On the Selection of Robust Tag SNPs

Yao-Ting Huang and Kun-Mao Chao
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
Email: {d92023, kmchao}@csie.ntu.edu.tw

Abstract

Recent studies have shown that the chromosome recombination only takes places at some narrow hotspots. Within segments between these hotspots, called haplotype block, little or even no recombination occurs and a small subset of SNPs, called tag SNPs, are sufficient to capture the entire block pattern. However, the tag SNP may be genotyped as missing data if it does not pass the threshold of data quality, and the DNA sample may fail to be identified due to the ambiguity caused by missing data. In this paper, we formulate this problem as finding a set of SNPs, called auxiliary tag SNPs, which is able to resolve the ambiguity caused by missing data. In addition, we also consider another set of SNPs, called robust tag SNPs, which guarantees no ambiguity regardless of the occurrence of missing data at any tag SNP. Both problems of finding minimum auxiliary and robust tag SNPs are shown to be NP-complete. Our study indicates that auxiliary tag SNPs can be found efficiently when robust tag SNPs have been computed in advance. To find robust tag SNPs, we propose two greedy approximation algorithms. These two approximation algorithms have approximation ratio $(m+1)\ln\frac{K(K-1)}{2}$ *and* $\ln((m+1)\frac{K(K-1)}{2})$ *respectively, where* m *is the number of missing data and* K *is the number of distinct block patterns.*

1 Introduction

In recent years, *Single Nucleotide Polymorphisms* (SNPs) [1] have become more and more popular for association studies[1] of genetic diseases or traits. Although the cost of genotyping SNPs is gradually decreasing, it is still uneconomical to genotype all SNPs for association study [2]. However, recent findings showed that the chromosomal recombination only occurs at some narrow hotspots. The chromosomal region between these hotspots is called a "haplotype block." Within a haplotype block, there is little or even no recombination occurred, and the SNPs (in the block) tend to be inherited together. Due to the low diversity of SNPs in a haplotype block, the information they carry is highly redundant. Thus, a small subset of SNPs (called "tag SNPs") is sufficient to capture the entire SNP pattern of the haplotype block. Haplotype blocks with corresponding tag SNPs are quite useful and cost-effective in association studies as it does not require genotyping all SNPs. Many studies have tried to minimize the number of tag SNPs required in each block. In a large-scale study of chromosome 21, Patil *et al.* [6] developed a greedy algorithm to partition the haplotypes into 4,135 blocks with 4,563 tag SNPs. Zhang *et al.* [7, 8] used a dynamic programming approach to reduce the number of blocks and tag SNPs to 2,575 and 3,562, respectively. Bafna *et al.* [2] showed that the problem of minimizing tag SNPs is NP-hard and gave efficient algorithms for special cases of this problem.

When identifying an unknown DNA sample, the tag SNPs of the DNA sample are genotyped and compared to those of diseases. However, the genotyped tag SNP is considered as the missing data if it does not pass the threshold of data quality [3, 6, 9]. In this case, the DNA sample may fail to be identified. Figure 1 illustrates the influence of the missing data on DNA samples. In this figure, a haplotype block[2] (Figure 1 (A)) defined

[1] To perform association study, scientists first collect DNA samples and extract (genotype) SNPs from diseased and non-diseased individuals. Next, the SNPs from diseased individuals are compared to those from non-diseased ones. Eventually, a profile that contains SNP patterns corresponding to diseases will be established.

[2] This haplotype block is redrawn from the haplotype database of chromosome 21 published by Patil *et al.* [6] at http://www.perlegen.com/haplotype/. We follow the same assumption as Patil *et al.*, Zhang *et al.*, and Bafna *et al.* that all SNPs are biallelic (i.e., taking on only two values).

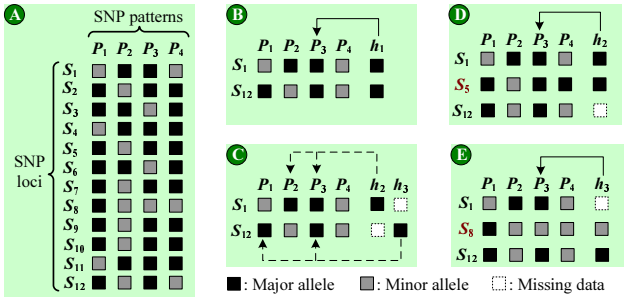

Figure 1: The influence of the missing data on DNA samples and corresponding auxiliary tag SNPs

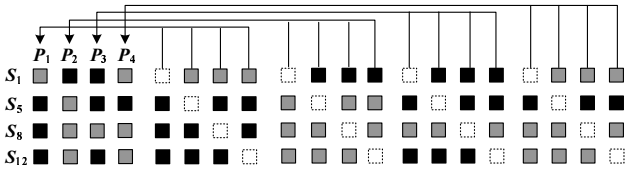

Figure 2: The robust tag SNPs and haplotype samples with missing data occurred at any locus

by 12 SNPs on chromosome 21 is presented. Each column represents a SNP pattern (P_1, P_2, P_3, and P_4) and each row represents a SNP locus (S_1, S_2, ..., and S_{12}). The black and grey boxes stands for the major and minor alleles at the SNP locus, respectively. Suppose we select SNPs S_1 and S_{12} as tag SNPs. The DNA sample h_1 is identified as SNP pattern P_3 unambiguously (Figure 1 (B)). Consider DNA samples D_2 and D_3 with one tag SNP genotyped as the missing data (Figure 1 (C)). h_2 can be identified as SNP patterns P_2 or P_3, and h_3 can be identified as P_1 or P_3. As a result, the missing data results in ambiguity when identifying DNA samples.

Although the selected tag SNPs fail to identify the sample due to missing data, the remaining SNPs within the haplotype block may provide abundant information to resolve the ambiguity. For example, suppose we genotype an additional SNP S_5 for h_2 (Figure 1 (D)). h_2 is identified as SNP pattern P_3 unambiguously. On the other hand, if SNP S_8 is genotyped (Figure 1 (E)), h_3 is also identified unambiguously. These additional SNPs are referred to "auxiliary tag SNPs," which can be found from the remaining SNPs in the block and are able to resolve the ambiguity caused by the missing data.

Alternatively, instead of re-genotyping auxiliary tag SNPs each time when encountering missing data, we can work on a set of SNPs which is not affected by the the occurrence of missing data. For example, suppose we select SNPs S_1, S_5, S_8, and S_{12} to be genotyped. Note that no matter which SNP is genotyped as missing data, the remaining three SNPs can still identify the DNA sample unambiguously (see Figure 1). We refer to these SNPs as "robust tag SNPs," which correctly identify the DNA sample regardless of the missing data occurred at any SNP locus. When the occurrence of missing data is frequently, the cost of re-genotyping processes can be reduced by using robust tag SNPs instead of auxiliary tag SNPs.

This paper studies the problems of finding robust and auxiliary tag SNPs. Our study indicates that auxiliary tag SNPs can be found efficiently if robust tag SNPs have been computed in advance. The result of the paper is organized as follows. In Section 2, we formulate the problem of finding the robust tag SNPs mathematically and prove its NP-hardness. Section 3 gives an efficient approximation algorithm to find robust tag SNPs. Its approximation ratio is $(m+1)\ln(\frac{K(K-1)}{2})$, where m is the number of SNPs genotyped as missing data and K is the number of patterns in the block. Section 4 illustrates the second approximation algorithm which achieves a better approximation ratio $\ln((m+1)\frac{K(K-1)}{2})$. In Section 5, we show the NP-hardness of finding auxiliary tag SNPs and describe an efficient algorithm when robust tag SNPs have been computed in advance. Section 6 presents the experimental result of our algorithms applied to the public haplotype database. Finally, concluding remarks are given in Section 7.

2 Finding Robust Tag SNPs

Assume we are given a haplotype block consisting of N SNPs and K SNP patterns, which is denoted by an $N \times K$ matrix M_h (see Figure 2 (A)). Let $M_h[i,j] \in \{1,2\}$ for each $i \in 1..N$ and $j \in 1..K$, where 1 and 2 represent the major and minor alleles, respectively. The set of robust tag SNPs C' which allows m SNPs genotyped as missing data must satisfy the following two properties: (1) each sample can be identified unambiguously (as one of the K patterns) by SNPs in C'; (2) when m SNPs in C' are genotyped as missing data, (1) still holds. This problem is referred to as *Minimum Robust Tag SNPs* (MRTS) and the formal definition is given below.

Problem: Minimum Robust Tag SNPs

Input: An $N \times K$ matrix M_h and an integer m.

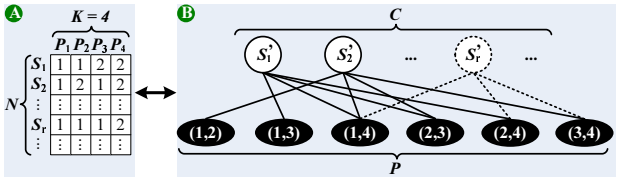

Figure 3: The haplotype matrix M_h and the corresponding bipartite graph G

Output: The minimum subset of rows (SNPs) C' in M_h which satisfies:
(1) for each pair of patterns P_i and P_j, these is a row k in C' such that $M_h[k,i] \neq M_h[k,j]$[3];
(2) when m rows are discarded from C' arbitrarily, (1) still holds.

Now we show that the MRTS problem can be reformulated as a variation of the set covering problem [5]. Each row k in M_h is reformulated as a set $S_k^{'} = \{(i,j) \mid M[k,i] \neq M[k,j] \text{ and } i < j\}$. For example, suppose the row k in M_h is {1,1,1,2}. The corresponding set $S_k^{'} = \{(1,4),(2,4),(3,4)\}$. Let C be the collection of $S_k^{'}$, where $1 \leq k \leq N$. Let P be the set that contains each pair of these K patterns (i.e., $P = \{(i,j) \mid 1 \leq i < j \leq K\} = \{(1,2),(1,3),\cdots,(K-1,K)\}$). The following lemma implies that the set of robust tag SNPs C' is a collection such that each element in P is covered by corresponding sets of C' for at least $(m+1)$ times.

Lemma 1 *$C' \subseteq C$ is the set of robust tag SNPs which allow m SNPs genotyped as missing data iff each element in P is covered by the sets in C' for at least $(m+1)$ times.*

Proof:
Consider each element (i,j) in P and each set $S_k^{'}$ in C as nodes in an undirected bipartite graph G (see Figure 2 (B)). There is an edge connecting the node (i,j) and $S_k^{'}$ *iff* $(i,j) \in S_k^{'}$. Consider a subset of nodes $C' \subseteq C$ such that each node in P has at least $(m+1)$ edges connected to some node in C' (i.e., C' covers P for at least $(m+1)$ times). Suppose the SNP S_r of the sample is genotyped as missing data, where $S_r^{'} \in C'$. This implies that the row r in M_h can not be used to distinguish the sample, which has the same effect as the removal of the node $S_r^{'}$ and its edges from C'. If m nodes in C' are removed (i.e., m SNPs genotyped as missing data), each node in P still connects to some node in C' (i.e., each pair of patterns can still be distinguished by the remaining SNPs in C). Thus, the SNPs corresponding to C' are the robust tag SNPs which allow m SNPs genotyped as missing data. The proof of the other direction is similar. □

[3]To identify the sample unambiguously, each pair of patterns must be distinguished by some row in C'. For example (see Figure 2 (A)), the patterns P_1 and P_2 can be distinguished by SNP S_2 since $M_h[2,1] \neq M_h[2,2]$.

Now we show that the NP-hardness of the MRTS problem.

Theorem 1 *The MRTS problem is NP-hard.*
Proof:
By Lemma 1, the set covering problem [5] can be reduced to a special case of MRTS when $m = 0$. Since the the set covering problem is NP-hard, MRTS is NP-hard. □

¿From Theorem 1, there is no polynomial time algorithm for solving MRTS unless $P = NP$. In Sections 3 and 4, we give two efficient approximation algorithms to solve MRTS.

3 The First Approximation Algorithm

In this section, we describe an approximation algorithm to solve MRTS by a greedy approach. By Lemma 1, we can solve MRTS by finding a subcollection $C^{'} \subseteq C$ that covers (distinguishes) each element (pair of patterns) in P for at least $(m+1)$ times. Assume that each element in P and the corresponding SNPs that distinguish it are stored in a $(m+1)\times|P|$ table (see Figure 3 (A)). If a SNP S_k is picked, S_k is written into the grid of the column (i,j), where (i,j) is the pair of patterns distinguished by S_k. At each step, the first algorithm picks a SNP that can write most grids in the row by a row-by-row manner. Figure 3 illustrates an example for this algorithm to cover P twice, where SNPs S_1, S_4, S_2, and S_3 are picked in order. Let R_i be the set of unwritten grids at row i. While writing R_i, this algorithm picks a set $S \in C$ that maximizes $|S \cap R_i|$) (i.e., the set that writes most unwritten grids in R_i). Then, S is added to $C^{'}$, and the rest elements in S (i.e., $S - R_i$) are written into the remaining unwritten grids in other rows. When all grids in this table are covered, $C^{'}$ is thus the collection that can cover P for $(m+1)$ times. The detail of this algorithm is given below.

Algorithm: GREEDY-1(C, P, m)

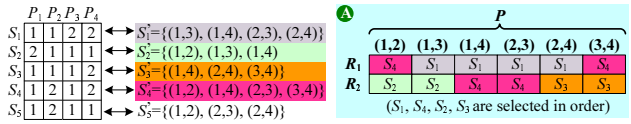

Figure 4: The sets picked by the first greedy algorithm

```
1   R_i ← P, for each 1 ≤ i ≤ m + 1
2   C' ← φ
3   for i = 1 to m + 1 do
4       while R_i ≠ φ do
5           select an S ∈ C that maximizes |S ∩ R_i|
6           C' ← C' ∪ S
7           j ← i
8           while S ≠ φ and j ≤ m + 1 do
9               S_tmp ← S ∩ R_j
10              R_j ← R_j − S_tmp
11              S ← S − S_tmp
12              j ← j + 1
13  endfor
14  return C'
```

The time complexity of this algorithm is analyzed as follows. At Line 4, the number of iterations of the intermediate loop is bounded by $|R_i| \leq |P|$. Within the loop body (Lines 5-12), Line 5 takes $O(|P||C|)$ because the we need to check each set in C and compare with each element in $R_i \leq |P|$. The inner loop (Lines 8-12) takes only $O(|S|)$. Thus, the entire program runs in $O(m|C||P|^2)$.

We now calculate the approximation ratio of $|C^{'}|$ to $|C^*|$, where C^* is the collection of the optimal solution. The approximation ratio is proved by assigning different scores to the sets picked by the greedy algorithm [4]. Let $|S_k^w|$ be the number of grids written by S_k in the row where S_k is picked by the algorithm. For example (see Figure 3), $|S_4^w| = 2$ since S_4 writes two grids, (1,2) and (3,4), in the first row where the greedy algorithm picks S_4. The score C_j^i is assigned to each grid put at the ith row and jth column, where

$$C_j^i = \begin{cases} \frac{1}{|S_k^w|} & \text{if the grid is written by } S_k \text{ while working the } i\text{th row;} \\ 0 & \text{Otherwise.} \end{cases}$$

Under this score assignment, the summation of the score C_j^i for each grid in the table is equal to $|C|$, that is,

$$\sum_{i=1}^{m+1} \sum_{j=1}^{\frac{K(K-1)}{2}} C_j^i = |C| \ . \qquad (1)$$

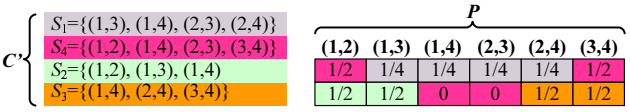

Figure 5: The score C_j^i for each set picked by the first greedy algorithm

Let R_k^i be the number of grids in the ith row remaining unwritten before the kth iteration (i.e., S_1, S_2, $\cdots$,S_{k-1} have been picked by the algorithm). Similar to [4], the summation of the score for each grid can be calculated as

$$\sum_{i=1}^{m+1} \sum_{j=1}^{\frac{K(K-1)}{2}} C_j^i = \sum_{i=1}^{m+1} \sum_{k=1}^{|C|} (R_{k-1}^i - R_k^i) \frac{1}{|S_k^w|} \ . \quad (2)$$

Lemma 2 $|S_k^w| \geq \frac{R_k^i}{|C^*|}$.

Proof:

Consider the beginning of the kth iteration. Let C_k^* be the collection of sets in C^* that has been picked by the greedy algorithm before the kth iteration, and the collection of remaining sets in C^* be $C_{\bar{k}}^*$. We claim that if this algorithm subsequently picks all sets in $C_{\bar{k}}^*$, the remaining unwritten grids in the table will be all written. Otherwise (i.e., some grids remain unwritten), since $C_k^* \cup C_{\bar{k}}^* = C^*$, this implies sets in C^* fail to write all grids in the table, which is a contradiction. By the pigeonhole principle, there exists one set in $C_{\bar{k}}^*$ with size at least $\frac{R_k^i}{|C_{\bar{k}}^*|}$.[4] Because sets in $C_{\bar{k}}^*$ are candidates to the greedy algorithm, $|S_k^w|$ must be at least $\frac{R_k^i}{|C_{\bar{k}}^*|}$, which implies $|S_k^w| \geq \frac{R_k^i}{|C^*|}$ since $|C^*| \geq |C_{\bar{k}}^*|$. □

Theorem 2 *The approximation ratio of the first greedy algorithm is* $(m+1)\ln\frac{K(K-1)}{2}$.

Proof:

From (2) and Lemma 2, we have

$$\begin{aligned} \sum_{i=1}^{m+1} \sum_{j=1}^{\frac{K(K-1)}{2}} C_j^i &= \sum_{i=1}^{m+1} \sum_{k=1}^{|C|} (R_{k-1}^i - R_k^i) \frac{1}{|S_k^w|} \\ &\leq \sum_{i=1}^{m+1} \sum_{k=1}^{|C|} (R_{k-1}^i - R_k^i) \frac{C^*}{R_{k-1}^i} \end{aligned}$$

[4] If each set in $C_{\bar{k}}^*$ has size less than $\frac{R_k^i}{|C_{\bar{k}}^*|}$, the summation of the size of all sets in $C_{\bar{k}}^*$ is $< \frac{R_k^i}{|C_{\bar{k}}^*|} \times |C_{\bar{k}}^*| = R_k^i$. Since $C_k^* \cup C_{\bar{k}}^* = C^*$, this means that C^* can not write all grids in R_k^i, which is a contradiction.

$$
\begin{aligned}
&= \sum_{i=1}^{m+1}\sum_{k=1}^{|C|}\Big(\sum_{l=R_k^i+1}^{R_{k-1}^i}\Big)\frac{C^*}{R_{k-1}^i} \\
&\le C^*\sum_{i=1}^{m+1}\sum_{k=1}^{|C|}\sum_{l=R_k^i+1}^{R_{k-1}^i}\frac{1}{l} \\
&= C^*\sum_{i=1}^{m+1}\sum_{k=1}^{|C|}\Big(\sum_{l=1}^{R_{k-1}^i}\frac{1}{l}-\sum_{l=1}^{R_k^i}\frac{1}{l}\Big) \\
&\le C^*\sum_{i=1}^{m+1}\sum_{k=1}^{|C|}(H(R_{k-1}^i)-H(R_k^i)) \\
&= C^*\sum_{i=1}^{m+1}[(H(R_0^i)-H(R_1^i))+\ldots \\
&\quad +(H(R_{|C|-1}^i)-H(R_{|C|}^i))] \\
&= C^*\sum_{i=1}^{m+1}[(H(R_0^i)-H(R_{|C|}^i))] \\
&\le C^*(m+1)max(H(R_0^i)) \\
&\le C^*(m+1)\ln|P| \ . \qquad (3)
\end{aligned}
$$

Combining (1) and (3), we get

$$\frac{C}{C^*} \le (m+1)\ln|P| = (m+1)\ln\frac{K(K-1)}{2} \ .$$

□

4 The Second Approximation Algorithm

This section gives a greedy algorithm which achieves a better approximation ratio than that in Section 3. Let U be the collection of R_i, which is defined in Section 3. In other words, U contains $m+1$ elements corresponding to each element in P. The greedy algorithm works by picking a set S that can cover the most uncovered elements from the union of R_i at each step (i.e., the S that maximizes $|S \cap (R_1 \cup \cdots \cup R_{m+1})|$) and adds S to C. Then, the elements in S are used to cover the remaining elements in R_i and we remove the R_i with no more uncovered elements from U. When all sets in U are covered, C can cover P $m+1$ times. The algorithm is described as follows:

Algorithm: GREEDY-2(C, P, m)
1 $R_i \leftarrow P$, for each $1 \le i \le m+1$
2 $U \leftarrow \{R_1, R_2, \ldots, R_{m+1}\}$
3 $C \leftarrow \phi$

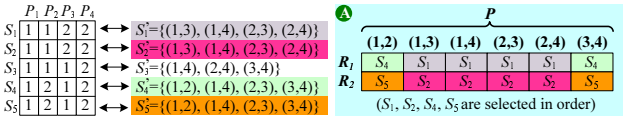

Figure 6: The sets picked by the second greedy algorithm

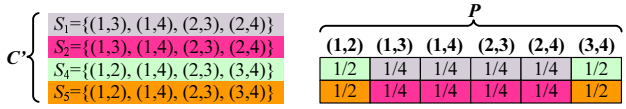

Figure 7: The score C_j^i for each grid in the table

4 **while** $U \neq \phi$ **do**
5 select and removed an $S \in C$ that maximizes $|S \cap (R_1 \cdots \cup R_{m+1})|$
6 $C \leftarrow C \cup S$
7 **for** each $R_i \in U$ **do**
8 $S_{tmp} \leftarrow S \cap R_i$
9 $R_i \leftarrow R_i - S_{tmp}$
10 $S \leftarrow S - S_{tmp}$
11 **if** $R_i = \phi$ **then** $U \leftarrow U - R_i$
12 **endfor**
13 **return** C

Figure 4 illustrates an example for this algorithm to cover P twice. The greedy algorithm picks the set S_1, S_2, S_4, and S_5, in order. At Line 4, the number of iterations of the loop is bounded by the number of elements contained in U, which is $(m+1)|P|$. Within the loop, Line 5 can be implemented to take $O(|P||C|)$ because the union of R_i only need to be calculated once and then stored in a separated variable. The inner loop (Lines 7-12) is also bounded at $O(|S| < |P|)$. Thus, the running time of this program is $O(m|C||P|^2)$.

We analyze the approximation ratio of the second greedy algorithm. Let C be the collection of sets chosen by this greedy algorithm, and C^* be the collection of sets chosen by the optimal solution. Let $|S_k'|$ be the number of elements covered by S_k which contribute to the greedy algorithm. For example (see Figure 4), $|S_4'| = 2$ since S_4 covers two elements, (1,2) and (3,4), in the table. With similar techniques as in Section 3, assign the score $C_j^i = \frac{1}{|S_k'|}$ to the grid positioned at the ith row and jth column.

Let T_k be the number of grids in the table remaining uncovered before the kth iteration. We have the following lemma:

Lemma 3 $|S_k'| \ge \frac{T_k}{|C^*|}$.
Proof:

C_k^* and $C_{\bar{k}}^*$ are defined the same as in the proof of Lemma 2. We claim that there exists a set in $C_{\bar{k}}^*$ which has size at least $\frac{T_k}{|C_{\bar{k}}^*|}$. Since the greedy algorithm always pick a set that covers maximal uncovered grids and all sets in $C_{\bar{k}}^*$ are available, $|S_k'| \geq \frac{T_k}{|C_{\bar{k}}^*|} \geq \frac{T_k}{|C^*|}$. □

Theorem 3 *The approximation ratio for the second greedy algorithm is* $\ln((m+1)\frac{K(K-1)}{2})$.

Proof:
The summation of the score for each grid would be

$$\begin{aligned} C &= \sum_{i=1}^{m+1} \sum_{j=1}^{\frac{K(K-1)}{2}} C_j^i \\ &= \sum_{k=1}^{|C|} (T_{k-1} - T_k) \frac{1}{|S_k'|} \\ &\leq \sum_{k=1}^{|C|} (T_{k-1} - T_k) \frac{C^*}{T_{k-1}} \\ &\leq C^* H(T_0) \\ &\leq C^* \ln((m+1)\frac{K(K-1)}{2}) \ . \end{aligned} \quad (4)$$

¿From (4), we get

$$\frac{C}{C^*} \leq \ln((m+1)\frac{K(K-1)}{2}) \ .$$

□

5 Finding Auxiliary Tag SNPs

This section describes the problem of finding the auxiliary tag SNPs corresponding to a set of tag SNPs S with missing data. M_h is defined as in Section 2. This problem is referred to as *Minimum Auxiliary Tag SNPs* (MATS) and defined as follows.

Problem: Minimum Auxiliary Tag SNPs

Input: An $N \times K$ matrix M_h, and a set S.

Output: The minimum set of auxiliary tag SNPs A such that $A \cup S$ can identify the haplotype sample without ambiguity.

Note that when N and K become larger and the number of missing data increases, it is more difficult to find the auxiliary tag SNPs.

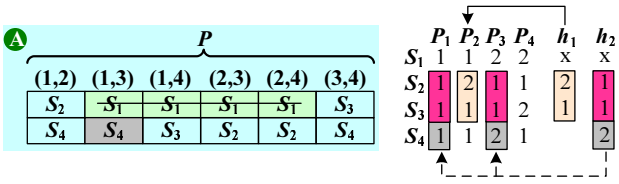

Figure 8: Examples to find the auxiliary tag SNPs by robust tag SNPs

Theorem 4 *MATS is NP-hard.*

Proof: Consider that all tag SNPs are genotyped as missing data. This problem is just like finding another set of tag SNPs to distinguish those K patterns, which is known as NP-hard. □

Although the MATS problem is NP-complete, we show that auxiliary tag SNPs can be found efficiently when robust tag SNPs have been computed in advance. According to Lemma 1, each element in P must be covered for $m+1$ times by these robust tag SNPs. Without loss of generality, assume that these robust tag SNPs are implemented and stored in an $(m+1) \times |P|$ matrix M_r (see Figure 5). Each column in M_r represents an element in P and each $M_r\{*, j\}$ stores the SNP that covers the jth element in P. With this matrix, we can apply the following algorithm to find auxiliary tag SNPs with respect to S. At first, we compare the rest SNPs (which are not missing data) in S with each of the K patterns. If there is only one pattern matched (e.g., h_1 in Figure 5), the haplotype sample is identified as that block pattern (e.g., P_2) and we are done. Otherwise (e.g., h_2 in Figure 5), each pair of the matched patterns (e.g., P_1 and P_3) stands for an uncovered element in P and requires further disambiguation. Then, for each pair of the ambiguous patterns, traverse the corresponding column in M_r to find a set which can distinguish the pair of patterns (e.g., S_4 can distinguish P_1 and P_3). Let the collection of these sets for each ambiguous pair of patterns be A. According to Lemma 1, since all uncovered elements in P can be covered by $A \cup S$, A is the set of auxiliary tag SNPs corresponding to S. The detail of this algorithm is described as follows:

Algorithm: FINDING-AUXILIARY-SNPS(M_h, M_r, S)
1 $A \leftarrow \phi$
2 *counter* $\leftarrow 0$
3 **for** each haplotype pattern $h \in M_h$ **do**
4 **if** these is no mismatch between each SNP in S and h **then**

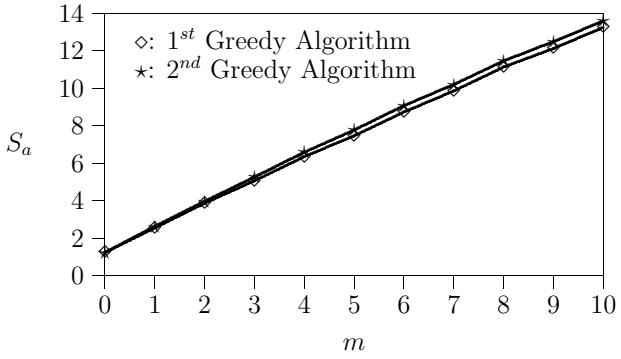

Figure 9: The average number of tag SNPs for each block with respect to m

```
5          counter ← counter + 1
6       endif
7    endfor
8    if counter = 1 then
9       return "No more SNPs required";
10   else
11      for each pair of the matched pattern j do
12         i ← 1
13         while M_r[i, j] ∈ S and i ≤ m do
14            i ← i + 1
15         Add M_r[i, j] to A
16      endfor
17   endif
18   return A
```

As for the running time of this algorithm, the worst case is that all SNPs in S are genotyped as missing data, and we need to traverse each column in the matrix M_r (Lines 11-16). Note that the size of P is $C_2^K = \frac{K(K-1)}{2}$. Thus, the running time of this algorithm is $O(m|P|)$=$O(m\frac{K(K-1)}{2})$=$O(mK^2)$.

6 Experimental Result

We apply our two approximation algorithms mentioned in Sections 3 and 4 on the public haplotype data of Human Chromosome 21 [6]. This data set includes 20 haplotypes of 24,047 SNPs spanning over about 32.4MB. We first compare the number of tag SNPs found by our algorithms with the optimal number found by Patil *et al.* over the same 4,135 blocks. The first and second algorithms both find 4,610 tag SNPs, where the optimal number they found is 4,563. Thus, the ratio is $\frac{4610}{4563} \simeq 1.01$, which indicates our approximation algorithms are quite close to the optimal solution.

Next, we evaluate these two algorithms with respect to m (i.e., the number of SNPs genotyped as missing data). Let S_a be the average number of robust tag SNPs for each block found by these two algorithms. Figure 6 plots S_a with respect to m. We observe that S_a grows linearly for both algorithms. Note that the lower bound of robust tag SNPs is $(m + \lg|K|)$, where K is the number of patterns. This phenomenon indicates that the number of robust tag SNPs does not grow too much when m increases and is quite close to the lower bound.

Although the theoretical approximation ratio of the second algorithm is better than that of the first one, our experimental result indicates that the first algorithm slightly outperforms the second one when m becomes large in the longer blocks[5]. This is because we search for SNPs to pick from the beginning of the long block but the SNPs within do not vary too much (e.g., the SNP with pattern (1,1,2,2) is repeated continuously at many loci). The second algorithm tends to pick the former SNPs in the block, where the first algorithm tends to pick the latter ones since it finds different patterns at each step. Thus, for long blocks where SNPs of the optimal solution distributed at two ends, the first algorithm is slightly better.

7 Conclusion

In this paper, we study the problems of finding robust and auxiliary tag SNPs. We describe two greedy approximation algorithms for finding robust tag SNPs. An efficient algorithm is presented to find auxiliary tag SNPs when robust tag SNPs have been computed in advance. Our experimental result shows that the solution found by both greedy algorithms is quite close to the optimal solution even when the number of SNPs allowed for missing data increases. Note that the first greedy algorithm tries to optimize SNPs in the first row of the table structure. Therefore, if the occurrence of missing data is infrequently, we can select the SNPs in the first row to genotype, and re-genotype auxiliary tag SNPs only when encountering missing data. Since the solution found by both greedy algorithms is similar, the first greedy algorithm may be more useful than the second one in practice.

[5]In the data set from Patil, there are many short blocks with very few SNPs, which are not biologically meaningful. We discard blocks that do not contain enough SNPs for the solution of MRTS in the experiment.

Acknowledgements. We thank Ting Chen, Chia-Yu Su, and the reviewer for helpful comments. Yao-Ting Huang and Kun-Mao Chao were supported in part by an NSC grant 92-2213-E-002-059.

References

[1] *http://www.ncbi.gov/dbsnp*.

[2] Bafna, V., Halldorsson, B.V., Schwartz, R., Clark, A.G., and Istrail, S. Haplotypes and Informative SNP Selection Algorithms: Don't Block Out Information. *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology*, pages 19–27, 2003.

[3] Bourgain, C., Genin, E., Ober, C., and Clerget-Darpoux, F. Missing data in haplotype analysis: a study on the MILC method. *Annals of Human Genetics*, 66(1):99–108, 2002.

[4] Cormen T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. *Introduction to Algorithms*. The MIT Press, 2001.

[5] Garey, M.R. and Johnson, D.S. *Computers and Intractability*. Freeman, New York, 1979.

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

[7] Zhang, K., Deng, M., Chen, T., Waterman, M.S., and Sun, F. A Dynamic Programming Algorithm for Haplotype Block Partitioning. *Proceedings of the National Academy of Sciences of the United States of America*, 99:7335–7339, 2002.

[8] Zhang, K., Sun, F., Waterman, M.S., and Chen, T. Dynamic Programming Algorithms for Haplotype Block Partitioning: Applications to Human Chromosome 21 Haplotype Data. *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology*, pages 332–340, 2003.

[9] Zhao, J.H., Lissarrague, S., Essioux, L., and Sham, P.C. GENECOUNTING: haplotype analysis with missing genotypes. *Bioinformatics*, 18(12):1694–1695, 2002.