Tiling Binary Matrices in Haplotyping: Complexity, Models and Algorithms

Giuseppe Lancia[1], Romeo Rizzi[1], and Russell Schwartz[2]

[1] D.I.M.I., University of Udine
Via delle Scienze 206, 33100 Udine, Italy
{lancia}{rizzi}@dimi.uniud.it
[2] Department of Biological Sciences and Lane Center for Computational Biology, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh 15213, PA, USA
russells@andrew.cmu.edu

Abstract. A tiling of a matrix is an exact cover of its elements by a set of row fragments, called tiles. A particular variant of the tiling problem has arisen in the context of computational biology for studying genetic variations between individuals, in which one wishes to find the minimum-cardinality tiling of a matrix whose rows correspond to genomic sequences of a set of individuals. In this case, the tiles define a set of *haplotype motifs* strings of consecutive variants that frequently co-occur on a single chromosome. By minimizing the number of tiles needed to explain a data set, we seek to identify frequent haplotypes that will be more amenable to statistical analysis than the raw variation data. Although the haplotype motif model was first proposed several years ago, the complexity of the associated optimization problem has never been settled. Here, we show that the minimum tiling problem is NP-hard. We also describe ILP models and Dynamic Programming procedures for its exact solution.

Keywords: haplotyping, tiling, computational biology, computational complexity

1 Introduction

The most common form of genetic variation between genomes of different people is the single nucleotide polymorphism (SNP, pronounced "snip") at which a single DNA base takes on two common variants *alleles* in a population. While in rare cases more than two of the four DNA bases (`A`, `T`, `C` and `G`) are commonly found at a single genomic site, these are generally excluded from analysis. A chromosome of any single individual can then be modeled as a binary string of SNP alleles, in which the i^{th} bit is zero if that individual has the more common (major) allele at the i^{th} SNP locus and is one if that individual has the less common (minor) allele. Nearly 18 million such SNPs are now known in the human genome [22,15,9,16] and there is great interest in them for studies of human ancestry (cf., [23]) and as markers for statistical tests of association between genotype and phenotype (cf., [10]).

SNP alleles are not independent of one another but rather tend to be strongly correlated when nearby on the genome. These correlations occur as a side-effect of the way in which we inherent DNA from our parents. Humans are *diploid* organisms, meaning that most of our DNA is organized in pairs of chromosomes of which we inherent one copy from the mother and one from the father. The copy inherited from a single parent is not identical to either of that parent's copies, though, but rather is assembled through a process called *recombination* (or *crossing-over*) that leads to a concentenation of pieces of both of the parent's chromosome copies. For instance, if a parent's haplotypes are

Giuseppe Lancia, Romeo Rizzi, Russell Schwartz: Tiling Binary Matrices in Haplotyping: Complexity, Models and Algorithms, pp. 89–102 .
Proceedings of PSC 2010, Jan Holub and Jan Žďárek (Eds.), ISBN 978-80-01-04597-8

```
C C G A G A A C C A T G C G
a c c g g a t g g a a t c g
```

a haplotype obtainable by cross-over could be

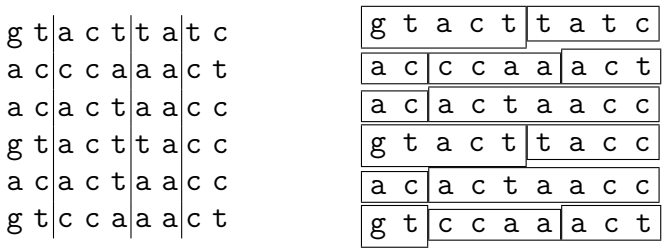
```
C C G A G a t g g a a G C G
```

As a result, when a new variation first appears in the genome through random mutation, that variation will remain strongly correlated with nearby SNPs for many generations, but will rapidly lose any detectable correlation with more physically distant SNPS. The result of this process is that when one examines patterns of variation across the human genome, one observes many strongly conserved sub-strings, called *haplotypes*, which are believed to represent sets of alleles that were found together in a subset of early human ancestors and which have largely escaped being broken up by recombination.

The haplotype structure of the genome is not merely an intellectual curiosity, but is the focus of intensive practical efforts to harness it for use in genetic association studies. Association studies, in which one seeks SNPs statistically associated with some phenotype (e.g., a disease), are hampered by the fact that the large number of SNPs means that corrections for multiple hypothesis obscure all but the strongest associations. It is hoped that testing for association with haplotypes instead of genotypes [6,21,1] will mitigate this problem by allowing one to identify any real associations in the data while performing many fewer tests. Several major studies are underway to examine haplotypes across human populations [25,26,11,12] for this purpose. These haplotypes also provide important information for applications in inferring ancestry and population substructure in human populations.

Attempts to analyze and use these data have led to several different approaches to mathematically model haplotype structure in ways that will be amenable to model inference and application in association study design and other contexts. At one extreme are "haplotype block" models [8], which assume that the genome can be decomposed into short regions (blocks) and that all haplotypes are broken at the boundaries of these blocks. The block model makes the simplifying assumption that cross-over has repeatedly occurred at the same boundaries. Block models are amenable to efficient computational inference [28] but at the expense of obscuring some information on correlations across block boundaries. While there are many roughly similar ways of optimizing for block boundaries (c.f., [20,28,27,13]), they appear to give relatively poor agreement between measures, population groups, or even sub-samples of a single population [20]. At the opposite extreme are models allowing for haplotypes to be broken arbitrarily within any given individual chromosome, effectively treating the genome as a Markov model in which each chromosome represents a unique path between a set of ancestral chromsomes [19,7]. These general models can more accurately capture true haplotype structure, but at the expense of being much more difficult to learn reliably and to apply to subsequent optimizations. A compromise between these two extremes is the *motif model* [17] (or the independently developed *dictionary model* [2]), which models each chromosome as a concentenation of a set of conserved DNA segments, called *motifs* or *tiles* but without the assumption that block model assumption that boundaries between conserved segments are shared across the population. Several studies have shown that these models are more effective than raw SNPs or block-based haplotypes for association testing [5,3] and for several associated optimization problems [18].

The following are examples of block decomposition (left) and motif decomposition (right) of six haplotypes:



A motif model, like a block model, can nonetheless be defined in many different ways depending on the criteria for which one optimizes the fit of motifs to observed haplotype data. The motif model was originally implemented by Schwartz using a heuristic method approximately optimizing for a likelihood model [17], and has since been studied using other probabilistic models [2] and minimum description length (MDL) models [14,24]. An obvious metric, with some practical motivation for the multiple hypothesis testing problem in association testing, is parsimony [17]: minimizing the total number of tiles needed to explain a data set. For instance, in the above example on the right, the given explanation consists of 8 tiles. Note that this is not the minimum tiling. In fact, a trivial tiling in which each row is a tile by itself has value 6 (if, on the other hand, we introduce restrictions on the minimum and/or maximum length allowed for a tile, the trivial tiling may not be feasible). The parsimony metric has, however, only been used heuristically [5]. There has been neither any efficient algorithm for this problem or any proof of its hardness.

In this paper we prove that this problem is APX-hard. We then proceed to describe an ILP formulation which can be used for its exact solution. The formulation has an exponential number of variables, but its LP relaxation can be solved in polynomial time by column-generation techniques. We also describe an alternative, but equivalent, polynomial-size ILP formulation based on a reduction to a multicommodity flow problem. We finally give an exact, dynamic programming, polynomial algorithm for the parsimony version of the block model problem (i.e., find a decomposition of the matrix in blocks so that the total number of tiles that the decomposition defines is minimum).

2 The problem

We are given a binary $m \times n$ matrix M. A *tile* t is specified by a first starting column $f(t)$, an ending column $e(t)$, and a string $s(t)$ of length $e(t) - f(t) + 1$. A tile t *applies to* (or is compatible with) any row M_i of M such that

$$s(t) = M[i, f(t)] \cdots M[i, e(t)]\,. \tag{1}$$

For any row M_i, let us denote by $\mathcal{T}(i)$ the set of tiles that apply to it. Furthermore, let $\mathcal{T} := \cup_i \mathcal{T}(i)$ the set of tiles which apply to some row of M. Notice that each triple r, c_f, c_e, with $1 \leq r \leq m$ and $1 \leq c_f \leq c_e \leq n$ identifies a unique tile in $\mathcal{T}$, namely the tile t for which $f(t) = c_f$, $e(t) = c_e$ and $s(t) = M[r, c_f] \cdots M[r, c_e]$. We denote this tile as $< r, c_f, c_e >$. Notice that it is possible for different triples $< r, c_f, c_e >$ to identify the same tile in $\mathcal{T}$, as long as they refer to different rows. (For example, $< 1, 1, 3 >$ and $< 4, 1, 3 >$ identify the same tile in the binary matrix M displayed in Fig. 1). Furthermore, $\mathcal{T}(r) = \{< r, c_f, c_e > \mid 1 \leq c_f \leq c_e \leq n\}$.

We say that a set $\hat{T}$ of tiles *covers* a row M_i of M if there exists a subset $T_i = \{t_1, \ldots, t_k\} \subseteq \hat{T}$, with $f(t_1) = 1$, $f(t_i) = e(t_{i-1}) + 1$ for $i = 2, \ldots, k$, and $e(t_k) = n$, such that

$$M_i = s(t_1) \cdot s(t_2) \cdots s(t_k)\,. \tag{2}$$

A *tiling* of M is a set $\hat{T}$ of tiles which covers every row of M.

In this paper we study the problem Tile where, given a binary matrix M as input, we seek for a tiling $\hat{T}$ of M with the minimum possible number of tiles. In Figure 1 we show an example of three different tilings for a same binary matrix.

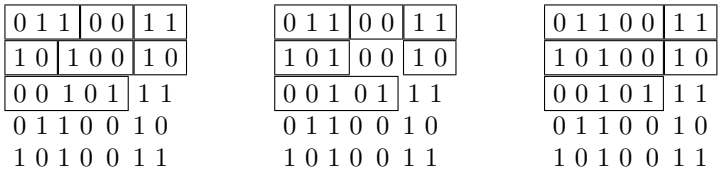


Figure 1. Three different tilings of a same binary matrix: one of size 7 (on the left), one of size 6 (in the middle), and an optimal tiling of size 5 (on the right).

Notice that, for any $m \times n$ input binary matrix M, the optimal value for problem Tile satisfies $OPT \leq \min\{m, 2n\}$, as both the whole rows of M and the single entries of M are considered as valid tiles.

3 Problem complexity

In this section we prove that the problem Tile is APX-hard.

The proof is split in two parts. First, in Subsection 3.1, we prove that Tile is APX-hard when matrices over general alphabets are considered. Next, in Subsection 3.2, we show that allowing non-binary matrices does not significantly affect the approximability of problem Tile.

We close this section by listing a few elementary facts which are useful both in establishing the reductions here proposed and also as a first aid in algorithmically managing the problem.

Fact 1. *If two rows are identical then we can remove one of them.*

Fact 2. *When the matrix is binary, then flipping the values in one column does not change the problem.*

Fact 3. *If two consecutive columns are identical (possibly after inversion of one of the two, in case of binary matrices) then we can remove one of them.*

Fact 4. *The problem can be solved in poly-time when the number of rows or the number of columns is bounded by a constant.*

3.1 APX-hardness of Tile in the general non-binary case

In this subsection, we prove that Tile is APX-hard for general non-binary matrices. This lemma is at the core of the result given in the next subsection (the APX-hardness of Tile for binary matrices) and is obtained by reducing Node-Cover

on cubic graphs to TILE. Explicit values of $\varepsilon > 0$ such that NODE-COVER on cubic graphs admits no $(1+\epsilon)$-approximation algorithm unless P=NP are given in [4].

Assume therefore to be given a cubic graph $G = (V, E)$ as instance of NODE-COVER. Let $m := |E|$ and $n := |V|$[1] Clearly, $m = \frac{3}{2}n$ since G is cubic. We assume the nodes in V to be labeled with the first naturals $0, 1, 2, \ldots, n-1$. In other words, $V = \mathbb{N}_n$. We can hence speak of the *small endnode* $s(e)$ and of the *big endnode* $b(e)$ for each edge $e \in E$. When we say that $e = uv$ is an edge in E we are implicitly assuming that $u < v$, that is, $u = s(e)$ and $v = b(e)$. We let $E = \{e_0, e_1, \ldots, e_{m-1}\}$.

We construct a matrix M with $\hat{n} := 3m$ columns and $\hat{m} := 4m + n + 2\hat{n}$ rows. The rows and columns of M are numbered starting from 0. All the entries of M are symbols from the alphabet $\Sigma := \{A, B, X, \sigma_1, \sigma_2\}$. For each $i, j \in \mathbb{N}_{\hat{n}}$ with $i \neq j$, let $M[4m+n+i, i] = M[4m+n+\hat{n}+i, j] = A$ and $M[4m+n+i, j] = M[4m+n+\hat{n}+i, i] = B$. That is, the last (second last) $\hat{n}$ rows of M are obtained from an $\hat{n} \times \hat{n}$ identity matrix by replacing each 0 with an A (respectively, with a B) and each 1 with a B (respectively, with an A). For each $i \in \mathbb{N}_n$, row i is associated to the node i of G and, for each $c \in \mathbb{N}_m$ and $t = 0, 1, 2$, the value of $M[i, 3c+t]$ is defined as follows.

$$M[i, 3c+t] = \begin{cases} \sigma_1 \text{ if } t = 0 \text{ and } i = s(e_c), \\ \sigma_1 \text{ if } t = 2 \text{ and } i = b(e_c), \\ A \text{ otherwise.} \end{cases}$$

Finally, for each $j \in \mathbb{N}_m$, rows $n+4j, n+4j+1, n+4j+2, n+4j+3$ are associated to the edge e_j of G and, for each $c \in \mathbb{N}_m$, $t = 0, 1, 2$ and $k = 0, 1, 2, 3$, the value of $M[n+4j+k, 3c+t)]$ is defined as follows.

$$M[n+4j+k, 3c+t)] = \begin{cases} X \text{ if } j = c \text{ and } t = 1, \\ \sigma_1 \text{ if } j = c \text{ and } (k,t) \in \{(0,0), (0,2), (1,0), (2,2)\}, \\ \sigma_2 \text{ if } j = c \text{ and } (k,t) \in \{(1,2), (2,0), (3,0), (3,2)\}, \\ A \text{ otherwise.} \end{cases}$$

Lemma 5. *Let X be a node cover of G. Then there exists a valid tiling T for M such that $|T| = 2\hat{n} + 4m + |X|$.*

Proof. Remember that each tile t is specified by a triple $(f(t), e(t), s(t))$ where $f(t)$ is the index of the first column, $e(t)$ is the index of the last column, and $s \in \Sigma^{e(t)-f(t)+1}$. We construct T in three phases. First, we place in T all the $\hat{n}$ tiles in the set $A_T := \{(i, i, A) : i \in \mathbb{N}_{\hat{n}}\}$ and all the $\hat{n}$ tiles in the set $B_T := \{(i, i, B) : i \in \mathbb{N}_{\hat{n}}\}$. Notice that the last $2\hat{n}$ rows of M are already covered by the tiles in $A_T \cup B_T$. Next, for each $e_j = uv \in E$, with $u < v$, we have two possible cases. If $u \in X$, then we add to T the 4 tiles $(3j, 3j+1, \sigma_1 X)$, $(3j, 3j+1, \sigma_2 X)$, $(3j+2, 3j+2, \sigma_1)$, $(3j+2, 3j+2, \sigma_2)$. Otherwise, if $u \notin X$, then we add to T the 4 tiles $(3j, 3j, \sigma_1)$, $(3j, 3j, \sigma_2)$, $(3j+1, 3j+2, X\sigma_1)$, $(3j+1, 3j+2, X\sigma_2)$. Notice here that, in either case, these 4 tiles plus the tiles in A_T suffice in covering the four rows $n+4j, n+4j+1, n+4j+2, n+4j+3$. Finally, for each $i \in X$, the whole row i of matrix M is placed as a tile in T. Notice that $|T| = 2\hat{n} + 4m + |X|$. It remains to check that, for each $i \in \mathbb{N}_n$, the i-th row of M is also covered by T. To see this, we distinguish between two cases: If $i \in X$, then row i appears in T as a tile. Otherwise, if $i \notin X$ and since X is a node cover of G,

[1] Notice that, within this section, m and n do not denote the number of rows and columns of the tiling matrix, but the number of edges and vertices of G.

then, for every edge $e_j = iu \in E$ (respectively, for every edge $e_j = ui \in E$) we have that $u \in X$ and hence the tile $(3j, 3j, \sigma_1)$ has been placed in T (respectively, the tile $(3j+2, 3j+2, \sigma_1)$ has been placed in T). Notice that row i is covered by these tiles (with $e_j \ni i$) plus the tiles in A_T. □

Lemma 6. *Let T be a feasible tiling for M. Then G admits a node cover X with $|X| = |T| - 2\hat{n} - 4m$.*

Proof. As in the proof of Lemma 5, each tile t is specified by a triple $(f(t), e(t), s(t))$, and $A_T := \{(i, i, A) : i \in \mathbb{N}_{\hat{n}}\}$, and $B_T := \{(i, i, B) : i \in \mathbb{N}_{\hat{n}}\}$. Let M_H (respectively, M_L) be the matrix comprising the first $4m + n$ (respectively, the last $2\hat{n}$) rows of matrix M. In other words, $M = \begin{pmatrix} M_H \\ M_L \end{pmatrix}$. For $i \in \{H, L\}$, let T_i be the set of tiles in T which are compatible with some row in M_i. Notice that there can be some tile $t \in T$ which belongs both to T_H and to T_L. However, for any such tile t, we have both that $s(t) \in \{A, X, \sigma_1, \sigma_2\}^*$, since $t \in T_H$, and that $s(t) \in \{A, B\}^*$, since $t \in T_L$. Indeed, all entries of M_H are in $\{A, X, \sigma_1, \sigma_2\}$ and all entries of M_L are in $\{A, B\}$. It follows that $s(t) \in \{A\}^*$ for each $t \in T_H \cap T_L$. Notice also that $A_T \cup B_T$ would be a tiling for M_L with $|A_T \cup B_T| = 2\hat{n}$. At the same time, $|T_L| \geq 2\hat{n}$ by Lemma 7 here below. These facts imply that we can always assume that $A_T \cup B_T \subseteq T$. Indeed, a tile $t \in T$ with $s(t) \in \{A\}^*$ which might possibly help in covering some rows of M_H can always be substituted with tiles in A_T.

Consider now the four rows $n+4j, n+4j+1, n+4j+2, n+4j+3$ associated with a generic edge e_j, $j \in \mathbb{N}_m$. Let $T(e_j)$ be the set of the tiles in $T \setminus A_T$ which are compatible with some of the above four rows. Notice that $|T(e_j)| \geq 4$. Notice furthermore that $T(e_p) \cap T(e_q) = \emptyset$ whenever $e_p, e_q \in E$ with $e_p \neq e_q$. Let $J_T := \{e_j : |T(e_j)| \geq 5\}$. We call a tiling T of M *standard* if $J_T = \emptyset$. We now show how to produce a standard tiling $\tilde{T}$ with $|\tilde{T}| \leq |T|$. To do so, it suffices to show how to obtain a tiling T' of M with $|T'| \leq |T|$ and such that $J_{T'}$ is strictly contained in J_T, whenever $J_T \neq \emptyset$. Indeed, where $|T(e_j)| \geq 5$, then let T' be obtained from T by removing all tiles in $T(e_j)$ and by adding the 5 tiles $t_1 = (3j, 3j+1, \sigma_1 X)$, $t_2 = (3j, 3j+1, \sigma_2 X)$, $t_3 = (3j+2, 3j+2, \sigma_1)$, $t_4 = (3j+2, 3j+2, \sigma_2)$ and $t_5 = (0, 3\hat{n} - 1, M_i)$ where $i = s(e_j)$ and M_i is the i-th row of M, i.e. the row of M associated with node i. Notice that T' is indeed a tiling of M, and indeed $|T'| \leq |T|$. Moreover, $T'(e_j) = \{t_1, t_2, t_3, t_4\}$, whence $J_{T'} \subset J_T$.

We hence assume T is standard, that is, $|T(e_j)| = 4$ for each $e_j \in E$. Since $A_T \subset T$, we can actually assume that either $T(e_j)$ comprises precisely the 4 tiles $(3j, 3j+1, \sigma_1 X)$, $(3j, 3j+1, \sigma_2 X)$, $(3j+2, 3j+2, \sigma_1)$, and $(3j+2, 3j+2, \sigma_2)$, or $T(e_j)$ comprises precisely the 4 tiles $(3j, 3j, \sigma_1)$, $(3j, 3j, \sigma_2)$, $(3j+1, 3j+2, X\sigma_1)$, and $(3j+1, 3j+2, X\sigma_2)$. For $i \in \mathbb{N}_n$, let $T(i)$ be the set of those tiles in $T \setminus A_T \setminus \cup_{e_j \in E} T(e_j)$ which are compatible with the i-th row of M. Notice that $T(i_1) \cap T(i_2) = \emptyset$ for each $i_1 \neq i_2$ with $i_1, i_2 \in \mathbb{N}_n$. Let now X be the set of those $i \in \mathbb{N}_n$ such that $T(i) \neq \emptyset$. Notice that X is a node cover of G. Finally, $|X| \leq |T| - 2\hat{n} - 4m$ is a consequence of the fact that the $T(e_j)$'s are disjoint sets of tiles and the $T(i)$'s are disjoint sets of tiles. □

We say that a set of tiles T *weakly covers* a matrix M if for every entry $M[i, j]$ of M there exists a tile t in T that is compatible with row i of M and such that $f(t) \leq j \leq e(t)$.

Lemma 7. *Let $M_{A/B}$ (respectively, $M_{B/A}$) be the $\hat{n} \times \hat{n}$ matrix whose entries are all B (respectively, all A) except for the entries on the diagonal which are all A*

(respectively, all B). Let $M_L = \begin{pmatrix} M_{B/A} \\ M_{A/B} \end{pmatrix}$ *and let* T_L *be a set of tiles which weakly covers* M_L*. Then* $|T_L| \geq 2\hat{n}$*.*

Proof. We prove a stronger claim: Let $\Sigma = \{A, B, C\}$. Let $N_{A/B}$ (respectively, $N_{B/A}$) be the $\hat{n} \times \hat{n}$ matrix obtained from matrix $M_{A/B}$ (respectively, $M_{B/A}$) by replacing with C all the entries below the main diagonal. Let $T_C := \{(i, i+1, C) : i \in \mathbb{N}_{\hat{n}}\}$. Let $N = \begin{pmatrix} N_{B/A} \\ N_{A/B} \end{pmatrix}$ and let T be a set of tiles such that $T \cup T_C$ weakly covers N. Then $|T| \geq 2\hat{n}$.

We prove this by induction on $\hat{n}$. Among the minimum-cardinality sets of tiles T such that $T \cup T_C$ weakly covers N, let T^* be one minimizing $\sum_{t \in T} |s(t)|$. Notice that $(\hat{n}-1, \hat{n}-1, B)$ and $(\hat{n}-1, \hat{n}-1, A)$ both belong to T^*. Indeed, where t is the tile in T compatible with the row $\hat{n}-1$ of N and with $e(t) = \hat{n}-1$, then $t = (\hat{n}-1-i, \hat{n}-1, C^i B)$ for some $i \in \mathbb{N}_{\hat{n}}\}$. Notice however that any tile t of this form can always be substituted by tile $(\hat{n}-1, \hat{n}-1, B)$, plus some tiles in T_C. The same argument also shows that $(\hat{n}-1, \hat{n}-1, A) \in T^*$. Notice now that $(T^* \setminus \{(\hat{n}-1, \hat{n}-1, B), (\hat{n}-1, \hat{n}-1, A)\}) \cup T_C$ weakly covers N', the matrix obtained from N by dropping the last column and by dropping the rows $\hat{n}-1$ and $2\hat{n}-1$. Notice that matrix N' is of the same form as matrix N, but with $\hat{n}' = \hat{n}-1$. Therefore, by induction, $|T^*| \geq 2 + 2(\hat{n}-1) = 2\hat{n}$. □

Theorem 8. *When we allow for general, possibly non-binary matrices, then the* Tile *problem is APX-hard.*

Proof. We proceed as follows: We assume to be given a $(1+\epsilon)$-approximation algorithm A for Tile and design a $(1+31\,\epsilon)$-approximation algorithm for Node-Cover which rests on algorithm A as a subroutine. The APX-harness of Tile then follows from the APX-harness of Node-Cover.

After receiving in input a cubic graph G, we construct the matrix M as described above. Assume the minimum node cover of G has size *opt*. Clearly, $opt \geq \frac{m}{3}$ since G is cubic. Moreover, by Lemma 5, there exists a tiling T_{opt} covering M with $|T_{opt}| = 2\hat{n} + 4m + opt = 10m + opt$. By running the $(1+\epsilon)$-approximation algorithm for Tile we are hence guaranteed to find a solution T_{apx} with $|T_{apx}| \leq (10\,m + opt)(1+\epsilon)$. And Lemma 6 (whose proof can be easily converted into a poly-time algorithm) shows how, starting from this tiling T_{apx}, one can obtain a node cover X of G of size at most

$$|X| \leq (10\,m + opt)(1+\epsilon) - 10\,m \leq 10\,\epsilon\,m + opt + \epsilon\,opt \leq 30\,\epsilon\,opt + opt + \epsilon\,opt$$
$$\leq (1 + 31\,\epsilon)opt.$$

□

3.2 The power of the binary case

In this subsection, we show that allowing non-binary matrices does not affect the approximability of the problem Tile. Formally stated, we prove the following result.

Lemma 9. *There exists an objective function preserving reduction from the* Tile *problem on general non-binary matrices to the* Tile *problem on binary matrices.*

Notice that, combining Lemma 9 here above with Theorem 8 from the previous section, we obtain the following result.

Theorem 10. *Even when restricted to binary input matrices, problem* Tile *is APX-hard.*

Assume the entries of the input matrix M are taken from the alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, where $k := |\Sigma|$. We can clearly assume that $k \leq m \cdot n$, where m and n are the number of rows and columns of matrix M. The objective function preserving reduction we are going to propose can be conveniently described as the composition of two objective function preserving transformations to be applied in series. First, an $m \times nm$ matrix M_σ over Σ is obtained from M by echoing each single column of M precisely m times. Notice that, by Fact 3, this does not affect the objective function value. Next and last, M_b is the $m \times nmk$ binary matrix obtained from M_σ by replacing each entry σ_i of M_σ with a row vector of i zero's followed by a row vector of $k - i$ one's. So, where M had m rows and n columns, both numbered starting from 0, then M_b has m rows and nmk columns, and

$$M_b[i,j] = \begin{cases} 0 \text{ if } M[i, j \,.\mathrm{div}.\, km] = \sigma_p \text{ and } p > j \,.\mathrm{mod}.\, k, \\ 1 \text{ otherwise.} \end{cases}$$

It should be clear that a tiling of M directly translates into a tiling of M_b involving the same number of tiles. Indeed, a feasible tiling for M gets converted into a feasible tiling for M_b if all tiles get stretched by a factor of mk. In the tiling of M_b obtained in this way, all tiles t have $f(t)$ which is a multiple of mk and $e(t) \equiv_k k - 1$. We call such a tiling of M_b *standard*.

Conversely, it is also clear that to any standard tiling of M_b corresponds a tiling of M involving the same number of tiles. Therefore, in order to prove Lemma 9, we only need to prove the following lemma.

Lemma 11. *Given any tiling T of M_b, we can produce in poly-time a standard tiling T' of M_b with $|T'| \leq |T|$.*

Proof. Clearly, since M_b has m rows, there is no difficulty in producing a standard tiling of M_b of size m. We can therefore assume that $|T| < m$. We also assume that T is a minimal tiling of M_b, that is, $T \setminus \{t\}$ is also a feasible tiling of M_b for no $t \in T$. Fix attention on any $c_1 = 0, 1, 2, \ldots, n-1$. Since, $|\{f(t) \,.\mathrm{div}.\, k : t \in T\}| \leq |T| < m$, then there exists a $c_2 = c_2(c_1) \in \{0, 1, 2, \ldots, m-1\}$ such that $f(t) \,.\mathrm{div}.\, k \neq c_1 m + c_2$ holds for every $t \in T$. This means that no tile starts within the k column positions of M_b corresponding to position $c_1 m + c_2$ of M_σ. More formally, for no $t \in T$ we have that $(c_1 m + c_2)k \leq f(t) < (c_1 m + c_2 + 1)k$. From this, and by the minimality of T, it also follows that for no $t \in T$ we have that $(c_1 m + c_2)k - 1 \leq e(t) < (c_1 m + c_2 + 1)k - 1$. Based on these facts, we can massage the tiles in T as follows.

1. **left extension** Let t be any tile in T with $f(t) \leq (c_1 m + c_2)k \leq e(t)$. If $f(t) > c_1 mk$, then t is replaced with a tile t' with $f(t') = c_1 mk$, $e(t') = e(t)$, $t'[p] = t[p]$ for each $p = f(t), f(t)+1, \ldots, e(t)$, and $t'[p] = t[(c_1 m + c_2)k + (p \,.\mathrm{mod}.\, k)]$ for each $p < f(t)$ with $p \geq f(t')$.
2. **right extension** Let t be any tile in T with $f(t) \leq (c_1 m + c_2)k \leq e(t)$. If $e(t) < (c_1 + 1)mk - 1$, then t is replaced with a tile t' with $f(t') = f(t)$, $e(t') = (c_1 + 1)mk - 1$, $t'[p] = t[p]$ for each $p = f(t), f(t) + 1, \ldots, e(t)$, and $t'[p] = t[(c_1 m + c_2)k + (p \,.\mathrm{mod}.\, k)]$ for each $p > e(t)$ with $p \leq e(t')$.

3. **right trim** Let t be any tile in T with $c_1mk \leq e(t) < (c_1m + c_2)k$. Then t is replaced with a tile t' with $f(t') = f(t)$, $e(t') = c_1mk - 1$, and $t'[p] = t[p]$ for each $p = f(t'), f(t) + 1, \ldots, e(t')$.
4. **left trim** Let t be any tile in T with $(c_1m + c_2 + 1)k \leq f(t) < (c_1 + 1)mk$. Then t is replaced with a tile t' with $e(t') = e(t)$, $f(t') = (c_1 + 1)mk$, and $t'[p] = t[p]$ for each $p = f(t'), f(t) + 1, \ldots, e(t')$.

Clearly, no one of the above four operations can possibly increase $|T|$. Furthermore, it can be checked that if a row r is covered by some sequence of tiles in T, then, after each one of the above 4 operations has been performed, row r can still be covered by a suitable sequence of tiles. Indeed, the new sequence of tiles can be obtained from the original one by performing the following operations:

(1) discard all tiles t with $f(t) \geq c_1mk$ and $e(t) < (c_1m + c_2)k$;

(2) discard all tiles t with $f(t)$ with $e(t) < (c_1 + 1)mk$ and $f(t) \geq (c_1m + c_2)k$;

(3) retain all other tiles; on each one of these remaining tiles, apply each one of the above four operations.

Notice that, after each one of the four operations above has been performed, no tile t can have $c_1mk < f(t) < (c_1 + 1)mk$ or $c_1mk \leq e(t) < (c_1 + 1)mk - 1$. Furthermore, if $f(t)$ (respectively, $e(t)$) has been affected by the above operations, then, after the operations have taken place, $f(t)$ (respectively, $e(t)+1$) is a multiple of mk. It follows that after the above 4 steps have been executed for each $c_1 = 0, 1, 2, \ldots, n$, and for the corresponding $c_2 = c_2(c_1)$, then T has become standard. □

4 ILP formulations and DP

Exponential formulation

As defined in Section 2, let m be the number of rows and n be the number of columns of the input matrix M for which we seek an optimal tiling.

In our first ILP formulation, we introduce a binary variable x_t for each possible tile $t \in \mathcal{T}$, and further, for every $i = 1, 2, \ldots, m$, we introduce a binary variable y_T for each minimal set of tiles which covers row i. For the porpouse of notation, we denote by $\mathcal{M}(i)$ the family of the minimal sets of tiles which cover row i. Notice that a set of tiles belongs to $\mathcal{M}(i)$ if and only if is contained in $\mathcal{T}(i)$ and has the form $\{t_1, \ldots, t_k\}$ with $f(t_1) = 1$, $f(t_i) = e(t_{i-1}) + 1$ for $i = 2, \ldots, k$, and $e(t_k) = n$.

We have

$$\min \sum_{t \in \mathcal{T}} x_t \tag{3}$$

$$\sum_{T \in \mathcal{M}(i)} y_T = 1 \qquad \forall i = 1, \ldots, m \tag{4}$$

$$\sum_{T \in \mathcal{M}(i) \,|\, t \in T} y_T \leq x_t \qquad \forall i = 1, \ldots, m \quad \forall t \in \mathcal{T}(i) \tag{5}$$

$$x_t, y_T \in \{0, 1\} \qquad \forall t \in \mathcal{T}, T \subseteq \cup_i \mathcal{M}(i) \tag{6}$$

Note that, for each i, it is $|\mathcal{T}(i)| = n(n+1)/2$ (we have to decide the first starting and the ending column), while $|\mathcal{M}(i)| = 2^{n-1}$ (we have to decide which of the first $n - 1$ columns are ending columns).

Thus, the above model has an exponential number of y variables. However, the LP relaxation can still be solved in polynomial time provided we can show how to solve the *pricing* problem for the y variables in polynomial time. The resulting approach is called column generation. The idea is to have all x variables in the model, and only a subset of the y variables. Then, given an optimal solution to the current LP, we see if there is any missing y variable that should be priced-in (i.e., added to the current variables).

Let $\gamma_1, \ldots, \gamma_m$ be the dual variables associated with constraints (4) and let λ_t^i, for $i = 1, \ldots, m$ and $t \in \mathcal{T}(i)$, be the dual variables associated with constraints (5).

To each primal variable y_T corresponds an inequality in the dual LP. The variable has negative reduced cost if and only if the corresponding dual constraints is violated by the current optimal dual solution. Assume T is a set in $\mathcal{M}(i)$. Then, the corresponding dual inequality for T is

$$\gamma_i - \sum_{t \in T} \lambda_t^i \leq 0 \tag{7}$$

If we consider λ_t^i to be the cost of tile t (relatively to a particular row i), and define $\lambda^i(T) := \sum_{t \in T} \lambda_t^i$, we have that the dual inequalities, for all $T \in \mathcal{M}(i)$, are of type

$$\lambda^i(T) \geq \gamma_i \tag{8}$$

A set of tiles violates the dual inequality if $\lambda^i(T) < \gamma_i$. If this happens, y_T should be added to the current set of primal variables. To identify a set which violates the dual inequality, it is enough to find the *smallest-cost* set. If $T^* \in \mathcal{M}(i)$ is such that $\lambda^i(T^*) = \min_{T \in \mathcal{M}(i)} \lambda^i(T)$, then, if $\lambda^i(T^*) < \gamma_i$ then y_{T^*} should be added to the LP variables, otherwise, no y_T variables, with $T \in \mathcal{M}(i)$ should be added to the LP. We should repeat this reasoning for all $i = 1, \ldots, m$.

Let us consider then the following problem:

– given i and costs λ_t^i, for $t \in \mathcal{T}(i)$, find T^* in $\mathcal{M}(i)$ with minimum λ-cost.

For each $1 \leq u \leq v \leq n$, denote in short $\Lambda(u, v)$ the value λ_t^i, where $t =< i, u, v >$.

We consider the following dynamic program. Denote by $V(r)$ the optimal (minimum) λ-cost of fragmenting row i in consecutive tiles, up to position r. We are interested in $V(n)$. We have the recurrence:

$$V(r) = \min_{p=\max\{1, r-l_M+1\}}^{r-l_m+1} (V(p-1) + \Lambda(p, r)) \tag{9}$$

where l_m is the minimum possible length of a tile, and l_M is the maximum possible length of a tile. We have $\lambda^i(T^*) = V(n)$. Base case is $V(j) = 0$, for $j \leq 0$, and $V(j) = +\infty$ for $0 < j < l_m$.

Polynomial-size formulation

Here we consider an alternative ILP formulation, which in fact yields the same bound as the previous one. The idea is to formulate the problem as a multicommodity flow problem. In principle, imagine to have a directed graph $G = (V, A)$, in which the vertices V are given by the column indexes, augmented with a dummy node $n + 1$, i.e., $V = 1, \ldots, n + 1$. The arcs are associated to the tiles. There is a directed arc for

each tile t. Assume $t =< i, u, v >$. Then, there is an arc $a_t = (u, v+1)$. Note that there can be parallel arcs, and, for each $1 \leq u \leq v \leq n+1$ there is at least one arc from u to v.

Now, for each commodity $i = 1, \ldots, m$, we want to send a unit of flow out of node 1, through the network as far as it can go (i.e., until it reaches node $n+1$). Each time an arc a_t is used by the flow f_i, for commodity i, it means that the tile t is used in the solution to cover row i.

We can associate flow variables to the arcs, and have flow conservation constraints. Furthermore, the activation variables for the tiles (i.e., the x_t variables) provide capacities for each arc (i.e., x_t is the capacity of the arc a_t).

Instead of actually building the network, we now describe a formulation that achieves the exact same purpose, and "builds" the network only implicitly.

As before, there are variables x_t for each tile $t \in \mathcal{T}$. Furthermore, for each row i and indices $1 \leq a \leq b \leq n$, we have a variable z^i_{ab}. This variable represents the i-th flow along the arc $a_{<i,a,b>}$ (i.e., one of the parallel arcs between a and $b+1$) in G.

We get the following formulation:

$$\min \sum_{t \in \mathcal{T}} x_t \tag{10}$$

$$\sum_{r=1,\ldots,n} z^i_{1r} = 1 \qquad \forall i = 1, \ldots, m \tag{11}$$

$$\sum_{1 \leq j < r} z^i_{jr} = \sum_{r < j \leq n} z^i_{rj} \qquad \forall i = 1, \ldots, m \quad \forall 1 < r < n \tag{12}$$

$$z^i_{ab} \leq x_{<i,a,b>} \qquad \forall i = 1, \ldots, m \quad \forall 1 \leq a \leq b \leq n \tag{13}$$

$$x_{<i,a,b>}, z^i_{ab} \in \{0, 1\} \qquad \forall 1 \leq i \leq m,\ 1 \leq a \leq b \leq n \tag{14}$$

Constraints (12) are flow conservation, saying that in each non-final column, the unit of flow coming in must also go out. Constraints (13) put capacities on the arcs, saying that an arc corresponding to a tile not activated ($x_t = 0$) cannot be used by the flows. Note that the z variables could be just declared real, as, when in a feasible solution the x are integer, there is always a way to make the z integer as well.

This model has $m \times n \times (n+1)/2$ z-variables ($\simeq mn^2/2$) and $|\mathcal{T}|$ x-variables. As for the constraints, there are m flow-out constraints, $m \times (n-2)$ conservation constraints and $m \times n(n+1)/2$ capacity constraints, for a total of $\simeq mn^2/2$ constraints.

Theorem 12. *For each solution (x, y) of the exponential formulation there is a solution (x, z) of the polynomial formulation, which achieves the exact same value, and vice versa.*

Proof. (Sketch) Just use the flow-decomposition theorem. Given a solution (x, y) each admissible set of tiles T for row i corresponds to (the arcs of) a path starting at vertex 1 and ending at vertex $n+1$. Sending along this path y_T units of flow, we get in the end a unit of flow out of 1. Vice versa, given a solution (x, z), each z^i identifies a flow of value 1 out of 1. This flow can be decomposed into paths, and each path identifies an admissible set T. The decomposition is based on finding a minimum-flow arc (say of value δ) and tracing it back to the source and to the sink, thus identifying a path. Then we subtract δ from the flow on the arcs of the path and iterate.

Dynamic Programming for Tiling into Strips

Let M be a binary $m \times n$ matrix. For each $i, j \in \{1, 2, \ldots, n\}$ with $i \leq j$, we denote by $M(i, j)$ the submatrix obtained from M by dropping all columns except those with index p with $i \leq p \leq j$, and let $M(i, -) = M(i, n)$ be obtained by removing only the first $i - 1$ columns.

A tiling T of M is called a *striping* of M if for every two tiles $t_1, t_2 \in T$ with $f(t_1) \leq e(t_2) \leq e(t_1)$ we have that $e(t_2) = e(t_1)$ and $f(t_2) = f(t_1)$.

In this section we show that the problem of finding a striping of minimum size can be solved in polynomial time by Dynamic Programming.

The *starting shadow* of a striping T is the set $F(T) := \{f(t) : t \in T\}$. Notice that $F(T)$ uniquely defines the striping T. Indeed, for every tile $t \in T$ we have that $f(t)$ and $e(t) + 1$ are two consecutive integers in $F(T)$, and, conversely, for every two consecutive integers f_1 and f_2 in $F(T)$, striping T contains precisely $variety(M(f_1, f_2 - 1))$ different tiles t with $f(t) = f_1$ and $e(t) = f_2 - 1$, where $Variety(M(i, j))$ are the equivalence classes over the rows of $M(i, j)$ under the identity relation, and $variety(M(i, j)) = |Variety(M(i, j))|$. In this section we show that the problem of finding a striping of minimum size can be solved by Dynamic Programming.

Denote by opt_i the minimum size of a striping for matrix $M(i, -)$. Then, since $M = M(1, -)$, the size of an optimum striping for M is given by opt_1. Moreover, $opt_n = variety(M(n, n))$, which amounts to the number of different symbols occurring in the last column of M (i.e. either 1 or 2). Finally, for $i = n - 1$ down to $i = 1$ we can iteratively compute opt_i by means of the recurrence

$$opt_i := \min_{j>i} opt_j + variety(M(i, j - 1)) .$$

We now introduce the data structures needed to efficienty compute all the values $variety(M(i, j))$ for $j \geq i$. Clearly, $variety(M(i, i))$ is the number of different symbols occurring in the column vector $M(i, i)$. We next show how $variety(M(i, j+1))$ can be computed from $variety(M(i, j))$ in $O(m)$ steps. Thanks to this, the total running time of the Dynamic Programming algorithm outlined above is clearly $O(mn^2)$. The idea is to store the partition of the rows of $M(i, j)$ associated to the identity equivalence relation as a vector $class$ of m entries. In each entry $class[r]$ the smallest index of a row identical to row r is reported. We now describe how the m-vector $class$ gets updated when going from $M(i, j)$ to $M(i, j + 1)$. This is done by using a second m-vector $newName$, initialized to all -1, and running the following algorithm.

```
for r := 1 to n,
  if M[r, j + 1] ≠ M[class[r], j + 1] then
    if newName[class[r]] = −1 then
        newName[class[r]] := r;
        class[r] := r;
    else class[r] := newName[class[r]].
```

Acknowledgments

R.S. was supported in this work by U.S. National Science Foundation award #0612099.

References

1. J. Akey, L. Jin, and M. Xiong: *Haplotypes versus single marker linkage disequilibrium tests: what do we gain?* European Journal of Human Genetics, 9 2001, pp. 291–300.
2. K. L. Ayers, C. Sabatti, and K. Lange: *Reconstructing ancestral haplotypes with a dictionary model.* Journal of Computational Biology, 13 2006, pp. 767–785.
3. K. L. Ayers, C. Sabatti, and K. Lange: *A dictionary model for haplotyping, genotype calling, and association testing.* Genetic Epidemiology, 31 2007, pp. 672–683.
4. P. Berman and M. Karpinski: *On some tighter inapproximability results*, ECCC report no. 29, University of Trier, 1998.
5. N. Castellana, K. Dhamdhere, S. Sridhar, and R. Schwartz: *Relaxing haplotype block models for association testing*, in Proc. Pacific Symposium on Biocomputing (PSB05), 2005.
6. N. H. Chapman and E. M. Wijsman: *Genome screens using linkage disequilibrium tests: optimal marker characteristics and feasibility.* American Journal of Human Genetics, 63 1998, pp. 1872–1885.
7. D. Falush, M. Stephens, and J. K. Pritchard: *Inference of population structure using multilocus genotype data: linked loci and correlated allele frequencies.* Genetics, 164 2003, pp. 1567–1587.
8. S. B. Gabriel, S. F. Schaffer, H. Nguyen, J. M. Moore, J. Roy, and *et al.*: *The structure of haplotype blocks in the human genome.* Science, 296 2001, pp. 2225–2229.
9. L. Helmuth: *Genome research: Map of the human genome 3.0.* Science, 293(5530) 2001, pp. 583–585.
10. J. N. Hirschhorn and M. J. Daly: *Genome-wide association studies for common diseases and complex traits.* Nature Reviews Genetics, 6 2005, pp. 95–108.
11. M. Jakobsson, S. W. Scholz, P. Scheet, J. R. Gibbs, J. M. VanLiere, and et al.: *Genotype, haplotype and copy-number variation in worldwide human populations.* Nature, 451 2008, pp. 998–1003.
12. J. Kaiser: *A plan to capture human diversity in 1000 genomes.* Science, 319 2008, p. 395.
13. G. Kimmel, R. Sharan, and R. Shamir: *Identifying blocks and sub-populations in noisy SNP data*, in Proceedings of the Third Workshop on Algorithms in Bioinformatics (WABI), 2003, pp. 303–319.
14. M. Koivisto et al.: *An MDL method for finding haplotype blocks and estimating the strength of haplotype block boundaries*, in Proceedings of the Pacific Symposium on Biocomputing (PSB), 2003, pp. 502–513.
15. E. Marshall: *Drug firms to create public database of genetic mutations.* Science Magazine, 284(5413) 1999, pp. 406–407.
16. E. W. Sayers, T. Barret, D. A. Benson, S. H. Bryant, K. Canese, and *et al.*: *Database resources of the National Center for Biotechnology Information.* Nucleic Acids Research, 37 2008, pp. D5–D15.
17. R. Schwartz: *Haplotype motifs: an algorithmic approach to locating evolutionarily conserved patterns in haploid sequences*, in Proceedings of the Second IEEE Computer Society Bioinformatics Conference, 2003, pp. 306–314.
18. R. Schwartz: *Algorithms for association study design using a generalized model of haplotype conservation*, in Proc. IEEE Computer Society Bioinformatics Conference, 2004, pp. 90–97.
19. R. Schwartz, A. Clark, and S. Istrail: *Methods for inferring block-wise ancestral history from haploid sequences: The haplotyping coloring problem*, in Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI), R. Guigo and D. Gusfield, eds., Lecture Notes in Computer Science, Springer, 2002.
20. R. Schwartz, B. Halldorsson, V. Bafna, A. G. Clark, and S. Istrail: *Robustness of inference of haplotype block structure.* Journal of Computational Biology, 10(1) 2003, pp. 13–20.
21. S. K. Service, D. W. Lang, N. B. Freimer, and L. A. Sandkuijl: *Linkage-disequilibrium mapping of disease genes by reconstruction of ancestral haplotypes in founder populations.* American Journal of Human Genetics, 64 1999, pp. 1728–1738.
22. S. T. Sherry, M.-H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin: *dbSNP:the NCBI database of genetic variation.* Nucleic Acids Research, 29 2001, pp. 308–311.
23. M. D. Shriver and R. A. Kittles: *Genetic ancestry and the search for personalized genetic histories.* Nature Reviews Genetics, 5 2004, pp. 611–618.

24. S. Sridhar, K. Dhamdhere, G. Blelloch, R. Ravi, and R. Schwartz: *Evaluation of the haplotype motif model using the principle of minimum description*, cmu computer science technical report cmu-cs-04-166, Carnegie Mellon University, Pittsburgh, USA, 2004.
25. The International HapMap Consortium: *The international HapMap project.* Nature, 426 2005, pp. 789–796.
26. The International HapMap Consortium: *A second generation human haplotype map of over 3.1 millions SNPs.* Nature, 449 2007, pp. 851–861.
27. K. Zhang, P. Calabrese, M. Nordborg, and F. Sun: *Haplotype block structure and its applications in association studies: power and study design.* The American Journal of Human Genetics, 71 2002, pp. 1836–1894.
28. K. Zhang, M. Deng, T. Chen, M. Watermanm, and F. Sun: *A dynamic programming algorithm for haplotype block partitioning.* PNAS, 99 2002, pp. 7335–7339.