Identifying Blocks and Sub-Populations in Noisy SNP Data

Gad Kimmel[1], Roded Sharan[2], and Ron Shamir[1]

[1] School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel.
{kgad,rshamir}@tau.ac.il
[2] International Computer Science Institute, 1947 Center St., Suite 600, Berkeley CA-94704.
roded@icsi.berkeley.edu

Abstract. 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 multi-population 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. [7]. On single-population real data, we generate a more concise block description than extant approaches, with better average LD within blocks. The algorithm also gives promising results on real 2-population genotype data.

Keywords: haplotype, block, genotype, SNP, sub-population, stratification, algorithm, complexity.

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 [17, 18], out of an estimated total of 10 millions common SNPs [9]. 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* [7, 16]. 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 70-90% of the haplotypes within a block belong to very few (2-5) *common haplotypes* [16]. The remaining haplotypes are called *rare haplotypes*. This finding is very important to disease association studies, since 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. [5] and Patil et al. [16] 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. [20] provided a dynamic programming algorithm for the same purpose. Koivisto et al. [14] provided a method based on Minimum Description Length to find haplotype blocks. Bafna et al. [2] proposed a combinatorial measure for comparing block partitions and suggested a different approach to find tag SNPs, that avoids the partition into blocks. For a recent review on computational aspects of haplotype analysis, see [12].

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, that within blocks in the human genome, only a few common haplotypes are observed [16, 5, 7]. 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 [11]. In this case, the problem was shown to be NP-hard [13]. This criterion was also proposed by Gusfield [10] as a secondary criterion in refinements to Clark's inference method [3]. 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. 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 binary *haplotype matrix* A with columns corresponding to SNPs in their order along the chromosome and rows corresponding to individual chromosomal segments typed. 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 in NP-hard, and provide a polynomial 2-approximation algorithm when the number of haplotypes is bounded. In a second variant, 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), which 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 replace each of them by zero or one, so that the total number of haplotypes is minimum. 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 [7]. When the partition of the sample haplotypes into sub-populations 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 sub-populations and provide a block structure for each one, so that the total number of distinct haplotypes over all sub-populations and their blocks is minimum. We show that MBH is NP-hard, but 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 sub-populations in simulated data, and is robust to noise sources. When compared to the LD-based algorithm of Gabriel et al. [7], 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. [5] we generate a more concise block description than extant approaches, with a better average value of the high LD-confidence fraction within blocks. As a final test, we applied our MBH algorithm to the two largest sub-populations reported in Gabriel et al. [7]. As this was genotype data, we treated heterozygotes as missing data. Nevertheless, the algorithm was able to determine that there are two sub-populations and correctly classified over 95% of the haplotypes.

The paper is organized as follows: In Section 2 we study the complexity of scoring a block under various noise sources and present our probabilistic scoring algorithm. In Section 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.

2 Scoring Noisy Blocks

In this section we study the problem of minimizing the number of distinct haplotypes in a 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 which incorporate errors into the model. We assume until Section 2.4 that there are no rare haplotypes. In the following we denote by v_i the i-th 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 block 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. Given an instance $(G = (V = \{1, \ldots, m\}, F = \{e_1, \ldots, e_n\}), k)$ 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 of 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 places.
2. For vertex $i \in V$ define the *vertex vector* u_i as the vector with '1' in its i-th position, and '0' otherwise. For each $i \in V$ we form a set U_i of n^2 identical copies of u_i.

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

($\Rightarrow$) Suppose that G has a vertex cover $\{v_1, \ldots, v_t\}$ with $t \leq k$. Partition the rows of A into the following subsets: For $1 \leq i \leq t$ the i-th 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 v_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 $n + n^2(m-t) \leq E$.

($\Leftarrow$) Suppose that A can be partitioned into at most $t+1$ subsets (with $t \leq k$) such that the number E^* of induced errors is at most E. 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 $i \in V$, 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 of errors induced by the rest $(m-l)$ sets of vertex vectors is $E' = (m-l)n^2 \leq (m-k)n^2 + n$. Hence, $k \leq l \leq t+1 \leq k+1$. Suppose to the contrary that $l = k+1$. Since the Hamming distance of any two distinct vertex vectors is 2, we get $E' \geq 2(m-k-1)n^2 > E$ (since $m > k+1$), a contradiction. Thus, $l = k$. We claim that these k vertices form a vertex cover of G. By the argument above each 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^*. Since 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. □

Thus, we study enumerative approaches to TBE. 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 to a variant of TBE, in which we wish to minimize the total number of errors induced by the solution. We devise a $(2 - \frac{2}{n})$-approximation algorithm for this variant, which takes $O(n^2 m + kn^{k+1})$ time.

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

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

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

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

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

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

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

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

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

□

Theorem 2. *TBE can be $(2 - \frac{2}{n})$-approximated in $O(n^2 m + kn^{k+1})$ 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: $P_{alg} = (A_1, \ldots, A_k)$, the one returned by our algorithm; and $P_{best} = (\hat{A}_1, \ldots, \hat{A}_k)$, a partition that induces a minimum number of errors. For $1 \leq i \leq k$ denote $v_b^i = \arg\min_{v \in A_i} \sum_{v_j \in A_i} h(v, v_j)$ and $\hat{v}_b^i = \arg\min_{v \in \hat{A}_i} \sum_{v_j \in \hat{A}_i} h(v, v_j)$. The number of errors induced by P_{alg} and P_{best} are $E_{alg} = \sum_{i=1}^{k} \sum_{v \in A_i} h(v_b^i, v)$ and $E_{best} = \sum_{i=1}^{k} \sum_{v \in \hat{A}_i} h(\hat{v}_b^i, v)$, 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 $\sum_{i=1}^{k} e_i = E_{best}$.

Since 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, $E_{alg} \leq E_{best}$. By Lemma 1, $\sum_{v \in \hat{A}_i} h(\hat{v}_b^i, v) \leq (2 - \frac{2}{n_i}) e_i$ for every $1 \leq i \leq k$. Summing this inequality over all $1 \leq i \leq k$ we get

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

Complexity: As a preprocessing step we compute the Hamming distance between every two rows in $O(n^2 m)$ 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. □

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 $h(u, v_i) \leq e$ for every i and every $u \in V_i$.

Theorem 3. *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 iff 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 $k+1$. Since $m > k+1$ 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. □

In the following we present an $O(\log n)$ approximation algorithm for the problem.

Theorem 4. *There is an $O(\log n)$ approximation algorithm for LBE that takes $O(n^2 m^e)$ 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 with respect to a matrix A as the set of row vectors in A 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 corresponding to row vectors of A. We reduce the LBE instance to a SET COVER instance $(V, \mathcal{S})$, where $\mathcal{S} \equiv \{e(v) \cap V : v \in U\}$. Clearly, there is a 1-1 mapping between solutions for the LBE instance and solutions for the SET COVER instance, and that mapping preserves the cardinality of the solutions. We now apply an $O(\log n)$-approximation algorithm for SET COVER (see, e.g., [4]) 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 5. *IH is NP-hard.*

Proof. By reduction from GRAPH COLORING [8]. Given an instance $(G = (V, E), k)$ of GRAPH COLORING we build an instance (A, k) of IH as follows: Let $V = \{1, \ldots, n\}$. Each $i \in V$ is assigned an n-dimensional row vector v_i in A with '1' in the i-th position, '0' in the j-th position for every $(i, j) \in E$ and '?' in all other positions.

Given a k-coloring of G, let $V_1, \ldots, V_k$ be the corresponding color classes. For each class $V_i = \{v_{j_1}, \ldots, v_{j_i}\}$ we complete the '?'-s in the vectors corresponding to its vertices as follows: Each '?' in one of the columns $j_1, \ldots, j_i$ is completed to 1, and all other 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 iff 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 since any two vectors corresponding to adjacent vertices must have a column with both '0' and '1' and, thus, represent two different haplotypes. $\square$

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 Sections 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 $\frac{1-p}{2}$, $A'_{ij} = 1$ with probability $\frac{1-p}{2}$ 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 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:

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

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

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. t^* is set to be the average of the expected dissimilarity between mates and the expected dissimilarity between independent vectors (see proof of Theorem 6). The algorithm produces a partition of the rows into mate classes of cardinalities $s_1 \geq s_2 \geq \ldots \geq s_l$. 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.

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) $H \leftarrow \{v\}$.
   (c) **For** every $v' \in V \setminus \{v\}$ **do**:
   If $d(v, v') < t^*$ **then** $H \leftarrow H \cup \{v'\}$.
   (d) $V \leftarrow V \setminus H$.
   (e) Insert(S,$|H|$).
4. Output S.

Fig. 1. An algorithm for scoring a block under a probabilistic model of the data. Procedure Insert(S,s) inserts a number s into a heap S.

Theorem 6. *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 which is 1 iff 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 Equation 1 we conclude:

$$(X|v' \parallel v) \sim Binom(m, f(1-p_I)) \ ,$$
$$(X|v' \approx v) \sim Binom(m, f(1-p_M)) \ .$$

We now require the following Chernoff bound (cf. [1]): If $Y \sim Binom(n,s)$ then for every $\epsilon > 0$ there exists $c_\epsilon > 0$ that depends only on ϵ, satisfying:

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

Let $\mu = mf(1-p_M)$. Define $\epsilon \equiv \frac{(1-p_I)-(1-p_M)}{2(1-P_M)}$ and $t^* \equiv \epsilon\mu$. Applying Chernoff bound we have that for all $c > 0$:

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

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

When using the algorithm as part of a practical heuristic (see Section 3), we do not report the rare haplotypes. Instead, we report only the smallest number of most abundant haplotypes as computed by the algorithm that together capture a fraction γ of all haplotypes. In applications in which the error rate α is not known, t^* cannot be directly computed. Instead, we calculate the ratio $\frac{d(v_1,v_2)}{fm}$ for any two row vectors, and keep these values in a sorted array: $d_1 \leq \ldots \leq d_{\binom{n}{2}}$. Next we find $(a,b) = \arg\max_{a=d_i, b=d_{i+1},\ 1 \leq i < \binom{n}{2}} [b-a]$. Then we set $t^* = \frac{a+b}{2}$. It can be shown that using this strategy the algorithm solves PMBS with high probability.

3 Minimum Block Haplotypes

Suppose that the matrix A contains haplotypes from several homogeneous populations. The partitioning into blocks can differ among populations [7]. Here, we study how to reconstruct the partitioning of the rows of A into sets called *sub-populations*, and the columns in each set into blocks, such that the sum of the scores of the submatrices corresponding to these blocks is minimized. Formally:

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

We usually know which populations the haplotypes came from, however, in certain situations, there may be a hidden stratification of the population, that can dramatically change the conclusions of association studies.

Given a partition of the rows, one can compute the score in the noiseless case using a simple adaptation of the dynamic programming algorithm of [20]. However, the general MBH problem is NP-hard.

Theorem 7. *MBH is NP-hard.*

For lack of space, the proof is omitted here. Interestingly, the problem can be solved in polynomial time if each sub-population is required to be a contiguous set of rows. This may be useful for designing heuristics that permute the matrix rows for local improvement.

We now 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 sub-population; and a simulated annealing algorithm to find an optimum partition into homogeneous sub-populations. We describe these components below.

The dynamic programming component computes the score for a given sub-population in a straightforward manner, similar to [20]. Let T_i, $0 \leq i \leq m$, be the minimum number of block haplotypes in the submatrix of A induced on the columns $1, \ldots, i$, where $T_0 = 0$. For a pair of columns i, j let B_{ij} be the score of the block induced by the row in S and the columns in $\{i, \ldots, j\}$. Then the following recursive formula can be used to compute T_m:

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

For scoring a block within the dynamic programming, we use the probabilistic algorithm described in Section 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 none of the vectors has '?', and setting $t^*_{v,v'} = \frac{l((1-p_M)+(1-P_I))}{2}$. Scoring an $n \times t$ block takes $O(tnk)$ time, where k is a bound on the number of common haplotypes. Hence, the dynamic programming 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 sub-populations. 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. 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. For each position i we check whether $d((u_i, \ldots, u_{i+w-1}), (v_i, \ldots, v_{i+w-1})) \leq w\alpha$. 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 sub-population should be more similar in blocks in which they share the same haplotypes and, thus, have better chance to hit good windows, and accumulate higher score in the scan. We next cluster the haplotypes based on their similarity values, using the K-means algorithm [15]. The resulting partition is taken to be the starting point for the process. To determine the number of sub-populations 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. Since an annealing step changes the haplotypes of two sub-populations only, it suffices to recompute the scores of these sub-populations only.

4 Experimental Results

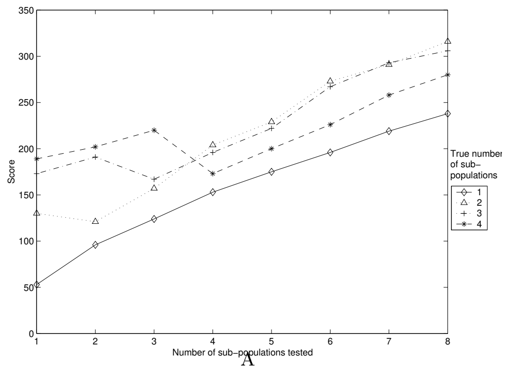
4.1 Simulations

We applied our algorithm to simulated and real haplotype data. First, we conducted extensive simulations to check the ability of our algorithm to detect sub-populations and recognize their block structure. Our simulation setup was as follows: Each simulated haplotype matrix contained 100 haplotypes and 300 SNPs. The number of sub-populations varied in the simulations. Sub-populations were of equal sizes. For each sub-population we generated block boundaries using a Poisson process with rate 20. Each block within a sub-population contained 2-5 common haplotypes covering 90% of the block's rows (with the rest 10% being rare haplotypes). Errors and missing data were introduced with varying rates up to 30%. The haplotype matrix was created according to the probabilistic model described in Section 2.4.

As a first test we simulated several matrices with 1-4 sub-populations and applied our algorithm with K ranging from 1 to 8. For each K we computed the score of the partition obtained, as described in Section 3. In each of the simulations the correct number got the lowest score (Figure 2.A). Next, we simulated several matrices with 3 sub-populations and different levels of errors and missing data. Figure 2.B summarizes our results in correctly assigning a haplotype to a sub-population (the set with the largest overlap with the true one was declared as correct). It can be seen that the MBH algorithm gives highly accurate results for missing data and error levels up to 10%.

For comparison, we also implemented the LD-based algorithm of Gabriel et al. [7] for finding blocks. We compared the block structures output by our algorithm and 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 [19, Sec. 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 3000 haplotypes, computed its block structure with both algorithms and compared them to the true one. We repeated this



A

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

B

Fig. 2. Simulation results. A: Determining the number of sub-populations. For each simulated matrix, containing 1-4 sub-populations, the figure shows the score assigned by the algorithm to partitions (y-axis) with different number of sub-populations (x-axis). Simulations were performed with 1% errors and no missing entries. B: Accuracy of haplotype classification by the MBH algorithm for different noise levels. Data are for 3 sub-populations.

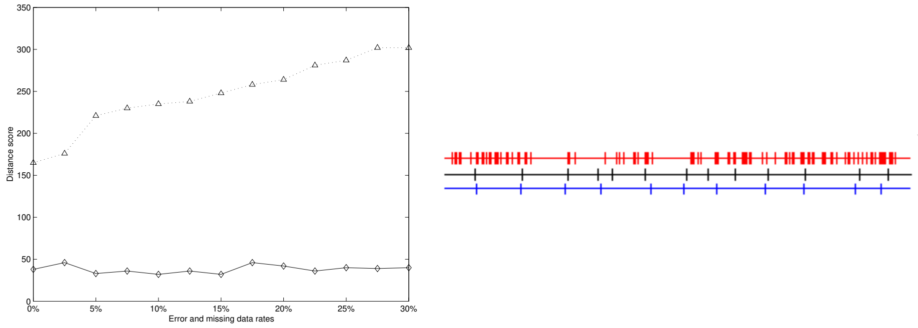
experiment with different error and missing data rates. The results are shown in Figure 3.A. It can be observed that our algorithm yields partitions that are closer to the true ones, particularly as the rate of errors and missing data rises. An example of the actual block structures produced is shown in Figure 3.B.

4.2 Real Data

We applied our algorithm to two published datasets. The first dataset of Daly et al. [5] 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 5% of rare haplotypes, i.e., in scoring a block we sought the minimum number of different haplotypes that together cover 95% of the rows.

In order to assess our block partitioning and compare it to the one reported by Daly et al. [5], 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. 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. is presented in Table 1. 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. the average fraction was



A B

Fig. 3. Block structure reconstruction. A: Accuracy in reconstruction by the PMBS algorithm (solid line) and the algorithm of Gabriel et al. [7] (dashed line). y-axis: the score of aligning the reconstructed structure with the correct one. x-axis: the noise rate. B: An example of the block structures produced for an error rate of 1% by our algorithm (bottom), the LD-based algorithm of [7] (top) and the true solution (middle). Each block boundary is denoted by a vertical line.

0.796. Another available partition for this data by Eskin et al. [6], was based on minimizing the number of representative SNPs. Their partition contained 11 blocks and its average fraction of high LD pairs was 0.814.

The second dataset we analyzed, due to Gabriel et al. [7], contains unresolved genotype data. In order to apply our algorithm to this data, we transformed it into haplotype data 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 (European) and D (individuals from Yoruba), consisting of 93 and 90 samples, respectively. Each population was genotyped in $\sim$60 different regions in the genome. We analyzed 6 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 2. 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. [20]. We devised several algorithms for different variants of the problem. In particular, we gave a simple algorithm, which, under an appropriate probabilistic

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 1. Comparison between the blocks of Daly et al. [5] and the blocks generated by our algorithm.

Chromosome: Region	#SNPs	Discovered blocks	% Correct classifications	Chromosome: Region	#SNPs	Discovered blocks	% Correct classifications
1: 3a	119	1: 1-35, 36-119 2: 1-46, 47-119	95	14: 41a	141	1: 1-48, 49-63, 64-141 2: 1-12, 13-63, 64-141	100
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

Table 2. Separation to populations and block finding on different regions in part of the data of [7], which includes populations A and D.

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 n/m of the data matrix is not too extreme: When the number of typed individuals n is very small and the number of SNPs 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 the LD-based method of Gabriel et al. [7]. 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. [5] we generated a slightly more concise block description than extant approaches, with a somewhat better fraction of high LD pairs.

We also treated the question of partitioning a set of haplotypes into sub-populations based on their different block structures, and devised a practical heuristic for the problem. On a genotype dataset of Gabriel et al. [7] we were able to identify sub-populations 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 sub-populations 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 sub-populations are independent. In practice, some boundaries may be common due to the common lineage of the sub-populations. A more detailed treatment of the block boundaries in sub-populations should be considered when additional haplotype data reveal the correct way to model this situation.

Acknowledgments

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

References

1. N. Alon and J. H. Spencer. *The Probabilistic Method.* John Wiley and Sons, Inc., 2000.
2. V. Bafna, B. V. Halldorsson, R. Schwartz, A. Clark, and S. Istrail. Haplotyles and informative SNP selection algorithms: Don't block out information. *Proc. of RECOMB*, pages 19–27, 2003.
3. A. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Molecular Biology and Evolution*, 7(2):111–22, 1990.
4. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Mass., 1990.
5. M.J. Daly et al. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29(2):229–232, 2001.
6. E. Eskin, E. Halperin, and R. M. Karp. Large scale reconstruction of haplotypes from genotype data. *Proc. of RECOMB*, pages 104–113, 2003.
7. S. B. Gabriel et al. The structure of haplotype blocks in the human genome. *Science*, 296:2225–2229, 2002.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, 1979.
9. L. Grugliyak and D. A. Nickerson. Variation is the spice of life. *Nature Genetics*, 27:234–236, 2001.
10. D. Gusfield. Inference of haplotypes in samples of diploid populations: Complexity and algorithms. *Journal of Computational Biology*, 8(3):305–323, 2001.

11. D. Gusfield. Haplotyping by pure parsimony. *Technical Report UCDavis CSE-2003-2, To appear in the Proceedings of the 2003 Combinatorial Pattern Matching Conference*, 2003.
12. B. V. Halldorsson et al. Combinatorial problems arising in SNP. *DMTCS '03 Conference.*
13. E. Hubbell. Finding a parsimony solution to haplotype phase is NP-hard. *Personal's communication.*
14. M. Koivisto et al. An MDL method for finding haplotype blocks and for estimating the strength of haplotype block boundaries. Proc. PSB 2003.
15. J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1965.
16. N. Patil et al. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294:1719–1723, 2001.
17. R. Sachidanandam et al. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature*, 291:1298–2302, 2001.
18. C. Venter et al. The sequence of the human genome. *Science*, 291:1304–51, 2001.
19. M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes.* Chapman and Hall, 1995.
20. K. Zhang, M. Deng, T. Chen, M.S. Waterman, and F. Sun. A dynamic programming algorithm for haplotype block partitioning. *Proc. Natl. Acad. Sci. USA*, 99(11):7335–9, 2002.