Applied Mathematics and Computation 217 (2011) 9798–9809

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Haplotyping populations by pure parsimony based on compatible genotypes and greedy heuristics

I-Lin Wang *, Hui-E Yang

Department of Industrial and Information Management, National Cheng Kung University, No. 1, University Rd., Tainan 701, Taiwan

ARTICLE INFO

Keywords:
Haplotype inference
Integer programming
Bioinformatics
Compatibility graph
Heuristics

ABSTRACT

The population haplotype inference problem based on the pure parsimony criterion (HIPP) infers an $m \times n$ genotype matrix for a population by a $2m \times n$ haplotype matrix with the minimum number of distinct haplotypes. Previous integer programming based HIPP solution methods are time-consuming, and their practical effectiveness remains unevaluated. On the other hand, previous heuristic HIPP algorithms are efficient, but their theoretical effectiveness in terms of optimality gaps has not been evaluated, either. We propose two new heuristic HIPP algorithms (MGP and GHI) and conduct more complete computational experiments. In particular, MGP exploits the compatible relations among genotypes to solve a reduced integer linear programming problem so that a solution of good quality can be obtained very quickly; GHI exploits a weight mechanism to selects better candidate haplotypes in a greedy fashion. The computational results show that our proposed algorithms are efficient and effective, especially for solving cases with larger recombination rates.

1. Introduction

In the post-genomic era, the development of a full haplotype map has high priority since Helmuth [1] suggests that the knowledge about the genetic constitution of an individual chromosome called haplotypes can be applied in linkage disequilibrium, inference of population evolutionary history, disease diagnosis, and customization of treatment for each individual. The haplotype is a sequence of closely linked single nucleotide polymorphisms (SNPs) in one copy of a chromosome. There are two haplotypes in a pair of chromosomes in all diploid organisms. 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. The computational problem to construct the haplotypes from genotype data is called haplotyping. Two categories of haplotyping problems are investigated in literature: one category is concerned with haplotyping a single individual based on the data and methodology of shotgun sequence assembly [2–4], and the other category is concerned with haplotyping a population [5,6]. We study the latter category in this paper.

Clark [5] first brings up the population haplotype inference (PHI) problem to infer haplotypes from genotypes for a group of individuals. He also gives an inference rule, assuming the genotypes with zero or one ambiguous sites contain the frequently observed haplotypes in the population. After that, many PHI solution methods and problems are proposed, including the estimation-maximization (EM) algorithm by Excoffier and Slatkin [7], Long et al. [8], and Hawley and Kidd [9], Bayesian method by Stephens et al. [10], Niu et al. [11] and Stephens and Donnelly [12], maximum resolution (MR) problem by Gusfield [6], perfect phylogeny haplotype (PPH) problem by Gusfield [13], and PHI problem that satisfies the pure parsimony

* Corresponding author.
E-mail address: ilinwang@mail.ncku.edu.tw (I.-L. Wang).

0096-3003/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2011.04.073

criterion (i.e. called the HIPP problem) proposed by Gusfield [14]. In this paper, we focus on solving the HIPP problem, where the number of distinct haplotypes for resolving a given genotype matrix is minimized.

Suppose we are given m genotype vectors, each has length n, and denote this set of vectors as an $m \times n$ 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 $G = \{g_1, g_2, \ldots, g_m\}$ be the set of all rows in G and $g_i = g_{i,1}g_{i,2}\cdots g_{i,n}$ denote the genotype data for individual i. Every element $g_{i,j}$ has value 0, 1, or 2 depending on whether the ith individual is homozygous wild type, homozygous mutant, or heterozygous, respectively, at the jth SNP. Any element $g_{i,j}$ in G is said to be resolved if its value is 0 or 1, and ambiguous if its value is 2. Suppose h_a and h_b are two $1 \times n$ haplotype vectors. We say h_a and h_b resolve a genotype g_i, denoted by $g_i = h_a \otimes h_b$, if and only if $h_{a,j} = h_{b,j} = g_{i,j}$ for each $g_{i,j} \in \{0, 1\}$ and $h_{a,j} + h_{b,j} = 1$ for each $g_{i,j} = 2$. See Fig. 1 for an example. In this case, we say h_a (or h_b) a candidate (or explaining) haplotype for g_i, $\{h_a, h_b\}$ a candidate (or explaining) haplotype pair for g_i, and h_a (or h_b) the conjugate (or complementary) haplotype for h_b (or h_a) in the pair $\{h_a, h_b\}$.

By definition, the number of candidate haplotype pairs equals to $2^{\kappa-1}$ for a genotype that contains κ ambiguous sites (i.e. heterozygous SNPs). Although there are many candidate haplotype pairs for resolving a given genotype matrix, the real-world haplotype pairs are induced by a very few amount of distinct haplotypes. For example, Drysdale et al. [15] identify 13 SNPs in the human β_2 *AR* gene, which has $2^{13} = 8192$ possible combinations. However, only 10 among those 8192 candidate haplotypes are related to asthmatic cohort. Thus Gusfield [14] suggests a combinatorial optimization problem called the haplotype inference based on pure parsimony (HIPP) which seeks the minimum amount of distinct haplotypes to resolve a given genotype matrix.

The objective of the HIPP problem is to find a $2m \times n$ haplotype matrix, in which the ith row (i.e. g_i) in the genotype matrix is resolved by the $(2i-1)$th and the $2i$th rows (i.e. $g_i = h_{2i-1} \otimes h_{2i}$) in the haplotype matrix, and the number of distinct haplotypes is minimized. See Fig. 1 for an example. Given the genotype matrix $G = \{202, 021, 212\}$, there are 2, 1, and 2 candidate haplotype pairs to resolve genotype 1, 2, and 3, respectively. Furthermore, there are 6, 5, 5, or 4 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 G, respectively. Using the pure parsimony criterion, (p_2, p_3, p_5) will be selected to resolve all the genotypes since this combination induces the minimum number of distinct candidate haplotypes.

The HIPP problem is APX-hard, as shown by Lancia et al. [16]. The HIPP solution methods in the literature are either based on the mathematical programming techniques such as integer linear programming (ILP) by Gusfield [14] and Brown and Harrower [17] and quadratic integer programming by Huang et al. [18] and Kalpakis and Namjoshi [19], or the heuristic algorithm by Li et al. [20]. In particular, Gusfield [14] gives the first ILP formulation called RTIP to model the HIPP problem. For each genotype, RTIP first enumerates all the candidate haplotype pairs and then gives constraints to ensure that exactly one among all the possible pairs for each genotype is selected. Take the 3-genotype case in Fig. 1 for example: for genotype $g_1 = 202$, we enumerate its two candidate haplotype pairs $p_1 = 000 \otimes 101$ and $p_2 = 001 \otimes 100$, and create two binary variables $y_{1,1}$ and $y_{1,2}$ to respectively represent whether p_1 or p_2 is selected in the HIPP solution; similarly, $y_{2,1}$ associated with $p_3 = 001 \otimes 011$ for $g_2 = 021$, as well as $y_{3,1}$ and $y_{3,2}$ corresponding to $p_4 = 010 \otimes 111$ and $p_5 = 011 \otimes 110$ for $g_3 = 212$ are also created. We then generate eight binary variables $x_1, x_2, \ldots, x_8$ to respectively represent whether to select the constituted haplotypes 000, 101, 001, 100, 011, 010, 111, and 110 in the optimal solution. Then, the RTIP formulation for this example is as follows:

$$
\begin{aligned}
\min \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \\
\text{s.t.} \quad & y_{1,1} + y_{1,2} = 1, y_{2,1} = 1, y_{3,1} + y_{3,2} = 1, \\
& y_{1,1} \leqslant x_1, y_{1,1} \leqslant x_2, y_{1,2} \leqslant x_3, y_{1,2} \leqslant x_4, y_{2,1} \leqslant x_3, \\
& y_{2,1} \leqslant x_5, y_{3,1} \leqslant x_6, y_{3,1} \leqslant x_7, y_{3,2} \leqslant x_5, y_{3,2} \leqslant x_8, \\
& x_1, \ldots, x_8, y_{1,1}, y_{1,2}, y_{2,1}, y_{3,1}, y_{3,2} \in \{0, 1\},
\end{aligned}
$$

where the objective function minimizes the number of selected candidate haplotypes, as long as one candidate haplotype pair is selected for each genotype, which forces its constituted candidate haplotypes to be selected as well. The RTIP formulation thus contains potentially exponential number of variables and constraints with respect to the number of SNPs.

$$
G = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 202 \\ 021 \\ 212 \end{bmatrix}, \quad \begin{array}{l} g_1 = 000 \otimes 101\ (p_1) \text{ or } 001 \otimes 100\ (p_2) \\ g_2 = 001 \otimes 011\ (p_3) \\ g_3 = 010 \otimes 111\ (p_4) \text{ or } 011 \otimes 110\ (p_5) \end{array}
$$

$$
\begin{aligned}
&\text{select } p_1, p_3, p_4 \Rightarrow 6 \text{ distinct haplotypes} \\
&\text{select } p_1, p_3, p_5 \Rightarrow 5 \text{ distinct haplotypes} \\
&\text{select } p_2, p_3, p_4 \Rightarrow 5 \text{ distinct haplotypes} \\
&\text{select } p_2, p_3, p_5 \Rightarrow 4 \text{ distinct haplotypes}
\end{aligned}
$$

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

Although RTIP is potentially exponential-sized, it is practically faster than PolyIP, the polynomial-sized ILP formulation proposed by Brown and Harrower [17]. Wang and Xu [21] give a branch and bound algorithm called HAPAR. HAPAR also takes a lot of computational time since it tries out all the combinations to solve the HIPP problem. RTIP, PolyIP, and HAPAR all calculate the exact optimal solution for the HIPP problem, but they usually consume a lot of computational resources and time, and may not be suitable for solving large-scale HIPP problems.

Recently, Boolean Satisfiability (SAT) has been proposed for solving the HIPP problem with success. According to Lynce and Marques-Silva [22,23], their SAT-based method called SHIPs can calculate exact optimal solutions for large-scale HIPP problems with sizes up to 50 genotypes and 100 sites, which are often unsolvable by RTIP, PolyIP, and HAPAR. Techniques such as local search (see Lynce et al. [24]) have also been used to improve the efficiency of SHIPs. Furthermore, Graca et al. [25,26] propose Pseudo-Boolean Optimization (PBO) models called PolyPB and its reduced version called RPoly, based on the PolyIP model. RPoly has smaller size and consistently better efficiency than PolyPB. Their experiments indicate RPoly is the state-of-the-art method since it can solve more large-scale HIPP cases and is also often faster than SHIPs. Although these optimal HIPP algorithms give the theoretical optimal HIPP solutions, whether these optimal solutions also have good practical effectiveness in terms of small error rates (i.e. the proportion of genotypes whose inferred haplotype pairs are different from the original ones) requires further evaluation.

Heuristics are important in solving the PHI problems since the methodologies for obtaining the exact optimal solutions are not only theoretically difficult but also time-consuming in practice. Based on different integer quadratic programming formulations, Huang et al. [18] give an approximation algorithm called SDPHapInfer, and Kalpakis and Namjoshi [19] propose another heuristic algorithm to solve their integer quadratic programming HIPP formulation. SDPHapInfer performs well for smaller cases but its error rates increase dramatically for larger cases (see Section 4 for details). The heuristic algorithm by Kalpakis and Namjoshi [19] requires further evaluation. On the other hand, the parsimonious tree growing heuristic algorithm called PTG by Li et al. [20] is very fast, but its theoretical effectiveness in terms of optimality gap (i.e. the difference in the number of distinct candidate haplotypes used, compared with the optimal solution) remains to be evaluated.

To design an efficient algorithm that can give a good solution to solve the HIPP problem, this paper investigates the compatible relations between genotypes. Based on the compatible relations between genotypes, we give estimates on the upper and lower bounds for the optimal HIPP objective value. Such compatible relations can also be used to reduce the solution space of Gusfield's formulation (RTIP) and obtain good solutions faster. Our first heuristic algorithm called MGP identifies common haplotypes for any two compatible genotypes, and then formulates an integer linear program to resolve the genotypes by those common haplotypes. MGP effectively reduces the solution space of the RTIP formulation so that it can obtain a good solution in a much shorter time than the original RTIP.

Since the HIPP problem seeks the minimum number of haplotypes to resolve a given genotype matrix, among all the candidate haplotypes, one that can resolve more genotypes should be more likely to appear in the solution since it may reduce the number of required haplotypes. To select such haplotypes, we design our second greedy heuristic algorithm called GHI by exploiting some weight mechanisms based on biological intuitions for each candidate haplotype and candidate haplotype pair so that the algorithm will select a candidate haplotype that seems to have higher probability to be in the optimal solution.

The remaining of this paper is organized as follows. Section 2 introduces the notations and compatibility graph used for our algorithms. In Section 3, we propose two new heuristic algorithms to solve the HIPP problem based on mathematical properties and biological insights. Several numerical experiments on both real-world and simulated genotype data for different formulations and algorithms are conducted and summarized in Section 4. Section 5 concludes the paper and suggests future research directions.

2. Preliminaries

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 ζ_a and ζ_b are compatible to each other, denoted as $\zeta_a \sim \zeta_b$, if and only if $(\zeta_{a,j}, \zeta_{b,j}) \notin \{(0,1), (1,0)\}$ for each column $j = 1,\ldots,n$. On the other hand, ζ_a and ζ_b are conflicting (or 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.

A compatibility graph G_C (see Fig. 2(a)) for a given genotype matrix G can consist of 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. Similarly, one may define a conflicting graph $\overline{G_C} = K_m \setminus G_C$ (see Fig. 2(b)), where $K_m$represents a complete graph of m nodes.

Lemma 1. *(a) Two conflicting genotypes share no common candidate haplotypes. (b) Let $A(G_C)$ denote the minimum number of arcs that covers all nodes in G_C. Then $3A(G_C)$ is an upper bound for the HIPP optimal objective value. (c) Let $\chi(\overline{G_C})$ denote the minimum number of node colors required for $\overline{G_C}$ such that adjacent nodes in $\overline{G_C}$ have different colors. Then $2\chi(\overline{G_C})$ is a lower bound for the HIPP optimal objective value.*

Proof

(a) By definition, if g_a conflicts g_b, there must exist a site j such that $(g_{a,j}, g_{b,j}) \in \{(0,1), (1,0)\}$. Since no haplotype on site j can have two different values, there exists no common candidate haplotypes to resolve g_a and g_b at the same time.

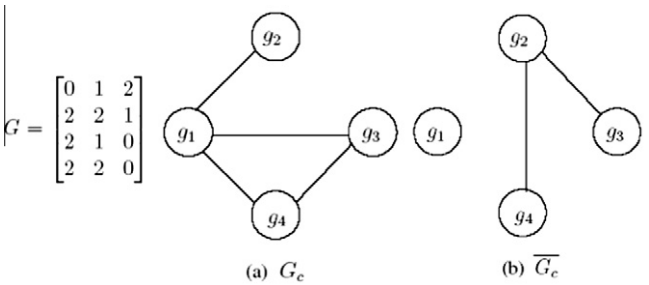

Fig. 2. An example of compatibility and conflicting graphs.

(b) Selecting an arc (g_a, g_b) in G_C can be thought as resolving the compatible genotype pair g_a and g_b using one of their common candidate haplotypes. Therefore, a selected arc in G_C will incur at most three 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.
(c) Since the nodes of the same color may at least require 2 candidate haplotypes, thus there are at least $2\chi(\overline{G_C})$ candidate haplotypes when all the nodes of the same color are resolved by the same haplotype pairs. □

Lemma 1(a) suggests a way to reduce an HIPP problem to several independent subproblems where each subproblem corresponds to a smaller HIPP problem whose genotypes form a component in G_C. Furthermore, the compatible relations can help to reduce the solution space of the RTIP model proposed by Gusfield [14]. In particular, one only requires enumerating those candidate haplotypes compatible with more than one genotype for the RTIP model, since those conflicting 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.

The compatible relations also provide a base for designing some intriguing heuristic algorithms that give good HIPP solutions. For example, Lemma 1(b) can be used to improve the objective upper bound from $2m$ to be $3A(G_C)$ by solving an arc covering problem on G_C which seeks $A(G_C)$, the minimum number of arcs that covers all the nodes. Similarly, Lemma 1(c) implies $2\chi(\overline{G_C})$ can be used to estimate the lower bound for the HIPP objective value. These estimated upper and lower bounds for the HIPP objective value may still be too loose. Similar compatible relations are also independently discussed by Lynce and Marques-Silva [23] and Lynce et al. [24], where they propose several techniques, different from ours, to improve the lower bounding procedures for their SHIPs method. In the next section, we introduce two heuristic algorithms that seek a good solution efficiently in practice based on the compatible relations.

3. Proposed heuristics

In this section, we give two heuristics to solve the HIPP problem. The first one exploits the compatible relations between genotypes and the second one selects popular haplotypes that can resolve more genotypes in a greedy fashion.

3.1. The method of merged genotype pairs (MGP)

Based on the compatible genotype pair relations, the common haplotypes for any two compatible genotypes can be easily identified and used to give a feasible solution to the HIPP problem. First, we construct a merged genotype pair, denoted as $mg_{\tilde{k}},\ \tilde{k} = 1, \ldots,\ \tilde{q}$, for each compatible genotype pair (g_a, g_b) as follows:

1. $mg_{\tilde{k}j} := g_{aj}$, if $g_{aj} = g_{bj} \in \{0, 1, 2\}$, for $j = 1, \ldots, n$.
2. $mg_{\tilde{k}j} := 1$ or 0, if $(g_{aj}, g_{bj}) \in \{(1,2), (2,1)\}$ or $(g_{aj}, g_{bj}) \in \{(0,2), (2,0)\}$, respectively, for $j = 1, \ldots, n$.

After conducting the pairwise comparisons, all the $\tilde{q}$ distinct merged genotype pairs $MG(G) := \{mg_{\tilde{k}} : \tilde{k} = 1, \ldots, \tilde{q}\}$ for a given genotype matrix G can be identified in $O(m^2n)$ time. For each g_i, we use $IVMG(i) := \{mg_{\tilde{k}} : g_a \otimes g_i = mg_{\tilde{k}}\ \forall g_a \in G\}$ to record the set of its compatible merged genotype pairs.

Similar to the minimum arc covering technique for estimating the upper bound that selects the minimum number of arcs (i.e. compatible relations) in G_C to cover all the nodes (i.e. genotypes), our first heuristic algorithm, called as MGP, selects the minimum number of merged pairs in G_C to cover all the nodes. Since a merged pair may contain more than one arc (i.e. compatible relation) in G_C, the objective value of MGP tends to be better than the objective value obtained by the minimum arc covering heuristics for upper bound as proposed in Lemma 1(b).

In particular, for each component $\hat{c}$ in G_C, we 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 the 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 component in G_C. The steps of MGP are described as follows:

Step 1. Conduct pairwise comparison to obtain $MG(G)$ and $IVMG(i)$ for each genotype g_i, $i = 1,\ldots,m$.
Step 2. For each component $\hat{c}$ in G_C, formulate $IP_{MGP1}(\hat{c})$, and then use its optimal solution to formulate $IP_{MGP2}(\hat{c})$
Step 3. Obtain the solution of MGP by taking the union of the optimal solution of $IP_{MGP2}(\hat{c})$ for each component $\hat{c}$.

Without loss of generality, suppose G_C only contains one component $\hat{c}$. For each merged genotype pair $mg_{\tilde{k}}$, assign a binary variable $\tilde{x}_{\tilde{k}}$ 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:

$$\begin{aligned}
\min \quad & \sum_{\tilde{k}=1}^{\tilde{q}} \tilde{x}_{\tilde{k}}(IP_{MGP1}(\hat{c})) \\
\text{s.t.} \quad & \sum_{\tilde{k}\in \arg IVMG(i)} \tilde{x}_{\tilde{k}} \geqslant 1, \quad \forall i = 1,\ldots,m, \\
& \tilde{x}_{\tilde{k}} \in \{0,1\}, \quad \forall \tilde{k} = 1,\ldots,\tilde{q}.
\end{aligned}$$

The optimal solution for $IP_{MGP1}(\hat{c})$ can be used to form a solution space for $IP_{MGP2}(\hat{c})$. In particular, we enumerate all the candidate haplotypes for the selected merged genotype pairs obtained from $IP_{MGP1}(\hat{c})$, and then use them to construct $IP_{MGP2}(\hat{c})$, which is a reduced RTIP formulation.

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

$$\begin{aligned}
\min \quad & \tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3 \\
\text{s.t.} \quad & \tilde{x}_1 + \tilde{x}_2 \geqslant 1,\ \tilde{x}_1 \geqslant 1,\ \tilde{x}_2 + \tilde{x}_3 \geqslant 1, \\
& \tilde{x}_1, \tilde{x}_2, \tilde{x}_3 \in \{0,1\}.
\end{aligned}$$

Multiple optimal solutions $(\tilde{x}_1^*, \tilde{x}_2^*, \tilde{x}_3^*) \in \{(1,1,0),(1,0,1)\}$ exist for this example, which may lead to different results. The first optimal solution shows that G can be resolved by five distinct candidate haplotypes {010, 011, 101, 110, 100} using mg_1 and mg_2, since $(g_1, g_2, g_3, g_4) = (010 \otimes 011,\ 011 \otimes 101,\ 010 \otimes 110,\ 010 \otimes 100)$. On the other hand, the second optimal solution shows that mg_1 and mg_3can be further expanded to a set of three haplotypes {011, 010, 110}, and then a set of six candidate haplotypes {010, 011, 101, 110, 100, 000} can be derived to resolve G. Let $\{p_1, p_2, p_3, p_4, p_5\}$ denote the set of candidate haplotype pairs {010 ⊗ 011, 011 ⊗ 101, 010 ⊗ 110, 010 ⊗ 100, 110 ⊗ 000} derived from those six candidate

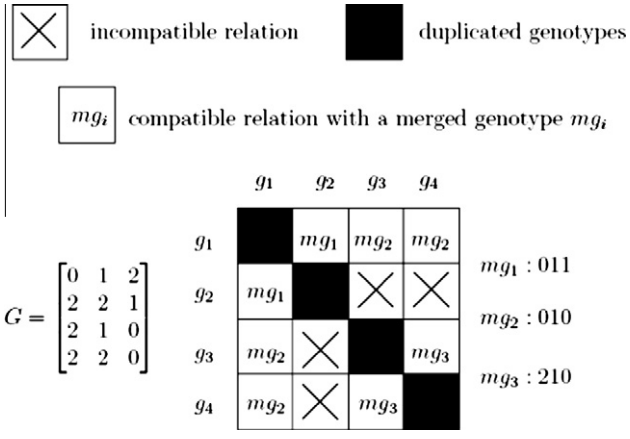

Fig. 3. An example of compatible genotype pairs.

haplotypes. Since g_4 can be resolved by more than one haplotype pair (i.e. p_4 and p_5), we have to form the second ILP $IP_{MGP2}(\hat{c})$ to decide the candidate genotype pairs that use fewest number of distinct haplotypes as follows:

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

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. $y = 0$).

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 genotype pairs to resolve all the genotypes. Since MGP can reduce the solution space of HIPP, it can give a good feasible solution for a large-scale HIPP problem that would be unsolvable in a reasonable time using the original RTIP formulation. We have implemented MGP and evaluated its performance in the next section.

3.2. The greedy heuristic inference method (GHI)

The pure parsimony criterion implies a popular candidate haplotype that can resolve more genotypes should be more likely to appear in the HIPP solution, since its appearance reduces the number of required haplotypes. To select popular haplotypes, our second greedy heuristic algorithm, called as GHI, gives larger weight for a candidate haplotype compatible with more genotypes, and then selects those haplotypes of larger weights.

In addition to the popularity intuition, we also take the inference rule proposed by Clark [5] into consideration to design the weight mechanism for each candidate haplotype. Since Clark's rule tries to infer heterozygote (i.e. genotypes containing more than one ambiguous site) from homozygote (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 containing fewer ambiguous sites should be assigned larger weights. Based on these weight mechanisms for each haplotype, GHI selects candidate haplotype pairs of larger weights to resolve all the genotypes.

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 each g_i, we define $CH(i) := \{h_a: h_a \sim g_i, \forall h_a \in H(G)\}$ to be the set of its candidate haplotypes, 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 its candidate haplotype pairs. Therefore, $\bigcup_{i=1,\ldots,m} CH(i) = H(G)$ and $\bigcup_{i=1,\ldots,m} CHP(i) = HP(G)$ for each $CH(i) \subset H(G)$ and $CHP(i) \subset HP(G)$. For each candidate haplotype h_k, $k = 1,\ldots,q$, 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 number of ambiguous sites and the weights associated with g_i, respectively for each $i = 1,\ldots,m$. By setting $gw(i) = \left(\sum_{j=1}^{m} ntwo(j)\right)/ntwo(i)$ for each $i = 1,\ldots,m$, we assign a larger weight for a genotype with fewer ambiguous sites. Note that for each genotype of zero ambiguous site, it can be resolved by a unique candidate haplotype pair (by the same haplotype), similar to a genotype of single ambiguous site. Therefore, we set $ntwo(i)$ to be one for each genotype i of zero or one ambiguous site. Then, we calculate the weight for each candidate haplotype h_k, $k = 1,\ldots,q$, by $hw(k) = \sum_{i \in \arg IG(k)} gw(i)$. The weight for each haplotype pair $hpw(\hat{k})$ equals to $hw(\hat{k}_a) \times hw(\hat{k}_b)$ for each $\hat{k} = 1,\ldots,\hat{q}$, if the $\hat{k}$th candidate haplotype pair is composed by the $\hat{k}_a$th and the $\hat{k}_b$th candidate haplotypes. Finally, our GHI algorithm selects the candidate haplotype pair that has the maximum pair weight to be the explaining pair for each g_i.

Besides the proposed weight mechanism for $gw(i)$, $hw(k)$, and $hpw(\hat{k})$, we have also tested other weight mechanisms such as equal weight, or linear incremental weight for $gw(i)$; or using addition rather than multiplication on the weights of the constituted haplotypes for $hpw(\hat{k})$. However, it turns out our proposed weight mechanism has the best performance. We give two points to explain the advantages of our proposed weight mechanism. First, the haplotype pair weight $hpw(\hat{k})$ can be thought as the frequency used in the EM algorithms by Excoffier and Slatkin [7], Long et al. [8], and Hawley and Kidd [9], where the idea of frequency is treated as a concept of probability. Therefore, the multiplication on the weights of the constituted haplotypes represents the probability for that candidate haplotype pair using those two constituted haplotypes. Similar to the EM algorithm which finds the haplotype probabilities to optimize the probability of the entire population, here we select the haplotype combinations with the largest estimated probability. Second, using $hw(\hat{k}_a) \times hw(\hat{k}_b)$ instead of $hw(\hat{k}_a) + hw(\hat{k}_b)$ to calculate $hpw(\hat{k})$ for each haplotype pair provides a better tie-breaking strategy.

GHI algorithm is straightforward and contains five procedures:

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

Step 2. Calculate$gw(i) = \left(\sum_{j=1}^{m} ntwo(j)\right)/ntwo(i)$ for each $g_i \in G$ using $ntwo(i)$.

Step 3. Calculate $hw(k) = \sum_{i \in \arg_{IG(k)}} gw(i)$ for each candidate haplotype $h_k \in H(G)$.
Step 4. Calculate $hpw(\hat{k}) = hw(\hat{k}_a) \times hw(\hat{k}_b)$ for each candidate haplotype pair $hp_{\hat{k}} = (hp_{\hat{k}_a}, hp_{\hat{k}_b}) \in HP(G)$.
Step 5. Select the candidate haplotype $hp_{i*} \in CHP(i)$ such that $i* = \arg\max_{hp_{\hat{k}}}\{hpw(\hat{k})\}$ for each genotype $g_i \in G$.

Take Fig. 4 for example. Given $G = \{202, 021, 212\}$, 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 g_1, g_2, and g_3, respectively. Since there are totally five ambiguous sites 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 weighs 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. Then, GHI will select p_2, p_3, and p_5 to resolve g_1, g_2, and g_3, respectively.

Suppose there are totally $|H(G)| = q$ candidate haplotypes and $|HP(G)| = \hat{q} = \sum_{i=1}^{m} 2^{ntwo(i)-1}$ candidate haplotype pairs. GHI takes $O(\hat{q} + mn + qm)$ time and $O(2^n)$ storage space using array to store $H(G)$ and $HP(G)$. Or, it takes $O(\hat{q}^3 + mn + qm)$ time and $O(\hat{q})$ storage space using linked list to store $H(G)$ and $HP(G)$. The efficiency of GHI thus depends very much in the total number of ambiguous sites that G contains. Note that the RTIP formulation by Gusfield [14] and the SDPHapInfer algorithm by Huang et al. [18] also require to enumerate all the necessary candidate haplotypes as does in the Step 1 of GHI, thus these three methods all consume similar time and storage space in their first step. However, considering the computational efforts afterwards, GHI only conducts simple calculation on weights, whereas RTIP has to solve an ILP problem and SDPHapInfer has to solve an SDP (i.e. semi-definite programming) problem. Therefore GHI should consume less computational time than RTIP and SDPHapInfer. Although GHI gives no theoretical guarantee on its solution quality, it includes the concept of Clark's rule and simple greedy intuition to obtain good solutions efficiently. Moreover, the practical performance of GHI has been shown to be good (see Section 4), especially for cases with recombination.

4. Computational experiments and discussion

This section summarizes the computational experiments for several HIPP algorithms. After introducing the experimental settings, one biological genotype dataset (i.e. β_2AR gene) and several simulated genotype datasets will be used for the experiments.

4.1. Settings for our computational experiments

All the computational experiments are conducted on a Pentium 4 PC with 3.2 GHz CPU, 1 GB RAM and Windows XP operating system. 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. [20], denoted as "PTG", is implemented using Borland Delphi 5.0 in Pascal; (4) the semidefinite programming relaxation algorithm by Huang et al. [18], denoted as "SDP", is implemented using MATLAB; (5) the integer linear programming model by Gusfield [14], 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 [5], denoted as "HAP", is implemented using Fortran. Note that although HAP is not designed for the pure parsimony criterion, we include it into our experiments since it is one of the haplotyping algorithms based on statistical models widely used in the community, and thus the error rates of HAP can serve as a base to compare the practical effectiveness of our proposed HIPP algorithms. Besides these six PHI algorithms, we have also implemented the basic PolyIP model (i.e. the one without branch and cut techniques) of Brown and Harrower [17]. Although PolyIP is an integer programming formulation with polynomial size which is theoretically better than the exponential-sized RTIP formulation by Gusfield [14], the results of our implementation indicates that PolyIP takes much more time than RTIP, even when solving small or medium sized HIPP problems. In fact, it performs the slowest in all of our experiments. Therefore, we do not include the results of PolyIP in this paper. We have not included the SAT-based HIPP methods such as SHIPs and RPoly into our computational tests, since their source codes are not publicly available for further modifications to meet our needs. Moreover, SHIPs and RPoly rely on some state-of-the-art SAT solver (e.g. MiniSAT by Eén and Sörensson, [27]) and PBO solver (e.g. MiniSAT+ by Eén and Sörensson, [28]), which are very different from the ILP solver (CPLEX) used by RTIP, PolyIP and MGP. Here in this paper, we focus on the performance of those ILP-based HIPP methods, and leave the experiments on those SAT-based HIPP methods for future research.

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

Fig. 4. Illustration on the proposed weight mechanism of GHI.

We record the computational time for several PHI algorithms in our experiments. Note that the amount of time spent by an algorithm can not really indicate its actual efficiency, since these algorithms are not implemented using the same programming language. However, for the practical purpose, a faster PHI algorithm is usually preferred.

We use the error rate, defined as the proportion of genotypes whose inferred haplotype pairs are different from the original ones, to evaluate the practical effectiveness for several classes of the HIPP algorithms. Take the 3-genotype case in Fig. 1 for example, suppose an HIPP algorithm gives p_2, p_3, p_4 as an answer for resolving genotypes g_1, g_2, g_3, while the original answer is p_2, p_3, p_5. Then, the error rate for this algorithm is 1/3 since only one (i.e. g_3) of the three genotypes is incorrectly inferred. Since the original inferred haplotype pairs usually can not be obtained in advance, the error rate is highly intractable, case-dependent, and may easily grow to a scale of 30% or more, even for those haplotyping algorithms based on statistical models by Stephens and Donnelly [12] and Niu et al. [11]. Intuitively, an algorithm that gives smaller error rates is usually considered to be more effective, although such an algorithm may not be the most effective one at all times.

For those non-optimal HIPP algorithms, we are the first to use the optimality gap, defined as the ratio of the difference in the number of selected inferred haplotypes obtained by an HIPP algorithm to the minimum number of inferred haplotypes (i.e. the optimal objective value of an HIPP problem), as a parameter to evaluate their theoretical effectiveness. Take the same 3-genotype example in previous paragraph, the optimal solution p_2, p_3, p_5 selects four haplotypes (i.e. 001, 100, 011, 110), while the solution p_2, p_3, p_4 by an HIPP algorithm selects five haplotypes (i.e. 001, 100, 011, 010, 111). Therefore, the optimality gap for this HIPP algorithm is $(5-4)/4 = 25\%$, representing the objective values by this HIPP algorithm is 25% deviated from the optimal objective value. Although the HIPP problem is an optimization problem used to model the haplotyping problem, whether an optimal HIPP algorithm does always give a more effective (e.g. of smaller error rate) solution than a non-optimal HIPP algorithm or not has never been evaluated. Here we first calculate the optimality gap for the solution of each non-optimal HIPP algorithm, and then check whether an HIPP algorithm that gives smaller optimality gap is indeed more effective (i.e. has a smaller error rate).

4.2. Experiments on β_2AR gene

We use a sample of the human β_2AR gene identified by Drysdale et al. [15] for our testing. Similar experiments have also been conducted in Stephens and Donnelly [12], Huang et al. [18], Li et al. [20], and Wang and Xu [21]. It contains 18 different genotypes and 13 SNPs from 121 individuals, and should be resolved by the 10 haplotypes related to asthmatic cohort. Among all algorithms we have tested, RTIP, MGP, GHI, and HAP all obtain 10 distinct haplotypes on human β_2 *AR* gene data, thus they are all optimal with respect to the pure parsimony criterion. The error rate of MGP, GHI, and HAP are all 0. However, RTIP has an error rate 6% due to the existence of multiple optimal solutions. Both SDP and PTG conduct randomized mechanism which may result in different solutions even for the same genotype matrix. Thus we apply SDP and PTG to solve this genotype matrix for three times. SDP obtains 12 haplotypes and has an error rate 11% on average. 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

We use the program by Hudson [29] 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 is repeatedly paired. Similar simulated techniques can also be found in Stephens et al. [10], Niu et al. [11], Gusfield [14], Brown and Harrower [17], Huang et al. [18], and Wang and Xu [21].

Recombination is a process during the formation of gametes where portions from the paternal and maternal genome are exchanged. Recombination may cause the haplotypes in offspring 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 r 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 tested recombination rate (i.e. $r = 0$, 4, or 16), 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 30 random test cases for each problem set have been generated. The size of the largest genotype matrix tested in our experiments (i.e. 30×30) is fairly large, compared with the largest cases, 40×10 and 25×10, tested in Wang and Xu [21] and Huang et al. [20], respectively. In fact, the number of SNP affects the running time more than the number of genotypes, especially for some algorithms (e.g. RTIP and GHI) that require enumeration of all candidate haplotype pairs.

Two parameters, optimality gap and error rate, for each algorithm in each test case are recorded for evaluation. For each parameter, we use their average over the 30 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 HAP, since RTIP is already optimal and HAP may not resolve all the genotypes which makes its optimality gap meaningless.

Although the computational time (see Tables 1–3) in our experiments are not for the purpose of efficiency comparison, the results indicate that all the heuristics (MGP, GHI, and PTG) solve the HIPP very quickly. Among these three heuristics, PTG is the most efficient algorithm. MGP is the second efficient since it solves two ILPs with sizes much smaller than the size of RTIP. GHI requires more time than the other two, since it sorts a set of exponential-sized candidate haplotype pairs to select the best one. On the other hand, both RTIP and SDP require much more time than the others. For our convenience, we only

Table 1
Computational time (in seconds) for cases without recombination ($r = 0$).

Problem sets	RTIP	MGP	GHI	PTG	HAP	SDP
10×10 (30/30)	0.06	0.04	0.02	0.01	0.03	1.58
20×10 (30/30)	0.06	0.04	0.03	0.01	0.03	7.69
30×10 (30/30)	0.09	0.06	0.05	0.01	0.03	7.42
10×20 (30/30)	405.85	0.05	126.42	0.01	0.03	15.75
20×20 (30/30)	84.83	0.06	58.65	0.02	0.03	16.71
30×20 (30/30)	9.46	0.16	6.85	0.02	0.04	26.02
10×30 (26/30)	37.87	0.08	14.95	0.02	0.04	17.72
20×30 (27/30)	19.16	0.11	7.70	0.03	0.03	6.95
30×30 (28/30)	26.27	0.09	19.71	0.04	0.03	11.98

Table 2
Computational time (in seconds) for recombination rate $r = 4$ cases.

Problem sets	RTIP	MGP	GHI	PTG	HAP	SDP
10×10 (30/30)	0.08	0.06	0.02	0.01	0.04	4.67
20×10 (30/30)	0.06	0.05	0.03	0.01	0.03	6.05
30×10 (30/30)	0.07	0.06	0.03	0.02	0.03	6.23
10×20 (30/30)	3.08	0.20	0.93	0.02	0.04	4.10
20×20 (30/30)	4.88	0.06	5.98	0.02	0.04	13.89
30×20 (30/30)	2.96	0.12	1.73	0.03	0.02	14.86
10×30 (29/30)	15.35	0.16	5.95	0.02	0.04	3.55
20×30 (28/30)	61.21	1.08	19.78	0.03	0.03	7.53
30×30 (30/30)	26.39	0.17	15.26	0.04	0.08	13.74

Table 3
Computational time (in seconds) for recombination rate $r = 16$ cases.

Problem sets	RTIP	MGP	GHI	PTG	HAP	SDP
10×10 (30/30)	0.05	0.06	0.01	0.01	0.03	2.62
20×10 (30/30)	0.06	0.04	0.02	0.02	0.04	5.71
30×10 (30/30)	0.11	0.04	0.05	0.02	0.04	24.73
10×20 (30/30)	377.54	0.08	29.05	0.02	0.03	4.88
20×20 (30/30)	0.61	0.08	0.25	0.03	0.03	8.96
30×20 (30/30)	1.32	0.16	1.05	0.04	0.03	20.26
10×30 (30/30)	317.46	292.35	18.54	0.02	0.03	8.02
20×30 (30/30)	2681.88	0.19	60.52	0.04	0.03	12.19
30×30 (30/30)	12.62	0.18	4.99	0.06	0.03	23.64

record the cases solvable within two hours for each algorithm. Although SDP can be terminated in two hours in our tests, RTIP may take longer. Thus we list the number of cases solvable within two hours by RTIP. For example, 10×30 (26/30) in the first column of Tables 1, 4, and 7 indicates that only 26 out of the 30 test cases for the 10×30 problem sets are solvable by RTIP within two hours for the cases without recombination. Note that if we include those 10×30, 20×30, and 30×30 test cases that take up to 2 h of computational time in Tables 1 and 2, the average solution time for some solution methods (e.g. RTIP, GHI, and SDP) will be longer than 7200 s (i.e. 2 h). In general, the complexity of an HIPP problem depends more on the number of heterozygous sites, and problem sets of larger n may contain more heterozygous sites and thus usually require more time to solve. On the other hand, problem sets of larger m may not necessarily take more solution

Table 4
Optimality gaps (in %) for cases without recombination ($r = 0$).

Problem sets	MGP	GHI	PTG	SDP
10×10 (30/30)	2.34	4.29	3.45	7.20
20×10 (30/30)	11.91	10.93	5.29	11.83
30×10 (30/30)	6.78	5.99	2.95	12.87
10×20 (30/30)	4.72	9.15	5.31	18.77
20×20 (30/30)	13.59	10.16	7.27	46.64
30×20 (30/30)	17.32	14.02	10.27	62.38
10×30 (26/30)	2.87	6.32	9.33	31.85
20×30 (27/30)	12.18	12.40	11.80	47.61
30×30 (28/30)	22.29	17.89	12.98	66.60

Table 5
Optimality gaps (in %) for recombination rate $r = 4$ cases.

Problem sets	MGP	GHI	PTG	SDP
10 × 10 (30/30)	2.64	4.12	5.09	4.95
20 × 10 (30/30)	8.98	6.11	5.14	6.25
30 × 10 (30/30)	9.46	7.78	5.43	10.51
10 × 20 (30/30)	4.15	8.83	11.10	25.74
20 × 20 (30/30)	14.74	10.20	13.48	46.54
30 × 20 (30/30)	17.08	14.07	19.11	50.10
10 × 30 (29/30)	4.79	9.87	13.88	37.58
20 × 30 (28/30)	11.12	12.05	13.66	50.96
30 × 30 (30/30)	16.04	14.80	16.73	67.63

Table 6
Optimality gaps (in %) for recombination rate $r = 16$ cases.

Problem sets	MGP	GHI	PTG	SDP
10 × 10 (30/30)	6.32	6.77	11.73	8.83
20 × 10 (30/30)	14.24	9.47	17.09	13.73
30 × 10 (30/30)	16.08	12.07	21.34	15.31
10 × 20 (30/30)	3.52	11.05	19.71	25.74
20 × 20 (30/30)	10.00	14.89	22.05	43.54
30 × 20 (30/30)	12.00	13.17	18.19	43.82
10 × 30 (30/30)	3.65	15.22	17.69	29.86
20 × 30 (30/30)	8.89	16.67	23.14	48.85
30 × 30 (30/30)	12.61	13.65	23.80	58.65

Table 7
Error rates (in %) for cases without recombination ($r = 0$).

Problem sets	RTIP	MGP	GHI	PTG	HAP	SDP
10 × 10 (30/30)	13.33	14.67	14.33	14.00	18.67	15.67
20 × 10 (30/30)	6.00	11.67	10.83	9.83	15.83	9.33
30 × 10 (30/30)	4.00	5.89	5.00	6.11	15.56	9.00
10 × 20 (30/30)	28.67	32.33	33.33	28.67	37.00	43.33
20 × 20 (30/30)	13.83	29.00	21.67	16.67	23.17	34.83
30 × 20 (30/30)	7.78	17.67	15.78	10.56	26.00	30.22
10 × 30 (26/30)	44.23	55.00	52.31	48.85	49.23	66.15
20 × 30 (27/30)	23.89	39.63	31.30	36.48	33.33	50.19
30 × 30 (28/30)	14.17	38.81	28.21	26.90	33.69	50.00

time. There is also no clear relationship between the computational time and the value of r in literature and from our testing results.

Tables 4 and 7 record the average performance in the optimality gap and error rate for different algorithms when there is no recombination (i.e. $r = 0$). The results for the cases with recombination rate $r = 4$ and $r = 16$ are summarized in Tables 5, 8, 6, 9, respectively.

For the cases without recombination (i.e. $r = 0$), among those non-optimal algorithms, the optimality gap in Table 4 shows that PTG performs the best, GHI performs the second, MGP performs the third, and SDP performs the fourth. Moreover, the optimality gap of SDP increases dramatically for larger cases. In terms of error rate, RTIP has the smallest error rate than all

Table 8
Error rates (in %) for recombination rate $r = 4$ cases.

Problem sets	RTIP	MGP	GHI	PTG	HAP	SDP
10 × 10 (30/30)	26.33	25.33	22.67	26.67	24.33	24.67
20 × 10 (30/30)	10.50	15.50	10.83	13.00	19.67	13.17
30 × 10 (30/30)	6.33	10.00	8.00	8.33	18.44	9.56
10 × 20 (30/30)	39.00	46.33	42.67	45.33	52.67	48.33
20 × 20 (30/30)	18.67	33.83	27.67	28.67	34.00	46.50
30 × 20 (30/30)	11.67	24.11	18.78	22.00	24.44	36.11
10 × 30 (29/30)	38.62	54.83	51.72	52.76	45.86	66.21
20 × 30 (28/30)	28.21	44.64	37.50	34.64	39.82	57.50
30 × 30 (30/30)	15.11	28.44	22.89	24.44	30.22	47.22

Table 9
Error rates (in %) for recombination rate $r = 16$ cases.

Problem sets	RTIP	MGP	GHI	PTG	HAP	SDP
10×10 (30/30)	24.33	31.00	27.33	35.67	29.33	33.00
20×10 (30/30)	20.17	27.8	23.00	28.67	28.83	24.33
30×10 (30/30)	16.00	20.33	16.78	28.11	24.78	22.44
10×20 (30/30)	40.00	44.00	48.33	57.67	59.67	57.67
20×20 (30/30)	24.83	34.33	29.83	42.50	40.00	50.83
30×20 (30/30)	20.33	33.33	27.22	36.22	31.89	44.78
10×30 (30/30)	57.00	56.67	57.00	63.33	59.33	73.00
20×30 (30/30)	35.17	47.00	41.67	47.50	49.50	58.17

Table 10
Comparative performance of six PHI algorithms.

Performance index	RTIP	MGP	GHI	PTG	HAP	SDP
Time (overall)	5	2	3	1	–	4
Optimality gap ($r = 0$)	–	3	2	1	–	4
Optimality gap ($r = 4$ or 16)	–	1	2	3	–	4
Error rate ($r = 0$)	1	4	3	2	5	6
Error rate ($r = 4$)	1	4	2	3	5	6
Error rate ($r = 16$)	1	3	2	5	4	6

other algorithms in all the cases without recombination (see Table 7), then PTG performs the second, GHI performs the third, MGP performs the fourth, HAP performs the fifth, and finally SDP performs the sixth. All the algorithms have consistent performance on both the optimality gaps and error rates for these cases without recombination. This indicates the pure parsimony criterion does serve its purpose for cases without recombination, so that algorithms that have smaller optimality gaps tend to have smaller error rates.

For the cases with recombination (i.e. $r = 4$ or $r = 16$), among those non-optimal algorithms, the optimality gap in Tables 5 and 6 shows that MGP performs the best, GHI performs slightly worse than MGP, PTG performs the third, whereas SDP has smaller optimality gap for cases up to 10 SNPs but the gap increases dramatically for larger cases. In terms of error rate (see Tables 8 and 9), RTIP again has the smallest error rate than all other algorithms in all the cases with recombination. For algorithm A and B, let $A \succ B$ represent A has smaller error rate than B. In general, $GHI \succ PTG \succ MGP \succ HAP \succ SDP$ for the cases of $r = 4$, and $GHI \succ MGP \succ HAP \succ PTG \succ SDP$ for the cases of $r = 16$. From Tables 7–9, we also find that the error rates are larger for cases with the same number of SNPs but fewer genotypes, due to the lack of sufficient information to infer haplotypes (similar conclusion can also be found in Huang et al. [18] and Wang and Xu [21]).

Although these algorithms do not perform consistently on the optimality gaps and error rates for cases with recombination, our heuristics GHI and MGP do perform very well consistently, and have better effectiveness in most cases, especially for cases with recombination rates. In particular, GHI and MGP have better error rates than HAP in most cases. Since HAP is a widely used haplotyping algorithm in the community, this indicates our proposed algorithms are also practically useful and competitive to the other algorithms based on statistical models. On the other hand, PTG performs very well for cases without recombination rate, but has consistently worse theoretical and practical effectiveness than GHI and MGP, especially when the recombination rate increases.

In summary, Table 10 lists the comparative performance for these six PHI algorithms, and ranks their relative performance along a given parameter (time, optimality gap, or error rate); as the relative performance worsens, the ranking gets higher. From the results, we recommend RTIP for researchers who can bear with longer running time but seek a solution with the smallest error rates. On the other hand, PTG is recommended for researchers seeking a very quick solution with promising quality. For researchers seeking a better solution within a promising schedule, we recommend both GHI and MGP.

5. Conclusions

This paper focuses on issues in solving the PHI problem based on pure parsimony criterion (HIPP). The major contributions of this paper are in two folds. First, we propose two new heuristic algorithms (MGP and GHI) based on the concept of compatible genotype pairs and a weight mechanism that takes the Clark's inference rule into consideration for selecting better candidate haplotypes in a greedy fashion. Approximated lower and upper bounds for the objective value of the HIPP problem can also be derived from the properties of the compatibility graph. Second, we conduct extensive computational experiments for six PHI algorithms on one biological dataset and several simulated datasets. Our experiments are more complete than others in the literature since we evaluate the optimality gap and error rate, which cover both the theoretical and practical effectiveness of HIPP algorithms at the same time. Our results show the pure parsimony criterion does serve its purpose in providing a practically good HIPP solution, since a HIPP algorithm that produces smaller optimality gaps does

usually give smaller error rates. Since seeking the exact optimal HIPP solution is too time-consuming, our results encourage the pursuit of developing more efficient and effective HIPP algorithms. To this end, our proposed heuristics (MGP and GHI) can successfully give very good solutions in a shorter time, and are especially effective for cases with larger recombination rates.

For future research, we suggest to evaluate the error rates for those SAT-based HIPP methods such as SHIPs and RPoly. We also encourage researchers to investigate or improve their HIPP methods, based on the compatibility graph and weight mechanism used by MGP and GHI, as proposed in this paper.

Acknowledgements

I-Lin Wang was partly supported by the National Science Council of Taiwan under Grant NSC96-2221-E-006-015.

References

[1] L. Helmuth, Map of the human genome 3.0, Science 293 (2001) 583–585.
[2] P. Bonizzoni, G.D. Vedova, R. Dnodi, et al, The haplotyping problem: an overview of computational models and solutions, J. Comput. Sci. Technol. 18 (2003) 675–688.
[3] W.Y. Qian, Y.J. Yang, N.N. Yang, L. Chun, Particle swarm optimization for SNP haplotype reconstruction problem, Appl. Math. Comput. 196 (2008) 266–272.
[4] J.L. Wu, J.X. Wang, J.N. Chen, A practical algorithm based on particle swarm optimization for haplotype reconstruction, Appl. Math. Comput. 209 (2009) 363–372.
[5] A.G. Clark, Inference of haplotypes from PCR-amplified samples of diploid populations, Mol. Biol. Evol. 7 (1990) 111–122.
[6] D. Gusfield, Inference of haplotypes from samples of diploid populations: complexity and algorithms, J. Comput. Biol. 8 (2001) 305–323.
[7] L. Excoffier, M. Slatkin, Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population, Mol. Biol. Evol. 12 (1995) 921–927.
[8] J.C. Long, R.C. Williams, M. Urbanek, An E–M algorithm and testing strategy for multiple-locus haplotypes, Am. J. Hum. Genet. 56 (1995) 799–810.
[9] M.E. Hawley, K.K. Kidd, HAPLO: a program using the EM algorithm to estimate the frequencies of multi-site haplotypes, J. Hered. 86 (1995) 409–411.
[10] M. Stephens, N.J. Smith, P. Donnelly, A new statistical method for haplotype reconstruction from population data, Am. J. Hum. Genet. 68 (2001) 978–989.
[11] T. Niu, Z.S. Qin, X. Xu, J.S. Liu, Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms, Am. J. Hum. Genet. 70 (2002) 157–169.
[12] M. Stephens, P. Donnelly, A comparison of Bayesian methods for haplotype reconstruction from population genotype data, Am. J. Hum. Genet. 73 (2003) 1162–1169.
[13] D. Gusfield, Haplotyping as perfect phylogeny: conceptual framework and efficient solutions, in: Annual Conference on Research in Computational Molecular Biology: Proceedings of the Sixth Annual International Conference on Computational Biology, 2002, pp. 166–175.
[14] D. Gusfield, Haplotype inference by pure parsimony, in: Combinatorial Pattern Matching: 14th Annual Symposium, 2003, pp. 144–155.
[15] C. Drysdale, D. McGraw, C. Stack, J. Stephens, R. Judson, K. Nandabalan, K. Arnold, G. Ruano, S. Liggett, Complex promoter and coding region β_2-adrenergic receptor haplotypes alter receptor expression and predict in vivo responsiveness, Proc. Natl. Acad. Sci. USA 97 (2000) 10483–10488.
[16] G. Lancia, C.M. Pinotti, R. Rizzi, Haplotyping populations by pure parsimony: complexity, exact and approximation algorithms, INFORMS J. Comput. 16 (2004) 348–359.
[17] D.G. Brown, I.M. Harrower, Integer programming approaches to haplotype inference by pure parsimony, IEEE ACM Trans. Comput. Biol. 3 (2006) 141–154.
[18] Y.T. Huang, K.M. Chao, T. Chen, An approximation algorithm for haplotype inference by maximum parsimony, J. Comput. Biol. 12 (2005) 1261–1274.
[19] K. Kalpakis, P. Namjoshi, Haplotype phasing using semidefinite programming, in: Proceedings of the Fifth IEEE Symposium on Bioinformatics and Bioengineering, 2005, pp. 145–152.
[20] Z. Li, W. Zhou, X. Zhang, L. Chen, A parsimonious tree-grow method for haplotype inference, Bioinformatics 21 (2005) 3475–3481.
[21] L. Wang, Y. Xu, Haplotype inference by maximum parsimony, Bioinformatics 19 (2003) 1773–1780.
[22] I. Lynce, J. Marques-Silva, SAT in bioinformatics: Making the case with haplotype inference, in: International Conference on Theory and Applications of Satisfiability Testing, Seattle, USA, 2006.
[23] I. Lynce, J. Marques-Silva, Haplotype inference with Boolean Satisfiability, Int. J. Artif. Intell. Trans. 17 (2) (2008) 355–387.
[24] I. Lynce, J. Marques-Silva, S. Prestwich, Boosting haplotype inference with local search, Constraints Int. J. 13 (2008) 1–2.
[25] A. Graca, J. Marques-Silva, I. Lynce, A. Oliveira, Efficient Haplotype Inference with Pseudo-Boolean Optimization, in: Algebraic Biology, Hagenberg, Austria, 2007.
[26] A. Graca, J. Marques-Silva, I. Lynce, A. Oliveira, Efficient haplotype inference with combined CP and OR techniques, in: International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 2008, pp. 308–312.
[27] N. Eén, N. Sörensson, An extensible SAT-solver, in: International Conference on Theory and Applications of Satisfiability Testing (SAT), 2003, pp. 502–518.
[28] N. Eén, N. Sörensson, Translating pseudo-Boolean constraints into SAT, J. Satisfiability Boolean Model. Comput. 2 (2006) 1–26.
[29] R.R. Hudson, Generating samples under a Wright-Fisher neutral model of genetic variation, Bioinformatics 18 (2002) 337–338.