Haplotype Inference by Pure Parsimony via Genetic Algorithm

Rui-Sheng Wang[1,*] Xiang-Sun Zhang[1] Li Sheng[2]

[1] Academy of Mathematics & Systems Science
Chinese Academy of Sciences, Beijing 100080, China
[2] Department of Mathematics
Drexel University, Philadelphia PA 19104, USA

Abstract *Haplotypes are specially important in the study of complex diseases since they contain more information about gene alleles than genotype data. However, getting haplotype data via experiments methods is techniquely difficult and expensive. Thus, haplotype inference through computational methods is practical and attractive. There are several models for inferrings haplotype from population genotypes, of which we are interested in the pure parsimony model. This problem has been proved to be an NP-hard problem, so the goal of this paper is to design a heuristic method to obtain good solutions within acceptable time. A heuristic method based on genetic algorithm is presented for haplotype inference under pure parsimony criterion. The algorithm was tested on a variety of biological data and simulated data. In comparison with the exact algorithm HAPAR (based on a branch and bound algorithm), experiment results show that the method can obtain optimal solutions in almost all cases and runs much faster than HAPAR when the number of genotypes or SNP sites is large. It is suited for haplotype inference in relative large haplotype blocks because the algorithm is fast and its running time is not exponentially increased with input size.*

Keywords haplotype inference; pure parsimony; genetic algorithm

1 Introduction

In post-genome era, the investigation of genetic differences in a population is attracting increasing attention and will be one of the main topics in genomics. It is generally accepted that some regions of variation in DNA sequences are responsible for genetic diseases and phenotype difference since all human share about 99.9% identity at the DNA level [12]. Genetic variation that involves a single nucleotide is called Single Nucleotide Polymorphism (SNP) which is the most frequent form among various genetic differences.

*Corresponding author. Email address: wangrsh@amss.ac.cn.

In diploid organism, a complete genome contains a pair of chromosomes, one inherited from each parent. A SNP is a particular nucleotide position in a pair of chromosomes and usually has two values (nucleotides) of four for different people in a population. We called these values *alleles*, using 0 to denote the wild type allele and 1 to denote the mutant type allele. The SNP sequence on each copy of a pair of chromosomes is called a *haplotype*. The mixed SNP data on the two copies is called a *genotype*. A genotype gives two nucleotides at each SNP site on a pair of chromosomes, but it does not indicate which chromosome each nucleotide is in. For a genotype, if alleles on both chromosomes at a SNP site are identical, we call this SNP site *homozygous*, otherwise *heterozygous*. Generally, haplotypes are more important in predicting disease because they have more information about gene alleles inheriting together than genotypes [17]. However, experimentally determining haplotype data is time-consuming and expensive while genotypes are much easier to get, so it is genotypes rather than haplotypes that are usually obtained. Thus, haplotype inference through computational methods is practical and attractive.

The haplotype inference problem is to resolve the heterozygous sites in a set of genotypes, i.e. to determine which copy of a pair of chromosomes each allele belongs to. As is well known, without any biological insight or genetic model, we can not recover the "true" haplotypes from genotypes, because there may be an exponential number of possible haplotypes. If we arbitrarily select a pair of haplotypes among them for a genotype, the haplotype inference problem is trivial and we do not know which one is true. Biological Experiments [4] show that human chromosomes in a population have block structure, where no or few recombinations could occur within each block. Blocks of limited haplotype diversity make the haplotype inference problem somewhat easier than the case that a lot of recombination events appear. So methods for haplotype inference are mostly concerned with the analysis of a specific block in the population. Some researchers have given several haplotype inference models based on these biological insights from different views. Among them, the inference method in [3,8] is a method by using a general inference rule and a parsimonious principle that a valid solution is usually the one resolves the largest number of genotypes. Another class of methods [1, 6, 9] for haplotype inference is based on specific biological models — the coalescent model and the infinite model. Besides, statistical methods [15, 18] have been also used to solve the haplotype inference problem. Reviews about the haplotype inference problem can be found in [2]

The pure parsimony criterion for the haplotype inference problem was proposed in [10] and its reasonability and biological meanings were illustrated in [10, 19]. This criterion is based on the fact that in natural populations, the number of observed distinct haplotypes is vastly smaller than the number of combinatorially possible haplotypes. Haplotype inference under the pure parsimony criteria (for convenience, HIPP) has been proved to be an NP-hard problem [14]. Literature [10] solves this problem via integer linear programming. Literature [19] gives a branch and bound algorithm called HAPAR. These methods are very efficient for small size problems, but their implementation time is increasing exponentially as the number of heterozygous sites in genotypes becomes large. It has been noted [11]

that for haplotype inference, exponential-time algorithms are only practical for SNPs around one gene, so approximation algorithms or meta-heuristic algorithms are needed for SNPs around multiple genes. Literature [14] gives approximation algorithms with error guarantee rate of 2^{k-1} (where k is the upper bound of the number of heterozygous sites in a genotype). In this paper, we design a heuristic method based on genetic algorithm for the HIPP problem (called GAHAP). It is intended to solve problems of relative large size — haplotype inference in blocks with large SNP sites or genotype set. Experimental results based on biological and simulated data show that the algorithm can achieve this goal. It can find optimal solutions returned by the exact algorithm HAPAR in almost all cases and runs much faster than HAPAR when the number of genotypes or SNP sites is relatively large.

The paper is organized as follows: Some notions and a detailed problem formulation are given in Section 2. In Section 3, we describe a genetic algorithm for the HIPP problem. Experimental results are shown in Section 4. Section 5 concludes the paper.

2 Formulation and Problem

The genotype data in a population of size m can be formulated as an $m \times n$ matrix $G = \{g_{ij}\}$ on $\{0, 1, 2\}$ with each row $g_i, i = 1, 2, \cdots, m$ of the matrix G corresponding to a genotype and each column $j,\ j = 1, 2, \cdots, n$ corresponding to a SNP site on the chromosome. For each genotype g_i, when the jth SNP site is wild type homozygous for g_i, $g_{ij} = 0$. When the jth SNP site is mutant type homozygous for g_i, $g_{ij} = 1$. When the jth SNP site is heterogenous for the genotype g_i, $g_{ij} = 2$. A position on a genotype is called *ambiguous* if the genotype on this position has value 2. A genotype is called ambiguous if it has at least two ambiguous positions.

A haplotype is a binary vector of length n on $\{0, 1\}$. A pair of haplotypes h_1 and h_2 is called a *resolution* of a genotype g if the following conditions hold: for each SNP site where $g_j = 0,\ h_{1j} = h_{2j} = 0$. For each SNP site where $g_j = 1,\ h_{1j} = h_{2j} = 1$. For each SNP site where $g_j = 2,\ h_{1j} = 0,\ h_{2j} = 1$ or $h_{1j} = 1,\ h_{2j} = 0$. The *resolution size* of a genotype is the number of all possible resolutions of this genotype. For a genotype with resolution size r, we order its resolutions and label them by $1, 2, \cdots, r$. A *realization* of a genotype matrix is a haplotype matrix H on $\{0, 1\}$ with each row corresponding to a haplotype and for each genotype $g_i, i = 1, 2, \cdots, m$, there exist two rows (a pair of haplotypes) $h_1,\ h_2$ of H such that $h_1,\ h_2$ form a resolution of g_i.

The haplotype inference problem is : *Given an $m \times n$ genotype matrix G, find a haplotype matrix H such that for each genotype there exists at least one pair of haplotypes which is a resolution of this genotype.*

Based on different genetic models, there are several kinds of assumptions on the haplotype matrix H. Perfect phylogeny haplotyping problem requires to find H as a realization of G and its rows form a haplotype perfect phylogeny [1, 6, 9], while haplotype inference by pure parsimony (HIPP) is a combinatorial optimization

problem that seeks a realization of G with least rows, i.e. a haplotype set with smallest cardinality. In this paper, we are concerned with designing a practical algorithm for the HIPP problem.

3 A Heuristic Method Based on Genetic Algorithm

Genetic algorithm [7] is a useful meta-heuristics algorithm and has successful applications in many areas including those of computational biology, e.g. protein structure prediction, promote sequence identification, primer design etc. In this section, we design a genetic algorithm for solving the HIPP problem. The input of the algorithm is a genotype matrix with all possible resolutions of each genotype. The output will be a haplotype matrix and pairs of halotypes associated with each genotype. The scheme of the designed genetic algorithm for HIPP is given in Table 1. The datails of the algorithm are given in the following subsections.

```
Haplotyping(G)
    Preprocessing(G);
    Generate a random initial population P_0, k = 0;
    while(k < GN) do
        Evaluate(P_k), i.e. compute the fitness of each individual in P_k.
        Select a fraction of individuals in P_k and add them into P_{k+1};
        Select some pairs of individuals in P_k and do crossover. Add all
offspring into P_{k+1};
        Mutation(P_{k+1});
        LocalOptimizition(P_{k+1});
        Record the best individual in the current population;
        k = k + 1;
    end do
return the best individual in the history.
```

Table 1: The scheme of the genetic algorithm for the HIPP problem.

3.1 Preprocessing

We adopt the ideas in [10, 19] to preprocess the genotype data. If four haplotypes of two resolutions of a genotype are not part of resolutions of any other genotypes, we can choose one as representative. If two genotypes have several combinatorial resolutions, we select one among those having same coverage (the number of genotypes that the haplotypes can resolve). This can reduce the resolution size of a genotype and the number of combinatorial resolutions of several genotypes and thus save computational time. In the rest part of the algorithm,

the size of the input data and the resolution size of a genotype mean that of the reduced one.

3.2 The population space

In order to shorten the length of code, we use real string code but not binary string code to represent an individual in the population space. For a given genotype matrix G with m rows, individuals representing feasible solutions to the HIPP problem are integer vectors of dimension m. For each genotype $g_i,\ i = 1, 2, \cdots, m$, we label its resolutions by $1, 2, \cdots, r_i$, where r_i is the resolution size of g_i. The value of the ith element of an individual is an integer k between 1 and r_i, denoting that in the feasible solution indicated by this individual, the kth resolution of the ith genotype is selected as its true resolution. The following set of integer vectors constitutes the population space:

$$P = \{(x_1, x_2, \cdots, x_m) \mid x_i \text{ is an integer and } 1 \leq x_i \leq r_i\}.$$

3.3 Fitness function

Every individual in the population space has a fitness degree. The goal of the HIPP problem is to find a haplotype set (a solution) with minimum cardinality to a given genotype matrix G, so the fitness of an individual is relevant to the cardinality of the feasible solution it corresponds to. We use $C_{\{x_1,x_2,\cdots,x_m\}}$ to denote the cardinality of the feasible solution that the individual $\{x_1, x_2, \cdots, x_m\}$ corresponds to, i.e. the number of distinct haplotypes in the haplotype set that this feasible solution corresponds to. Then, we can design such a fitness function:

$$f(x_1, x_2, \cdots, x_m) = \frac{2m - C_{\{x_1,x_2,\cdots,x_m\}}}{2m}.$$

Note that $0 < C_{\{x_1,x_2,\cdots,x_m\}} \leq 2m$ and $0 \leq f < 1$. The fitness of an individual is reversely proportional to the cardinality of the corresponding feasible solution, which embodies the goal of the HIPP problem.

3.4 Genetic operator

Selection operator: In order to conserve the diversity of a population and at the same time not to destroy the goodness of the population, the individuals in the current population are selected using the tournament selection method to be added into a new population and the standard roulette wheel scheme is used to select pairs of individuals intended to accept crossover operation.

Tournament selection: select an individual with highest fitness among randomly selected N individuals and let it survive to the next generation. Generally, the size of tournament selection N is 2, i.e. randomly select two individuals among the current population and let the individual with higher fitness survive. Repeat this process M times and get M individuals of the next generation.

Roulette wheel selection: the probability that an individual will be selected is proportional to its own fitness and reversely proportional to the fitness of the other individuals in the current population. Let *popsize* denote the population size and the fitness of the individual i is f_i, then the probability p_i with which i will be selected is:

$$p_i = \frac{f_i}{\sum_{i=1}^{popsize} f_i}, \quad i = 1, 2, \cdots, popsize.$$

Crossover operation: p_c (crossover rate) percent individuals of a population will be intended to do crossover. We use the combination of the single-point crossover method and the uniform crossover method in our algorithm. Single-point crossover exchanges two parts of two mated individuals at a randomly selected position. Uniform crossover is to exchange with probability 1/2 two values at each position of two individuals intended to accept crossover.

Mutation operation: The mutation is performed with a probability p_m (mutation rate) on each individual in the new population created by selection operation and crossover operation. Due to the speciality of real string code, we can not adopt swap mutation and invert mutation suited for binary string code. The mutation operation in the above algorithm is done as follows: Randomly select a position on the individual intended to accept mutation, e.g. i_0. Then generate randomly an integer between 1 and r_{i_0}. Replace the original value on the selected position with this integer.

3.5 Local optimization

The resolution size of a genotype increases exponentially with the number of heterozygous sites of this genotype. If the input genotype matrix has many ambiguous sites, the population size in the genetic algorithm must be large in order to find a good solution. This may affect the speed of each iteration. Thus, we introduce a local optimization mechanism whose idea is somewhat similar to that in [13]. After doing selection, crossover and mutation operation, we adopt a self-adaptive local optimization strategy according to the standard deviation σ of fitness in the population. σ determines the chance of accepting a bad mutation operation according to a Bolzmann distribution. By introducing the local optimization strategy, we can use a relatively small population sizes. The local optimization operation is illustrated as follows, where P_{LO} is the probability for doing local optimization and LS is the number of local search steps.

3.6 Parameter settings

Generally, genetic algorithm has four parameters to be determined: population size *popsize*, crossover rate p_c, mutation rate p_m and the maximum number of generation GN. Settings of these parameters are relevant to concrete instances to be solved and especially the size of instances. The larger the population size is, the earlier the algorithm finds a good solution, but the algorithm will consume more time. There is a trade-off between these factors. In the genetic algorithm for

```
LocalOptimization(P)
  { T = 1/σ;
    i = 0;
    repeat
        if (Random(0,1)< P_LO)
            Localsearch(indi[i]);
        end if
        i = i + 1;
    until (i > popsize)
  }

Localsearch(indi)
  { repeat
        tempindi = indi;
        mutation(indi);
        δ = fitness(tempindi)− fitness(indi);
        if (δ > 0)
            if (Random(0,1)> e^(−kδ/T))
                indi = tempindi;
                return;
            end if
        end if
        s = s + 1;
    until(s > LS)
  }
```

HIPP, the population size has closer relevance to the number of ambiguous sites in the genotype data than to the number of genotypes. Since we will use multiple datasets and their sizes are different, we will select values for these parameters according to concrete dataset. Experiment results show that the designed genetic algorithm is robust with crossover rate, mutation rate and the maximum number of generation, so we always set crossover rate as 0.8, mutation rate as 0.6 and the maximum number of generation as 150. In addition, the tournament selection size is set as 2. In local optimization mechanism, local optimization rate is set as 0.3. The number of local search steps is set as 5 and parameter k is set as 0.9. The population size will be set according to concrete data sets.

4 Experiment Results

In this section, we will test our algorithm on multiple datasets (including real datasets and simulation datasets) in comparison with the branch and bound algorithm (called HAPAR) in [19]. Our algorithm (called GAHAP) is implemented on a 1.8G Hz Pentium 4 processor PC using Microsoft Visual C++ compiler 6.

As mentioned previously, the designed genetic algorithm is suited to solve problems of large size since it has advantage in implementation time over exact algorithms. We first use two datasets of relatively small size to test the effectiveness of the algorithm, then use relatively large datasets to test the efficiency of the algorithm. Both algorithms are evaluated by three criteria. *hap_number* is the number of haplotypes that an algorithm returns. *correct_hap* is the number of correct haplotypes among them. *error_rate* is the proportion of genotypes whose original haplotype pairs are inferred incorrectly.

4.1 Experiment on human β_2-adrenergic receptor gene

Literature [5] reported 13 varying sites within a region of 1.6kb in the human β_2-adrenergic receptor (β_2AR) in the population consisting of 121 individuals, 18 distinct genotypes and 12 distinct haplotypes (we consider 10 haplotypes in the asthmatic cohort) were identified in the studied region. The number of identified haplotypes is far less than theoretically possible combinations. Implementation of the exact algorithm HAPAR [19] for the HIPP problem shows that the minimum number of haplotypes needed to resolve the 18 genotypes is 10. And the haplotype set output by HAPAR is exactly the original one. For this instance, we choose 50 as the population size of GAHAP. Other parameter settings are the same to those in Section 3. The designed genetic algorithm GAHAP also returns the same set of haplotypes in several seconds.

4.2 Experiment on chromosome 5q31

Literature [4] reported 103 SNPs across 500kb on chromosome 5q31 in a population of 129 trios. Their results show a picture of discrete haplotype blocks with limited diversity within each block. We consider the block 7 in which there are 4 common haplotypes with 31 SNP sites. We select 9 genotypes generated by these haplotypes as the input data (after removing 8 missing SNP sites). The parameter settings of genetic algorithm for this instance are the same to those in last subsection. Both algorithms return the results indentical to the common haplotype set in the block 7 in several seconds.

4.3 Experiment on angiotensin converting enzyme (ACE)

Literature [16] completed the genomic sequencing of the DCP1 gene (encode angiotensin converting enzyme) from individuals and reported 78 SNP sites in 22 chromosomes. 52 out of the 78 varying sites are non-unique polymorphic sites. There are 13 distinct haplotypes from 11 individuals in this region. For this instance, we choose 300 as the population size of GAHAP. Both algorithm again return the same optimal value, i.e. the same number of haplotypes, but the haplotype sets returned by two algorithms are different. So we run GAHAP ten times and the average results are listed in Table 2. GAHAP obtains higher accuracy of haplotype inference than HAPAR in each of ten runs.

	hap_number	*correct_hap*	*error_rate*	*run._time*
HAPAR	11	7	0.273	229(s)
GAHAP	11	8.5	0.227	36(s)

Table 2: Comparison results of HAPAR and GAHAP on ACE

4.4 Experiments on simulated data sets

Given a group of parameters m, n, k, we randomly generate m haplotypes with k SNPs. n genotypes are created by randomly combining two haplotypes among these haplotypes. Firstly, 6 instances are generated under the parameter settings that $m = 10$, $k = 20$, $n = 10, 11, 12, 13, 14, 15$. We set the population size of GAHAP as 600. The results of two algorithms are listed in Table 3, where F denotes that the HAPAR fails to return a solution within two hours and the results of GAHAP are averaged over 10 runs.

	hap_number		*correct_hap*		*error_rate*	
	HAPAR	GAHAP	HAPAR	GAHAP	HAPAR	GAHAP
$n = 10$	10	10	10	10	0.000	0.000
$n = 11$	9	9.3	9	8.1	0.000	0.100
$n = 12$	10	10	10	10	0.000	0.000
$n = 13$	F	9.5	F	8.6	F	0.065
$n = 14$	9	9	9	9	0.000	0.000
$n = 15$	F	10.6	F	9.7	F	0.037

Table 3: Comparison results of HAPAR and GAHAP on simulated data.

From Table 3 we can see that even for instances of small size, HAPAR can not obtain a solution within acceptable time, while GAHAP can always return a solution at most several minutes for such instances. In addition, GAHAP has comparable accuracy of haplotype inference with HAPAR.

Due to the randomness of simulated data, the execution time of HAPAR is quite unstable. Even for the same size of instances, HAPAR is able to output a solution to one instance very quickly and fails to find a solution to another within several hours. Especially in the data set with large SNP sites or genotypes, HAPAR is much slower that GAHAP. On the other hand, GAHAP can always output solutions in a stable and consistent way.

5 Conclusions

In this paper, we developed a heuristic method based on genetic algorithm (GAHAP) to solve a kind of haplotype inference problem — the HIPP problem.

We tested our algorithm on a variety of datasets including biological data and simulated data. Experimental results show the effectiveness and efficiency of this algorithm. The designed algorithm can return the same optimal values as the exact algorithm HAPAR in almost all cases and has comparable accuracy of haplotype inference with HAPAR. For large instances, our algorithm consumes much less time than the HAPAR. So GAHAP is suited for haplotype inference in haplotype blocks with large number of SNP sites or genotypes.

From experiment results in Section 4, we can see that sometimes even an optimal solution of the HIPP problem does not correspond to haplotype inference with 100% accuracy. This is due to the pure parsimony criterion, or there are a few recombinations in a practical haplotype block. We can use some other information (e.g. SNP fragments) of a genotype to modify the pure parsimony criterion and improve the accuracy of haplotype inference by parsimony. This is our further work.

Acknowledgement

This work is supported by the National Natural Science Foundation of China under grant No.10471141.

References

[1] Bafna,V., Gusfied,D., Lancia,G. and Yooseph,S. Haplotyping as perfect phylogeny: a direct approach. *Journal of computational biology*, **10**:323-340, 2003.

[2] Bonizzoni,P., Vedova,G.D., Dnodi,R., and Li,J. The haplotyping problem: An overview of computational models and solutions. *Journal of Computer Science and Technology*, **18**(6):675-688, 2003.

[3] Clark, A.G. Inference of haplotypes from PCR-amplified samples of dipoid populations. *Molecular Biology and Evolution*, **7**(2):111-122, 1990.

[4] Daly,M., Rioux,J., Hudson,T., and Lander,E. High-resolution haplotype structure in human genome. *Nat. Genet,* **29**:229-232, 2001.

[5] Drysdale,C., McGraw,D., *et al.* Complex promoter and coding region β_2-adrenergic receptor haplotypes alter receptor expression and predict in vivo responsiveness. *Proceedings of National Academy Sciebce USA*,**97**:10483-10488, 2000.

[6] Eskin,E., Halperin,E. and Karp,R. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, **1**(1):1-20, 2003.

[7] Goldberg,D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading: Addison-Wesley, 1989.

[8] Gusfield, D. Inference of haplotypes from samples of diploid populations: complexity and algorithms. *Journal of Computational Biology*, **8**(3): 305-323, 2001.

[9] Gusfield, D. Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *Proc. RECOMB'02*, 166-175, 2002.

[10] Gusfield, D. Haplotype inference by pure parsimony. *Lecture Notes in Computer Science*, **2676**:144-155. Springer-verlag, 2003.

[11] Halldórsson,B.V., Bafna,V. *et al.* A survey of computational methods for determining haplotypes, *Lecture Notes in Computer Science*, 2983:26-47, Springer-Verlag, 2004.

[12] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the Human genome. *Nature*, **409(6822)**:860-921, 2001.

[13] Krasnogor,N. and Smith,J. A memetic algorithm with self-adaptive local search: TSP as a case study. In *Proc. GECCO'00*, 987-994, 2000.

[14] Lancia,G., Pinotti,C. and Rizzi,R. Haplotyping populations: complexity, exact and approximation algorithms. *INFORMS Journal on Computing*, 16(4):348-359, 2004

[15] Niu,T., Qin,Z.S., and Liu,J.S. Bayesian haplotype inference for multiple linked single-nucleotide polymorphism. *American Journal of Human Genetics,* **70**(1):157-159, 2002.

[16] Rieder,M., Taylor,S., Clark,A. and Nickerson,D. Sequence variation in the human angiotensim converting enzyme. *Nature genetics*, **22**:59-62, 1999.

[17] Stephens,J.C., Schneider,J.A., Tanduay,D.A. *et al.* Haplotype variation and linkage disequilibrium in 313 human genes. *Science*, **293**:489-493, 2001.

[18] Stephens,M., Smith,N.J. and Donnelly,P. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics,* **68**(4):978-979, 2001.

[19] Wang L.S. and Xu Y. Haploptye inference by maximum parsimony, *Bioinformatics*, **19**(14):1773-1780, 2003.