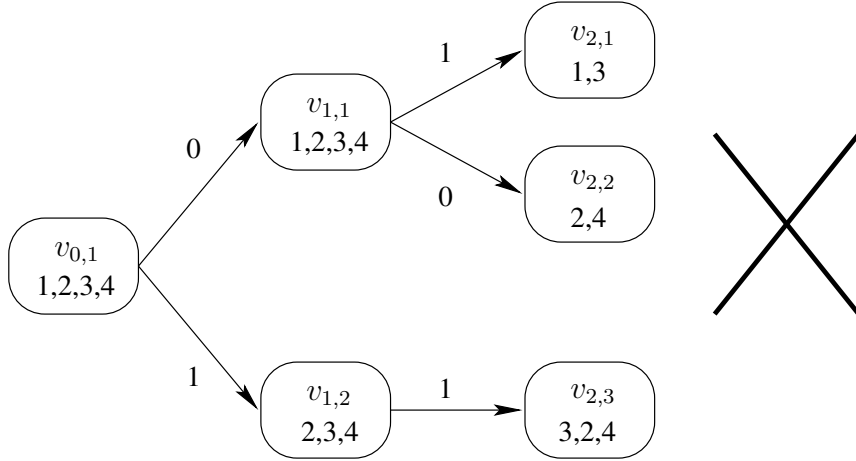


Figure 5.2: An example of ImprovedPTG - the first tree

require to enumerate all the candidate haplotypes and are suffered by the problem of insufficient memory storage. We provide an intuitive divide-and-concur technique which divides the genotype matrix G into several submatrices of the same number of rows, iteratively solves each submatrix using any PHI algorithm, and then merges the resolved haplotype submatrices into a complete resolved haplotype matrix H . We have

Stop to grow this tree, since genotype 2 and genotype 1 does not have common haplotype

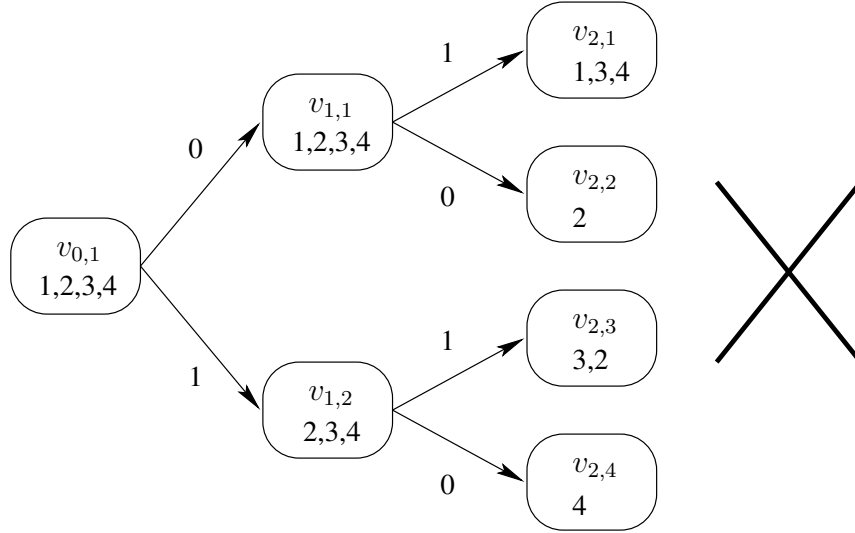


Figure 5.3: An example of ImprovedPTG - the second tree

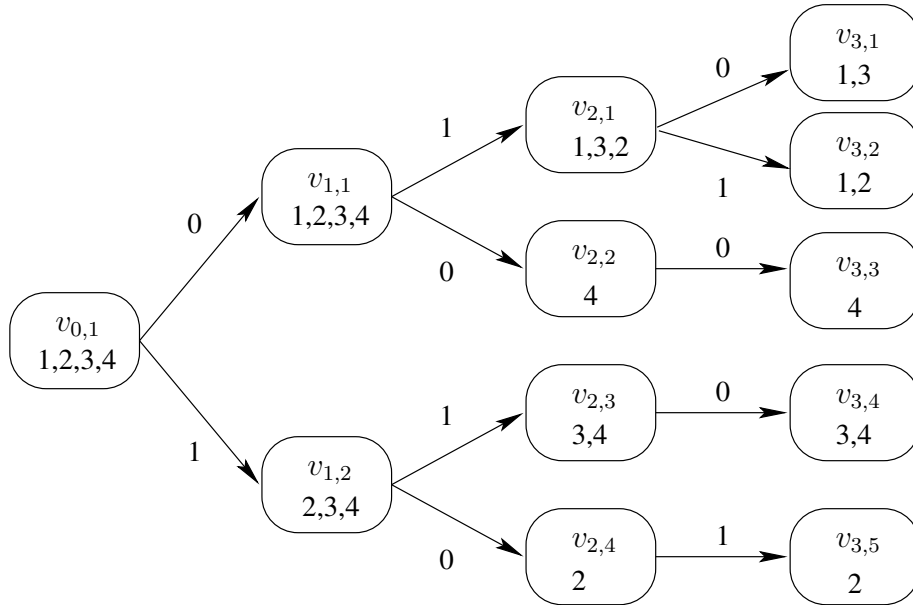


Figure 5.4: An example of ImprovedPTG - the third tree

observed that such a divide-and-concur method would not necessarily give an optimal solution, and give reasons to explain how the number of partitions might affect the quality of the solution obtained.

According to our computational experiment in Chapter 4, PTG can give a good

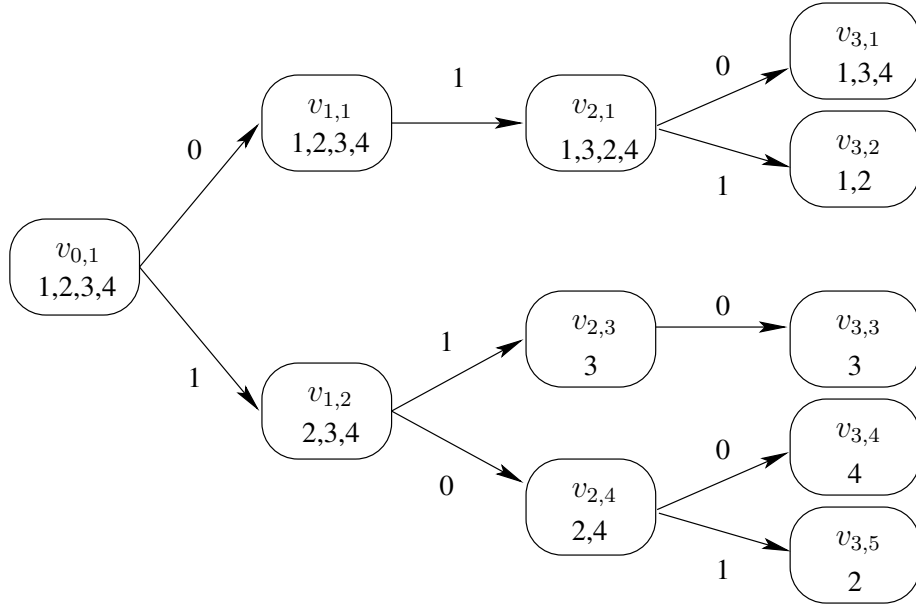


Figure 5.5: An example of ImprovedPTG - the fourth tree

solution in a very short time, which gives it more room in exploring more possibilities to improve its effectiveness. We modify PTG to consider more branches of opportunities which lead to the multiple global optimal solutions. Our technique can be viewed as an efficient way of exhaustive enumeration. Its practical efficiency remains to be evaluated.

CHAPTER VI

CONCLUSION

Population Haplotype Inference (PHI) problem which infers haplotypes from genotypes for a group of person is important since we have known more about the structure and sequence, but not yet the function of human DNA and the function of DNA may be a key for human disease. This thesis focuses on issues in solving the PHI problem, and more specifically, the PHI problem based on pure parsimony criterion (HIPP).

The major contributions of this thesis are in two folds. First, we propose two new heuristic algorithms (MGP and GHI) based on the concept of compatible genotype pairs. Approximated lower and upper bounds for the objective value of the HIPP problem can also be derived from the compatible graph constructed from those compatible genotype pairs. Second, we conduct extensive computational experiments for six PHI algorithms on one biological dataset and several simulated datasets, and three parameters (optimality gap, error rate, and computational time) are recorded and compared. Our experiments are more complete than others in the literature in the following two senses: first, since most heuristic algorithms in the literature do not compare with the optimal algorithms but we compare with all of them; second, most heuristic algorithms only use error rates as their performance index; on the other hand, most optimal algorithms compare their computational time; here in this thesis, we use all of them and

introduce the optimality gap as another performance index for evaluation.

Other minor contributions of this thesis includes: (1) a comprehensive survey on the state-of-the-art progress in solving the PHI problem, (2) techniques as well as their weakness in solving large-scale PHI problems, and (3) improvement for the PTG heuristic algorithm to obtain all the global optimal solutions of the HIPP problem.

With our limited experience, we suggest the following topics for future research in the related fields:

1. Modification of our heuristic algorithm in weight assignment mechanism:

Since different weight assignment mechanism of our heuristic algorithm leads to different result, one might further improve our weight assignment mechanism to reduce both error rate and the optimality gap. More knowledge in biology may be required to design more reasonable and better heuristics.

2. Better solution methods for the large-scale HIPP problem:

The divide-and-concur technique we used in Chapter 5 is intuitive but comes with many possible disadvantages. Since different partition may lead to different result, a preprocess that can suggest a better partition strategy after probing the structure of genotype matrix should be helpful. Furthermore, from our analysis, there are so many possible combination of the solutions for all the genotype segments, how to design a better and reasonable combination decision is also a very challenging problem.

3. Mathematical structure of the HIPP problem:

There are other interesting mathematical properties related with the structure of the HIPP problem. For example, what if G contains duplicated columns or rows? Is an entire column or row of value 2 in G removable

without affecting the optimal solution? What are the key factors (m , n , or total number of sites with value 2) to the difficulty of the HIPP problem and how does it affect the complexity of the solution methods? Answers to these questions would help us understand more on the mathematical structure of the HIPP problem and therefore help design more efficient and effective solution methods.

4. Tie-breaking mechanisms:

Many solution methods to the HIPP problem would require a good tie-breaking mechanism in their procedures. Our GHI heuristic provides several tie-breaking mechanisms which may be helpful for other heuristics. In general, it is difficult to design a good tie-breaking mechanism to achieve a better error rate. Techniques aiming at improving the error rate should require more inputs in the background knowledge in biology.

5. Mathematical programming formulations:

The RTIP formulation by Gusfield (2003) is straightforward, but will be suffered by the limited memory storage, when the number of ambiguous sites is large. The PolyIP formulation by Brown and Harrower (2004) is practically not efficient. The semi-definite programming formulation by Huang et al. (2005) and Kalpakis and Namjoshi (2005) are only solved approximately using approximation algorithms. All of these mathematical programming based models usually consume a lot of computational time. It would be a challenging research topic to propose a better mathematical programming formulation. To this end, we suggest to take a closer look in the compatible graph as proposed in Chapter 3. Our MGP heuristic is just a quick start and we think the compatible relations should help design better mathematical

formulations.

REFERENCES

- Blain, P., Holder, A., Silva, J. and Vinzant, C. 2005. *Mathematical approaches to the pure parsimony problem* (Tech. Rep.). Trinity University, Mathematics Technical Report S40, San Antonio.
- Brown, D. G. and Harrower, I. M. 2004. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In I. Jonassen and J. Kim (Eds.), *Algorithms in Bioinformatics: 4th international workshop, WABI 2004* (pp. 254–265). Springer Berlin / Heidelberg.
- Brown, D. G. and Harrower, I. M. 2005. *A new formulation for haplotype inference by pure parsimony* (Tech. Rep.). School of Computer Science, University of Waterloo.
- Clark, A. G. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution*, **7**(2), 111–122, 1990.
- Clark, A. G., Weiss, K. M., Nickerson, D. A., Taylor, S. L., Buchanan, A., Stengard, J., Salomaa, V., Vartiainen, E., Perola, M., Boerwinkle, E. and Sing, C. F. Haplotype structure and population genetic inferences from nucleotide-sequence variation in human lipoprotein lipase. *American Journal of Human Genetics*, **63**(2), 595–612, 1998.
- Daly, M. J., Rioux, J. D., Schaffner, S. F., Hudson, T. J. and Lander, E. S. High-

- resolution haplotype structure in the human genome. *Nature Genetics*, **29**(2), 229–232, 2001.
- Davis, C. and Holder, A. 2003. *Haplotyping and minimum diversity graphs* (Tech. Rep.). Trinity University, Mathematics Technical Report S73, San Antonio.
- Donnelly, P. and Tavaré, S. Coalescents and genealogical structure under neutrality. *Annual Review of Genetics*, **29**(1), 401–421, 1995.
- Drysdale, C., McGraw, D., Stack, C., Stephens, J., Judson, R., Nandabalan, K., Arnold, K., Ruano, G. and Liggett, S. Complex promoter and coding region β_2 -adrenergic receptor haplotypes alter receptor expression and predict *in vivo* responsiveness. *Proceedings of the National Academy of Sciences of the United States of America*, **97**(19), 10483–10488, 2000.
- Excoffier, L. and Slatkin, M. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, **12**(5), 921–927, 1995.
- Futatsugawa, Y., Kubota, T., Ishiguro, A., Suzuki, H., Ishikawa, H. and Iga, T. PCR-based haplotype determination to distinguish CYP2B6*1/*7 and *5/*6. *Clinical Chemistry*, **50**(8), 1472–1473, 2004.
- Gusfield, D. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of Computational Biology*, **8**(3), 305–323, 2001.
- Gusfield, D. 2002. Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In G. Myers, S. Hannenhalli, D. Sankoff, S. Istrail, P. Pevzner, and M. Waterman (Eds.), *Annual Conference on Research in Computational*

Molecular Biology: Proceedings of the sixth annual international conference on computational biology (pp. 166–175). ACM Press New York, NY, USA.

Gusfield, D. 2003. Haplotype inference by pure parsimony. In *Combinatorial Pattern Matching: 14th Annual Symposium* (pp. 144–155). Springer Berlin / Heidelberg.

Gusfield, D. 2004. An overview of combinatorial methods for haplotype inference. In A. C. Sorin Istrail, Michael Waterman (Ed.), *Computational Methods for SNPs and Haplotype Inference: DIMACS/RECOMB Satellite workshop* (pp. 9–25). Springer Berlin / Heidelberg.

Halldórsson, B. V., Bafna, V., Edwards, N., Lippert, R., Yooseph, S. and Istrail, S. 2002. A survey of computational methods for determining haplotypes. In *Computational methods for SNPs and haplotype inference* (pp. 26–47). Springer Berlin / Heidelberg.

Hawley, M. E. and Kidd, K. K. HAPLO: A program using the EM algorithm to estimate the frequencies of multi-site haplotypes. *Journal of heredity*, **86**(5), 409–411, 1995.

Helmuth, L. Map of the human genome 3.0. *Science*, **293**(5530), 583–585, 2001.

Huang, Y. T., Chao, K. M. and Chen, T. An approximation algorithm for haplotype inference by maximum parsimony. *Journal of Computational Biology*, **12**(10), 1261–1274, 2005.

Hudson, R. R. Gene genealogies and the coalescent process. *Oxford survey of evolutionary biology*, **7**, 1–44, 1990.

Hudson, R. R. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, **18**(2), 337–338, 2002.

- Kalpakis, K. and Namjoshi, P. 2005. Haplotype phasing using semidefinite programming. In *Bibe '05: Proceedings of the fifth ieee symposium on bioinformatics and bioengineering* (pp. 145–152). Washington, DC, USA: IEEE Computer Society.
- Lancia, G., Bafna, V., Istrail, S., Lippert, R. and Schwartz, R. 2001. SNPs problems, algorithms and complexity. In *European symposium on algorithms (ESA 01)* (pp. 182–193). Springer-Verlag.
- Lancia, G., Pinotti, C. M. and Rizzi, R. Haplotyping populations by pure parsimony : Complexity, exact and approximation algorithms. *INFORMS Journal on Computing*, **16**(4), 348–359, 2004.
- Li, Z., Zhou, W., Zhang, X. and Chen, L. A parsimonious tree-grow method for haplotype inference. *Bioinformatics*, **21**(17), 3475–3481, 2005.
- Lippert, R., Schwartz, R., Lancia, G. and Istrail, S. Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Briefings in Bioinformatics*, **3**(1), 23–31, 2002.
- Long, J. C., Williams, R. C. and Urbanek, M. An E-M algorithm and testing strategy for multiple-locus haplotypes. *American journal of human genetics*, **56**(3), 799–810, 1995.
- Michalatos-Beloin, S., Tishkoff, S. A., Bentley, K. L., Kidd, K. K. and Ruano, G. Molecular haplotyping of genetic markers 10 kb apart by allele-specific long-range PCR. *Nucleic Acids Research*, **24**(23), 4841–4843, 1996.
- Nickerson, D. A., Taylor, S. L., Weiss, K. M., Clark, A. G., Hutchinson, R. G., Sten-gard, J., Salomaa, V., Vartiainen, E., Boerwinkle, E. and Sing, C. F. DNA

- sequence diversity in a 9.7-kb region of the human lipoprotein lipase gene. *Nature Genetics*, **19**, 233–240, 1998.
- Niu, T., Qin, Z. S., Xu, X. and Liu, J. S. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *American Journal of Human Genetics*, **70**(1), 157–169, 2002.
- Rizzi, R., Bafna, V., Istrail, S. and Lancia, G. 2002. Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. In R. Guig'o and D. Gusfield (Eds.), *Algorithms in Bioinformatics: Second international workshop* (pp. 29–43). Springer Berlin / Heidelberg.
- Ruano, G. and Kidd, K. K. Direct haplotyping of chromosomal segments from multiple heterozygotes via allele-specific PCR amplification. *Nucleic Acids Research*, **17**(20), 8392–8392, 1989.
- Ruano, G., Kidd, K. K. and Stephens, J. C. Haplotype of multiple polymorphisms resolved by enzymatic amplification of single DNA molecules. *Proceedings of the National Academy of Sciences of the United States of America*, **87**(16), 6296–6300, 1990.
- Stephens, M. and Donnelly, P. A comparison of bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, **73**(5), 1162–1169, 2003.
- Stephens, M., Smith, N. J. and Donnelly, P. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, **68**(3), 978–989, 2001.

- Wang, L. and Xu, Y. Haplotype inference by maximum parsimony. *Bioinformatics*, **19**(14), 1773–1780, 2003.
- Wang, R. S., Wu, L. Y., Li, Z. P. and Zhang, X. S. Haplotype reconstruction from SNP fragments by minimum error correction. *Bioinformatics*, **21**(10), 2456–2462, 2005.
- Zhang, X. S., Wang, R. S., Wu, L. Y. and Chen, L. Models and algorithms for haplotyping problem. *Current Bioinformatics*, **1**(1), 105–114, 2006.

BIOGRAPHY

姓名：楊惠娥

生日：1982 年 10 月 04 日

學歷：國立成功大學資訊管理研究所畢業（2004.09 ~ 2006.06）

國立台灣科技大學資訊管理學系畢業（2002.09 ~ 2004.06）

國立台北商業技術學院畢業（1997.09 ~ 2002.06）

E-mail：r7693103@ccmail.ncku.edu.tw