Computational Experiments on Algorithms for Haplotype Inference Problems by Pure Parsimony

I-Lin Wang[1] Hui-E Yang[1]

[1] Institute of Information Management, National Cheng Kung University

Abstract

To analyze the function of DNA, researchers have to obtain each haplotype, the genetic constitution of an individual chromosome, of an individual for analysis. Due to the significant efforts required in collecting haplotypes, the descriptions of one conflated pair of haplotypes called genotypes are usually 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 problem by pure parsimony criterion which uses the minimum number of haplotypes to infer the haplotype data from genotype data for a population. Previous researches use mathematical programming methods such as integer programming and semidefinite programming models to solve the population haplotype inference problem. However, no computational experiment has ever been conducted to evaluate the algorithmic effectiveness. This paper thus conducts the first computational experiments on four haplotyping algorithms, including our new greedy heuristic and three pervious haplotyping algorithms.

Keywords: Haplotype inference, Genotype, Pure parsimony, Mathematical programming, Heuristic

1. Introduction

Among all variations defined by biologists, the Single Nucleotide Polymorphism (SNP) is the most frequently observed one and may be a key to disease analysis. A haplotype is a sequence of closely linked SNPs in one copy of chromosome. Since the collection of haplotype 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. In this paper, we focus on the population haplotype inference (PHI) problem, which infers haplotypes from genotypes for a group of individuals. In particular, the PHI problem can be defined as follows:

Given an $m \times n$ genotype matrix G, each row in the genotype matrix corresponds to a genotype data for one individual, whereas each column stands for one SNP, and each element in G has value 0, 1, or 2. A site is called homozygous wild type if it has value 0, homozygous mutant type if it has value 1, and heterozygous if it has value 2. A site is resolved if it has value 0 or 1, and ambiguous if it has value 2. A genotype is called resolved if there are two haplotypes such that for every site with value 0 or 1 in that genotype, the value of that site in the two haplotypes are either both with value 0 or both with value 1; for every site with value 2 in that genotype, one of the haplotype must have value 0 and the other haplotype must have value 1 in that site. The goal of a PHI is to find a $2m \times n$ haplotype matrix, in which the i th row in the genotype matrix is resolved by the $(2i-1)$th and $2i$ th rows in the haplotype matrix.

$$g_1 = 202 \Rightarrow p_1 = (000,101); p_2 = (001,100)$$

$$g_2 = 021 \Rightarrow p_3 = (001,011)$$

$$g_3 = 212 \Rightarrow p_4 = (010,111), p_5 = (011,110)$$

select $p_1, p_3, p_4 \Rightarrow$ 6 distinct haplotypes
select $p_1, p_3, p_5 \Rightarrow$ 5 distinct haplotypes
select $p_2, p_3, p_4 \Rightarrow$ 5 distinct haplotypes
select $p_2, p_3, p_5 \Rightarrow$ 4 distinct haplotypes

Fig. 1: A PHI example by pure parsimony criterion.

In order to solve the PHI problem with meaningful biological solutions, a popular biological model for PHI problem called the *pure parsimony criterion*, which seeks the fewest distinct haplotypes to resolve a given genotype matrix, has been proposed by Gusfield [2]. Take Fig. 1 for example. Given a genotype matrix G ={202,021,212}, there are two, one, and two possible haplotype pairs to resolve genotype 1, 2, and 3, respectively. Furthermore, there are six, five, five, or four distinct haplotypes if we select (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 the genotype matrix,

respectively. Using the pure parsimony criterion, (p_2, p_3, p_5) will be selected to resolve all genotypes, since this combination induces the minimum number of distinct haplotypes. Pure parsimony criterion to the PHI problem is NP-hard, as first formally shown by Lancia et al. [6].

The major contributions of this paper are in two folds. First, we propose a new greedy heuristic whose solution achieves a good error rate in a time shorter than the optimal solution by Pure Parsimony, where the error rate is the proportion of genotypes whose original haplotype pairs are inferred incorrectly. Second, we have conducted computational experiments for three other mathematical programming algorithms, in addition to our own heuristic, to evaluate their effectiveness on several classes of haplotype inference problems. The rest of this paper is organized as follows: Section 2 reviews the literature. Section 3 illustrates our new heuristic algorithm to PHI problem by pure parsimony. Section 4 describes settings of our computational experiments, as well as their results. Section 5 concludes the paper and suggests future research topics.

2. Literature Review

The mathematical programming methodologies to resolve the PHI problem by pure parsimony in the literature can be divided into three categories: integer programming, integer quadratic programming, and dynamic programming.

Gusfield [2] defines the Pure Parsimony criterion and proposes an integer linear programming formulation to the PHI problem. For each genotype, Gusfield first enumerates all possible inferred haplotype pairs and then applies constraints to ensure exactly one pair among all the possible pairs for that genotype to be selected. The formulation contains exponentially many variables and constraints with respect to the number of SNP (n) and genotypes (m), and has to be solved using mathematical programming software such as CPLEX [5]. Although its optimal solution solves the PHI problem by pure parsimony, there usually exist multiple optimal solutions with the same number of selected haplotype pairs.

Brown and Harrower [1] propose a different and polynomial-sized integer linear programming formulation to solve the optimal solution for the PHI problem by pure parsimony. Using different formulation techniques without enumerating all possible haplotype pairs as the model by Gusfield [2], their formulation has polynomial number of constraints and variables. Nevertheless, their model usually requires much more computational efforts to solve than the model by Gusfield [2].

Huang et al. [3] propose another nonlinear integer formulation for this problem. Similar to the model by Gusfield [2], they also enumerate all possible haplotype pairs for each genotype but use different techniques to formulate an integer quadratic problem. Since solving an integer quadratic programming is NP-hard, they use a semidefinite programming relaxation algorithm to find an $O(\log m)$ approximation solution, where m is the number of genotypes (rows) in a given genotype matrix. A Matlab code, SDPHapInfer, has been developed by Huang et al. [3] and is reported to have small error rate performance, compared with two statistical methods by Niu et al. [8] and Stephens and Donnelly [9].

Besides those methods which solve for the exact or approximated solutions, Li et al. [7] propose another heuristic algorithm called *parsimonious tree-grow* (PTG) method which resolves the genotype matrix in a column-by-column manner. This algorithm first creates a root node and adds indices of all genotypes into the root, then grows this tree with minimal branching principle. In particular, whenever a site with value 0 (or 1) is encountered, the branch of one node is set to be 0 (or 1). On the other hand, whenever a site with value 2 is encountered, PTG will select an appropriate branch such that the number of haplotypes is minimized at that moment. PTG is shown to resolve a given genotype matrix in $O(m^2 n)$ time, although such a solution may not be optimal with respect to the pure parsimony criterion.

In summary, the PHI problem by pure parsimony is a very new research topic. Most solution methods in the literature are based on mathematical programming techniques, and are row-based where all the possible haplotype pairs may have to be enumerated, which limits the number of SNPs (columns) applicable in algorithmic implementation. The computational experiments are usually conducted on genotype matrices with columns up to 20, except for the column-based PTG method which is capable of resolving genotype matrices of large size. Effectiveness on solution methods are usually evaluated by their error rates.

3. Proposed Methodology

Although the methods designed to solve the PHI problem by pure parsimony aim at resolving a genotype matrix using the fewest number of haplotypes, the exact optimal solutions obtained by those mathematical programming methodologies usually consume a lot of computational time and lack computational error rate analysis. This suggests the need for developing a solution method that may not require solving this problem to optimality, but should

be fast and capable of computing a solution with satisfactory quality. To this end, we propose a greedy heuristic based on the idea of the model in Gusfield [2]. In particular, among those candidate haplotypes, one that can resolve more genotypes should be more likely to appear in the solution since its use reduces the required haplotypes.

Our new heuristic contains four procedures: First, for each genotype in a given genotype matrix, we enumerate all possible inferred haplotype pairs. Second, for each haplotype, we count the number of its resolved genotypes and use this number as its weight. Third, for each candidate haplotype pair, we multiply the weights of its two haplotypes and use this product of weights as the weight associated with this candidate haplotype pair. Fourth, for each genotype, we select the candidate haplotype pair with the largest weight to be its inferred haplotype pair. The complexity of our heuristic is $O(q)$, where q is the number of total candidate inferred haplotype pairs.

$$
\begin{aligned}
g_1 = 202 \Rightarrow p_1 &= (000,101), \text{ weight } = 1\times 1 = 1 \\
p_2 &= (001,100), \text{ weight } = 2\times 1 = 2 \\
g_2 = 021 \Rightarrow p_3 &= (001,011), \text{weight } = 2\times 2 = 4 \\
g_3 = 212 \Rightarrow p_4 &= (010,111), \text{weight } = 1\times 1 = 1 \\
p_5 &= (011,110), \text{weight } = 2\times 1 = 2
\end{aligned}
$$

Fig. 2: Haplotype weight calculation using our heuristic.

Take Fig. 2 for example, if the input genotype matrix is $G = \{202, 021, 212\}$, three inferred haplotype pairs: one of p_1 and p_2, p_3, 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 the weights for haplotypes 000, 101, 001, 100, 011, 010, 111, and 110 are 1, 1, 2, 1, 2, 1, 1, and 1, respectively, the weighs for haplotype pairs p_1, p_2, p_3, p_4 and p_5 become 1, 2, 4, 1, and 2, respectively, as illustrated in Fig. 2. Then, our heuristic will select p_2, p_3, and p_5 to resolve g_1, g_2, and g_3, respectively. This heuristic is straightforward, easy to implement, and only require a little more computational efforts after enumerating all the possible inferred haplotype pairs as also required in algorithms by Gusfield [2] and Huang et al.[3].

There may be several ways to assign the weights for the candidate haplotype pairs. The rule of thumb is to give larger weights to candidate haplotype pairs that contain "popular" individual haplotypes capable of resolving more genotypes. To avoid selecting a candidate haplotype pair which contains both a very popular haplotype and a very unpopular individual haplotype, we use multiplication instead of addition of the weights for those two individual haplotypes to be the weight of a candidate haplotype pair.

4. Computational experiments and analysis

Besides evaluating the error rates and computational time, it should also be interesting to see how well the solutions obtained by some non-optimal algorithms (e.g. PTG and SDPHapInfer) could achieve this objective. For this purpose, we define and use the optimality gap, the proportion of the difference in the number of distinct inferred haplotypes appeared in the solution to the minimum number of the inferred haplotypes, as an evaluating parameter in our computational experiments.

We simulate the input genotypes and haplotypes using the program by Hudson [4] which produces a group of haplotypes, and then randomly pair two haplotypes as a genotype. Nine problem sets of different genotype matrix sizes (10x10, 20x10, 30x10, 20x10, 20x20, 30x20, 30x10, 30x20, and 30x30) are simulated, where 15 random datasets for each problem set have been generated.

Four algorithms are implemented and evaluated: (1) our greedy heuristic, denoted as "GHI", is implemented in C++ and compiled by g++ compiler ; (2) the tree-grow heuristic algorithm directly imported from Li et al.[7], denoted as "PTG", is implemented using Borland Delphi 5.0 in Pascal; (3) the semidefinite programming relaxation algorithm by Huang et al.[3], denoted as "SDPHapInfer", is implemented using Matlab; and (4) the integer linear programming model by Gusfield [2], denoted as "OPT", is implemented in C++, compiled by Visual C++, and linked with CPLEX 9.0 callable library [5].

All the computational experiments are conducted on a Pentium4 PC with 3.2 GHz CPU, 1GB RAM and Windows XP operating system. For each test we record the optimality gap (%), error rate (%), and computational time (sec) for each algorithm, and then compute the average of these three parameters for 15 cases of the same problem set for each algorithm as illustrated in Fig. 3, Fig. 4, and Fig 5.

About the solution quality of three non-optimal algorithms in terms of the optimality gap using the pure parsimony criterion as shown in Fig. 3, PTG performs the best, GHI performs the middle, whereas SDPHapInfer has lower optimality gap for cases up to 10 SNPs but the gap increases dramatically for larger cases. OPT has the lowest error rate in all cases tested (see Fig. 4). However, the error rate performances of the other three algorithms are similar to their performances in optimality gap. As for the computational time (see Fig. 5), OPT consumes the most time, whereas PTG is the quickest algorithm. Theoretically, SDPHapInfer should still require more time than GHI even implemented using the same

language since GHI consumes much less time than the complicated semidefinite programming relaxation algorithm.

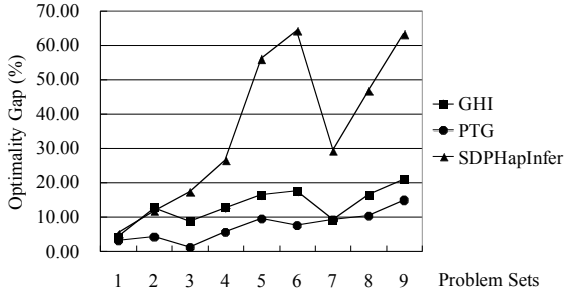

Fig. 3: Optimality gap comparison

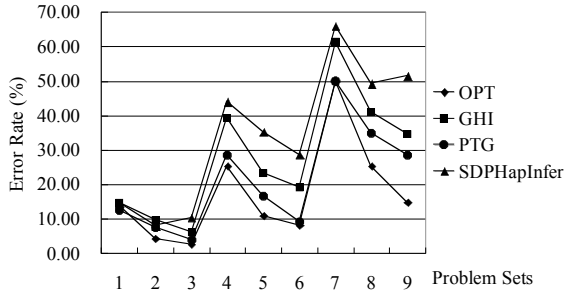

Fig. 4: Error rate comparison

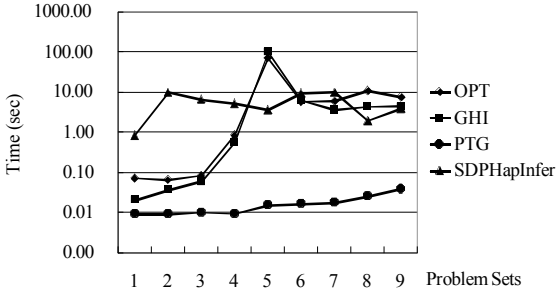

Fig. 5: Computational time comparison

5. Conclusion

In this paper, we propose a new heuristic algorithm (GHI) for solving the PHI problem by pure parsimony criterion and conduct experiments to compare our greedy heuristic algorithm to three methodologies, including a tree-grow heuristic (PTG), a linear quadratic programming approximation method (SDPHapInfer), and a linear integer programming model (OPT). Our greedy heuristic is straightforward, easy to implement, and capable of producing a solution with small error rate and small optimality gap in a short computational time. We conduct the first computational experiments in evaluating and comparing the efficiency and effectiveness for four algorithms. The results suggest the method by Huang et al. [3] is suitable for smaller but not good for larger cases. The PTG by Li et al. [7] almost performs the best in all of the three aspects that we tested.

The existence of multiple optimal solutions may lead to larger error rates of the exact optimal solution to the pure parsimony criterion. Furthermore, this also suggests the need for a better tie-breaking mechanism to improve the solution quality of many algorithms, including ours.

6. Acknowledgements

I-Lin Wang was supported by NSC93-2213-E006-096.

7. References

[1] D. G. Brown, and I. M. Harrower, "A new integer programming formulation for the pure parsimony problem in haplotype analysis", *Algorithms in Bioinformatics: 4th International Workshop*, *Springer Lecture Notes in Computer Science*, Vol. 3240, pp. 254-265, 2004.

[2] D. Gusfield, "Haplotype inference by pure parsimony ", *Annual Symposium Combinatorial Pattern Matching*, *Springer Lecture Notes in Computer Science*, No. 2676, pp.144-155, 2003.

[3] Y. T. Huang, K. M. Chao, and T. Chen, "An approximation algorithm for haplotype inference by maximum parsimony", *Journal of Computational Biology*, Vol. 12, No. 10, pp. 1261-1274, 2005.

[4] R. Hudson, "Generating samples under a Wright-Fisher neutral model of genetic variation", *Bioinformatics*, Vol. 18, pp. 337–338, 2002.

[5] ILOG, ILOG CPLEX 9.0 User's Manual, 2003.

[6] G. Lancia, C. M. Pinotti, and R. Rizzi, "Haplotyping populations by pure parsimony: complexity, exact and approximation algorithms", *INFORMS Journal on Computing*, Vol. 16, pp. 348-359, 2004.

[7] Z. Li, W. Zhou, X. Zhang, and L. Chen, "A parsimonious tree-grow method for haplotype inference." *Bioinformatics*, Vol. 21, No. 17, pp. 3475-3481, 2005.

[8] T. Niu, Z. Qin, X. Xu, and J. S. Liu, "Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms", *Am. J. Human Genet.*, Vol. 70, pp. 157–159, 2002.

[9] M. Stephens, N. J. Smith, and P. Donnelly, "A new statistical method for haplotype reconstruction from population data", *Am. J. Human Genet.*, Vol. 68, No. 4, pp. 978–989, 2001.