INFORMS Journal on Computing
Vol. 16, No. 4, Fall 2004, pp. 360–370
ISSN 0899-1499 | EISSN 1526-5528 | 04 | 1604 | 0360

DOI 10.1287/ijoc.1040.0088

Computational Problems in Noisy SNP and Haplotype Analysis: Block Scores, Block Identification, and Population Stratification

Gad Kimmel
School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel, kgad@tau.ac.il

Roded Sharan
International Computer Science Institute, 1947 Center St., Suite 600, Berkeley, California 94704, USA, roded@icsi.berkeley.edu

Ron Shamir
School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel, rshamir@tau.ac.il

The study of haplotypes and their diversity in a population is central to disease-association research. We study several problems arising in haplotype block partitioning. Our objective function is the total number of distinct haplotypes in blocks. We show that the problem is NP-hard when there are errors or missing data, and provide approximation algorithms for several of its variants. We also give an algorithm that solves the problem with high probability under a probabilistic model that allows noise and missing data. In addition, we study the multipopulation case, where one has to partition the haplotypes into populations and seek a different block partition in each one. We provide a heuristic for that problem and use it to analyze simulated and real data. On simulated data, our blocks resemble the true partition more than the blocks generated by the LD-based algorithm of Gabriel et al (2002). On single-population real data, we generate a more concise block description than do extant approaches, with better average LD within blocks. The algorithm also gives promising results on real two-population genotype data.

Key words: haplotype; block; genotype; SNP; subpopulation; stratification; algorithm; complexity
History: Accepted by Harvey J. Greenberg, Guest Editor; received August 2003; revised March 2004; accepted April 2004.

1. Introduction

The availability of a nearly complete human genome sequence makes it possible to look for telltale differences between DNA sequences of different individuals on a genome-wide scale, and to associate genetic variation with medical conditions. The main source of such information is single nucleotide polymorphisms (SNPs). Millions of SNPs have already been detected (Sachidanandam et al. 2001, Venter et al. 2001), out of an estimated total of 10 million common SNPs (Kruglyak and Nickerson 2001). This abundance is a blessing, as it provides very dense markers for association studies. Yet, it is also a curse, as the cost of typing every individual SNP becomes prohibitive. Haplotype blocks allow researchers to use the plethora of SNPs at a substantially reduced cost.

The sequence of alleles in contiguous SNP positions along a chromosomal region is called a *haplotype*. A major recent discovery is that haplotypes tend to be preserved along relatively long genomic stretches, with recombination occurring primarily in narrow regions called *hot spots* (Gabriel et al. 2002, Patil et al. 2001). The regions between two neighboring hot spots are called *blocks*, and the number of distinct haplotypes within each block that are observed in a population is very limited: typically, some 70% to 90% of the haplotypes within a block belong to very few (two to five) *common haplotypes* (Patil et al. 2001). The remaining haplotypes are called *rare haplotypes*. This finding is very important to disease-association studies because once the blocks and common haplotypes are identified, one can hopefully obtain a much stronger association between a haplotype and a disease phenotype. Moreover, rather than typing every individual SNP, one can choose few representative SNPs from each block that suffice to determine the haplotype. Using such *tag SNPs* allows a major saving in typing costs.

Due to their importance, blocks have been studied quite intensively recently. Daly et al. (2001) and Patil et al. (2001) used a greedy algorithm to find a partition into blocks that minimizes the total number of

SNPs that distinguish a prescribed fraction of the haplotypes in each block. Zhang et al. (2002) provided a dynamic-programming algorithm for the same purpose. Koivisto et al. (2003) provided a method based on minimum description length to find haplotype blocks. Bafna et al. (2003) proposed a combinatorial measure for comparing block partitions and suggested a different approach to find tag SNPs that avoids the partition into blocks. For an excellent, recent review on computational aspects of haplotype analysis, see Halldorsson et al. (2003).

In this paper we address several problems that arise in haplotype studies. Our starting point is a very natural optimization criterion: we wish to find a block partition that minimizes *the total number of distinct haplotypes* that are observed in all the blocks. This criterion for evaluating a block partition follows naturally from the above-mentioned observation: within blocks in the human genome, only a few common haplotypes are observed (Patil et al. 2001, Daly et al. 2001, Gabriel et al. 2002). The same criterion is used in the pure-parsimony approach for haplotype inference, where the problem is to resolve genotypes into haplotypes, using a minimum number of distinct haplotypes (Gusfield 2003). In this case, the problem was shown to be NP-hard (Hubbell 2003, cf. Halldorsson et al. 2003). This criterion was also proposed by Gusfield (2001) as a secondary criterion in refinements to Clark's inference method (Clark 1990). Minimizing the total number of haplotypes in blocks can be done in polynomial time (if there are no data errors) using a dynamic-programming algorithm. As we shall show, the problem becomes hard when errors are present or some of the data are missing. In fact, the problem of scoring a single given block turns out to be the bottleneck. Note that in practice, one has to account for rare haplotypes and hence minimize the total number of *common* haplotypes.

The input to all the problems we address is a *haplotype matrix A* with columns corresponding to SNPs in their order along the chromosome and rows corresponding to individual chromosomal segments typed. Because virtually all SNP sites have two alleles, we adopt the common assumption that the matrix is binary after we transform the two distinct alleles at each site arbitrarily to 0 and 1. A_{ij} is the allele type of chromosome i in SNP j. The first set of problems that we study concerns the scoring of a single block in the presence of errors or missing data. In one problem variant, we wish to find a minimum number of haplotypes such that by making at most E changes in the matrix, each row vector is transformed into one of them. We call this problem *total block errors* (TBE). We show that the problem is NP-hard, and provide a polynomial 2-approximation algorithm to a variant of TBE, where one wishes to minimize the total number of errors induced by the solution and the number of common haplotypes is bounded. In a second problem, we wish to minimize the number of haplotypes when the *maximum* number of errors between a given row and its (closest) haplotype is bounded by e. We call this problem *local block errors* (LBE). This problem is shown to be NP-hard too, and we provide a polynomial algorithm (for fixed e) that guarantees a logarithmic approximation factor. In a third variant, some of the data entries are missing (manifested as *question marks* in the block matrix), and we wish to complete each of them by zero or one so that the total number of resulting haplotypes is minimized. Again, we show that this *incomplete haplotypes* (IH) problem is NP-hard. To overcome the hardness we resort to a probabilistic approach. We define a probabilistic model for generating haplotype data, including errors, missing data, and rare haplotypes, and provide an algorithm that scores a block correctly with high probability under this model.

Another problem that we address is stratifying the haplotype populations. It has been shown that the block structure in different populations is different (Gabriel et al. 2002). When the partition of the sample haplotypes into subpopulations is unknown, determining a single block structure for all the haplotypes can create artificial solutions with far too many haplotypes. We define the *minimum block haplotypes* (MBH) problem, where one has to partition the haplotyped individuals into subpopulations and provide a block structure for each one so that the total number of distinct haplotypes over all subpopulations and their blocks is minimum. We show that MBH is NP-hard, but we also provide a heuristic for solving it in the presence of errors, missing data, and rare haplotypes. The algorithm uses ideas from the probabilistic analysis.

We applied our algorithm to several synthetic and real datasets. We show that the algorithm can identify the correct number of subpopulations in simulated data, and that it is robust to noise sources. On simulated data, when compared to the LD-based algorithm of Gabriel et al. (2002), we show that our algorithm forms a partition into blocks that is much more faithful to the true one. On a real dataset of Daly et al. (2001) we generate a more concise block description than do extant approaches, with a better average value of high LD-confidence fraction within blocks. As a final test, we applied our MBH algorithm to the two largest subpopulations reported in Gabriel et al. (2002). As these were genotype data, we treated heterozygotes as missing data. Nevertheless, the algorithm determined that there are two subpopulations and correctly classified over 95% of the haplotypes.

The paper is organized as follows. In §2 we study the complexity of scoring a block under various noise sources and present our probabilistic scoring algorithm. In §3 we study the complexity of the MBH problem and describe a practical algorithm for solving it. Section 4 contains our results on simulated and real data.

A preliminary version of the results of this paper is to appear in Proceedings of the Third Workshop on Algorithms in Bioinformatics (WABI) (Kimmel et al. 2003).

2. Scoring Noisy Blocks

In this section we study the problem of minimizing the number of distinct haplotypes in a single block under various noise sources. This number will be called the *score* of the block. The scoring problem arises as a key component in block partitioning in single- and multiple-population situations.

The input is a haplotype matrix A with n rows (haplotypes) and m columns (SNPs). A may contain errors (where 0 is replaced by 1 and vice versa), resulting from point mutations or measurement errors, and missing entries, denoted by "?". Clearly, if there are no errors or missing data then a block can be scored in time proportional to its size by a hashing algorithm. Below we define and analyze several versions of the scoring problem that incorporate errors into the model. We assume until §2.4 that there are no rare haplotypes. In the following we denote by v_i the ith row vector (haplotype) of A, and by $V = \{v_1, \ldots, v_n\}$ the set of all n row vectors.

2.1. Minimizing the Total Number of Errors

First we study the following problem: We are given an integer E, and wish to determine the minimum number of (possibly new) haplotypes, called *centroids*, such that by changing at most E entries in A, every row vector is transformed into one of the centroids. Formally, let $h(\cdot, \cdot)$ denote the Hamming distance between two vectors. Define the following problem:

Problem 1 (Total Block Errors (TBE)). Given a binary haplotype matrix A and an integer E, find a minimum number k of centroids $v_1, \ldots, v_k$, such that $\sum_{u\in V} \min_i h(u, v_i) \leq E$.

Determining if $k = 1$ can be done trivially in $O(nm)$ time by observing that the minimum number of errors is obtained when choosing v_1 to be the consensus vector of the rows of A. The general problem, however, is NP-hard, as shown below:

Theorem 1. *TBE is NP-hard.*

Proof. We provide a reduction from VERTEX COVER (Garey and Johnson 1979). Given an instance $(G = (W = \{w_1, \ldots, w_m\}, F = \{e_1, \ldots, e_n\}), k)$ of VERTEX COVER, where w.l.o.g. $k < m - 1$, we form an instance $(A, k+1, E)$ of TBE. A is an $(n + mn^2) \times m$ matrix, whose rows are constructed as follows:

(1) For each edge $e_i = (s, t) \in F$, we form a binary vector v_{e_i} with 1 in positions s and t, and 0 in all other positions.

(2) For vertex $w_i \in W$ define the *vertex vector* u_i as the vector with 1 in its ith position, and 0 otherwise. For each $w_i \in W$ we form a set U_i of n^2 identical copies of u_i.

Finally, define $E = n + n^2(m - k)$. We shall prove that G has a vertex cover of size at most k if and only if there is a solution to TBE on A with at most $k + 1$ centroids and E errors.

($\Rightarrow$) Suppose that G has a vertex cover $\{w_1, \ldots, w_t\}$ with $t \leq k$. Take some cover with $t = k$. Partition the rows of A into the following subsets: for $1 \leq i \leq t$ the ith subset will contain all vectors corresponding to edges that are covered by v_i (if an edge is covered by two vertices, choose one arbitrarily), along with the n^2 vectors in U_i. Its centroid will be w_i. The $(t+1)$st subset will contain all vectors corresponding to vertices of G that are not members of the vertex cover, with its centroid being the all-0 vector. It is easy to verify that the number of errors induced by this partition is exactly $n + n^2(m - k) = E$.

($\Leftarrow$) Suppose that A can be partitioned into at most $t+1$ subsets with corresponding centroids (with $t \leq k$) such that the number E^* of induced errors is at most E. In particular, examine a partition that induces a minimum number of errors. W.l.o.g., we can assume that for each i all vectors in U_i belong to the same set in the partition. For each vertex $i \in W$, the set U_i induces at least n^2 errors, unless u_i is one of the centroids. Let l be the number of centroids that correspond to vertex vectors. Then the number E' of errors induced by the remaining $m - l$ sets of vertex vectors is at least $(m-l)n^2$. But because $E' \leq E^*$, it follows that $(m - l)n^2 \leq E = (m - k)n^2 + n$. Hence, $k \leq l + 1/n$ and by integrality $k \leq l$. Now, $l \leq t + 1 \leq k + 1$. Suppose to the contrary that $l = k + 1$. Because the Hamming distance of any two distinct vertex vectors is 2, we get $E' \geq 2(m-k-1)n^2 > E$ (because $m > k+1$), a contradiction. Thus, $l = k$. We claim that these k vertices form a vertex cover of G. By the argument above, every other vertex vector must belong to the $(k+1)$st subset and, moreover, its centroid must be the all-0 vector. Consider a vector w corresponding to an edge (u, w). If w is assigned to the $(k+1)$st subset, it adds 2 to E^*. Similarly, if w is assigned to one of the first k subsets corresponding to a vertex v, and $u, w \neq v$, then w adds 2 to E^*. Because there are n edges and the assignment of vertex vectors induced $E' = n^2(m-k) \geq E - n$ errors, each edge can induce at most one error. Hence, each edge induces exactly one error, implying that every edge is incident to one of the k vertices. □

Due to the hardness of TBE, we resort to enumerative approaches. We study the optimization version where E is to be minimized. A straightforward approach is to enumerate the centroids in the solution and assign each row vector of A to its closest centroid. Suppose there are k centroids in an optimum solution. Then the complexity of this approach is $O(kmn2^{mk})$, which is feasible only for very small m and k. In the following we present an alternative approach. We devise a $(2-2/n)$-approximation algorithm, which takes $O(n^2m+kn^{k+1})$ time.

To describe the algorithm and prove its correctness we use the following lemma, that focuses on the problem of seeking a single centroid $v \in W$ for the set of vectors $W=\{v_1,\ldots,v_n\}$. Denote $\tilde{v}_b \equiv \arg\min_{v\in\{0,1\}^m}\sum_{i=1}^n h(v,v_i)$, and let $E\equiv\sum_{v\in W}h(v,\tilde{v}_b)$.

Lemma 2. *Let* $v_b=\arg\min_{v\in W}\sum_{i=1}^n h(v,v_i)$. *Then* $\sum_{i=1}^n h(v_b,v_i)\le(2-2/n)E$.

Proof. Define $s\equiv\sum_{1\le i<j\le n}h(v_i,v_j)$. We first claim that $s\le E(n-1)$. Then,

$$s=\sum_{i<j}h(v_i,v_j)\le\sum_{i<j}[h(v_i,\tilde{v}_b)+h(\tilde{v}_b,v_j)]$$
$$=(n-1)\sum_i h(v_i,\tilde{v}_b)=(n-1)E.$$

The first inequality follows because the Hamming distance satisfies the triangle inequality. The last equality follows by using $\tilde{v}_b$ as the centroid. This proves the claim.

By the definition of v_b, for every $v_c\ne v_b$ we have

$$\sum_{v_i\in V}h(v_b,v_i)\le\sum_{v_i\in V}h(v_c,v_i).$$

Summing the above inequality for all n vectors, noting that $h(v,v)=0$, we get

$$n\sum_{v_i\in V}h(v_b,v_i)\le 2\sum_{1\le i<j\le n}h(v_i,v_j)=2s\le 2E(n-1).\quad\square$$

Theorem 3. *TBE can be* $(2-2/n)$*-approximated in* $O(n^2m+kn^{k+1})$ *time.*

Proof. Algorithm: Our algorithm enumerates all possible subsets of k rows in A as centroids, assigns each other row to its closest centroid, and computes the total number of errors in the resulting solution.

Approximation Factor. Consider two (possibly equal) partitions of the rows of A: $P_{\text{alg}}=(A_1,\ldots,A_k)$, the one returned by our algorithm; and $P_{\text{best}}=(\hat{A}_1,\ldots,\hat{A}_k)$, a partition that induces a minimum number of errors. For $1\le i\le k$ denote

$$v_b^i=\arg\min_{v\in A_i}\sum_{v_j\in A_i}h(v,v_j),\quad \hat{v}_b^i=\arg\min_{v\in\hat{A}_i}\sum_{v_j\in\hat{A}_i}h(v,v_j).$$

The number of errors induced by P_{alg} and P_{best} are

$$E_{\text{alg}}=\sum_{i=1}^k\sum_{v\in A_i}h(v_b^i,v)\quad\text{and}\quad E_{\text{best}}=\sum_{i=1}^k\sum_{v\in\hat{A}_i}h(\hat{v}_b^i,v),$$

respectively. Finally, let $n_i=|\hat{A}_i|$ and denote by e_i the minimum number of errors induced in subset $\hat{A}_i$, by the optimal solution. In particular, $\sum_{i=1}^k n_i=n$ and $\sum_{i=1}^k e_i=E$.

Because our algorithm checks all possible solutions that use k of the original haplotypes as centroids and chooses a solution that induces a minimal number of errors, $E_{\text{alg}}\le E_{\text{best}}$. By Lemma 2, $\sum_{v\in\hat{A}_i}h(\hat{v}_b^i,v)\le(2-2/n_i)e_i$ for every $1\le i\le k$. Summing this inequality over all $1\le i\le k$ we get

$$E_{\text{alg}}\le E_{\text{best}}=\sum_{i=1}^k\sum_{v\in\hat{A}_i}h(\hat{v}_b^i,v)\le\sum_{i=1}^k\left(2-\frac{2}{n_i}\right)e_i$$
$$\le\sum_{i=1}^k\left(2-\frac{2}{n}\right)e_i=\left(2-\frac{2}{n}\right)E.$$

Complexity. As a preprocessing step we compute the Hamming distance between every two rows in $O(n^2m)$ time. There are $O(n^k)$ possible sets of centroids. For each centroid set, assigning rows to centroids and computing the total number of errors takes $O(kn)$ time. The complexity follows. □

We note that Ostrovsky et al. (2002) presented a probabilistic algorithm for the above problem, with an approximation ratio of $(1+4\sqrt{\epsilon})^2$, where $\frac{1}{4}\ge\epsilon>0$.

2.2. Handling Local Data Errors

In this section we treat the question of scoring a block when the *maximum* number of errors between a haplotype and its centroid is bounded. Formally, we study the following problem.

Problem 2 (Local Block Errors (LBE)). Given a block matrix A and an integer e, find a minimum number k of centroids $v_1,\ldots,v_k$ and a partition $P=(V_1,\ldots,V_k)$ of the rows of A, such that $h(u,v_i)\le e$ for every i and every $u\in V_i$.

Theorem 4. *LBE is NP-hard even when* $e=1$.

Proof. We use the same construction as in the proof of Theorem 1. We claim that the VERTEX COVER instance has a solution of cardinality at most k if and only if the LBE instance has a solution of cardinality at most $k+1$, such that at most one error is allowed in each row. The "only if" part is immediate from the proof of Theorem 1. For the "if" part, observe that any two vectors corresponding to a pair of independent edges cannot belong to the same subset in the partition, and so is the case for a vertex vector and any vector corresponding to an edge that is not incident on that vertex. This already implies a vertex cover of size at most $k+1$. Because $m>k+1$ there must be a subset in the partition that contains at least two vectors corresponding to distinct vertices.

But then either it contains no edge vector, or it contains exactly one edge vector and the vectors corresponding to its endpoints. In any case we obtain a vertex cover of the required size. □

Theorem 5. *There is an $O(\log n)$ approximation algorithm for LBE that takes $O(n^2m^e)$ time.*

Proof. Our approximation algorithm for LBE is based on a reduction to SET COVER. Let V be the set of row vectors of A. Define the *e-set* of a vector v as the set of vectors of the same length that have Hamming distance at most e to v. Denote this e-set by $e(v)$. Let U be the union of all e-sets of row vectors of A. We reduce the LBE instance to a SET COVER instance $(V, \mathcal{S})$, where $\mathcal{S} \equiv \{e(v) \cap V \colon v \in U\}$. Clearly, there is a 1-1 correspondence between solutions for the LBE instance and solutions for the SET COVER instance, and that correspondence preserves the cardinality of the solutions. We now apply an $O(\log n)$-approximation algorithm for SET COVER (see, e.g., Cormen et al. 1990) to $(V, \mathcal{S})$ and derive a solution to the LBE instance, which is within a factor of $O(\log n)$ of optimal. The complexity follows by observing that $|U| = O(nm^e)$. □

2.3. Handling Missing Data

In this section we study the problem of scoring an *incomplete matrix*, i.e., a matrix in which some of the entries may be missing. The problem is formally stated as follows.

Problem 3 (Incomplete Haplotypes (IH)). Given an incomplete haplotype matrix A, complete the missing entries so that the number of haplotypes in the resulting matrix is minimum.

Theorem 6. *IH is NP-hard.*

Proof. We present a reduction from GRAPH COLORING (Garey and Johnson 1979). Given an instance $(G = (W, E), k)$ of GRAPH COLORING we build an instance (A, k) of IH as follows. Let $W = \{1, \dots, n\}$. Each $i \in W$ is assigned an n-dimensional row vector v_i in A with 1 in the ith position, 0 in the jth position for every $(i, j) \in E$, and "?" in all other positions.

Given a k-coloring of G, let $W_1, \dots, W_k$ be the corresponding color classes. For each class $W_i = \{v_{j_1}^{(i)}, \dots, v_{j_l}^{(i)}\}$ we complete the ?s in the vectors corresponding to its vertices as follows. Each ? in one of the columns $v_{j_1}^{(i)}, \dots, v_{j_l}^{(i)}$ is completed to 1, and all the others are completed to 0. The resulting matrix contains exactly k distinct haplotypes. Each haplotype corresponds to a color class, and has 1 in position i if and only if i is a member of the color class.

Conversely, given a solution to IH of cardinality at most k, each of the solution haplotypes corresponds to a color class in G. This follows because any two vectors corresponding to adjacent vertices must have a column with both 0 and 1 and, thus, represent two different haplotypes. □

2.4. A Probabilistic Algorithm

In this section we define a probabilistic model for the generation of haplotype block data. The model is admittedly naive, in that it assumes equal allele frequencies and independence between different SNPs and distinct haplotypes. However, as we shall see in §§3 and 4, it provides useful insights towards an effective heuristic that performs well on real data. We give a polynomial algorithm that computes the optimal score of a block under this model with high probability (w.h.p.). Our model allows for all three types of confusing signals mentioned earlier: rare haplotypes, errors, and missing data.

Denote by T the hidden true haplotype matrix, and by A the observed one. Let T' be a submatrix of T, which contains one representative of each haplotype in T (common and rare). We assume that the entries of T' are drawn independently according to a Bernoulli distribution with parameter 0.5. T is generated by duplicating each row in T' an arbitrary number of times. This completes the description of the probabilistic model for T. Note that we do not make any assumption on the relative frequencies of the haplotypes. We now introduce errors to T by independently flipping each entry of T with probability $\alpha < 0.5$. Finally, each entry is independently replaced with a ? with probability p. Let A be the resulting matrix, and let A' be the submatrix of A induced by the rows in T'. Under these assumptions, the entries of A' are independently identically distributed as follows: $A'_{ij} = 0$ with probability $(1-p)/2$, $A'_{ij} = 1$ with probability $(1-p)/2$ and $A'_{ij} = ?$ with probability p.

We say that two vectors x and y have a *conflict* in position i if one has value 1 and the other has value 0 in that position. Define the dissimilarity $d(x, y)$ of x and y as the number of their conflicting positions (in the absence of ?s, this is just the Hamming distance). We say that x is *independent* of y and denote it by $x \parallel y$, if x and y originate from two different haplotypes in T. Otherwise, we say that x and y are *mates* and denote it by $x \approx y$. Intuitively, independent vectors will have higher dissimilarity compared to mates. In particular, for any i:

$$\begin{aligned} p_I &\equiv \mathrm{Prob}(x_i = y_i \mid x \parallel y;\, x_i, y_i \in \{0,1\}) = 0.5, \\ p_M &\equiv \mathrm{Prob}(x_i = y_i \mid x \approx y;\, x_i, y_i \in \{0,1\}) \\ &= \alpha^2 + (1-\alpha)^2 > 0.5. \end{aligned} \tag{1}$$

Problem 4 (Probabilistic Model Block Scoring (PMBS)). Given an incomplete haplotype block matrix A, find a minimum number k of centroids $v_1, \dots, v_k$, such that under the above probabilistic model, with high probability, each vector $u \in A$ is a mate of some centroid.

```
Score($A$):
    1. Let $V$ be the set of rows in $A$.
    2. Initialize a heap $S$.
    3. While $V \neq \emptyset$ do:
        (a) Choose some $v \in V$.
        (b) $H \leftarrow \{v' \in V \mid d(v, v') \leq t^*\}$.
        (c) $V \leftarrow V \setminus H$.
        (d) Insert($S$,$|H|$).
    4. Output $S$.
```

Figure 1 **An Algorithm for Scoring a Block Under a Probabilistic Model of the Data**

Note. Procedure insert(S, s) inserts a number s into a heap S.

Our algorithm for scoring a block A under the above probabilistic model is described in Figure 1. It uses a threshold t^* on the dissimilarity between vectors to decide on mate relations. We set t^* to be the average of the expected dissimilarity between mates and of the expected dissimilarity between independent vectors (see proof of Theorem 7). The algorithm produces a partition of the rows into mate classes of cardinalities $s_1 \geq s_2 \geq \cdots \geq s_l$. Given any lower bound γ on the fraction of rows that need to be covered by the common haplotypes, we give A the score $h = \arg\min_j \sum_{i=1}^{j} s_i \geq \gamma n$. We prove below that w.h.p. h is the correct score of A.

Theorem 7. *If $m = \omega(\log n)$ then w.h.p. the algorithm computes the correct score of A.*

Proof. We prove that w.h.p. each mate relation decided by the algorithm is correct. Applying a union bound over all such decisions will give the required result. Fix an iteration of the algorithm at which v is the chosen vertex and let $v' \neq v$ be some row vector in A. Let X_i be a binary random variable that is 1 if and only if v_i and v'_i are in conflict. Clearly, all X_i are independent identically distributed Bernoulli random variables. Define $X \equiv d(v, v') = \sum_{i=1}^{m} X_i$ and $f \equiv (1-p)^2$. Using (1) we conclude:

$$(X \mid v' \parallel v) \sim \text{Binom}(m, f(1-p_I)),$$
$$(X \mid v' \approx v) \sim \text{Binom}(m, f(1-p_M)).$$

We now require the following Chernoff bound (cf. Alon and Spencer 2000). If $Y \sim \text{Binom}(n, s)$ then for every $\epsilon > 0$ there exists $c_\epsilon > 0$ that depends only on ϵ, satisfying:

$$\text{Prob}[|Y - ns| \geq \epsilon ns] \leq 2e^{-c_\epsilon ns}.$$

Let $\mu = mf(1-p_M)$. Define $\epsilon \equiv [(1-p_I) - (1-p_M)]/(2(1-P_M))$ and $t^* \equiv \epsilon\mu$. Applying the Chernoff bound and using the assumption that $m = \omega(\log n)$, we have that for all $c > 0$:

$$\text{Prob}(X > t^* \mid v' \approx v) \leq 2e^{-c_\epsilon m} < \frac{1}{n^c},$$
$$\text{Prob}(X \leq t^* \mid v' \parallel v) < \frac{1}{n^c}.$$

Because we check whether $d(v, v') < t^*$ a total of $O(n^2)$ times, by applying a union bound we conclude that the probability that throughout the algorithm some implied mate relation is incorrect and is bounded by a polynomial in $1/n$. □

When using the algorithm as part of a practical heuristic (see §3), we do not report the rare haplotypes. Instead, we report only the smallest number of the most abundant haplotypes as computed by the algorithm that together capture a fraction γ of all haplotypes.

3. The Multipopulation Case

Suppose that the matrix A contains haplotypes from several homogeneous populations. The partitioning into blocks can differ among populations (Gabriel et al. 2002). Here, we study the question of reconstructing the partition of the rows of A into sets called *subpopulations*, and the columns in each set into blocks, such that the sum of the scores of the submatrices corresponding to these blocks is minimized.

Problem 5 (Minimum Block Haplotypes (MBH)). Given a haplotype matrix A, find a partition of its rows into subpopulations so that the total number of block haplotypes is minimized.

In practice, we usually have full information on the population from which each of the haplotypes originates. However, in certain situations there may be a hidden stratification of a population that can affect the conclusions of association studies on it. Problem 5 aims to address such situations.

3.1. Minimum Block Haplotypes

For a haplotype matrix A and a subset S of its rows, we denote by H_S^A the (minimum) total number of block haplotypes in an optimal partition of S into blocks. Our goal is to find $H^A = H_V^A$. Given a partition $P = (P_1, \ldots, P_r)$ of the rows of A into subpopulations, we let $H^A(P) = \sum_{i=1}^{r} H_{P_i}^A$, that is, the (minimum) total number of block haplotypes in an optimal partition of each subpopulation into blocks. In the following we omit the superscript A when it is clear from the context. Given a partition P, $H(P)$ can be polynomially computed in the noiseless case using a simple adaptation of the dynamic-programming algorithm of Zhang et al. (2002). However, the general MBH problem is NP-hard.

Theorem 8. *MBH is NP-hard.*

Proof. We provide a reduction from VERTEX COVER (Garey and Johnson 1979). Let $(G = (V, E), k)$ be an instance of VERTEX COVER where $|V| = n$, $|E| = m$, and w.l.o.g. $n < m$. We build an instance $(A, n(8m+4+2m^2)+12m+2k)$ of (the decision version of) MBH as follows. We associate with the vertices and edges of G row vectors of dimension $c = (2n+1)m^{10}$. These vectors will constitute the matrix A. Each of the row vectors v is partitioned into segments where the segment of length m^{10} between positions $i^- \equiv (i-1)2m^{10}+1$ and $i^+ \equiv (i-1)2m^{10}+m^{10}$ corresponds to vertex i. The m^{10} last positions in v are called its *tail*.

The content of each segment will be a periodic binary sequence. For an integer k let S_k be the sequence $(0, \ldots, 0, 1)$ of length k where $S_0 = (0)$ and $S_1 = (1)$. For convenience we denote S_k also as S_k^1, and use S_k^{-1} to denote the complement of that sequence. Each of the vector segments consist of repetitions of some S_k or its complement. We denote by $S_k(l)$ the sequence formed by concatenating copies of S_k up to a total length of l where the last copy may be truncated.

For an ordered sequence of integers $1 = i_1 < \cdots < i_{l+1} = c+1$, inducing a partition of $[1, \ldots, c]$, we define the following vector set:

$$\begin{aligned}&U_{i_1, \ldots, i_{l+1}}(k_1, \ldots, k_l)\\ &\qquad \equiv \bigcup_{r_1, \ldots, r_l \in \{1, -1\}} \big(S_{k_1}^{r_1}(i_2 - i_1), \ldots, S_{k_l}^{r_l}(i_{l+1} - i_l)\big).\end{aligned}$$

In words, $U_{i_1, \ldots, i_{l+1}}(k_1, \ldots, k_l)$ is a set of 2^l vectors of dimension c, where the sth vector contains in its tth segment copies of $S_{k_t}^r$ with $r = 1$ iff the tth bit of s is 0.

With each vertex v_i we associate the set of $2 \cdot (2 \cdot 4m) \cdot 2 \cdot 2m^2 = 64m^3$ vectors:

$$V_i = \bigcup_{1 \le j \le 4m,\, im^2 \le k < (i+1)m^2} U_{1,\, i^-,\, i^+ + 1,\, c - m^{10} + m^9 i,\, c+1}(0, j, 0, k).$$

Thus, each vertex vector has four segments: until position i^- it is all zeros or all ones; between i^- and i^+ it has one of $4m$ possible sequences or their complements; until the beginning of its tail it is again all zeros or all ones; and then at a unique position, which depends on the vertex identity, starts one of $2m^2$ possible tail sequences for that vertex.

With each edge e_l: $1 \le l \le m$ connecting vertices i and j, where $i < j$, we associate a set of $2 \cdot (2 \cdot 4) \cdot 2 \cdot (2 \cdot 4) \cdot 2 = 512$ vectors

$$E_l = \bigcup_{p = 4l-3}^{4l} U_{1,\, i^-,\, i^+ + 1,\, j^-,\, j^+ + 1,\, c+1}(0, p, 0, p, 0).$$

Thus, each edge vector contains one of eight possible sequences in its (i^-, i^+) and (j^-, j^+) segments, and these sequences are unique for each edge.

By construction, $H_{V_i} = 2 + 8m + 2 + 2m^2 = 8m + 4 + 2m^2$ and $H_{E_l} = 16 + 6 = 22$. We now prove that G has a vertex cover of size at most k if and only if A has a partition P with $H(P) \le n(8m+4+2m^2)+12m+2k$.

($\Rightarrow$) W.l.o.g., let $\{1, \ldots, t\}$ be a vertex cover of size $t \le k$ for G. Let C_i be the set of edges covered by vertex i (for an edge covered by two vertices, choose the one with smaller index) where $C_i = \varnothing$ for $i > t$. Define $A_i \equiv V_i \cup \bigcup_{j \in C_i} E_j$ for $1 \le i \le n$. Let $P = (A_1, \ldots, A_n)$. We shall prove that $H(P)$ is of the required size. Fix i and let $C_i = \{e_1, \ldots, e_p\}$ where e_j connects i to s_j and, w.l.o.g., $i < s_1 < \cdots < s_p$. We claim that $H_{A_i} = (8m+4+2m^2)+12p+2\delta$ where δ is an indicator that equals 1 if and only if $i \le t$. Consider the partition of A_i into the following blocks: $(1, i^{-}1)$, (i^-, i^+), $(i^+ + 1, s_1^{-}1)$, (s_1^-, s_1^+), $\ldots$, $(s_{p-1}^+ + 1, s_p^{-}1)$, (s_p^-, s_p^+), $(s_p^+ + 1,\ c - m^{10} + m^9 - 1)$, $(c - m^{10} + m^9, c)$. Due to V_i, A_i has two haplotypes in the first block, $8m$ haplotypes in the second block (which corresponds to the segment of vertex i), two haplotypes in the segment before last, and $2m^2$ haplotypes in the tail block. In addition, if we add the sets E_j one by one to the same subpopulation, then every such set, corresponding to the edge (i, s_j), adds two new blocks and 12 haplotypes (two haplotypes in $((j-1)^+ + 1, j^{-}1)$ and $8+2$ in (j^-, j^+)). The only exception is $j = 1$, for which two more haplotypes are added in the tail segment. Thus, if $|C_i| = p > 0$ then $H_{A_i} = (8m+4+2m^2)+12p+2$ and if A_i contains no edge vectors then $H_{A_i} = 8m+4+2m^2$. The claim follows.

($\Leftarrow$) Suppose that A has a partition $P = (A_1, \ldots, A_t)$ so that $H(P) \le n(8m+4+2m^2)+12m+2k$. In particular, examine the partition P^* for which $H \equiv H(P^*)$ is minimal. W.l.o.g. every one of V_i and E_j is completely contained in some A_k. We first claim that no set in the partition contains both V_i and V_j for $i \ne j$. Suppose this is not the case. Define a new partition P' in which V_j is moved into a new set. Then $H - H(P') \ge (2m^2+2) - 8m - 4 > 0$ where the first term is due to the tail segments of i and j and the second is due to edge vectors corresponding to edges incident on j that are possibly present in the same partition set as V_i and V_j. Thus, we arrive at a contradiction.

Now consider an edge l connecting vertices i and j, and let $A_r \supseteq E_l$. We claim that $V_i \subset A_r$ or $V_j \subset A_r$ (in P^*). To see that, observe that in the first case l adds at most 14 haplotypes to H (similar to the argument in the "only if" part of the proof), while in the second case it adds at least 16 haplotypes to H because each of the segments (i^-, i^+) and (j^-, j^+) contains eight unique haplotypes.

Finally, suppose there are t sets in P^* that contain edge vectors. Then $H \ge n(8m+4+2m^2)+12m+2t$, implying that $t \le k$ and G has a vertex cover of size at most k. □

3.2. A Polynomial Case

We now give a polynomial algorithm for a restricted version of MBH in which each subpopulation is required to be a contiguous set of rows. We call this variant *minimum contiguous block haplotypes* (*MCBH*). Its solution may be useful for designing heuristics that permute the matrix rows for local improvement. For clarity, in the discussion below we shall assume that there exists an oracle that scores a given block in $O(1)$ time. Denote the optimal solution of MCBH on A by H^A.

Theorem 9. *MCBH can be solved in* $O(n^2m^2)$ *time.*

Proof. Algorithm: Let A be an input haplotype matrix. We give a dynamic-programming procedure to solve MCBH. A key component of the algorithm is a dynamic-programming algorithm, which computes the score for a given subpopulation S in a straightforward manner, similar to Zhang et al. (2002). Let T_i^S, $0 \le i \le m$, be the minimum number of block haplotypes in the submatrix of A induced on the rows in S and the columns $1, \ldots, i$, where $T_0^S = 0$. For a pair of columns i, j let B_{ij}^S be the score of the block induced by the rows in S and the columns in $\{i, \ldots, j\}$. Then the following recursive formula can be used to compute T_m^S:

$$T_i^S = \min_{0 \le j \le i-1} T_j^S + B_{ji}^S.$$

We now use a second dynamic-programming algorithm to compute H^A. Define P_i, $0 \le i \le n$ as the minimum number of block haplotypes in any row partition of $A_{\{1,\ldots,i\}}$. Clearly, $P_0 = 0$ and $P_n = H^A$. The computation of P_i uses the following recursive formula:

$$P_i = \min_{1 \le j \le i} P_{j-1} + T_m^{\{j,\ldots,i\}}.$$

Complexity. Computing T_m^S for any S takes $O(m^2)$ time. Hence, computing H^A takes $O(n^2m^2)$ time in total. □

3.3. A Heuristic

Next, we present an efficient heuristic for MBH. The algorithm has three components: a block-scoring procedure, a dynamic-programming algorithm to find the optimum block structure for a single subpopulation, and a simulated-annealing algorithm to find an optimum partition into homogeneous subpopulations. We describe these components below.

The dynamic-programming component is as described in the first part of the proof of Theorem 9. For scoring a block within the dynamic-programming procedure we use the probabilistic algorithm described in §2.4 with a small modification: instead of using a fixed threshold t^*, we compute a different threshold $t^*_{v,v'}$ for every two vectors v, v'. This is done by counting the number l of positions in which neither of the vectors has ?, and setting $t^*_{v,v'} = \frac{1}{2}l[(1-p_M)+(1-P_I)]$. Scoring an $n \times t$ block takes $O(tnk)$ time where k is a bound on the number of common haplotypes. Hence, the dynamic program takes $O(mb^2nk)$ total time where b is an upper bound on the allowed block size. Additional saving may be possible by precomputing the pairwise distances of rows in contiguous matrix segments of size up to b.

The goal of the annealing process is to optimize the partition of the haplotypes into subpopulations. We define a *neighboring partition* as any partition that can be obtained from the current one by moving one haplotype from one group to another. The process proceeds through a sequence of neighboring partitions depending on their scores and the temperature parameter in a standard annealing fashion. A crucial factor in obtaining a good solution is the initialization of the annealing process. We perform the initialization as follows. We compute pairwise similarities between every two haplotypes. The similarity S_{uv} of vectors u and v is calculated as follows. Initially we set $S_{uv} = 0$. We then slide a window of size $w = 20$ along u and v (20 is the average size of a block). For each position i we check whether $d((u_i, \ldots, u_{i+w-1}), (v_i, \ldots, v_{i+w-1})) \le w\alpha$ (for a parameter α). If this is the case, we increment S_{uv} and jump to $i + w$ for the next iteration. Otherwise, we jump to $i + 1$. The intuition is that rows from the same subpopulation should be more similar in blocks in which they share the same haplotypes and, thus, have a better chance to hit good windows and accumulate a higher score in the scan. Next we cluster the haplotypes based on their similarity values using the K-means algorithm (MacQueen 1965). The resulting partition is taken as the starting point for the annealing process. To determine the number of subpopulations K, we try several choices and pick the one that results in the lowest score.

The running time of the practical algorithm is dominated by the cost of each annealing step. Because this step changes the haplotypes of two subpopulations only, it suffices to recompute the scores of these subpopulations only.

4. Experimental Results

4.1. Simulations

We applied our heuristic algorithm to simulated and real haplotype data. First we conducted extensive simulations to check the ability of our algorithm to detect subpopulations and recognize their block structure. Our simulation setup is as follows. We generated simulated haplotype matrices with 100 haplotypes and 300 SNPs. The number of subpopulations varied in the simulations. Subpopulations were of equal sizes. For each subpopulation we generated block

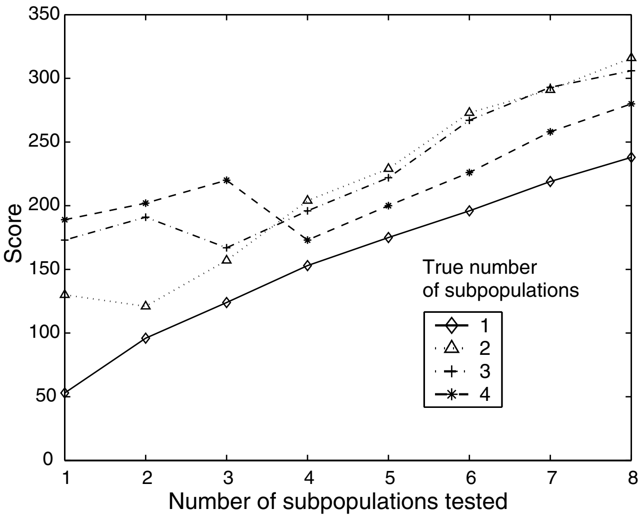

Figure 2 **Simulation Results: Determining the Number of Subpopulations**

Note. For each simulated matrix, containing one to four subpopulations, the score assigned by the algorithm to partitions (y axis) with different numbers of subpopulations (x axis). Simulations were performed with error level 1% and no missing entries.

boundaries using a Poisson process with rate 20. Each block within a subpopulation contained two to five common haplotypes, covering 90% of the block's rows (with the remaining 10% being rare haplotypes). Within each block of each subpopulation, the haplotype matrix was created according to the probabilistic model described in §2.4. Errors and missing data were introduced with varying rates of up to 30%.

As a first test we simulated several matrices with one to four subpopulations and applied our algorithm with K ranging from 1–8. For each K we computed the score of the partition obtained (as described in §3.3). In each of the simulations the correct number got the lowest score (Figure 2). Next we simulated several matrices with three subpopulations and different levels of errors and missing data. Table 1 summarizes our results in correctly assigning haplotypes to subpopulations (the set with the largest overlap with the true subpopulation was declared correct). It can be seen that the MBH algorithm gives highly accurate results for missing data and error levels up to 10%.

Table 1 **Accuracy of Haplotype Classification by the MBH Algorithm for Different Noise Levels (Data Are for Three Subpopulations)**

% Errors	% Missing entries	% Correct classifications
0	0	99
5	5	98
10	10	95
15	15	84
20	20	71

For comparison, we also implemented the LD-based algorithm of Gabriel et al. (2002) for finding blocks. We compared the block structures produced by our algorithm and by the LD-based algorithm to the correct one, using an alignment score similar to the one used in comparison of two DNA restriction enzyme maps (Waterman 1995, §9.10). The score of two partitions P_1 and P_2 of m SNPs is computed as follows. We form two vectors of size $m-1$, in which 1 in position i denotes a block boundary between SNPs i and $i+1$, and 0 denotes that the two SNPs belong to the same block. We then compute an alignment score of these vectors using an affine gap penalty model with penalties 3, 2, and 0.5 for mismatch, gap open, and gap extension, respectively, and a match score of zero.

We simulated one population with 3,000 haplotypes, computed its block structure with both algorithms, and compared them to the true one. We repeated this experiment with different error and missing-data rates. The results are shown in Figure 3. It can be observed that our algorithm yields partitions that are closer to the true ones, particularly as the rate of errors and missing data rises. An example of the actual block structures produced is shown in Figure 4.

4.2. Real Data

We applied our algorithm to two published datasets. The first dataset of Daly et al. (2001) consists of 258 haplotypes and 103 SNPs. We applied our block partitioning algorithm with the following parameters: the maximal allowed error ratio between two vectors to be considered as resulting from a single haplotype was 0.02. In addition, we allowed up to 5% rare

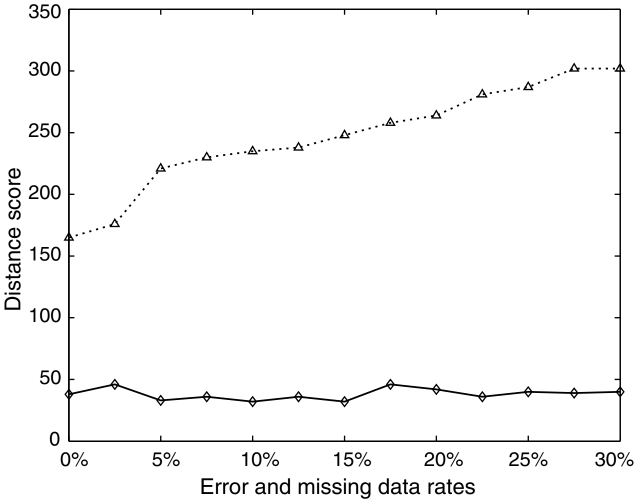

Figure 3 **Accuracy in Block Reconstruction by our Algorithm (Solid Line) and the Algorithm of Gabriel et al. (2002) (Dashed Line)**

Note. y axis: the score of aligning the reconstructed structure with the correct one. x axis: the noise rate.

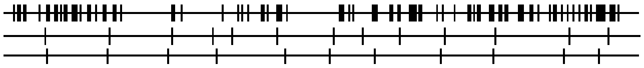

Figure 4 **An Example of the Block Structures Produced for an Error Rate of 1% by Our Algorithm (Bottom), the LD-Based Algorithm of Gabriel et al. (2002) (Top), and the True Solution (Middle)**

Note. Each block boundary is denoted by a vertical line.

haplotypes, i.e., in scoring a block we sought the minimum number of different haplotypes that together cover at least 95% of the rows.

In order to assess our block partitioning and compare it to the one reported by Daly et al. (2001), we calculated LD-based measures for both partitions. Specifically, we calculated the LD-confidence values between every pair of SNPs inside the same block, using a χ^2 test, as follows. For a pair i, j of SNPs, let $P_{a,b}$, where $a, b \in \{0, 1\}$, be the frequency of occurrence of a in position i and b in position j of a haplotype. Let p_0, p_1 (q_0, q_1) denote the frequencies of haplotypes with 0 and 1 in the ith (jth) SNP, respectively. Define $D \equiv P_{00}P_{11} - P_{01}P_{10}$. D is a measure of linkage disequilibrium, and $nD^2/(p_0p_1q_0q_1)$ is distributed as χ^2, with one degree of freedom.

For each block, we calculated the fraction of SNP pairs in the block whose LD-confidence value exceeded 95% (*high LD pairs*). The average fraction over all blocks was computed as the ratio of the total number of high LD pairs inside blocks to the total number of SNP pairs within blocks.

A comparison between our block partition to the one obtained by Daly et al. (2001) is presented in Table 2. Overall, the two block partitions have similar boundaries and similar scores. The average fraction of high LD pairs in blocks for our partition was 0.823. For the partition of Daly et al. (2001) the average fraction was 0.796. Another partition was produced for these data by Eskin et al. (2003) based on minimizing the number of representative SNPs. Their partition contained 11 blocks and its average fraction of high LD pairs was 0.814.

Table 2 **Comparison Between the Blocks of Daly et al. (2001) and the Blocks Generated by Our Algorithm**

Daly et al. blocks	Fraction of high LD pairs	Our blocks	Fraction of high LD pairs
1: 1–9	0.78	1: 1–15	0.81
2: 10–15	1		
3: 16–24	0.78	2: 16–24	0.78
4: 25–35	0.95	3: 25–36	0.94
5: 36–40	0.70	4: 37–44	0.68
6: 41–45	1		
7: 46–77	0.77	5: 45–67	0.84
		6: 68–78	0.71
8: 78–85	0.50	7: 79–81	0.33
9: 86–91	0.93	8: 82–90	0.89
10: 92–98	0.95	9: 91–95	1
11: 99–103	1	10: 96–103	0.75
Average	0.796		0.822

Table 3 **Separation to Subpopulations and Block Finding on Different Regions in Part of the Data of Gabriel et al. (2002)**

Chromosome: region	#SNPs	Discovered blocks	% Correct classifications
1: 3a	119	1: 1–35, 36–119 2: 1–46, 47–119	95
2: 8a	73	1: 1–73 2: 1–73	99
6: 24a	121	1: 1–52, 53–121 2: 1–44, 45–121	98
8: 29a	104	1: 1–27, 28–104 2: 1–40, 41–104	100
9: 32a	110	1: 1–25, 26–110 2: 1–38, 39–110	99
14: 41a	141	1: 1–48, 49–63, 64–141 2: 1–12, 13–63, 64–141	100

Note. Includes subpopulations A and D

The second dataset we analyzed, of Gabriel et al. (2002) contains unresolved genotype data. In order to apply our algorithm to these data, we transformed them into haplotypes by treating heterozygous SNPs as missing data. Notably, the fraction of heterozygous sites was relatively small, so the loss in information was moderate. We considered the two largest populations in the data, A (Europeans) and D (individuals from Yoruba), consisting of 93 and 90 samples, respectively. Each population was genotyped in ~60 different regions in the genome. We analyzed six of those regions that contained over 70 SNPs. In all cases we were able to detect two different populations in the data and classify correctly over 95% of the haplotypes. The results are shown in Table 3. The results with three populations were poorer, due to the smaller size of the third population.

5. Concluding Remarks

We have introduced a simple and intuitive measure for scoring and detecting blocks in a haplotype matrix: the total number of distinct haplotypes in blocks. Using this measure along with several error models, we have studied the computational problems of scoring of a block, and of finding an optimal block structure. Most versions of the scoring problem that address imperfect data are shown to be NP-hard. A similar situation occurred with the f score function of Zhang et al. (2002). We devised several algorithms for different variants of the problem. In particular, we gave a simple algorithm, which, under an appropriate probabilistic model, scores a block correctly with high probability in the presence of errors, missing data, and rare haplotypes.

Note that our measure is adequate only when the ratio of the number n of typed individuals to the number m of SNPs is not too extreme. When n is very small and m is large, our measure might be optimized by the trivial solution of a single block.

In simulations, our score leads to more accurate block detection than does the LD-based method of Gabriel et al. (2002). While the simulation setup is quite naive, it seems to act just as favorably for the LD-based methods. The latter methods apparently tend to over-partition the data into blocks, as they demand a very stringent criterion between every pair of SNPs in the same block. This criterion is very hard to satisfy as block size increases, and the number of pairwise comparisons grows quadratically. On the data of Daly et al. (2001) we generated a slightly more concise block description than do extant approaches, with a somewhat better fraction of high-LD pairs.

We also treated the question of partitioning a set of haplotypes into subpopulations based on their different block structures, and devised a practical heuristic for the problem. On a genotype dataset of Gabriel et al. (2002) we were able to identify two subpopulations correctly, in spite of ignoring all heterozygous types. A principled method of dealing with genotype data remains a computational challenge. While in some studies the partition into subpopulations is known, others may not have this information, or further, finer partition may be detectable using our algorithm. In our model we implicitly assumed that block boundaries in different subpopulations are independent. In practice, some boundaries may be common due to the common lineage of the subpopulations. A more detailed treatment of the block boundaries in subpopulations should be considered when additional haplotype data reveal the correct way to model this situation.

Acknowledgments

R. Sharan was supported by a Fullbright grant. R. Shamir was supported by a grant from the Israel Science Foundation (Grant 309/02). The authors thank Chaim Linhart and Dekel Tsur for their comments on the manuscript.

References

Alon, N., J. H. Spencer. 2000. *The Probabilistic Method.* John Wiley and Sons, Inc., New York.

Bafna, V., B. V. Halldorsson, R. Schwartz, A. Clark, S. Istrail. 2003. Haplotyles and informative SNP selection algorithms: Don't block out information. *Proc. Seventh Annual Internat. Conf. Res. Comput. Molecular Biol.* (*RECOMB*). The Association for Computing Machinery. New York, 19–27.

Clark, A. 1990. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biol. Evolution* **7** 111–122.

Cormen, T. H., C. E. Leiserson, R. L. Rivest. 1990. *Introduction to Algorithms.* MIT Press, Cambridge, MA.

Daly, M. J., J. D. Rioux, S. F. Schaffner, T. J. Hudson, E. S. Lander. 2001. High-resolution haplotype structure in the human genome. *Nature Genetics* **29**(2) 229–232.

Eskin, E., E. Halperin, R. M. Karp. 2003. Large scale reconstruction of haplotypes from genotype data. *Proc. Seventh Annual Internat. Conf. Res. Comput. Molecular Biol.* (*RECOMB*). The Association for Computing Machinery. New York, 104–113.

Gabriel, S. B., S. F. Schaffner, H. Nguyen, J. M. Moore, J. Roy, B. Blumenstiel, J. Higgins, M. DeFelice, A. Lochner, M. Faggart, S. N. Liu-Cordero, C. Rotimi, A. Adeyemo, R. Cooper, R. Ward, E. S. Lander, M. J. Daly, D. Altshuler. 2002. The structure of haplotype blocks in the human genome. *Science* **296** 2225–2229.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, CA.

Gusfield, D. 2001. Inference of haplotypes in samples of diploid populations: Complexity and algorithms. *J. Comput. Biol.* **8**(3) 305–323.

Gusfield, D. 2003. Haplotype by pure parsimony. *Proc. Fourteenth Annual Sympos. Combin. Pattern Matching (CPM), Morelia, Mexico*. Springer, Berlin, 144–155.

Halldorsson, B. V., V. Bafna, N. Edwards, R. Lippert, S. Yooseph, S. Istrail. 2003. Combinatorial problems arising in SNP. *Discrete Math. Theoret. Comput. Sci. Lecture Notes in Computer Science*, No. 2731. Springer-Verlag, Heidelberg, Germany, 26–47.

Hubbell, E. 2003. Finding a parsimony solution to haplotype phase is NP-hard. Unpublished manuscript, Affymetrix Inc., Santa Clara, CA.

Kimmel, G., R. Sharan, R. Shamir. 2003. Identifying blocks and subpopulations in noisy SNP data. *Proc. Third Workshop Algorithms in Bioinformatics* (*WABI*). Springer-Verlag, Berlin, 303–319.

Koivisto, M., M. Perola, T. Varilo, W. Hennah, J. Ekelund, M. Lukk, L. Peltonen, E. Ukkonen, H. Mannila. 2003. An MDL method for finding haplotype blocks and for estimating the strength of haplotype block boundaries. *Proc. Pacific Sympos. Biocomputing* (*PSB*), *Big Island of Hawaii, Hawaii*, Vol. 8. World Scientific, Singapore, 502–513.

Kruglyak, L., D. A. Nickerson. 2001. Variation is the spice of life. *Nature Genetics* **27** 234–236.

MacQueen, J. 1965. Some methods for classification and analysis of multivariate observations. *Proc. Fifth Berkeley Sympos. Math. Statist. Probab.*, University of California Press, Berkeley, CA, 281–297.

Ostrovsky, R., Y. Rabani. 2002. Polynomial time approximation schemes for geometric k-clustering. *J. Assoc. Comput. Mach.* **49** 139–156.

Patil, N., A. J. Berno, D. A. Hinds, W. A. Barrett, J. M. Doshi, C. R. Hacker, C. R. Kautzer, D. H. Lee, C. Marjoribanks, D. P. McDonough, B. T. Nguyen, M. C. Norris, J. B. Sheehan, N. Shen, D. Stern, R. P. Stokowski, D. J. Thomas, M. O. Trulson, K. R. Vyas, K. A. Frazer, S. P. Fodor, D. R. Cox. 2001. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science* **294** 1719–1723.

Sachidanandam, R., D. Weissman, S. C. Schmidt, J. M. Kakol, L. D. Stein, G. Marth, S. Sherry, J. C. Mullikin, B. J. Mortimore, D. L. Willey, S. E. Hunt, C. G. Cole, P. C. Coggill, C. M. Rice, Z. Ning, J. Rogers, D. R. Bentley, P. Y. Kwok, E. R. Mardis, R. T. Yeh, B. Schultz, L. Cook, R. Davenport, M. Dante, L. Fulton, L. Hillier, R. H. Waterston, J. D. McPherson, B. Gilman, S. Schaffner, W. J. Van Etten, D. Reich, J. Higgins, M. J. Daly, B. Blumenstiel, J. Baldwin, N. Stange-Thomann, M. C. Zody, L. Linton, E. S. Lander, D. Altshuler. 2001. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature* **291** 1298–2302.

Venter, J. Craig, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne et al. 2001. The sequence of the human genome. *Science* **291** 1304–1351.

Waterman, M. S. 1995. *Introduction to Computational Biology: Maps, Sequences and Genomes.* Chapman and Hall.

Zhang, K., M. Deng, T. Chen, M. S. Waterman, F. Sun. 2002. A dynamic programming algorithm for haplotype block partitioning. *Proc. National Acad. Sci. USA* **99** 7335–7339.