Haplotype Phasing using Semidefinite Programming

Konstantinos Kalpakis and Parag Namjoshi
Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
Email: {kalpakis,nam1}@csee.umbc.edu

Abstract

*Diploid organisms, such as humans, inherit one copy of each chromosome (haplotype) from each parent. The conflation of inherited haplotypes is called the genotype of the organism. In many disease association studies, the haplotype data is more informative than the genotype data. Unfortunately, getting haplotype data experimentally is both expensive and difficult. The haplotype inference with pure parsimony (*HPP*) problem is the problem of finding a minimal set of haplotypes that resolve a given set of genotypes.*

We provide a Quadratic Integer Programming (QIP) formulation for the HPP *problem, and describe an algorithm for the* HPP *problem based on a semi–definite programming (SDP) relaxation of that QIP program. We compare our approach with existing approaches. Further, we show that the proposed approach is capable of incorporating a variety of additional constraints, such as missing or erroneous genotype data, outliers etc.*

1. Introduction

Humans and other mammals are diploid organisms, i.e. they inherit one copy of each chromosome from each parent. Each copy of the chromosome is called a *haplotype*. The conflation of the two inherited haplotypes, which may be identical, is called a *genotype*.

In many disease association studies, the haplotype data is far more informative than the genotype data [12]. However, due to the diploid nature of (most) human cells, finding the haplotypes experimentally is expensive and technically difficult. Thus, the question of inferring haplotypes from genotypes computationally becomes an important and challenging problem.

An allele is a nucleotide at the given site (locus) of a chromosome, and hence a haplotype is a sequence of alleles. A Single Nucleotide Polymorphism (*SNP*) is a mutation (polymorphism) of a single nucleotide. In humans, most of the SNPs are *biallelic*, i.e. only two of the four possible nucleotides are observed in the majority of population. The haplotypes are usually constructed from SNPs instead of a complete DNA sequence. A locus where the same allele is present on both haplotypes is homozygous, otherwise it is heterozygous. Similarly, a genotype where all loci are homozygous is homozygous, otherwise it is heterozygous.

Genotypes and haplotypes in the context of haplotype inference problems are typically represented as vectors of $\{0, 1, 2\}$ [5, 6]. Arbitrarily label the two alleles at any SNP site (locus) with 0 or 1. A haplotype with m SNP sites is represented by an m–dimensional vector, all of whose entries are 0 or 1. A genotype with m sites is represented by an m–dimensional vector g, such that g_i is set to 0, 1, or 2 depending on whether the observation for genotype at site i is $\{0\}$, $\{1\}$, or $\{0, 1\}$, respectively. A heterozygous genotype locus takes the value 2, while the homozygous locus takes value 0 or 1 depending upon the allele at the site. Two haplotypes h and h' of size m conflate to produce a genotype g as follows: $g_i = 0, 1,$ or 2 when $h_i = h'_i = 0$, $h_i = h'_i = 1$, and $h_i \neq h_j$ respectively, for each site i.

Two haplotypes are said to resolve a genotype if their conflation is that genotype. The fundamental problem in SNP and Haplotype Analysis is the genotype phasing problem: finding a set a haplotypes that resolve a given set of genotypes. This problem has many equally plausible solutions and thus as stated is not very meaningful. Observe that a homozygous genotype g can only be resolved by two haplotypes identical to g, while a heterozygous genotype can only be resolved by pairs of distinct haplotypes. A genotype with k heterozygous loci can be resolved by 2^{k-1} distinct haplotype pairs.

Many studies show that the number of haplotypes observed in natural populations are far smaller than all the combinatorial possibilities. This suggests the Haplotype phasing with Pure Parsimony (**HPP**) problem: find a minimal set of haplotypes that resolve a given set of genotypes [7].

In this paper, we consider the HPP problem. We provide a quadratic integer program (QIP) for the HPP problem and describe a simple heuristic that is based on a vector program

relaxation of that QIP that is solved via semi–definite programming techniques. We provide preliminary results of an experimental comparison between the proposed approach, Clark's rule, and a relaxation of Gusfield's [7] TIP scheme. Moreover, we show that the proposed scheme has high expressive power, by showing that it extends easily to handle a number of variations to the basic HPP problem, such as: partial genotype and/or haplotype knowledge, shared haplotype information, errors in the input genotypes, and outlier genotypes.

The rest of the paper is organized as follows. We review previous work in section 2, and provide preliminaries definitions and notations in section 3. In section 3 we present the quadratic integer program for HPP and the SDP–based heuristic. Experimental results are given in section 4. Extensions to the basic QIP are presented in section 5. We conclude in section 6.

2. Previous Work

Clark [2], in his seminal work, describes a greedy inference rule to get a set of haplotypes resolving a set of genotypes. Starting with a set of haplotypes H that resolves all the homozygous genotypes, Clark's rule does the following: for each still unresolved genotype g, if there is a pair (h, h'), $h \in H$, that resolves g, then add h' to H. The algorithm terminates if no progress can be made. Clark's rule may terminate with some genotypes unresolved (orphans); the rule can be modified to include a pair of haplotypes that resolve an orphan genotype, and continue as before. The solution obtained by Clark's rule is sensitive to the order genotypes are resolved.

Gusfield [7] introduces the pure parsimony approach for the haplotype inference problems and provides the TIP scheme to HPP: enumerating all distinct haplotypes that resolve each heterozygous genotype, and then use an Integer linear Program (IP) to select a minimal set haplotypes for HPP. The TIP scheme is practical for genotypes with a small number of heterozygous loci. Wang and Xu [16] give a branch–and–bound implementation (HAPAR) of the TIP scheme.

Brown and Harrower [1] give an alternate polynomial sized IP–formulation for the HPP problem (HB-IP). Using 2n haplotype vectors, they include linear constraints to ensure that haplotype $2i-1$ and $2i$ explain genotype i. 0–1 variables are introduced for each pair of haplotypes which take value 1 if the two haplotypes i, j are distinct and 0 otherwise. Finally, 0–1 variables are introduced for each haplotype to count the number of distinct haplotypes, which take value 1 if haplotype i is distinct from haplotypes 1 to $i-1$ and 0 otherwise. The objective function is to minimize the sum of the counting variables i.e. minimize the number of distinct haplotypes used to explain the genotypes. Further the cuts are given which strengthen the formulation. The experimental results in [1] show that larger problems can be solved using this approach compared to Gusfield's IP.

EM and MCMC based algorithms for haplotype inference are given in [3, 11, 13]. Halldórson et al [8] give a nice comprehensive review of problems arising in SNP and haplotype inference.

3. Preliminaries

We provide a brief introduction to Semi–definite programming. For a more thorough introduction the interested reader is referred to [15, 9] and references therein.

A *vector program* is the problem of optimizing a linear function of inner products of vector variables subject to constraints that are also linear functions of inner products of those vector variables.

An $n \times n$ real symmetric matrix A is *positive semidefinite* matrix, and we write $A \succeq 0$, if all its eigenvalues are non–negative. For any positive semidefinite matrix A there is a real matrix V such that $A = V^T V$; such a V can be computed from a Cholesky factorization of A or as $A^{1/2}$ from its Singular Value Decomposition (SVD). A *semi definite program* (SDP) is a problem of optimizing a linear function of the elements of a variable $n \times n$ matrix X, subject to linear constraints on the elements of X and the constraint that $X \succeq 0$. SDP programs can be solved numerically to within additive error ϵ in time polynomial in n, $\log(1/\epsilon)$, and the number of constraints [4].

For any vector program on n vector variables $v_1, v_2, \ldots, v_n$ there is an equivalent SDP program on matrix variable $X = (x_{i,j}) \succeq 0$, by requiring that each $x_{i,j}$ equals the inner product $v_i^T v_j$ of v_i and v_j. Given $X = W^T W$, the vectors $v_1, v_2, \ldots, v_n$ will be given by the columns of W. There are many ways to get a matrix W from X. In this paper, we assume that $W = X^{1/2}$ and is computed via X's SVD in a standard way. Moreover, since SDP's can be solved efficiently, it follows that vector programs can also be solved efficiently.

A *quadratic function* (or constraint) is a (multi–variate) polynomial with real coefficients and of total degree at most 2. A quadratic function/constraint is *strict* if all monomials have total degree either 0 or 2. A *quadratic integer program* (QIP) is a problem of optimizing a quadratic objective function of integer variables subject to quadratic constraints. If the objective functions and all the constraints are strict, then we have a *strict* QIP.

A strict QIP P can be relaxed into a vector program as follows. With each integer variable x_i of P, associate a vector variable $\underline{x_i}$, and then replace each non–constant monomial $x_i x_j$ in P, with the inner product $\underline{x_i}^T \underline{x_j}$. This vector program relaxation of P can be solved efficiently. Vector program relaxations to strict QIP programs have provided

good approximation algorithms for various computationally hard problems (e.g. MAX-CUT, MAX-k-SAT, etc) [15].

A *0–1 variable* (± 1 *variable*) takes values in $\{0, 1\}$ ($\{-1, 1\}$). We use the terms 0–1 and boolean variables interchangeably, based on the natural mapping $1 \equiv \text{true}$ and $0 \equiv \text{false}$. It is easy to show the following equivalences: logical **not** $\overline{x} \equiv 1 - x$, logical **and** $x \wedge y \equiv xy$, logical **or** $x \vee y \equiv x + y - xy$, and logical **xor** $x \oplus y \equiv x + y - 2xy$.

A bijection between a 0–1 variable x and a ± 1–variable $\widehat{x}$ is given by

$$x = \frac{1}{2}\left(1 + z\widehat{x}\right), \tag{1}$$

where z is a new ± 1 variable [15]. Observe that for any two 0–1 variables x, y, $x \cdot y = \frac{1}{4}\left(1 + \widehat{x} \cdot \widehat{y} + z \cdot \widehat{x} + z \cdot \widehat{y}\right)$, since $z^2 = 1$. Therefore, every QIP with n 0–1 variables can be converted into an equivalent strict QIP with $(n + 1)$ ± 1 variables.

Randomized rounding is used for obtaining an approximate solution to the MAX–CUT and MAX–2–SAT problems from an vector program (i.e. SDP) relaxation of these problems (see [15] and references therein). The idea is to choose a random hyperplane through the origin and assign values ± 1 to the ± 1 variables depending on which side of the hyperplane each variable's vector falls into. We call this the *standard randomized rounding* for SDP relaxations of QIPs. It can be shown that two unit vectors v_1, v_2 lie on different sides of a random hyperplane through the origin with probability $\theta/\pi = \cos^{-1}(a_1^T a_2)/\pi$. Moreover, when the standard randomized rounding is applied to the variables $\widehat{x}$ that correspond to the 0–1 variables x of a QIP, x is rounded to 1 with probability equal to

$$\mathrm{P}_{\mathrm{srr}}[x = 1] = 1 - \frac{\cos^{-1}\left(\underline{z}^T \underline{x}\right)}{\pi} \tag{2}$$

where $\underline{x}$ is the vector variable corresponding to $\widehat{x}$ (and x as well).

4. An SDP–based Heuristic for HPP

We give a convenient algebraic formulation of the HPP problem, describe a QIP for it, and then describe a heuristic for the HPP problem that relies on a vector program (SDP) relaxation of the QIP.

4.1. Arithmetic genotypes and the k–HPP problem.

We associate with any genotype g an *arithmetic genotype* δ, which is an m–dimensional vector with $\delta_i = 0$, 1, or 2 if g_i is equal to 0, 2, or 1 respectively, for $i = 1, 2, \ldots, m$. Arithmetic genotypes are beneficial due to the following important observation: if the conflation of haplotypes h and h' is genotype g, then $\delta = h + h'$, where $+$ is the standard vector addition. Hereafter, for brevity, we refer to an arithmetic genotype simply as genotype, and assume without loss of generality (w.l.o.g.) that all genotypes are distinct.

A set of n genotypes with m sites is represented by an $n \times m$ matrix $G = (g_{i,j})$, whose rows correspond to the genotypes. Let $\Delta = (\delta_{i,j})$ be the $n \times m$ matrix whose rows are the corresponding arithmetic genotypes. Similarly, a set of k haplotypes with m sites is represented by an $k \times m$ haplotype matrix $H = (h_{t,j})$, whose rows correspond to the haplotypes. Observe that if the haplotypes in H resolve the genotypes in G then

$$\Delta = S \cdot H, \tag{3}$$

where $S = (s_{i,t})$ is an $n \times k$ matrix with elements in $\{0, 1, 2\}$, and with each row having a row–sum of 2. We call S the *selector matrix*. Each row of S has either two 1's or a single 2, with all other entries 0. Observe that the non–zero entries of the ith row of S provide us with the two haplotypes that conflate to resolve the ith genotype. If these haplotypes are distinct, then the row has exactly two 1's.

The **k–HPP problem** is: given $n \times m$ genotype matrices G and Δ, find a $n \times k$ selector matrix S and an $k \times m$ haplotype matrix H such that $\Delta = S \cdot H$, such that S has the smallest number of non–zero columns. Since n genotypes can always be resolved by $2n$ haplotypes, the HPP problem is the same as the $2n$–HPP problem.

There is a simple method to find the selector matrix S given the (arithmetic) genotype matrix Δ and the haplotype matrix H. Just compute all the pairwise conflations of haplotypes and for each genotype, choose one pair that resolves it.

To find a haplotype matrix H given Δ and S, we efficiently solve a 2–SAT problem with km variables and $2nm$ clauses constructed as follows. Start with an empty set of clauses F. For each locus j of genotype i, resolved by haplotypes t and l as specified by S, add add to F the two clauses $\overline{h_{t,j}} \wedge \overline{h_{l,j}}$, $h_{t,j} \wedge h_{l,j}$, or $(h_{t,j} \vee h_{l,j}) \wedge (\overline{h_{t,j}} \vee \overline{h_{l,j}})$ if $\delta_{i,j}$ is equal to 0, 1, or 2 respectively.

Bounds on the number of Haplotypes:

The lower bound of $\sqrt{n}$ for the HPP problem is well known. Similarly $2\,n$ is the well known upper bound obtained by resolving each genotype independently of others. Arithmetic genotypes give an alternate lower bound on the number of haplotypes required to explain the given genotypes.

Proposition 1 *The rank of the genotype matrix Δ is a lower bound on the number of haplotypes required to explain the genotypes.*

Proof. Let H be an optimal solution to the HPP problem

and let S be the corresponding selector matrix.

$$\begin{aligned} \Delta &= S\,H, \\ rank(\Delta) &\leq min\{rank(S), rank(H)\}, \\ rank(\Delta) &\leq rank(H). \end{aligned}$$

Since the number of distinct rows (haplotypes) in matrix H is at least $rank(H)$ and $rank(\Delta) \leq rank(H)$, $rank(\Delta)$ is a lower bound on the number of haplotypes required to explain the given genotypes. ■

It should be noted that $rank(\Delta) \leq min\{n, m\}$.

4.2. A QIP and a heuristic based on its SDP relaxation

Given an $n \times m$ genotype matrix Δ and either a $n \times k$ selector matrix S or an $k \times m$ haplotype matrix H it is fairly easy to find the other matrix so that $\Delta = S \cdot H$. Next, we describe a QIP for the considerably more difficult k–HPP problem.

Our QIP is given in Table 4.2 and is using the following variables: (1) $s_{i,t} \in \{0, 1, 2\}$ = number of times haplotype t is used in a resolution of genotype i, (2) $\tilde{s}_{i,t} \in \{0, 1\} = 1$ if haplotype t is used in a resolution of genotype i, (3) $h_{t,j} \in \{0, 1\}$ = value of locus j of haplotype t, (4) $\phi_t \in \{0, 1\} = 1$ if haplotype t is used to resolve any genotype, (5) $r_i \in \{0, 1\} = 1$ if genotype i resolved, (6) $\epsilon_{i,j} \in \{0, 1\} = 1$ if an edit/correction is done at locus j of genotype i, (7) $\epsilon^+_{i,j}$ ($\epsilon^-_{i,j}$) $\in \{0, 1, 2\}$ the amount of positive (negative) correction at locus j of genotype i,

The objective function is to minimize the number of haplotypes used (which is equivalent to the number of non–zero columns of the selector matrix). Any solution of this QIP must satisfy the following constraints:

Constraint (5): the locus j of genotype i should be correctly resolved. For convenience we call this constraint the locus resolution constraint. This constraint is omitted if the value of $g_{i,j}$ is missing.

Constraint (6): two haplotypes (not necessarily distinct) should be used for resolving a genotype.

Constraint (7): haplotype t should be selected, i.e. $\tilde{s}_{i,t}$ should be 1, if it is used in a resolution of genotype i. If the haplotypes used in a resolution of a genotype are always distinct, then $\tilde{s}_{i,t}$ is always equal to $s_{i,t}$ and this constraint and the variable $\tilde{s}_{i,t}$ are not needed. For example this happens whenever a genotype is heterozygous. Though homozygous genotypes can be resolved immediately, rendering this constraint and the variables $\tilde{s}_{i,t}$ redundant, we choose to include them because it enables a simple way to describe various extensions to the HPP problem.

Constraint (8): haplotype t is selected, i.e. ϕ_t equals 1, if it is used to resolve any genotype.

Constraint (9): two genotypes i and j can not share a haplotype if they differ in any homozygous locus. This constraint, though is implied by the others, is included since is useful in the numerical solution of the QIP. We define the *homozygous Hamming distance* $d_{\text{homo}}(i, j)$ between two genotypes g_i and g_j as the number of homozygous loci/sites that the two genotypes differ.

Note that this QIP has $\Theta(nk + km)$ variables and constraints.

We can easily incorporate partial knowledge in QIP: simply replace the known variable with its value. Note that the QIP already models the k–HPP problem with partial genotypes. Further, alternate quadratic objective functions are possible.

Since the HPP problem is NP–hard, it is unlikely that an optimal solution to the k–HPP and this QIP can be obtained efficiently.

A equivalent strict QIP can be obtained from this QIP by introducing a new ± 1 variable z and replacing each 0–1 variable with its corresponding ± 1 variable (as discussed in 3). This strict QIP has a vector program relaxation, and thus an SDP relaxation, that can be solved efficiently to any desired degree of accuracy. We describe next a heuristic for obtaining an approximate solution to the QIP from a solution to its SDP relaxation.

The idea of the heuristic is to repeatedly fix entries of the selector matrix until all its entries are fixed to 0 or 1. After all the entries of the selector matrix are fixed, we find a haplotype matrix by solving a 2–SAT problem as described earlier. We do not use the standard randomized rounding, since the likelihood of obtaining feasible selector and haplotype matrices is decreasing fast with n, m and k due to constraints (5)– (8). Instead, we fix entries $s_{i,t}$ (and thus $\tilde{s}_{i,t}$), one at a time, in decreasing order of their probability $\mathrm{P}_{\text{srr}}[\tilde{s}_{i,t}]$ of being rounded to 1 under the standard randomized rounding. The details of the heuristic are given in Table 4.2.

Due to an incorrect fixing of $\tilde{s}_{i,t}$ and $s_{i,t}$ it may happen that the SDP relaxation of the the QIP problem P is infeasible at step 8, or the 2–SAT problem at step 15 is unsatisfiable. * This situation can be handled by incorporating backtracking into the heuristic. The theoretical analysis of performance ratio (ounds on the optimality) of the proposed heuristic remains open.

5. Experiments

We experiment with three algorithms for the HPP problem: Clark's rule, linear relaxation of the Gusfield's TIP scheme (TIP-APPROX), and our proposed approach. For

*In our experiments in section 5, backtracking was needed in 18 and 22 times in two datasets each consisting of 80 randomly generated instances.

Table 1. Quadratic Integer Program for the k–HPP problem.

Objective :

$$\text{minimize } \sum_{t=1}^{k} \phi_t \tag{4}$$

Constraints :

$$\sum_{t=1}^{k} s_{i,t} h_{t,j} = \delta_{i,j}, \quad \text{for } i = 1, 2, \ldots, n \text{ and } j = 1, 2, \ldots, m. \tag{5}$$

$$\sum_{t=1}^{k} s_{i,t} = 2, \quad \text{for } i = 1, 2, \ldots, n. \tag{6}$$

$$\mathrm{s}_{i,t} = \begin{cases} 2\tilde{s}_{i,t}, & \text{if } i \text{ is homozygous} \\ \tilde{s}_{i,t}, & \text{if } i \text{ is heterozygous} \end{cases} \quad \text{for } i = 1, 2, \ldots, n \text{ and } t = 1, 2, \ldots, k. \tag{7}$$

$$\sum_{i=1}^{n} \tilde{s}_{i,t} \leq n\phi_t, \quad \text{for } t = 1, 2, \ldots, k. \tag{8}$$

$$\sum_{t=1}^{k} \tilde{s}_{i,t} \tilde{s}_{j,t} = 0, \quad \text{for all } 1 \leq i, j \leq n \text{ whenever } d_{\mathrm{homo}}(g_i, g_j) \geq 1. \tag{9}$$

simplicity, we refer to the proposed approach as the QIP approach.

The first algorithm is Clark's inference rule with the following extension: if there is an unresolved (orphan) genotype that can only be resolved by a pair (h, h') with $h, h' \notin H$, add both h and h' to H, and continue.

The second method implemented is the TIP-APPROX. Gusfield's TIP enumerates all haplotypes that can resolve each genotype, and then uses an integer program (IP) to select a minimal set of haplotypes for HPP. Gusfield [7] describes RTIP, which is TIP with certain efficiency improvements. Using a linear programming (LP) relaxation of TIP's IP, and then rounding each haplotype indicator variable to the nearest integer (0 or 1), we obtain a TIP-APPROX solution to the HPP problem. The LP relaxation gives integer solutions in many instances as observed in [7], but the method becomes computationally expensive for genotypes with moderately many heterozygous loci due to the enumeration stage of the scheme.

The third method is an *non–backtracking* implementation of the for k–HPP in section 4.2, with $k = 2n$. The MATLAB package SDPT 3.02 [14] is used to solve the SDP relaxation of the QIP in our algorithm. All experiments are done on a single CPU MATLAB on a Dual Xeon 2.4 GHz desktop with 1 GB memory.

Synthetic dataset A is generated as follows. For each triplet $(n, m, k_*) = (5, 5, 5)$, $(8, 8, 8)$, $(10, 10, 10)$, and $(15, 15, 15)$, 20 instances of the HPP problem were generated as follows. The genotypes were generated by random mating of k_* haplotypes. Each locus of each haplotype took value 0 or 1 independently and with equal probability.

We generate the synthetic dataset B using Hudson's program [10]. We generate haplotypes to simulate a population of k_* haplotypes with m loci "sampled" from a diploid population of size 10^6 with neutral mutation rate 1.5×10^{-6}, and at recombination levels $\rho = 0, 16$ and 40, as in Gusfield [7]. The k_* haplotypes are then randomly mated to produce the n genotypes. The dataset consists of 20 instances for each ρ above and $(n, m, k_*) = (5, 5, 5)$ and $(10, 10, 10)$.

Table 3 shows the average number k_o of haplotypes found by each approach as well as the average elapsed time in seconds for the instances in datasets A and B. We observe that, with respect to the QIP approach, the CLARK approach uses up to 100% more haplotypes. Furthermore, TIP is limited by the number of heterozygous loci (and thus $\|E\|$) of its genotypes —- TIP-APPROX, our implementation of the TIP scheme, did not solve any instances with 15 genotypes in dataset A due to insufficient memory.

Our current implementation of the QIP approach does not use backtracking, and it could not solve 18 and 22 out of the 80 and 120 instances in datasets A and B respectively.

Moreover, observe that the rank for the (arithmetic) genotype matrix Δ, provides a tighter lower bound on the HPP than $\sqrt{n}$. In addition, the QIP approach provides solutions that, on the average, are within 10% of the rank lower–bound, and thus optimal within 10% for the two datasets in the experiments.

6. QIP extensions for the k–HPP variants.

The scheme described previous section is quite flexible and capable of handling some useful variations of k–HPP problem. To do so, we just need to append and/or modify constraints of the QIP in Table 4.2, and use the heuristic in Table 4.2 with the extended QIP. Observe that the QIP in Table 4.2 handles partial genotypes by simply omitting the locus resolution constraints for the loci with missing data.

6.1. Shared haplotype information.

If we know that genotypes i and j must share exactly one haplotype, then we include the constraint (10). This is useful when it is known that i and j have a parent–child

Table 2. Backtracking algorithm for the k–HPP problem using the SDP relaxation of the QIP.

```
1   let P be the QIP program in Table 4.2
2   let V be an initially empty set of s̃_{i,t} variables that have been fixed to 0 or 1
3   let F be an initially empty stack of s̃_{i,t} variables in V
4   while V does not contain all the s̃_{i,t} variables in P do
5       set backtrack to false
6       let P' be the program P modified to take into consideration all the fixed variables in V
7       solve the vector program relaxation of P' (via its corresponding SDP relaxation)
8       if the relaxation is feasible then
9           find a variable s̃_{i,t} with the maximum P_srr[s̃_{i,t} = 1] among those not in V
10          fix s̃_{i,t} to 1, add it to V, and push it onto the stack F
11          // Infer values for additional entries of S that can be fixed
12          for each row of S whose fixed entries sum to 2 do
13              fix the remaining s̃_{i,t} variables of that row to 0 and add them to V.
14          end–for
15          if the 2-SAT for the genotypes whose S rows are completely fixed is unsatisfiable then
16              set backtrack to true
17          endif
18      else
19          set backtrack to true
20      endif
21      if backtrack is true then
22          while the variable x at the top of the stack F is fixed to 0 do
23              remove x from V and pop it off the stack
24          end–while
25          if the variable x at the top of the stack F is fixed to 1 then
26              remove from V all variables that were fixed to 0 due to x being fixed to 1
27              fix x to 0 instead
28          else
29              stack is empty – return INFEASIBLE
30          endif
31      endif
32  end–while
33  find a haplotype matrix H by solving the 2–SAT problems that corresponds to G and S.
34  return S and H
```

Table 3. Experimental comparison of approaches on synthetic instances solved by the QIP approach.

					CLARK		TIP–APPROX			QIP	
n	m	k_*	ρ	rank(Δ)	k_o	time	k_o	time	$\|E\|$	k_o	time
Synthetic dataset A (Random)											
5	5	5		3.60	5.95	0.01	5	0.19	19.50	4.56	17.48
8	8	8		6.00	11.80	0.02	7.80	7.88	117.95	7.29	164.02
10	10	10		9.00	16.60	0.04	10.05	131.62	411.55	9.33	444.71
15	15	15		12.35	26.85	0.15	N/A	N/A	N/A	13.13	7241.20
Synthetic dataset B (Hudson)											
5	5	5	0	2.00	4.15	0.01	3.45	0.22	23.60	3.65	16.08
5	5	5	16	4.00	4.80	0.00	4.05	0.14	16.00	4.00	14.83
5	5	5	40	3.60	5.95	0.01	5.00	0.19	19.50	4.56	17.48
10	10	10	0	5.13	7.67	0.01	6.10	32.23	156.15	5.80	251.00
10	10	10	16	5.40	9.20	0.02	6.27	50.46	188.35	5.80	259.83
10	10	10	40	6.87	11.27	0.02	8.00	45.57	182.85	6.90	258.40

relationship. Note that this constraint allows *no* mutations.

$$\sum_{t=1}^{k} \tilde{s}_{i,t}\tilde{s}_{j,t} = 1, \tag{10}$$

for all applicable genotypes $1 \leq i, j \leq n$.

6.2. Input errors and loci editing.

It is possible that there are errors in the input genotype data. Corrections/edits to loci can be dealt with by modifying the locus resolution constraint and including some additional constraints and variables. The amount of editing at a locus $g_{i,j}$ is a number in $\{-2, -1, 0, 1, 2\}$, and can captured by the expression $(2\psi_{i,j} - 1)(\epsilon^+_{i,j} + \epsilon^-_{i,j})$ where $\psi_{i,j}, \epsilon^+_{i,j}$, and $\epsilon^-_{i,j}$ are all $\{0, 1\}$ variables. The $\epsilon^+_{i,j} + \epsilon^-_{i,j}$ is the amount of the correction to be applied to the site i, j while $(2\psi_{i,j} - 1)$ gives the sign of that correction. First, we replace the locus resolution constraint with the constraint (11).

$$\sum_{t=1}^{k} s_{i,t}h_{t,j} = \delta_{i,j} + (2\psi_{i,j} - 1)(\epsilon^+_{i,j} + \epsilon^-_{i,j}) \tag{11}$$

for $i, j = 1, 2, \ldots, n$. Second, we include the constraint (12) to force the correct value for the locus edit indicator variable $\epsilon_{i,j}$.

$$\epsilon^+_{i,j} + \epsilon^-_{i,j} \leq 2\epsilon_{i,j}, \quad \text{for } i, j = 1, 2, \ldots, n. \tag{12}$$

Third, we include the constraint (13) for limiting the number of loci that are edited/corrected to at most $\epsilon_{\max}$, a user specified constant.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \epsilon_{i,j} \leq \epsilon_{\max}. \tag{13}$$

Note that minimizing $\epsilon_{\max}$ can also become the objective function of the QIP instead of minimizing the number of haplotypes selected.

6.3. Outliers.

To permit for up to n_{outliers} genotypes to be skipped and not resolved, treating them as outliers, we need to ensure that constraints on the selector and haplotype matrix elements are enforced only when a genotype is resolved (and thus is not treated as an outlier). We achieve that by doing the following:

- For every genotype i, include a $0-1$ variable r_i that is 1 iff that genotype is resolved (and thus it is not treated as an outlier)
- include the constraint (14)

$$\sum_{i=1}^{n} r_i \geq n - n_{\text{outliers}} \tag{14}$$

- multiply the right hand–side of constraint (6), on the number of haplotypes used to resolve a genotype, with r_i. Note that whenever r_i is 0, the ith row of the S matrix is forced to be 0.
- replace $\delta_{i,j}$ with $r_i \cdot \delta_{i,j}$ in the locus resolution constraint (5) (or constraint (11) if applicable).

- replace the constraint (9) with the constraint

$$\sum_{t=1}^{k} \tilde{s}_{i,t}\tilde{s}_{j,t} \leq 2(1 - r_i r_j), \tag{15}$$

  for all $1 \leq i, j \leq n$, and whenever $d_{\text{homo}}(g_i, g_j) \geq 1$. Note that this constraint is enforced only if both genotypes are resolved.

- multiply the right hand–side of constraint (10), on sharing a haplotype between two genotypes, with $r_i \cdot r_j$.

7. Conclusion

We present a QIP for the k–HPP problem, together with a heuristic that is based on an SDP relaxation of that QIP. The proposed approach is capable of incorporating simultaneously a variety of additional constraints and prior information (such as shared partial genotypes, haplotypes, input errors, outliers etc.) We provide a preliminary experimental comparison of the proposed approach with two existing approaches.

Future work includes theoretical analysis and a thorough experimental evaluation of the proposed approach.

References

[1] D. Brown and I. Harrower. A new integer programming formulation for the pure parsimony problem in haplotype analysis. In *WABI*, pages 254–265, 2004.

[2] A. Clark. Inference of haplotypes from PCR amplified samples of diploid populations. *Molecular Biology and Evolution*, 7(2):111–133, 1990.

[3] L. Excoffier and M. Slatkin. Maximum likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5):921–927, 1995.

[4] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer–Verlag, second corrected edition edition, 1993.

[5] D. Gusfield. Inference of Haplotypes in Samples of Diploid Populations: Complexity and Algorithms. *Journal of Computational Biology*, 8(3):305–323, 2001.

[6] D. Gusfield. Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *Proc. of the 6th Intnl. Conference on Computational biology*, pages 166–175. ACM Press, 2002.

[7] D. Gusfield. Haplotyping by Pure Parsimony. In *Combinatorial Pattern Matching (LNCS 2676)*, pages 144–155. Springer–Verlag, June 2003.

[8] B. Halldórsson, V. Bafna, N. Edwards, R. Lippert, S. Yooseph, and S. Istrail. Combinatorial problems arising in SNP and Haplotype Analysis. In *4th Conf. on Discrete Math. and Theoretical Computer Science*, pages 26–47, 2003.

[9] C. Helmberg. Semidefinite Programming for Combinatorial Optimization. Technical report, ZIB Report 00–34, October 2000.

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

[11] J. C. Long, R. C. Williams, and M. Urbanek. An E–M algorithm and testing strategy for multiple-locus haplotypes. *American Journal of Human Genetics*, 56(799-810), 1995.

[12] National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov/ about/primer/snps.html.

[13] M. Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68:978–989, 2001.

[14] K.-C. Toh, M. J. Todd, and R. H. Tutuncu. SDPT3–a MATLAB software package for semidefinite-quadratic-linear programming, version 3.0. Technical report, National University of Singapore, August 2001.

[15] V. V. Vazirani. *Approximation Algorithms*. Springer–Verlag, 2003.

[16] L. Wang and Y. Xu. Haplotype Inference by Parsimony. *Bioinformatics*, 19(14), 2003.