Pattern Identification in a Haplotype Block

Kun-Mao Chao[1,2,3]

[1]Graduate Institute of Biomedical Electronics and Bioinformatics
[2]Department of Computer Science and Information Engineering
[3]Graduate Institute of Networking and Multimedia
National Taiwan University, Taipei, Taiwan 106
Email: kmchao@csie.ntu.edu.tw

May 24, 2011

Abstract

A Single Nucleotide Polymorphism (SNP, pronounced *snip*) is a single nucleotide variation in the genome that recurs in a significant proportion of the population of a species. In recent years, the patterns of Linkage Disequilibrium (LD) observed in the human population reveal a block-like structure. The entire chromosome can be partitioned into high LD regions, referred to as haplotype blocks, interspersed by low LD regions, referred to as recombination hotspots. Within a haplotype block, there is little or no recombination and the SNPs are highly correlated. Consequently, a small subset of SNPs, called tag SNPs, is sufficient to distinguish the haplotype patterns of the block. Using tag SNPs for association studies can greatly reduce the genotyping cost since it does not require genotyping all SNPs. We illustrate how to recast the tag SNP selection problem as the set-covering problem and the integer-programming problem – two well-known optimization problems in computer science. Greedy algorithms and LP-relaxation techniques are then employed to tackle such optimization problems. We conclude the chapter by mentioning a few extensions.

Keywords: SNP, haplotype block, set cover, integer programming, greedy algorithm, approximation.

1 Introduction

A DNA sequence is a string of the four nucleotide "letters" `A` (adenine), `C` (cytosine), `G` (guanine), and `T` (thymine). The genetic variations in DNA sequences have a major impact on genetic diseases and phenotypic differences. Among various genetic variations, the *Single Nucleotide Polymorphism* (SNP, pronounced *snip*) is one of the most frequent forms and has fundamental importance for disease association and drug design. A SNP is a single nucleotide variation in the genome that recurs in a significant proportion of the population of a species. Specifically, a single nucleotide mutation is called a SNP if its minor allele frequency is no less than a given threshold, say 1%. For example, a mutation in the genome in which 85% of the population have a `G` and the remaining 15% have an `A` is a SNP. Since tri-allelic and tetra-allelic SNPs are very rare, we often refer to a SNP as a bi-allelic marker: major allele vs. minor allele. Millions of SNPs have been identified and made publicly available.

In recent years, the patterns of *Linkage Disequilibrium* (LD) observed in the human population reveal a block-like structure. LD refers to the association that particular alleles at nearby sites are more likely to occur together than would be predicted by chance. The entire chromosome can be partitioned into high LD regions interspersed by low LD regions. The high LD regions are usually called "haplotype blocks," and the low LD ones are referred to as "recombination hotspots." Since there is little or no recombination within a haplotype block, these SNPs are highly correlated. Consequently, a small subset of SNPs, called tag SNPs or haplotype tagging SNPs, is sufficient to categorize the haplotype patterns of the block. It is thus possible to identify genetic variation without genotyping every SNP in a given haplotype block. This can greatly reduce the genotyping cost for genome-wide association studies.

In this study we assume that the haplotype blocks have been delimited in advance, and our objective is to find a minimum set of SNPs which can distinguish all pairs of haplotype patterns in a given block. Figure 1 depicts a haplotype block containing five SNPs and four haplotype patterns. To determine which haplotype pattern category a sample belongs to, we may genotype all five SNPs in this block. However, it works just as well if we only genotype SNPs S_1 and S_4 since their combinations can distinguish all pairs of haplotype patterns. For example, if both S_1 and S_4 are major alleles, the sample is categorized as haplotype pattern P_3.

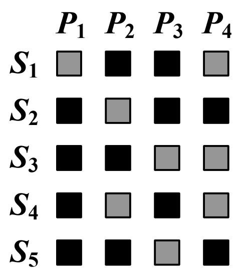

Figure 1: A haplotype block containing five SNPs and four haplotype patterns. In this figure, a black square stands for a major allele and a gray square stands for a minor allele.

We show that the tag SNP selection problem is analogous to the minimum test collection problem. We then illustrate how to recast the tag SNP selection problem as the set-covering problem and solve it approximately by a greedy algorithm. Furthermore, it can be formulated as an integer-programming problem, and a simple rounding algorithm can be employed to find its near-optimal solutions. We conclude this chapter by mentioning a few extensions.

2 The Tag SNP Selection Problem

Assume that we are given a haplotype block containing n SNPs and h haplotype patterns. Let $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ denote the SNP set and let $\mathcal{P} = \{P_1, P_2, ..., P_h\}$ denote the pattern set. A haplotype block is represented by an $n \times h$ binary matrix M whose entries are either a black square or a gray square, representing the major and minor alleles, respectively. Figure 1 depicts a 5×4 haplotype block.

We say that SNP S_i can distinguish the pattern pair P_j and P_k if $M[i,j] \neq M[i,k]$, where $1 \leq i \leq n$ and $1 \leq j < k \leq h$. In other words, if one pattern contains a major allele of SNP S_i, and the other contains a minor allele of SNP S_i, then the two patterns can be distinguished by S_i. For instance, in Figure 1, SNP S_1 can distinguish patterns P_1 and P_4 from P_2 and P_3 since P_1 and P_4 contain a minor allele of S_1, and P_2 and P_3 contain a major allele of S_1. The goal of the *tag SNP selection problem* is to find a minimum number of SNPs that can distinguish all possible pairwise combinations of patterns. In Figure 2, S_1 and S_4 form a set of tag SNPs since they can distinguish all

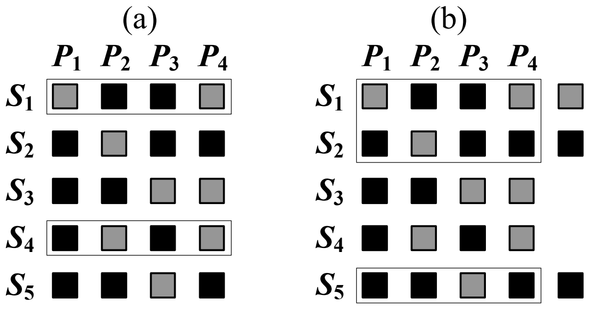

Figure 2: Selecting tag SNPs that can distinguish all pairs of haplotype patterns. (a) SNPs S_1 and S_4 form a minimum set of tag SNPs. (b) SNPs S_1, S_2, and S_5 do not form a set of tag SNPs since they cannot distinguish the pair P_1 and P_4.

pairs in $\mathcal{P}$ whereas S_1, S_2, and S_5 do not form a set of tag SNPs since they cannot distinguish the pair P_1 and P_4.

In fact, the tag SNP selection problem is analogous to the minimum test collection problem, which arises naturally in fault diagnosis and pattern identification. Given a collection C of subsets of a finite set $\mathcal{A}$ of "possible diagnoses," the minimum test collection problem is to ask for a subcollection $\mathcal{C}' \subseteq \mathcal{C}$ such that $|\mathcal{C}'|$ is minimized and, for each pair $a_j, a_k \in \mathcal{A}$, there exists some set (i.e., a *test*) in $\mathcal{C}'$ that contains exactly one of them. In other words, such a test can *distinguish* the pair a_j, a_k. Take Figure 1 for example. SNP S_1 can distinguish patterns P_1 and P_4 from others, thus we include $\{P_1, P_4\}$ in $\mathcal{C}$. Similarly, each of SNPs S_2, S_3, S_4, and S_5 can distinguish a particular set of patterns from others. It follows that the instance of the minimum test collection problem for Figure 1 is $\mathcal{A} = \{P_1, P_2, P_3, P_4\}$ and $\mathcal{C} = \{\{P_1, P_4\}, \{P_2\}, \{P_3, P_4\}, \{P_2, P_4\}, \{P_3\}\}$. Its minimum subcollection $\mathcal{C}'$ is $\{\{P_1, P_4\}, \{P_2, P_4\}\}$ since $|\mathcal{C}'| = 2$ is minimal and $\mathcal{C}'$ can distinguish all pairs in $\mathcal{A}$. The corresponding set of tag SNPs for $\mathcal{C}'$ is $\{S_1, S_4\}$.

Unfortunately, the minimum test collection problem has been proved to be NP-hard, which is a technical term that stands for a class of intractable problems for which no efficient algorithms have been found. Nevertheless, we may employ some algorithmic strategies to tackle NP-hard problems by finding near-optimal solutions; in practice, these solutions are often good enough. In the next section, we show that the tag SNP selection problem can be reformulated as the set-covering problem, which is well-studied in the field of approximation algorithms. By this reformulation, a simple greedy method for the set-covering problem can be employed for solving the tag SNP selection problem. The algorithm may not always deliver an optimal solution, but we will show that the ratio of its heuristic solution to an optimal solution is bounded by a certain factor.

3 A Reduction to the Set-Covering Problem

We now recast the tag SNP selection problem as the *set-covering* problem. Given a universal set $\mathcal{U}$ and a collection $\mathcal{C}$ of subsets of $\mathcal{U}$, the set-covering problem is to find a minimum-size subcollection of $\mathcal{C}$ that covers all elements of $\mathcal{U}$. It is an abstraction of many naturally arising combinatorial problems, such as crew scheduling, committee forming, and service planning. For example, a universal set $\mathcal{U}$ could represent a set of skills required to perform a task. Each person in the candidate pool has certain skills in $\mathcal{U}$. The objective is to form a task force with as few people as possible so that all the required skills are owned by at least one person in the task force. In other words, we wish to recruit a minimum number of persons to *cover* all the requisite skills.

Recall that a haplotype block is represented by an $n \times h$ binary matrix M whose entries are either a black square (representing a major allele) or a gray square (representing a minor allele). To reformulate the tag SNP selection problem as a set-covering problem, let $\mathcal{U} = \{(j,k) \mid 1 \leq j < k \leq h\}$ be the set of all possible pairwise haplotype pattern indexes. Let $\mathcal{C} = \{C_1, C_2, ..., C_n\}$, where $C_i = \{(j,k) \mid M[i,j] \neq M[i,k] \text{ and } 1 \leq j < k \leq h\}$ stores the index pairs of haplotype patterns that SNP $S_i \in \mathcal{S}$ can distinguish. We show that a subset of $\mathcal{S}$ forms a set of tag SNPs if and only if its corresponding subset of $\mathcal{C}$ covers all the elements in $\mathcal{U}$. Each element in $\mathcal{U}$ represents a pair of haplotype patterns needed to be distinguished. If a subset of $\mathcal{C}$ covers all the elements in $\mathcal{U}$, then its corresponding SNP subset of $\mathcal{S}$ forms a set of tag SNPs since all pairs of haplotype patterns can be distinguished. Conversely, if a subset of $\mathcal{S}$ forms a set of tag SNPs, it can distinguish all pairs of haplotype patterns, which yields that its corresponding subset of $\mathcal{C}$ covers all the elements in $\mathcal{U}$.

Now let us consider the example given in Figure 1. We have four haplotype patterns, so the universal set $\mathcal{U}$ is $\{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$, which contains all the elements to be covered. Since SNP S_1 can distinguish patterns P_1 and P_4 from P_2 and P_3, we set C_1 to be $\{(1,2),(1,3),(2,4),(3,4)\}$ (see Figure 3). SNP S_2 can distinguish pattern P_2 from P_1, P_3, and P_4, so

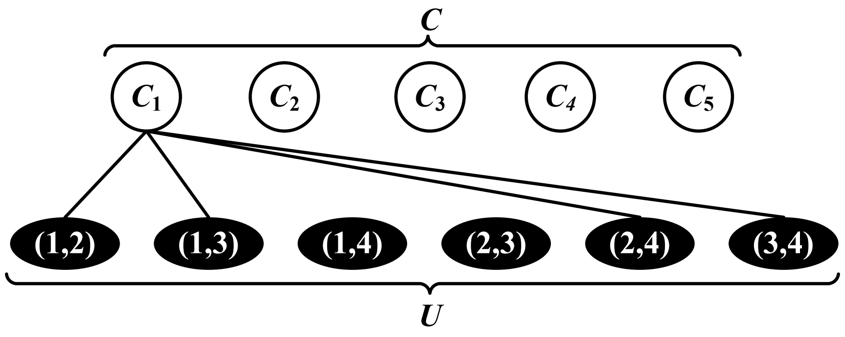

Figure 3: The elements covered by C_1, which correspond to the pairs of haplotype patterns distinguished by SNP S_1.

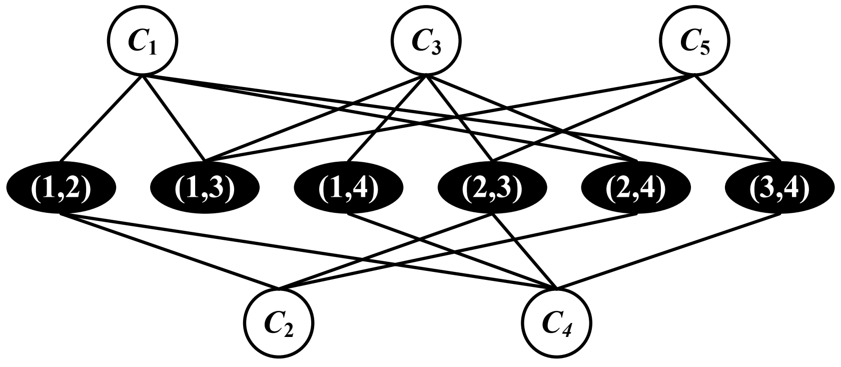

Figure 4: The elements covered by each C_i in $\mathcal{C}$.

we set C_2 to be $\{(1,2),(2,3),(2,4)\}$. Figure 4 depicts the pairs of haplotype patterns distinguished by each SNP. As a consequence, the collection $\mathcal{C}$ of subsets is $\{C_1,\ C_2,\ C_3,\ C_4,\ C_5\}$, where

$$\begin{aligned}
C_1 &= \{(1,2),(1,3),(2,4),(3,4)\},\\
C_2 &= \{(1,2),(2,3),(2,4)\},\\
C_3 &= \{(1,3),(1,4),(2,3),(2,4)\},\\
C_4 &= \{(1,2),(1,4),(2,3),(3,4)\}, \text{and}\\
C_5 &= \{(1,3),(2,3),(3,4)\}.
\end{aligned}$$

As shown in Figure 2(b), S_1, S_2 and S_5 do not form a set of tag SNPs since they cannot distinguish the pair P_1 and P_4. In the corresponding set-covering instance, element $(1,4)$ is not covered by C_1, C_2, and C_5 (see Figure 5).

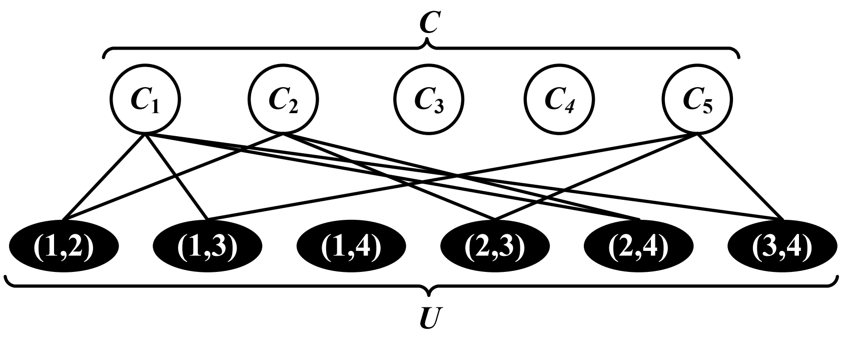

Figure 5: An invalid set cover. Element $(1,4)$ is not covered by C_1, C_2, and C_5.

On the contrary, S_1 and S_4 form a set of tag SNPs since they can distinguish all pairs in $\mathcal{P}$. In the corresponding set-covering instance, each element is covered by at least one set in $\mathcal{C}$ (see Figure 6).

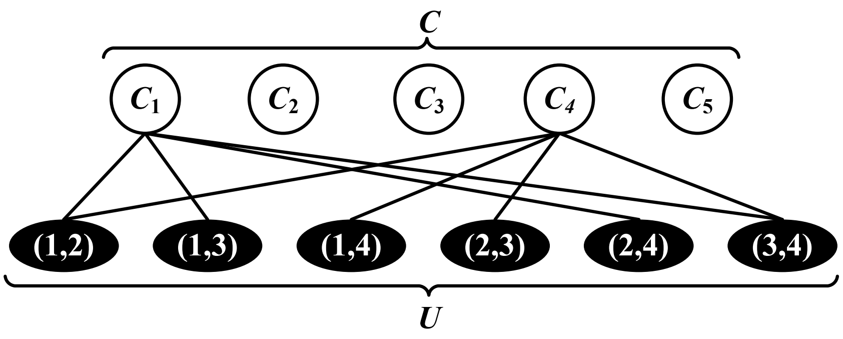

Figure 6: A valid set cover. All elements are covered by C_1 and C_4.

Now let us consider a greedy method for the set-covering problem. The greedy algorithm iteratively picks the set that covers the most remaining uncovered elements until all elements are covered. In the context of the tag SNP selection problem, the algorithm iteratively chooses the SNP that distinguishes the most remaining undistinguished pairs until all pairs of haplotype patterns are distinguished.

The Set-Cover-Greedy algorithm takes an input a universal set $\mathcal{U}$ and a colletion $\mathcal{C}$ of subsets of $\mathcal{U}$. Let $\mathcal{R}$ store the uncovered elements in $\mathcal{U}$, which is initially set to be $\mathcal{U}$ because all elements are uncovered at the beginning of the procedure. $\mathcal{C}'$ stores the selected sets and is initialized as an empty set. While $\mathcal{R}$ is not empty, we choose the set $C_i \in \mathcal{C}$ that can cover the most elements in $\mathcal{R}$. C_i would essentially cover the most uncovered elements in $\mathcal{U}$. Then we include C_i in $\mathcal{C}'$ and remove from $\mathcal{R}$ the elements that are covered by it. Repeat this procedure until all elements are covered.

```
Algorithm: SET-COVER-GREEDY($\mathcal{U}, \mathcal{C}$)
1  $\mathcal{R} \leftarrow \mathcal{U}$
2  $\mathcal{C}' \leftarrow \phi$
3  while $\mathcal{R} \neq \phi$ do
4      Select a set $C_i$ from $\mathcal{C}$ that maximizes $|C_i \cap \mathcal{R}|$
5      $C' \leftarrow \mathcal{C}' \cup \{C_i\}$
6      $R \leftarrow \mathcal{R} - C_i$
7  endwhile
8  return $\mathcal{C}'$
```

The subcollection of sets, $\mathcal{C}'$, returned by the SET-COVER-GREEDY algorithm is valid as long as each element of $\mathcal{U}$ is covered by at least one set in $\mathcal{C}$. However, the size of $\mathcal{C}'$ may not always be minimal over all possible valid set covers. For example, let $\mathcal{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $\mathcal{C} = \{C_1, C_2, C_3\}$, where $C_1 = \{2, 3, 4, 5, 6, 7\}$, $C_2 = \{1, 2, 3, 4, 5\}$, and $C_3 = \{5, 6, 7, 8, 9\}$. The greedy algorithm will first pick C_1 since it covers the most elements. After this choice, it will also need to pick C_3 followed by C_2 to form a valid set cover. The resulting $\mathcal{C}'$ is $\{C_1, C_2, C_3\}$. But for this instance, the minimum set cover is $\{C_2, C_3\}$ since all the elements in $\mathcal{U}$ can be covered by C_2 and C_3 without including C_1.

Although the SET-COVER-GREEDY algorithm may not always deliver the minimum set cover, its solution is in fact not too far away from an optimal one. Assume that $\mathcal{C}^*$ is an optimal set cover. Let $|X|$ denote the size (cardinality) of a given set X. We show that $|\mathcal{C}'|$ can be bounded by $|\mathcal{C}^*|$ times a reasonable factor. To calculate the bound, we distribute the covering cost of a selected set to the elements it covers. For the example given in the previous paragraph, the covering order of the elements by the greedy algorithm might be $[2, 3, 4, 5, 6, 7, 8, 9, 1]$ because each of the elements in $\{2, 3, 4, 5, 6, 7\}$ is covered for the first time by C_1 in the first iteration, and then $\{8, 9\}$ by C_3 in the second iteration, and $\{1\}$ by C_2 in the last iteration. Since C_1 covers six uncovered elements, each element in $\{2, 3, 4, 5, 6, 7\}$ shares a cost of 1/6. Similarly, each element in $\{8, 9\}$ shares a cost of 1/2, and the element in $\{1\}$ shares a cost of 1. The covering cost for each element in order is $[1/6, 1/6, 1/6, 1/6, 1/6, 1/6, 1/2, 1/2, 1]$. Summing these costs would get 3, which is the size of the set cover, $\mathcal{C}'$, delivered by the greedy algorithm.

Let $[u_1, u_2, ..., u_{|\mathcal{U}|}]$ be the elements in the order in which they are covered by the Set-Cover-Greedy algorithm. A key observation here is that the cost shared by u_k is at most $|\mathcal{C}^*|/(|\mathcal{U}|-k+1)$ for $1 \leq k \leq |\mathcal{U}|$. In the iteration when u_k is covered, there are at least $|\mathcal{U}| - k + 1$ elements still uncovered, and for sure these uncovered elements can be covered by $\mathcal{C}^*$, which gives an average shared cost of $|\mathcal{C}^*|/(|\mathcal{U}| - k + 1)$. Since the greedy algorithm covers the most uncovered elements, its shared cost for each element in any iteration is the minimum. It follows that the cost shared by u_k is no more than $|\mathcal{C}^*|/(|\mathcal{U}| - k + 1)$. In other words, the covering cost for $[u_1, u_2, ..., u_{|\mathcal{U}|}]$ is no more than $[|\mathcal{C}^*|/|\mathcal{U}|, |\mathcal{C}^*|/(|\mathcal{U}| - 1), \ldots, |\mathcal{C}^*|]$, respectively. Since the size of $\mathcal{C}'$ is the sum of the costs shared by u_k for $1 \leq k \leq |\mathcal{U}|$, we have

$$|\mathcal{C}'| \leq (1 + \frac{1}{2} + \cdots + \frac{1}{|\mathcal{U}|}) \times |\mathcal{C}^*|.$$

The series $1+\frac{1}{2}+\cdots+\frac{1}{|\mathcal{U}|}$ is called the harmonic series. It grows very slowly. For instance, it sums approximately to 2.929 when $|\mathcal{U}| = 10$, to 5.187 when $|\mathcal{U}| = 100$, to 7.485 when $|\mathcal{U}| = 1000$, and to 14.393 when $|\mathcal{U}| = 1,000,000$. As a matter of fact, the harmonic series $1 + \frac{1}{2} + \cdots + \frac{1}{|\mathcal{U}|}$ is bounded by $1 + \int_1^{|\mathcal{U}|} \frac{1}{x}dx$, which yields the bound $\log_e |\mathcal{U}| + 1$. Furthermore, this factor is only a worst-case analysis, and the real approximation ratio could be even better.

4 A Reduction to the Integer-Programming Problem

Linear programming is a general formulation of problems involving maximizing or minimizing a linear objective function subject to certain linear constraints. The following is a simple example.

$$\textbf{Minimize} \quad x_1 + x_2$$

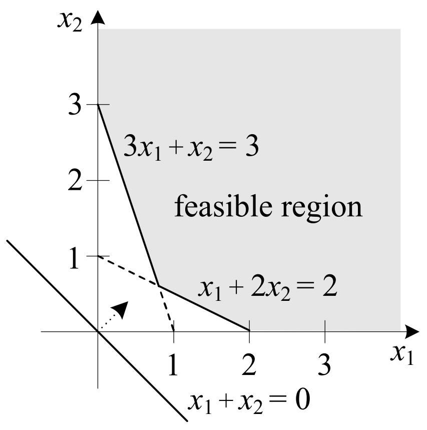

Figure 7: A feasible region defined by the four linear constraints $x_1+2x_2 \geq 2$, $3x_1 + x_2 \geq 3$, $x_1 \geq 0$, and $x_2 \geq 0$.

$$\begin{array}{ll} \textbf{Subject to} & x_1 + 2x_2 \geq 2, \\ & 3x_1 + x_2 \geq 3, \\ & x_1 \geq 0, \\ & x_2 \geq 0. \end{array}$$

Here the linear objective function is $x_1 + x_2$, and there are four linear constraints $x_1 + 2x_2 \geq 2$, $3x_1 + x_2 \geq 3$, $x_1 \geq 0$, and $x_2 \geq 0$. By graphing the constraints on the plane, we observe that the objective function $x_1 + x_2$ (lines with slope -1, see Figure 7) is minimized when $x_1 = \frac{4}{5}$ and $x_2 = \frac{3}{5}$, a corner point where the line $x_1 + 2x_2 = 2$ and the line $3x_1 + x_2 = 3$ intersect.

If we impose the extra constraints that the values of the variables are integers, then the problem is called *integer linear programming* or simply *integer programming*. In the above example, if both x_1 and x_2 are required to be integers, the problem becomes an integer-programming problem.

Now we show how to formulate the tag SNP selection problem as an integer-programming problem. Recall that we are given a haplotype block containing n SNPs and h haplotype patterns. Let us assign a variable x_i for each SNP $S_i \in \mathcal{S}$. Variable x_i is set to be 1 if SNP S_i is selected and set to be 0 otherwise. Define $D(P_j, P_k)$ as the set of SNPs which can distinguish between patterns P_j and P_k, $1 \leq j < k \leq h$. Each pair of patterns must be distinguished by at least one SNP. Therefore, for each set $D(P_j, P_k)$, at least one SNP has to be selected to distinguish between patterns P_j and P_k. The following integer program formulates the tag SNP selection problem whose objective is to minimize the number of selected SNPs.

$$\begin{aligned}
&\textbf{Minimize} && \sum_{i=1}^{n} x_i \\
&\textbf{Subject to} && \sum_{S_i \in D(P_j,P_k)} x_i \geq 1, \quad \text{for all } 1 \leq j < k \leq h, \\
& && x_i = 0 \text{ or } 1, \quad \text{for all } 1 \leq i \leq n.
\end{aligned}$$

In Figure 1, the pair P_1 and P_2 can be distinguished by SNPs S_1, S_2, and S_4. Thus, we have $D(P_1, P_2) = \{S_1, S_2, S_4\}$, which yields the constraint $x_1 + x_2 + x_4 \geq 1$. Similarly, $D(P_1, P_3) = \{S_1, S_3, S_5\}$, $D(P_1, P_4) = \{S_3, S_4\}$, $D(P_2, P_3) = \{S_2, S_3, S_4, S_5\}$, $D(P_2, P_4) = \{S_1, S_2, S_3\}$, and $D(P_3, P_4) = \{S_1, S_4, S_5\}$. By examining all possible pairs of haplotype patterns, we obtain the following integer program for Figure 1.

$$\begin{aligned}
&\textbf{Minimize} && x_1 + x_2 + x_3 + x_4 + x_5 \\
&\textbf{Subject to} && x_1 + x_2 + x_4 \geq 1, \\
& && x_1 + x_3 + x_5 \geq 1, \\
& && x_3 + x_4 \geq 1, \\
& && x_2 + x_3 + x_4 + x_5 \geq 1, \\
& && x_1 + x_2 + x_3 \geq 1, \\
& && x_1 + x_4 + x_5 \geq 1, \\
& && x_1, x_2, x_3, x_4, x_5 = 0 \text{ or } 1.
\end{aligned}$$

In the above integer program, if we set x_1 and x_4 to be 1 and the rest of the x_i's to be 0, then all constraints are satisfied. Consequently, the set of SNPs S_1 and S_4 can distinguish all pairs of haplotype patterns and its size is minimized. However, if we set x_1, x_2, and x_5 to be 1 and set x_3 and x_4 to be 0, then the third constraint $x_3 + x_4 \geq 1$ (for distinguishing P_1 and P_4) is not satisfied. This implies that SNPs S_1, S_2, and S_5 do not form a set of tag SNPs since patterns P_1 and P_4 cannot be distinguished.

All variables x_i's are required to be 0 or 1. Such an integral constraint makes the problem much harder to solve. In fact, both integer programming and 0-1 integer programming have been shown to be NP-hard as has the set-covering problem. It should be noted, however, that without the integral constraint, this integer program becomes a linear program in which variables can be fractional numbers, and fast algorithms, such as the simplex algorithm by George Dantzig, are available for solving it. A general strategy for solving the 0-1 integer-programming problems is thus to replace the integral constraint that each variable must be 0 or 1 by a weaker constraint that each variable be a number in the interval [0,1]. This process is referred to as a *linear-programming relaxation*. After the relaxation, the solution to the relaxed linear program may assign fractional values to the variables. For the above integer program, if we set x_1, x_3, and x_4 to be 0.5 and set x_2 and x_5 to be 0, all the constraints can be satisfied except the last integral constraint. Several techniques, such as randomized rounding, can cope with the linear-programming relaxation to derive heuristic integral solutions for the original unrelaxed integer program. A widely-used idea for rounding a fractional solution is to use their fractions as probabilities for rounding. The heuristic solutions may not be optimal, but often their quality can be assured by a logarithmic approximation ratio.

5 Discussion

In this chapter, we reformulate the tag SNP selection problem as two well-known optimization problems in computer science – the set-covering problem and the integer-programming problem. Both problems are hard to solve, yet efficient approximation algorithms can be used to find their near-optimal solutions.

In reality, some tag SNPs may be missing, and we may fail to distinguish two haplotype patterns due to the ambiguity caused by missing data. To conquer this, either we genotype a larger set of tag SNPs for tolerating missing data or re-genotype some auxiliary tag SNPs to resolve the ambiguity on the fly. We can handle these extensions by modifying the formulations.

It should be noted that selecting tag SNPs *within* a haplotype block is only one of the models for selecting tag SNPs. An alternative is to identify a minimum set of bins, each of which contains highly-correlated SNPs. Such an approach identifies a minimum set of tag SNPs that can represent all other SNPs which might be far apart, whereas the block-based methods considered in this chapter are mainly focused on representing all other SNPs in a short contiguous region. Furthermore, some methods may assume that the number of tag SNPs is specified as an input parameter and identify tag SNPs which can reconstruct the haplotype of an unknown sample with high accuracy.

Acknowledgements

This work was supported in part by NSC grants 97-2221-E-002-097-MY3 and 98-2221-E-002-081-MY3 from the National Science Council, Taiwan. The author would like to thank Yao-Ting Huang and Tandy Warnow for making several valuable comments that improved the presentation.

Bibliographic Notes and Further Reading

This chapter presents two algorithmic approaches for solving the tag SNP selection problem. Readers can refer to algorithm textbooks for more algorithmic details. For instance, the algorithm book (or "The White Book") by Cormen et al. [3] is a comprehensive reference of data structures and algorithms with a solid mathematical and theoretical foundation. The minimum test collection problem was shown to be NP-hard via a reduction from the 3-dimensional matching problem by Garey and Johnson [4].

An early review paper by Brookes [1] provides a good orientation for readers who are not familiar with SNPs. Millions of SNPs have been identified, and these data are now publicly available [5, 6, 8]. The Phase II HapMap has characterized over 3.1 million human SNPs genotyped in 270 individuals from four geographically diverse populations [6]. The dbSNP database is a public-domain archive for a broad collection of SNPs [8].

In a large-scale study of human Chromosome 21, Patil et al. [7] developed a greedy algorithm to partition the haplotypes into 4,135 blocks with 4,563 tag SNPs. It was later refined by Zhang et al. [9, 10] and Chang et al. [2].

References

[1] A.J. Brookes. The Essence of SNPs. *Gene*, 234: 177–86, 1999.

[2] C.-J. Chang, Y.-T. Huang, and K.-M. Chao. A Greedier Approach for Finding Tag SNPs. *Bioinformatics*, 22: 685–691, 2006.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 2009.

[4] M.R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-completeness*. W.H. Freeman and Co., 1979.

[5] D.A. Hinds, L.L. Stuve, G.B. Nilsen, E. Halperin, E. Eskin, D.G. Ballinger, K.A. Frazer, and D.R. Cox. Whole-Genome Patterns of Common DNA Variation in Three Human Populations. *Science*, 307: 1072–1079, 2005.

[6] The International HapMap Consortium. A Second Generation Human Haplotype Map of over 3.1 Million SNPs. *Nature*, 449: 851–861, 2007.

[7] N. Patil, A.J. Berno, D.A. Hinds, W.A. Barrett, J.M. Doshi, C.R. Hacker, C.R. Kautzer, D.H. Lee, C. Marjoribanks, D.P. McDonough, B.T. Nguyen, M.C. Norris, J.B. Sheehan, N. Shen, D. Stern, R.P. Stokowski, D.J. Thomas, M.O. Trulson, K.R. Vyas, K.A. Frazer, S.P. Fodor, and D.R. Cox. Blocks of Limited Haplotype Diversity Revealed by High-Resolution Scanning of Human Chromosome 21. *Science*, 294: 1719–1723, 2001.

[8] 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 Res.*, 29: 308–11, 2001.

[9] K. Zhang, F. Sun, M.S. Waterman, and T. Chen. Haplotype Block Partition with Limited Resources and Applications to Human Chromosome 21 Haplotype Data. *Am. J. Hum. Genet.*, 73: 63–73, 2003.

[10] K. Zhang, Z.S. Qin, J.S. Liu, T. Chen, M.S. Waterman, and F. Sun. Haplotype Block Partition and Tag SNP Selection Using Genotype Data and Their Applications to Association Studies. *Genome Research*, 14: 908–916, 2004.