BIOINFORMATICS

Vol. 19 no. 14 2003, pages 1773–1780
DOI: 10.1093/bioinformatics/btg239

Haplotype inference by maximum parsimony

Lusheng Wang[1,*] and Ying Xu[2]

[1]Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, People's Republic of China and [2]Department of Computer Science, Peking University, Beijing 100871, People's Republic of China

Received on December 26, 2002; revised on March 10, 2003; accepted on March 31, 2003

ABSTRACT
Motivation: Haplotypes have been attracting increasing attention because of their importance in analysis of many fine-scale molecular-genetics data. Since direct sequencing of haplotype via experimental methods is both time-consuming and expensive, haplotype inference methods that infer haplotypes based on genotype samples become attractive alternatives.
Results: (1) We design and implement an algorithm for an important computational model of haplotype inference that has been suggested before in several places. The model finds a set of minimum number of haplotypes that explains the genotype samples. (2) Strong supports of this computational model are given based on the computational results on both real data and simulation data. (3) We also did some comparative study to show the strength and weakness of this computational model using our program.
Availability: The software HAPAR is free for non-commercial uses. Available upon request (lwang@cs.cityu.edu.hk).
Contact: lwang@cs.citu.edu.hk

1 INTRODUCTION

Single nucleotide polymorphisms (SNPs) are the most frequent form of human genetic variation. The SNP sequence information from each of the two copies of a given chromosome in a diploid genome is a *haplotype*. Haplotype information has been attracting great attention in recent years because of its importance in analysis of many fine-scale molecular-genetics data, such as in the mapping of complex disease genes, inferring population histories and designing drugs (Hoehe *et al*., 2000; Clark *et al*., 1998; Schwartz *et al*., 2002). However, current routine sequencing methods typically provide genotype information rather than haplotype information. (It consists of a pair of haplotype information at each position of the two copies of a chromosome in diploid organisms. However, the connection between two adjacent positions is not known.) Since direct sequencing of haplotype via experimental methods is both time-consuming and expensive, *in silico* haplotyping methods become attractive alternatives.

The *haplotype inference* problem is as follows: given an $n \times k$ genotype matrix, where each cell has value 0, 1, or 2. Each of the n rows in the matrix is a vector associated with sites of interest on the two copies of an allele for diploid organisms. The state of any site on a single copy of an allele is either 0 or 1. A cell (i, j) in the ith row has a value 0 if the site is homozygous wild type, a value 1 if it is homozygous mutant, and a value 2 if it is heterozygous. A cell is resolved if it has value 0 or 1 (homozygous), and ambiguous if it has value 2 (heterozygous). The goal here is to determine which copy of the allele has a value 1 and which copy of the allele has avalue 0 at those heterozygous sites based on some mathematical models.

In 1990, Clark first discovered that genotypes from population samples were useful in reconstructing haplotypes and proposed an inference method. After that, many algorithms and programs have been developed to solve the haplotype inference problem. The existing algorithms can be divided into four primary categories. The first category is Clark's inference rule approach that is exemplified in Clark (1990) and extended in Gusfield (2001) by trying to maximize the number of resolved vectors. The second category is expectation-maximization (EM) method which looks for the set of haplotypes maximizing the posterior probability of given genotypes (Excoffier and Slatkin, 1995; Hawley and Kidd, 1995; Long *et al*., 1995; Chiano and Clayton, 1998). The third category contains several statistical methods based on Bayesian estimators and Gibbs sampling (Stephens *et al*., 2001; Niu *et al*., 2002; Zhang *et al*., 2001). Finally, adopting the no-recombination assumption, Gusfield proposed a model that finds a set of haplotypes forming a perfect phylogeny (Gusfield, 2002; Bafna *et al*., 2002). Besides the haplotype inference model, an alternative method for haplotyping based on the data and methodology of shotgun sequence assembly was discussed in Lancia *et al*. (2001), Lippert *et al*. (2002) and Rizzi *et al*. (2002).

The model studied in this paper finds a set of minimum number of haplotypes that explains the genotype samples. (See the formal definition in Section 2.) The model was suggested in several places. Earl Hubell first proposed the model and proved that the problem was NP-hard. (The results have not been published.) The formal definition of the model first appeared in Gusfield (2001). Gusfield proposed an integer linear programming approach that can give optimal solutions for

* To whom correspondence should be addressed.

practical sized problems of this model (Gusfield, 2003). [The results were announced in Gusfield (2001).] In this paper, we design and implement an algorithm for this parsimony model. We adopt a branch-and-bound method that is different from Gusfield's integer linear programming approach. Based on the computational results of both real data and simulation data, we provide strong support for this computational model. Comparative studies show the strength and weakness of this approach.

2 SUPPORT OF THE PARSIMONY MODEL

Given an $n \times k$ genotype matrix M, where each cell has a value 0, 1, or 2, a $2n \times k$ *haplotype matrix* Q that *explains* M is obtained as follows: (1) duplicate each row i in M to create a pair of rows i, and i' (for $i = 1, 2, \ldots, n$) in Q and (2) re-set each pair of cells $Q(i, j) = 2$ and $Q(i', j) = 2$ to be either $Q(i, j) = 0$ and $Q(i', j) = 1$ or $Q(i, j) = 1$ and $Q(i', j) = 0$ in the new resulting $2n \times k$ matrix Q. Each row in M is called a *genotype*. For a genotype m_i (the ith row in M), the pair of finally resulted rows Q_i and $Q_{i'}$ form a *resolution* of m_i. We also say that Q_i and $Q_{i'}$ *resolve* the genotype m_i. For a given genotype matrix M, if there are h 2s in M, then any of the 2^h possible haplotype matrices can explain M. Thus, without any further assumptions, it is hard to infer the haplotypes.

The computational model we study finds a set of minimum numberof haplotypes that explains the genotype samples. The problem is defined as follows: given an $n \times k$ genotype matrix M, find a $2n \times k$ haplotype matrix Q such that the number of *distinct* rows in Q is minimized. n is often referred to as the *sample size*. The model was suggested in several places (Gusfield, 2001, 2003).

The approach is based on the parsimony principle that attempts to minimize the total number of haplotypes observed in the sample. The parsimony principle is one of the most basic principles in nature, and has been applied in numerous biological problems. In fact, Clark's inference algorithm (Clark, 1990), which has been extensively used in practice and shown to be useful (Clark *et al*., 1998; Rieder *et al*., 1999; Drysdale *et al*., 2000), can also be viewed as a sort of parsimony approach. However, to apply Clark's algorithm, there must be homozygote or single-site heterozygote in the sample. The computational model studied here overcomes this obstacle by proposing a global optimization goal.

The characteristics of real biological data also provide justifications for the method. The number of haplotypes existing in a large population is actually very small whereas genotypes derived from these limited number of haplotypes behave a great diversity. Theoretically, given m haplotypes, there are $m(m-1)/2$ possible pairs to form genotypes. (Even if Hardy–Weinberg equilibrium is violated and some combinations are rare, the number is still quite large.) When some population is to be studied, the haplotype number can be taken as a fixed constant, while the number of distinct genotypes is decided by the sample size sequenced, which is relatively large. Intuitively, when genotype sample size n is large enough, the corresponding number of haplotypes m would be relatively small. Thus, this computational model has a good chance to recover the haplotype set.

A real example strongly supports the above arguments.

2.1 A real example exactly fitting the model

β_2-Adrenergic receptors (β_2ARs) are G protein-coupled receptors that mediate the actions of catecholamines in multiple issues. In Drysdale *et al*. (2000), 13 variable sites within a span of 1.6 kb were reported in the human β_2AR gene. Only 10 haplotypes were found to exist in the studied asthmatic cohort, far less than theoretically possible $2^{13} = 8192$ combinations. Eighteen distinct genotypes were identified in the sample consisting of 121 individuals. Those 10 haplotypes and 18 genotypes are illustrated in Tables 1 and 2, respectively. In this data set, the genotype number (18) is relatively large with respect to the haplotype number (10). Computation shows that the minimum number of haplotypes needed to generate the 18 genotypes is 10, and given the 18 genotypes as input, the set of haplotypes inferred by our algorithm (see next section) is exactly the original set.

3 AN EXACT ALGORITHM

We design an exact algorithm to find the optimal solution for this parsimony model. Our algorithm adopts the branch-and-bound approach. A good (small size) initial solution is critical for any branch-and-bound algorithms. Here we propose a greedy algorithm.

3.1 The greedy algorithm

The *coverage of a haplotype* is the number of genotypes that the haplotype can resolve. The *coverage of a resolution* is the sum of the coverage for the two haplotypes of the resolution. The greedy algorithm simply chooses from each genotype a resolution with maximum coverage to form a solution. This heuristic algorithm can often give a solution with size close to the optimum.

3.2 The branch-and-bound algorithm

Let $M = \{m_1, m_2, \ldots, m_n\}$ be a set of genotypes (rows in the genotype matrix). The sketch of the algorithm is illustrated in Figure 1. To ease the presentation, the code is for the case, where the number of rows n in the genotype matrix is fixed. Our implementation treats n as part of the input and is more sophisticated.

Basically, the branch-and-bound algorithm searches all possible solutions and finds the best one. When the size of a partial solution is greater than or equal to the current bound, it skips the bad subspace and moves to next possible good choice. Theoretically, the running time of the algorithm is exponential in terms of the input size. In order to make our

Table 1. Ten haplotypes of β_2AR genes

Nucleotide	−1023	−709	−654	−468	−367	−47	−20	46	79	252	491	523	
Alleles	G/A	C/A	G/A	C/G	T/C	T/C	T/C	G/A	C/G	G/A	C/T	C/A	
h_1	A	C	G	C	T	T	T	A	C	G	C	C	100000010000
h_2	A	C	G	G	C	C	C	G	G	G	C	C	100111101000
h_3	G	A	A	C	T	T	T	A	C	G	C	C	011000010000
h_4	G	C	A	C	T	T	T	A	C	G	C	C	001000010000
h_5	G	C	A	C	T	T	T	G	C	G	C	C	001000000000
h_6	G	C	G	C	T	T	T	G	C	A	C	A	000000000101
h_7	G	C	G	C	T	T	T	G	C	A	T	A	000000000111
h_8	G	C	A	C	T	T	T	A	C	A	C	A	001000010101
h_9	G	C	G	C	T	T	T	G	C	A	C	C	000000000100
h_{10}	G	C	G	C	T	T	T	G	C	G	C	C	000000000000

Nucleotide number is the position of the site, based on the first nucleotide of the starting codon being +1. Allele is the two nucleotide possibilities at each SNP site. These data are from Drysdale *et al*. (2000). The original paper gave 12 haplotypes of 13 SNP sites. In this table, we only listed 10 haplotypes which were found in the asthmatic cohort, and one rare SNP site which did not show ambiguity in the sample was excluded. The last column of each haplotype is the representation of that haplotype. Each haplotype is a vector of SNP values. For each SNP, we assume the first nucleotide in 'Alleles' to be wild type (represented with 0), and second one to be mutant (represented with 1).

Table 2. Eighteen genotypes of β_2AR genes

Genotype	Resolution	Value
m_1	(h_2, h_4)	202222222000
m_2	(h_2, h_2)	100111101000
m_3	(h_2, h_6)	200222202202
m_4	(h_4, h_4)	001000010000
m_5	(h_4, h_6)	002000020202
m_6	(h_2, h_5)	202222202000
m_7	(h_4, h_9)	002000020200
m_8	(h_1, h_4)	202000010000
m_9	(h_1, h_6)	200000020202
m_{10}	(h_2, h_{10})	200222202000
m_{11}	(h_2, h_3)	222222222000
m_{12}	(h_2, h_7)	200222202222
m_{13}	(h_2, h_8)	202222222202
m_{14}	(h_3, h_4)	021000010000
m_{15}	(h_4, h_5)	001000020000
m_{16}	(h_4, h_7)	002000020222
m_{17}	(h_4, h_8)	001000010202
m_{18}	(h_6, h_7)	000000000121

The second column of each genotype is the (true) resolution to that genotype. For example, genotype m_1 is resolved by haplotypes h_2 and h_4. The third column is the representation of that genotype. Each genotype is a vector of SNP values and the value of each SNP is 0, 1 or 2, standing for homozygous wild type, homozygous mutant, or heterozygous.

program efficient enough for practical use, we made another improvement to reduce the running time.

3.3 Improvement

Reducing the size of the resolution lists. Since we only report one optimal solution, if some possible resolutions are 'equally good', we can keep only one representative and discard the others. We only consider two cases in our program.

Case 1. Two resolutions to the same genotype m_i both have coverage 2. In this case, none of the four haplotypes contained in the two resolutions appears in any other genotypes. Thus, we just keep one of the resolutions in the resolution list of m_i. The implementation detail is similar to that in Gusfield (2003).

Case 2. Consider two genotypes m_i and m_j. Suppose m_i has two resolutions (h_1, h_2) and (h_4, h_5) and m_j has two resolutions (h_2, h_3) and (h_5, h_6). If h_1, h_3, h_4 and h_6 have coverage 1, and h_2 and h_5 have coverage 2, then we only have to keep the combination (h_1, h_2) and (h_2, h_3) and ignore the combination (h_4, h_5) and (h_5, h_6). Thus, (h_4, h_5) and (h_5, h_6) will not show up in the resolution lists.

An example is given as follows: $h_1 = 0101$, $h_2 = 1001$, $h_3 = 1111$, $h_4 = 0001$, $h_5 = 1101$, $h_6 = 1011, m_1 = 2201$ and $m_2 = 1221$. There are another two given genotypes $m_3 = 2220$ and $m_4 = 1020$ that have nothing to do with the six haplotypes. In this case, we only need to put (h_1, h_2) into Array(1) and (h_2, h_3) into Array(2), and ignore (h_4, h_5) in Array(1) and (h_5, h_6) in Array(2).

The idea can be extended to more sophisticated cases. However, those cases do not happen very often and may not help much in practice. After applying this trick, the number of possible resolutions and the number of candidate haplotypes (haplotypes appeared in resolution arrays) are dramatically cut down. For example, when we run our program on ACE data containing 11 individuals of 52 SNPs (see Section 4 for details), the number of candidate haplotypes is only 483, which is far less than the total of 2^{52} possible haplotypes.

We implement the above algorithm with C++. The program, HAPAR, is now available upon request. It takes a file containing genotype data as input, and outputs resolved haplotypes. With all these improvements, our program is fairly efficient. For example, it takes HAPAR only 2.25 min on our computer to compute ACE data. In contrast, it takes PHASE,

```
Illustration of the Branch-and-bound Algorithm
Input: a genotype matrix $M$ containing $n$ rows
1.  List all possible resolutions for each genotype.
    Let Array(1), Array(2), ..., Array(n) be the arrays of resolutions for the
    $n$ rows in $M$. Denote $s_i$ to be the length of Array(i).
    Let $S$ be a set of resolutions. ($S$ is empty at the beginning.)
    $f(S)$ is the number of distinct haplotypes in $S$.
2.  Use the greedy algorithm to get a solution and set $f^*(S)$ to be the size
    of the solution.
3.  Search for an optimal solution as follows:
3.1 for $j_1 = 1$ to $s_1$ do begin
        $S = \{Array(1)[j_1]\}$.
        if $(f(S) > f^*(S))$ then go to 3.1 and try next $j_1$;
3.2     for $j_2 = 1$ to $s_2$ do begin
            $S = \{Array(1)[j_1], Array(2)[j_2]\}$.
            if $(f(S) > f^*(S))$ then go to 3.2 and try next $j_2$;
            ......
3.n         for $j_n = 1$ to $s_n$ do begin
                $S = \{Array(1)[j_1], Array(2)[j_2], \ldots, Array(n)[j_n]\}$.
                if $(f(S) < f^*(S))$ then $f^*(S) = f(S)$;
    end end ... end
```

Fig. 1. The branch-and-bound algorithm.

a program based on Gibbs sampling, 12 min to compute ACE in the same environment. Programs based on EM method cannot even handle this set of data.

4 RESULTS

We ran our program HAPAR for a large amount of real biological data as well as simulation data to demonstrate the performance of our program. We also compared our program with four widely used existing programs, HAPINFERX, Emdecoder, PHASE, and Haplotyper. HAPINFERX is an implementation of Clark's algorithm (Clark, 1990), and was kindly provided by Clark. Emdecoder uses an EM alogrithm, and was downloaded at Liu's homepage (http://www.people.fas.harvard.edu/~junliu/). PHASE is a Gibbs sampler for haplotyping (Stephens *et al.*, 2001), and was downloaded at Stephens' homepage. Haplotyper is a Bayesian method, and was downloaded at Liu's homepage. We will discuss different sets of data in the following subsections.

Throughout our experiment, we measure performance by the *error rate*, a commonly used criterion in haplotype inference problem (Stephens *et al.*, 2001; Niu *et al.*, 2002). The *error rate* is the proportion of genotypes whose original haplotype pairs were incorrectly inferred by the program. For example, if the error rate is 0.2, then 20% of genotypes are resolved incorrectly (whose haplotype pairs are incorrect) and the other 80% are resolved correctly.

4.1 Angiotensin converting enzyme

Angiotensin converting enzyme (encoded by the gene *DCP*1, also known as ACE) catalyses the conversion of angiotensin I to the physiologically active peptide angiotensin II. Due to its key function in the renin–angiotensin system, many association studies have been performed with *DCP*1. Rieder *et al.* (1999) completed the genomic sequencing of *DCP*1 from 11 individuals, and identified 78 varying sites in 22 chromosomes. Fifty two out of the 78 varying sites are non-unique polymorphic sites, and complete data on these 52 biallelic markers are available. Thirteen distinct haplotypes were resolved from the sample.

We ran the four programs, HAPAR, Haplotyper, HAPINFERX and PHASE, on ACE data set (11 genotypes, 52 SNPs, 13 haplotypes). (Emdecoder is limited in the number of SNPs in genotype data and thus is excluded.) The result is summarized in Table 3.

From our experiments, most programs can resolve 8 out of the 11 genotypes correctly, having an error rate of 3/11. The relatively low performance was due to the small sample size. In fact, three genotypes are composed by six haplotypes each

Table 3. Comparison of performance of four programs on ACE data set

Program	Error rate	Number of haplotype correctly inferred
HAPAR	0.273	7
Haplotyper	0.182 or 0.273	7 or 9
HAPINFERX	0.273 or 0.364	5 or 7
PHASE	0.273	7

Haplotyper has two different error rates because it gives different results in multiple runs. So does HAPINFERX.

of which appears only once, so there is not enough information for any of the four programs to resolve those three genotypes successfully. In some runs, Haplotyper can guess one resolution correctly (thus reducing the error rate to be 2/11), but it cannot get a consistent result.

4.2 Simulations on random data sets

In this subsection, we use simulation data to evaluate different programs. First, m haplotypes, each containing k SNP sites, were randomly generated. Then a sample of n genotypes were generated, each of which was formed by randomly picking two haplotypes and conflating them. (Here n is the sample size, which may be larger than the number of distinct genotypes.) A haplotyping program resolved those genotypes and output the inferred resolutions, which were then compared with the original resolutions to get the error rate of the program.

Throughout the simulations, 100 data sets were generated for each parameter setting, and the error rate was calculated by taking the average of the error rates in the 100 runs.

We conducted simulations with different parameter settings and compared the performance of the five programs HAPAR, HAPINFERX, Haplotyper, Emdecoder and PHASE. Two sets of parameters were used: (1) $m = 10, k = 10, n$ ranges from 5 to 24 [see Fig. 2a and b], and (2) $m = 8$, $k = 15$, n ranges from 5 to 11 [see Fig. 2c and d]. As shown by the figures, HAPAR outperforms the other four programs in many cases. When n is as small as $m/2$ (every haplotype appears in only one genotype), any resolution combination would be an optimal solution for the minimum number of origins problem, so HAPAR cannot identify the correct resolutions. In this case, other programs also have poor performance due to the lack of information. When n becomes larger, all five programs improve their performance, and HAPAR shows an advantage over the others. When n is large enough ($n = 12$ for $m = 10$, $k = 10$; and $n = 10$ for $m = 8$, $k = 15$), HAPAR, Haplotyper, Emdecoder and PHASE can all resolve the genotypes successfully with high probability.

Besides, standard deviations of the error rates are shown to reflect the stability [Fig. 2b and d]. In our simulations, the standard deviations typically range from 0 to 0.5. It is small when the mean error rate is very small (near 0) or very large (near 1), and reaches its maximum when the error rate is near 0.5. The standard deviations of HAPAR are slightly lower than Haplotyper, Emdecoder and PHASE in most cases, which shows that our program has good stability. HAPINFERX is the most unstable among all.

4.3 Simulations on maize data set

The maize data were used in Wang and Miao (2002) as one of the benchmarks to evaluate haplotyping programs. The data was from Ching *et al*. (2002). The locus 14 of maize profile containing 17 SNP sites and 4 haplotypes (with frequency 9, 17, 8 and 1) were identified. We randomly generated a sample of n genotypes from these haplotypes, each of which was formed by randomly picking 2 haplotypes according to their frequencies and conflating them. The error rates and standard deviations of the four programs were summarized in Table 4. (Emdecoder is limited in the number of SNPs in genotype data and thus is excluded.) According to our experiments, HAPAR, Haplotyper and PHASE are able to resolve most of the genotypes correctly when the sample size n is greater than or equal to 4, and HAPAR and Haplotyper behave better than PHASE. HAPINFERX has a relatively high error rate even when sample size n is as large as 10.

4.4 Simulations based on the coalescence model

The coalescence model is one of the evolutionary model most commonly used in population genetics (Gusfield, 2002; Hudson, 2002). Here we conduct a careful study.

4.4.1 Simulations on real data forming a perfect phylogeny The coalescent model of haplotype evolution says that without recombination, haplotypes can fit into a perfect phylogeny (Gusfield, 2002). Jin *et al*. (1999) found a 565 bp chromosome 21 region near the $MX1$ gene, which contains 12 polymorphic sites. This region is unaffected by recombination and recurrent mutation. The genotypes determined from sequence data of 354 human individuals were resolved into 10 haplotypes, the evolutionary history of which can be modelled by a perfect phylogeny. These 10 haplotypes were used to generate genotype samples of different size for evaluation.

The error rates of the five programs on MX1 data are compared in Figure 3. As expected, PHASE outperforms the other programs because it incorporates the coalescent model which fits these data sets. In the remaining four programs which do not adopt the coalescent assumption, HAPAR has the lowest error rate. When the sample size is large ($n \geq 22$), all the five programs have low error rates, and the difference between the error rates of the programs, especially those of HAPAR, Haplotyper and PHASE, is very small. The pattern of standard deviations is similar to those in Figure 2 and Table 4.

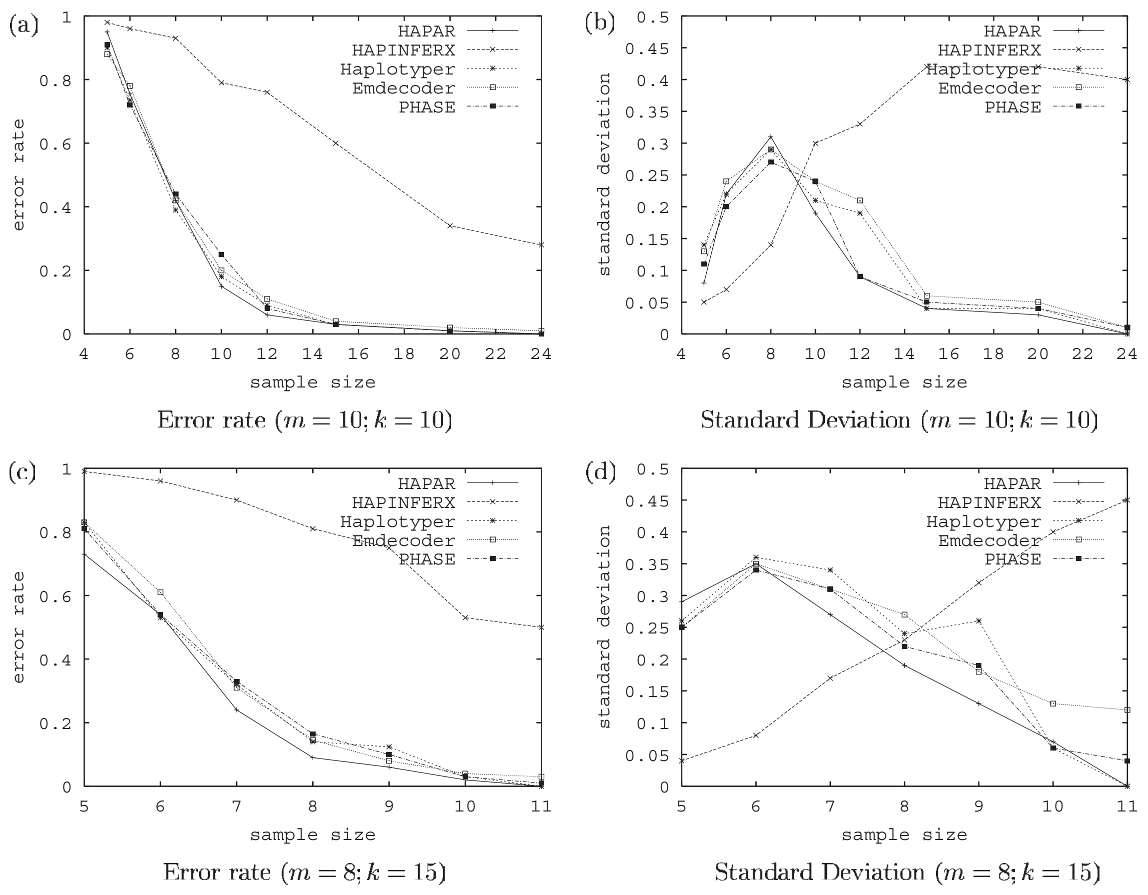

Fig. 2. Comparison of the performance of five programs on random data. In (**a**) and (**b**), originally there are 10 haplotypes of 10 SNPs; in (**c**) and (**d**), originally there are 8 haplotypes of 15 SNPs. For each parameter setting, 100 data sets were generated. (**a**) and (**c**) show the relationship of the mean error rate with the sample size, while (**b**) and (**d**) show the standard deviation.

Table 4. Comparison of performance of four programs on Maize data set ($m = 4,\ k = 17$)

Sample size	HAPAR	Haplotyper	HAPINFERX	PHASE
3	0.510 (0.372)	0.473 (0.347)	0.860 (0.165)	0.527 (0.340)
4	0.096 (0.228)	0.140 (0.277)	0.640 (0.360)	0.154 (0.226)
7	0.046 (0.172)	0.046 (0.172)	0.430 (0.388)	0.070 (0.198)
10	0 (0)	0 (0)	0.282 (0.370)	0 (0)

Each item shows the error rate followed by the standard deviation in brackets.

4.4.2 Coalescence-based simulations without recombination In this scheme, samples of haplotypes were generated according to a neutral mutation model. We use the software (ms) in Hudson (2002) to generate $2n$ gametes. After that we randomly pair gametes into n genotypes, which were used as input for those haplotype inference programs. The number of SNP loci is fixed as 10, and a total of 100 replications were made for each sample size. When generating gametes, we specify recombination parameter to be 0.

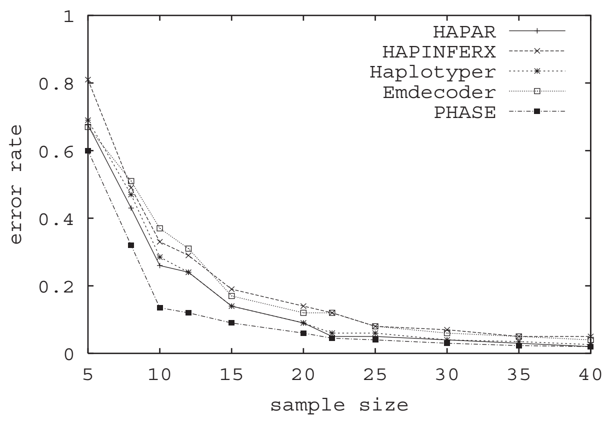

Fig. 3. Comparison of performance of five programs on MX1 data set ($m = 10, k = 12$).

The error rates of the five programs are illustrated in Figure 4. Again, PHASE is the best. In the remaining four programs, HAPAR is the best and the error rate of HAPAR is only slightly lower than PHASE when sample size is large.

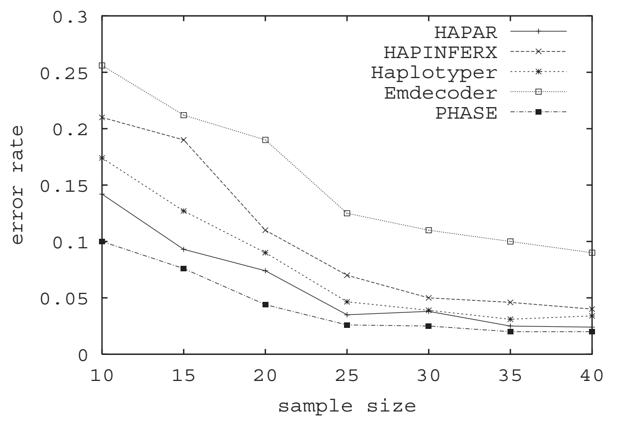

Fig. 4. Comparison of performance of five programs on data based on coalescence model without recombination ($k = 10$).

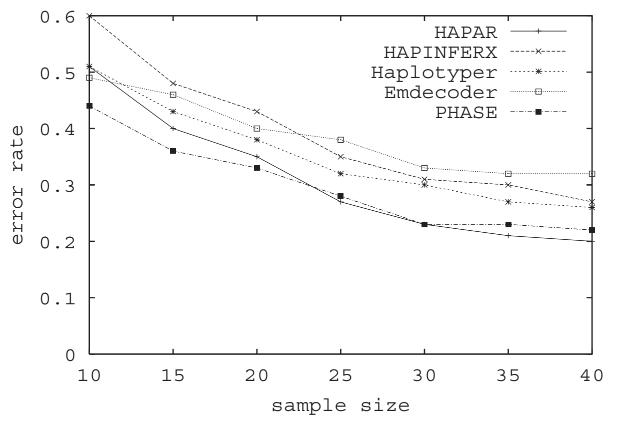

Fig. 5. Comparison of performance of five programs on data based on coalescence model with recombination ($k = 10$).

Surprisingly, HAPINFERX performs fairly well on the data, even beating Emdecoder.

4.4.3 Coalescence-based simulations with recombination Now we introduce recombination into the model when generating simulated haplotypes. The simulation scheme is similar, except that we set recombination parameter ρ to be 100.0 when generate gametes using Hudson's software (Hudson, 2002). [$\rho = 4N_0r$, where r is the probability of crossing-over per generation between the ends of the locus being simulated, and N_0 is the diploid population size. See Hudson (2002) for details.]

The error rates of the five programs are illustrated in Figure 5. Overall, the error rates are greater than those without recombination. The performance of PHASE is still the best among the five programs when the sample size is small, but as the sample size grows large ($n > 25$), HAPAR begins to outperform PHASE.

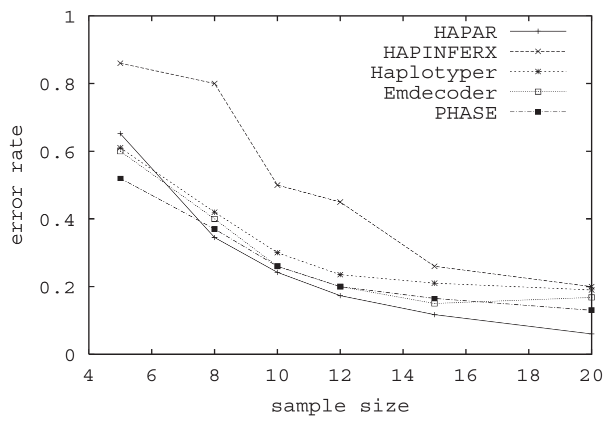

Fig. 6. Comparison of performance of five programs on $5q31$ data set ($m = 9, k = 12$).

4.5 Simulations on real data with recombination hotspots

To further compare the performance of different programs in the presence of recombination events, we conduct simulations using the data on chromosome $5q31$ studied in Daly *et al.* (2001). They reported a high-resolution analysis of the haplotype structure across 500 kb on chromosome $5q31$ using 103 SNPs in a European-derived population. The result showed a picture of discrete haplotype blocks (of tens to hundreds of kilobases), each with limited diversity. The discrete blocks are separated by intervals in which several independent historical recombination events seem to have occurred, giving rise to greater haplotype diversity for regions spanning the blocks.

We use the haplotypes from block 9 (with 6 sites, and 4 haplotypes) and block 10 (with 7 sites, 6 of which are complete, and 3 haplotypes). There is a recombination spot between the two blocks, which is estimated to have a haplotype exchange rate of 27%. Nine new haplotypes with 12 sites are generated by connecting two haplotypes from block 9 and block 10 which were observed to have common recombination events, and their frequencies were normalized. Genotype samples of different sizes were randomly generated and used for evaluation. According to the experiment results illustrated in Figure 6, HAPAR has lower error rates than the other programs, when the sample size is large.

5 CONCLUSION

We design and implement an algorithm for a computational model that was suggested in several places (Gusfield, 2001, 2003). Our program performs well (has the lowest error rates) in some simulations using real haplotype data as well as randomly generated data, when the sample size is big enough. Under the coalescence model, PHASE has the best performance when there is no recombination. When recombinations occur under the coalescence model, PHASE has the best performance when the sample is small, whereas our program is slightly better when the sample size is big. This shows that our program (in fact, the computational model) has limitations on different evolutionary models. Our program also performs well on some real data haplotypes with recombination.

ACKNOWLEDGEMENTS

We thank referees for their helpful suggestions. In particular, we thank Dan Gusfield for pointing out the origins of the computational model and the suggestion of the performance measure used in the present version of the paper.

We are grateful to A. Clark for kindly giving us HAPINFERX. We also thank R. Hudson, J. Liu and M. Stephens for providing software on their web pages. The work is fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 1087/00E).

REFERENCES

Bafna,V., Gusfield,D., Lancia,G. and Yooseph,S. (2002) Haplotyping as perfect phylogenty: a direct approach. Technical Report UCDavis CSE-2002-21.

Chiano,M. and Clayton,D. (1998) Fine genetic mapping using haplotype analysis and the missing data problem. *Am. J. Hum. Genet.*, **62**, 55–60.

Ching,A., Caldwell,K.S., Jung,M., Dolan,M., Smith,O.S., Tingey,S., Morgante,M. and Rafalski,A.J. (2002) SNP frequency, haplotype structure and linkage disequilibrium in elite maize inbred lines. *BMC Genet.*, **3**, 19.

Clark,A. (1990) Inference of haplotypes from PCR-amplified samples of diploid populations. *Mol. Biol. Evol.*, **7**, 111–122.

Clark,A., Weiss,K., Nickerson,D., Taylor,S., Buchanan,A., Stengard,J., Salomaa,V., Vartiainen,E., Perola,M., Boerwinkle,E. and Sing,C. (1998) Haplotype structure and population genetic inferences from nucleotide-sequence variation in human lipoprotein lipase. *Am. J. Hum. Genet.*, **63**, 595–612.

Drysdale,C., McGraw,D., Stack,C., Stephens,J., Judson,R., Nandabalan,K., Arnold,K., Ruano,G. and Liggett,S. (2000) Complex promoter and coding region β_2-adrenergic receptor haplotypes alter receptor expression and predict in vivo responsiveness. *Proc. Natl Acad. Sci. USA*, **97**, 10483–10488.

Daly,M., Rioux,J., Schaffner,S., Hudson,T. and Lander,E. (2001) High-resolution haplotype structure in the human genome. *Nat. Genet.*, **29**, 229–232.

Excoffier,L. and Slatkin,M. (1995) Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Mol. Biol. Evol.*, **12**, 921–927.

Gusfield,D. (2001) Inference of haplotypes from samples of diploid populations: complexity and algorithms. *J. Comput. Biol.*, **8**, 305–323.

Gusfield,D. (2002) Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *Proceedings of the Sixth Annual International Conference on Computational Biology (RECOMB'02)*, pp. 166–175.

Gusfield,D. (2003) Haplotyping by pure parsimony. In *14th Symposium on Combinatorial Pattern Matching*, to appear.

Hawley,M. and Kidd,K. (1995) HAPLO: a program using the EM algorithm to estimate the frequencies of multi-site haplotypes. *J. Hered.*, **86**, 409–411.

Hoehe,M., Kopke,K., Wendel,B., Rohde,K., Flachmeier,C., Kidd.K, Berrettini,W. and Church,G. (2000) Sequence variability and candidate gene analysis in complex disease: association of μ opioid receptor gene variation with substance dependence. *Hum. Mol. Genet.*, **9**, 2895–2908.

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

Jin,L., Underhill,P., Doctor,V., Davis,R., Shen,P., Cavalli-Sforza,L. and Oefner,P. (1999) Distribution of haplotypes from a chromosome 21 region distinguished multiple prehistoric human migrations. *Proc. Natl Acad. Sci. USA*, **96**, 3796–3800.

Lancia,G., Bafna,V., Istrail,S., Lippert,R. and Schwartz,R. (2001) SNPs problems, complexity, and algorithms. In *Proceedings of the Ninth Annual European Symposium on Algorithms (ESA'01)*, pp. 182–193.

Lippert,R., Schwartz,R., Lancia,G. and Istrail,S. (2002) Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Briefings in Bioinformatics*, **3**, 23–31.

Long,J., Williams,R. and Urbanek,M. (1995) An E–M algorithm and testing strategy for multi-locus haplotypes. *Am. J. Hum. Genet.*, **56**, 799–810.

Niu,T., Qin,Z., Xu,X. and Liu,J. (2002) Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *Am. J. Hum. Genet.*, **70**, 157–169.

Rieder,M., Taylor,S., Clark,A. and Nickerson,D. (1999) Sequence variation in the human angiotensin converting enzyme. *Nat. Genet.*, **22**, 59–62.

Rizzi,R., Bafna,V., Istrail,S. and Lancia,G. (2002) Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. In Guigo,R. and Gusfield,D. (eds.), *Algorithms in Bioinformatics, Second International Workshop (WABI'02)*, pp. 29–43.

Schwartz,R., Clark,A. and Istrail,S. (2002) Methods for inferring block-wise ancestral history from haploid sequences. In Guigo,R. and Gusfield,D. (eds.), *Algorithms in Bioinformatics, Second International Workshop (WABI'02)*, pp. 44–59.

Stephens,M., Smith,N. and Donnelly,P. (2001) A new statistical method for haplotype reconstruction. *Am. J. Hum. Genet.*, **68**, 978–989.

Wang,X. and Miao,J. (2002) In-silico haplotyping: state-of-the-art. UDEL Technical Reports.

Zhang,J., Vingron,M. and Hoehe,M. (2001) On haplotype reconstruction for diploid populations. EURANDOM Report 2001-026.