A Linear Space Algorithm for Haplotype Blocks Partitioning Using Limited Number of Tag SNPs

Yaw-Ling Lin* Wen-Pei Chen Hsiang-Sheng Shin

Dept. of Comput. Sci. and Info. Management,
College of Computing and Informatics, Providence University,
200 Chung Chi Road, Shalu, Taichung, Taiwan 433.
yllin@pu.edu.tw, wenpei.keynes@gmail.com, g9571065@pu.edu.tw

Abstract

The pattern of linkage disequilibrium (LD) plays a central role in genome-wide association studies of identifying genetic variation responsible of common human diseases. *A Single Nucleotide Polymorphism* or SNP is a DNA sequence variation occurring when a single nucleotide in the genome differs between members of species. Recent studies show that the patterns of linkage disequilibrium observed in human chromosome reveal a block-like structure; the high LD regions are called *haplotype blocks*, and furthermore, a small subset of SNPs, called *tag SNPs*, is sufficient to capture the haplotype patterns in each haplotype block. Both Patil [18] and Zhang et al. [24] have proposed algorithms to partition haplotype sample into blocks fully under the circumstances of requiring minimal number of tag SNPs. However, when resources are limited, investigators and biologists may not be able to genotype all the tag SNPs and instead must restrict the number of tag SNPs used in their studies. In this paper, we examine several haplotype block diversity evaluation functions and propose dynamic programming algorithms for haplotype block partitioning with using the limited number of tag SNPs. We implement these algorithms and analyze the chromosome 21 haplotype data given by Patil et al. [18]. When the sample is partitioned into blocks fully, we identify a total of 2,266 blocks and 3,260 tag SNPs which is smaller than those identified by Zhang et al. [24]. We demonstrate that Zhang's algorithm does not find the optimal solution due to ignoring the non-monotonic property of common haplotype evaluation function. The algorithms described have been implemented in the web-based system as the analysis tools for bioinformaticists and geneticists.

Keywords:SNP, Diversity, haplotype block, tag SNP, dynamic programming.

1 Introduction

A Single Nucleotide Polymorphism, or *SNP*, is a small genetic variation, that occur within a person's DNA sequence. It occurs when a single nucleotide is replaced by others. Mutation in DNA is the principle factor resulted in the phenotypic differences among human beings, and SNPs are the most common mutations. They are useful polymorphic markers to investigate genes susceptible to diseases or those related to drug responsiveness.

*This work is supported by grants from the Taichung Veterans General Hospital and Providence University (TCVGH-PU-968110) and in part by the National Science Council (NSC-96-2221-E-126-002) Taichung, Taiwan, Republic of China.

Furthermore, a small subset of SNPs directly influences the quality or quantity of the gene product, and increase a risk to certain diseases and to severe side effect by drugs. Alleles of SNPs that are close together tend to be inherited together. A *haplotype* refers to a set of SNPs found to be statistically associated on a single chromosome. Haplotypes defined by common SNPs have important implications for identifying disease association and human traits [3, 19]. Recent studies have shown that the patterns of linkage disequilibrium (LD) observed in human chromosome reveal a block-like structure [3, 4, 18], and therefore the entire chromosome can be partitioned into high LD regions interspersed by low LD regions. The high LD regions are called *haplotype blocks* and the low LD ones are referred to as *recombination hotspots*. Within a haplotype block, there is little or no recombination that occurs and the SNPs are highly correlated. There are only a few common haplotypes, that account for most of the variation from person to person, in a haplotype blocks. Furthermore, each haplotype block, in which the genome is largely made up of regions of low diversity, can be characterized by a small number of SNPs, which are referred to as *tag SNPs* [13]. Tag SNPs are aimed at characterizing candidate genes avoiding redundancies in genotyping. Most of the tag SNP selection strategies are haplotype based. The aim is to identify a minimal subset of SNPs that can characterize the most common haplotypes [18, 24]. The characteristics of haplotype blocks and tag SNPs are very important and useful for medicine and therapy. Studying on haplotype blocks and tag SNPs not only decrease the cost for detecting inherited diseases but also has many contributions for classifying the race of human and researching on species evolution.

Diversity functions

Several operational definitions have been used to identify haplotype-block structures, including LD-based [4, 22], recombination-based [12, 23], information-complexity-based [1, 14, 6] and diversity-based [2, 18, 25] methods. The result of block partition and the meaning of each haplotype block may be different by using different measuring formula. For simplicity, haplotype samples can be converted into haplotype matrices by assigned major alleles to 0 and minor alleles to 1.

Definition 1 (haplotype block diversity) *Given an interval $[i, j]$ of a haplotype matrix A, a* diversity function, $\delta : [i, j] \to \delta(i, j) \in \mathbb{R}$ *is an evaluation function measuring the diversity of the submatrix $A(i, j)$.*

Haplotype blocks are the genome regions with high LD, thus it implies that no matter what kinds of haplotype block definition we used, the patterns of haplotype within the block will be small, and the diversity of the block will be low. In terms of diversity functions, the block selection problem can be viewed as finding a segmentation of given haplotype matrix such that the diversities of chosen blocks satisfy certain value constraint. Following we examine several haplotype block diversity evaluation functions. Given an $m \times n$ haplotype matrix A, a *block* $S(i, j)$ (i, j are the block boundaries) of matrix A is viewed as m haplotype strings; they are partitioned into groups by merging identical haplotype strings into the same group. The probability p_i of each haplotype pattern s_i, is defined accordingly such that $\sum p_i = 1$.

As an example, Li [15] proposes a diversity formula defined by

$$\delta_D(S) = 1 - \sum_{s_i \in S} p_i^2. \quad (1)$$

Note that $\delta_D(S)$ is the probability that two haplotype strings chosen at random from S are different from each other. Other measurements of diversity can be obtained by choosing different diversity function; for example, to measure the information-complexity one can choose the information entropy (negative-log) function [1, 14, 6]:

$$\delta_E(S) = -\sum_{s_i \in S} p_i \log p_i. \quad (2)$$

In the literatures [18, 24, 25], Patil and Zhang et al. define a haplotype block as a region where at least 80% of observed haplotypes within a block must be common haplotype. As the same definition of common haplotype in the literatures, the coverage of common haplotype of the block can be formulated as a form of diversity:

$$\delta_C(S) = 1 - \frac{\sum_{s_i \in C} p_i}{\sum_{s_i \in U} p_i} = \frac{\sum_{s_i \in M} \frac{1}{m}}{\sum_{s_i \in U} p_i}. \quad (3)$$

Here U denotes the unambiguous haplotypes, C denotes the common haplotypes, and M denotes the singleton haplotypes. In other words, Patil et al. require that $\delta_C(S) \leq 20\%$.

Some studies [4, 28, 26] propose the haplotype block definition based on LD measure D'; however, there is no consensus definition for it so far. Zhang and Jin [28] define a haplotype block as a region in which all pair-wise $|D'|$ values are not lower than a threshold α. Let S denote a haplotype interval $[i, j]$. We define the diversity as the complement of minimal $|D'|$ of S. By the definition, S is a haplotype block if its diversity is lower than $1 - \alpha$.

$$\delta_{L1}(S) = 1 - \min\{(|D'_{i'j'}|) | i \leq i' < j' \leq j\}. \quad (4)$$

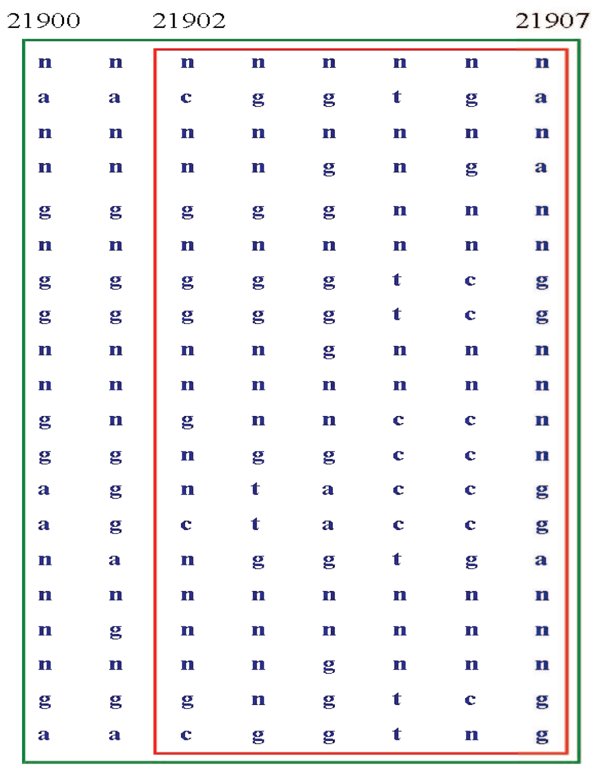
Zhang et al. [26] also propose the other definition for haplotype block; they require at least α proportion of SNP pairs having strong LD (the pairwise $|D'|$ greater than a threshold) in each block. Similarly, we can use the diversity to redefine the function. We define the diversity as the proportion of SNP pairs that do not have strong LD. Therefore, haplotype interval S is a feasible haplotype block if its diversity is smaller than a threshold. We can use the following diversity function to calculate the diversity of S. Here $N(i, j)$ denotes the number of SNP pairs that do not have strong LD in the interval $[i, j]$.

$$\delta_{L2}(S) = \frac{N(i,j)}{\binom{(j-i)+1}{2}} = \frac{N(i,j)}{\frac{1}{2}[(j-i)^2 + j - i]}. \quad (5)$$

Diversity measurement usually reflects the activity of recombination events occurred during the evolutionary process. Generally, haplotype blocks with low diversity indicates conserved regions of genome.

Definition 2 (monotonic diversity) *A diversity function δ is said to be* monotonic *if, for any haplotype block (interval) $I = [i, j]$ of A, it follows that $\delta(i', j') \leq \delta(i, j)$ whenever $[i', j'] \subset [i, j]$; that is, the diversity of any subinterval of I is always no larger than the diversity of I.*

It is easily verified that many diversity functions, including the diversity functions (1) and (2), are monotonic. However, the evaluative function of common haplotype proposed by Patil et al. [18] does not satisfy the monotonic property when the haplotype sample has missing data. For example, in Figure 1, it is a small portion of human chromosome 21 haplotype sample provided by Patil, here n denotes the missing data. We can observe that the coverage of common haplotype of interval [21900,21907] is 9/10, more than 80%. Therefore, according to the definition proposed by Patil et al., it is a feasible haplotype block. On the other hand, the coverage of common haplotype of interval [21902,21907] is 3/7, less than 80%, so it is not a feasible haplotype block. Note that interval [21900,21907] and interval [21902,21907] are two intervals terminated at the same SNP locus, and interval [21900,21907] which has more SNPs is a feasible haplotype block but interval [21902,21907] is not.



21900		21902					21907
n	n	n	n	n	n	n	n
a	a	c	g	g	t	g	a
n	n	n	n	n	n	n	n
n	n	n	n	g	n	g	a
g	g	g	g	g	n	n	n
n	n	n	n	n	n	n	n
g	g	g	g	g	t	c	g
g	g	g	g	g	t	c	g
n	n	n	n	g	n	n	n
n	n	n	n	n	n	n	n
g	n	g	n	n	c	c	n
g	g	n	g	g	c	c	n
a	g	n	t	a	c	c	g
a	g	c	t	a	c	c	g
n	a	n	g	g	t	g	a
n	n	n	n	n	n	n	n
n	g	n	n	n	n	n	n
n	n	n	n	g	n	n	n
g	g	g	n	g	t	c	g
a	a	c	g	g	t	n	g

Figure 1: The evaluative function of common haplotype does not satisfy the monotomic property when the haplotype sample has missing data.

Tag SNPs can capture most of the haplotype diversity in the blocks, and therefore could potentially capture most of the information for association between a trait and the SNP marker loci. We can figure out the diversity and features of each haplotype block easily and economically by using tag SNPs. For these reasons, we want to define the haplotype structure by using tag SNPs as fewer as possible. In previous studies, Patil et al. [18] defined a haplotype block as a region in which a fraction of percent or more of all the observed haplotypes are represented at least n times or at a given threshold in the sample. They applied the optimization criteria outlined by Zhang et al. [24, 25] and describe a general algorithm that defines block boundaries in a way that minimizes the number of tag SNPs that are required to uniquely distinguish a certain percentage of all the haplotypes in a region. Patil et al. have developed a greedy algorithm and identified a total of 4,563 tag SNPs and a total of 4,135 blocks to define the haplotype structure of human chromosome 21. In each block they required at least 80% of haplotype must be represented more than once in the block. In addition, Zhang et al. [24] used a dynamic programming approach to reduce the numbers of blocks and tag SNPs to 2,575 and 3,582, respectively.

Patil and Zhang's algorithms both partition the haplotype sample in to blocks fully under the circumstances of requiring minimal number of tag SNPs. However, when resources are limited, investigators and biologists may not be able to genotype all the tag SNPs and instead must restrict the number of tag SNPs used in their studies. In this paper, we propose several dynamic programming algorithms concerning haplotype block partition problems.

Problem 1 (longest-blocks-t-tags) *Given a haplotype matrix A and a diversity upper* *limit D, we wish to find a list of feasible blocks whose total tag SNP numbers is less than t such that the total length is maximized. That is, output the set $S = \{B_1, B_2, \ldots, B_{|S|}\}$ such that $(\forall B_i \in S)(\delta(B_i) \leq D)$ and $\sum tag(B_i) \leq t$; $tag(B_i)$ denote the number of tag SNPs required for block B_i, so that $|B_1| + |B_2| + \cdots + |B_{|S|}|$ is maximized.*

In our previous study [16], we show that assuming all of the feasible blocks and tag SNPs required for each block have been preprocessed, the longest-blocks-t-tags problem can be solved in $O(tL)$ time and $O(tn)$ space, here L denotes the total number of feasible blocks and n represents the total number of SNPs. In this paper, we propose a linear space algorithm for the same problem.

2 Method

In this section, we show dynamic programming algorithms to partition haplotype blocks with constraints on diversity and tag SNP number. That is, we want to find the longest segmentation S consist of some blocks with the diversity of each block is less than an upper limit D and the total number of tag SNPs required for these blocks does not exceed a specific number t. This problem had been discussed by Zhang et al. [27], but here we propose a more time-efficiency and linear space algorithm to solve the problem. The problem definition is shown in Problem 1.

Our algorithm begin with the preprocessing of the *farthest site* (good partner) [17] for each SNP marker. According to the haplotype block definition in Patil [18] and the discussion in previous section, we know that the common haplotypes coverage evaluation function is not monotonic. That is, for each SNP marker j there will be a left farthest marker i so that $[i, j]$ is the longest haplotype block among all feasible blocks that terminated at site j, but some interval $[i', j] \subset [i, j]$ are not feasible blocks. Thus, before the computation of finding the longest segmentation using limited tag SNPs, we need to preprocess the set of left good partners L_i for each SNP marker i, $L_i = \{x | [x, i] \text{ is a feasible haplotype block}\}$. We can use the event list [17, 20] which is computed by techniques of suffix tree [7, 21] and lowest common ancestor (LCA) [9] to find the L_i for each SNP locus i in $O(mn)$, linear proportional to the input size of haplotype matrix, time. Furthermore, we also need to pre-compute the number of tag SNPs required for each feasible haplotype block. This problem can be solve by using the algorithm described in section 2.1. The time complexity of the algorithm is $O(2^{t_0} L)$; here L is the number of all feasible blocks, and t_0 is the maximum number of tag SNPs required among all feasible blocks. In our experience, the preprocessing will need much time, however, we just need to compute it once.

2.1 Tag SNP Selection Algorithms

According to the haplotype block definition defined by Patil et al. [18], they require that at least $\rho = 80\%$ of unambiguous haplotypes are represented more than once. Using the same criteria as in Patil [18], for each block, we want to minimize the number of SNPs that distinguish uniquely at least ρ percentage of the unambiguous haplotypes in the block. Those SNPs can be thought of as a signature of the haplotype block partition.

It is interesting to note that, although the num-

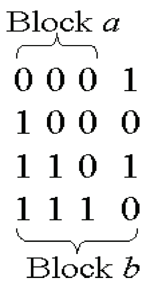


Figure 2: An example of a longer block but required few tag SNPs.

ber of tag SNPs required increases as the length of haplotype block increases in general, there are exceptions to the case. As an example shown in Figure 2, the block a which consisted of 3 SNP markers needs 3 tag SNPs to distinguish each haplotype uniquely, but the block b which consisted of 4 SNP markers just needs 2 tag SNPs (*i.e.* column 2 and column 4.)

The problem of finding the minimum number of tag SNPs within a block to uniquely distinguish all the haplotypes is known as the MINIMUM TEST SET problem, which has been proven to be NP-Complete [5]. Thus, there are no polynomial time algorithms that guarantees to find the optimal solution for any input. Although, some approximation algorithms such as the greedy algorithm have been proposed but may fail to find the optimal solution [11, 30, 29]. In order to find the optimal solution, we adopt the brute force method to find tag SNPs within a block. Our strategy for selecting the tag SNPs in haplotype blocks is as the following. First, the common haplotypes are grouped into k distinct patterns by merging the compatible haplotypes in each block. After the missing data are assigned in each group, we decide the least number of tag SNPs required based on the least number of haplotype groups needed to be distinguished such that haplotypes in these groups contain at least ρ percentage of the unambiguous haplotypes in the block. Finally, we select a loci set consisted of minimum number of SNPs on the haplotypes such that at least ρ percentage of the unambiguous haplotypes can be uniquely distinguished; the exhaustive searching method can be used very efficiently since the number of tag SNPs needed for each block is usually modest in the situation. The exhaustive searching algorithm shown in Figure 3 enumerates next t-combination in lexicographic order to generate the next candidate tag SNP loci set until each pattern can be uniquely distinguish.

2.2 Longest Blocks Partition Using Limited Number of Tag SNPs

In our previous study [16], given an $m \times n$ haplotype matrix, after the preprocessing of left farthest site for each SNP marker and tag SNPs required for each feasible blocks, we show that finding the longest blocks covered by t tag SNPs can be found in $O(tL)$(or $O(tnl)$) time; here t denotes the number of tag SNPs used, and $L = \sum_{i=1}^{n} |L_i|$ denotes the total number of feasible blocks. The results are summarized as following.

Let $f(i,t)$ define the length of the longest segmentation of haplotype matrix $A(1,i)$ covered by t tag SNPs, and $tag(i,j)$ denote the number of tag SNPs required for block which is bounded by sites i and j. It is interesting to note that $f(i,t)$ can be computed by the following recurrence relation:

$$\begin{array}{rl} f(1,0) = & \begin{cases} 0 & \text{if } tag(1,1) > 0 \\ 1 & \text{if } tag(1,1) = 0 \end{cases} \\ f(1,t) = & 1 \text{ if } t \geq 1 \\ f(i,t) = & -\infty \text{ if } t < 0 \end{array}$$

$f(i,t) =$

```
FindTag($B, \rho$)        ▷ Find the number of tag SNPs required for haplotype block $B$
                           such that $\rho$ percentage of unambiguous haplotypes in $B$ can be
                           distinguished uniquely.
Input: A percentage $\rho$ and haplotype block $B$ with unambiguous haplotype pattern
       $P = \{p_1, p_2, \ldots, p_k\}$; the haplotypes number in each pattern is $n_1, n_2, \ldots, n_k$.
Output: The tag SNPs required for haplotype block $B$.
1 Sort $k$ haplotype patterns in $P = \langle p_1, p_2, \ldots, p_k \rangle$.
     ▷ $p_i$'s are listed in decreasing order of the number of haplotype strings.
2 $U = \rho \cdot (\sum_{i=1}^{k} n_i)$      ▷ $U$ is the number of $\rho$ percentage of unambiguous haplotypes.
3 Find the minimum number $g$ such that $\sum_{i=1}^{g} n_i \geq U$
4 $t \leftarrow \lceil \log_2 g \rceil$      ▷ $t$ is the minimum number of tag SNPs required.
5 for $i \leftarrow 1$ to $t-1$ do      ▷ initiate the tag SNP loci set $\{a[1], a[2], \ldots, a[t]\}$.
6     $a[i] \leftarrow i$
7 $a[t] \leftarrow a[t-1]$; $h \leftarrow 0$
8 while $h < U$ do      ▷ $h$ is the total haplotype strings that can be distinguished.
9         $i \leftarrow t$      ▷ generate the next $t$-combination in lexicographic order.
10        while $a[i] = l - t + i$ do      ▷ $l$ is the length of haplotype string .
11              $i \leftarrow i - 1$
12        if $i = 0$
13           $t \leftarrow t + 1$
14           for $i \leftarrow 1$ to $t$ do
15                $a[i] \leftarrow i$
16        else
17           $a[i] \leftarrow a[i] + 1$
18           for $j \leftarrow i + 1$ to $t$ do
19                $a[j] \leftarrow a[i] + j - i$
20        $h \leftarrow \sum n_x$, $x \in \{x | p_x$ is the haplotype that can be distinguished by tag SNP.$\}$
21 return $t$
```

Figure 3: The exhaustive searching algorithm for tag SNPs selection.

$$\max \left\{ \begin{array}{l} f(i-1, t) \\ \max_{k \in L_i} \left\{ \begin{array}{l} (i-k+1)+ \\ f(k-1, t - tag(k,i)) \end{array} \right\} \end{array} \right. \tag{6}$$

The maximized segmentation S between sites 1 and i will have two cases, either the site i is included in the last block of S or not. If site i is not included in the last block of S, it will find S between sites 1 and $i-1$, otherwise there will exist a site $k \in L_i$ such that $[k, i]$ is the last block of S. In the latter case, the tag SNPs required for the bock $[k, i]$ is $tag(k, i)$ which has been calculated in preprocessing, so we can find other blocks which are covered by other $t - tag(k, i)$ tag SNPs between sites 1 and $k - 1$.

Note that if l is the average number of $|L_i|$ for each SNP marker i, $f(i, t)$ will be able to be determined in $O(l)$ time suppose $f(1..(i-1), t)$'s and $f(\cdot, 1..(t-1))$'s being ready. It follows that $f(\cdot, t)$'s can be calculated from $f(\cdot, 1..(t-1))$'s totally in $O(nl)$ time. Thus a computation ordering from $f(\cdot, 1)$'s, $f(\cdot, 2)$'s, …, to $f(\cdot, t)$'s leads to the following result.

Theorem 1 (longest-blocks-t-tags) *Given a haplotype matrix A, a diversity upper limit D and the number of tag SNP t, find a segmentation S consisted of k feasible blocks such that $(\forall_i)(\delta(B_i) \leq D)$ and $\sum tag(B_i) \leq t$, so that the total length of S is maximized can be done in $O(tnl)$ time after the preprocessing of L_i and*

$tag(k,i)$'s, $k \in L_i$, for each SNP marker i.

The recurrence relation has been used to develop a dynamic programming algorithm, the algorithm also can be viewed in [16]. We must point out that the algorithm also can be used to partition haplotypes into blocks with a fixed genome coverage, that has been discussed by Zhang et al. [27]. We can define the length of block $B = [i,j]$ as the actual length of the genome spanning form the i-th SNP to the j-th SNP. Given an $m \times n$ haplotype matrix of a chromosome of length G and a percentage $\alpha \leq 1$, find a segmentation $S = \{B_1, B_2, \ldots, B_{|S|}\}$ with the total length $\ell(S) \geq \alpha G$ such that the total tag SNPs required is minimized; the recurrence relation $f(1,n,t)$ is a block length evaluation function, given n, we can increase the value of $f(1,n,t)$ by increasing the value of t until $f(1,n,t) \geq \alpha G$. After finding the t such that $f(1,n,t) \geq \alpha G$, we can get the boundaries of each block by tracing back.

2.3 Linear Space Algorithms

In section 2.2, given an $m \times n$ haplotype matrix A, a diversity upper limit D, and a specific number of tag SNPs t, we propose an $O(tnl)$ (or $O(tL)$) time algorithm for finding the longest segmentation S containing blocks with the diversity of each block is no greater than D and the total tag SNPs number required for these blocks does not exceed t. We apply the dynamic programming technique to general case and obtain the following recurrence relation. Note that $f(i,j,t)$ define the length of the longest segmentation of haplotype matrix $A(i,j)$ covered by t tag SNPs.

$$\begin{array}{rl} f(i,i,0) = & \begin{cases} 0 & \text{if } tag(i,i) > 0 \\ 1 & \text{if } tag(i,i) = 0 \end{cases} \\ f(i,i,t) = & 1 \text{ if } t \geq 1 \\ f(i,j,t) = & -\infty \text{ if } t < 0 \\ f(i,j,t) = & -\infty \text{ if } j < i \end{array}$$

$$f(i,j,t) = \max \begin{cases} f(i,j-1,t) \\ \max_{k \in L_j} \begin{Bmatrix} (j-k+1)+ \\ f(i,k-1,t-tag(k,j)) \end{Bmatrix} \end{cases} \quad (7)$$

Lemma 1 *Given an $m \times n$ haplotype matrix A, a diversity upper limit D, and the number of tag SNPs t, for an any constrained interval $[i,j]$, $1 \leq i \leq j \leq n$, find a segmentation consisted of k feasible blocks such that $(\forall_i)(\delta(B_i) \leq D)$ and $\sum tag(B_i) \leq t$, so that the total length of S is maximized can be done in $O(|j-i|lt)$ time after the preprocessing of L_i and $tag(k,i)$'s, $k \in L_i$, for each SNP marker i.*

Clearly, the space complexity of our algorithm shown in [16] is $O(tn)$ for the retrieval of the boundaries of each block by tracking back. Such space requirement results in practiced difficulties in cases where t and n become too large. For the reason, we need to have a space-efficiency algorithm to solve the problem. Using the similar concept as in [10], we find a cut-point x^* to divide n SNP sites into two parts, n_1 and n_2, and use t^* tag SNPs for n_1 and the other $t - t^*$ tag SNPs for n_2 such that the total size of blocks covered by t^* tags in n_1 and blocks covered by $t - t^*$ tags in n_2 is maximized. We obtain the following recurrence relation.

$$f(i,j,t) = f(i,x^*,t^*) + f(x^*+1,j,t-t^*) \quad (8)$$

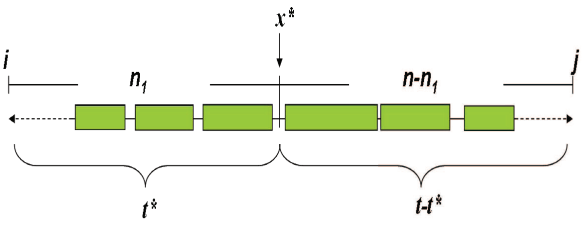


Figure 4: Illustration of the idea of recurrence $f(i, j, t)$.

The idea behind the recurrence relation is illustrated at Figure 4. Note that in order to make the total size of blocks tagged by t^* SNPs and $t - t^*$ SNPs maximized, we can not assign a half of t to t^* directly, because in some case, the $\lfloor \frac{t}{2} \rfloor$-th and the $(\lfloor \frac{t}{2} \rfloor + 1)$-th SNP will be used to tag the same block which is the member of the longest segmentation. If we use the first to the $\lfloor \frac{t}{2} \rfloor$-th SNPs to tag the blocks in n_1, and use the $(\lfloor \frac{t}{2} \rfloor + 1)$-th to the t-th SNPs to tag the blocks in n_2, we will not get the longest segmentation on n SNPs. In general case, there will be many pairs of t^* and x^* solutions which fit our request. For the purpose of time efficiency, we want to make x^* and t^* to approach the half of n and t as far as possible. Let t_0 denote the maximum number of tag SNPs required among all feasible blocks. In order to find the appropriate value of t^* and x^*, we can examine t^* in t_0 continuous possible values, $\lfloor \frac{t}{2} \rfloor - \lfloor \frac{t_0}{2} \rfloor \le t^* \le \lfloor \frac{t}{2} \rfloor + \lceil \frac{t_0}{2} \rceil$, and examine x^* in all SNPs loci for each selection of t^*. Since t_0 is small in general case, we can find the t^* and the x^* quickly. After finding the appropriate values of t^* and x^*, we can execute the steps recursively to partition the original problem to two subproblems repeatedly. Until $t \le t_0$, we just use the dynamic programming algorithm shown in [16] to solve each subproblem. Here we can trace back to output the boundaries of each block. The algorithm is shown in Figure 5.

Theorem 2 (longest-blocks-t-tags) *Assume the maximum number of tag SNPs required among all feasible blocks, t_0, is a fixed constant. Given a haplotype matrix A, a diversity upper limit D, and the number of tag SNPs t, find a segmentation S consisted of k feasible blocks such that $(\forall_i)(\delta(B_i) \le D)$ and $\sum tag(B_i) \le t$, so that the total length of S is maximized can be done in $O(tnl)$ time and using linear space after the preprocessing of R_i, L_i and $tag(k, i)$'s, $k \in L_i$, for each SNP marker i.*

Proof. We propose an $O(tnl)$ time algorithm, ListTag(i, j, T), shown in Figure 5. The correctness of the algorithm can be shown as follow. When $T \le t_0$, the algorithm uses the dynamic programming algorithm shown in [16] to compute $f(i, j, T)$ and output the boundaries of each blocks by tracing back. If $T > t_0$, we must find a x^* between sites i and j, and find a t^* between 0 and T. Subsequently, we can find a maximum segmentation S_1 tagged by t^* SNPs between sites i and x^* and a maximum segmentation S_2 tagged by $T - t^*$ SNPs between sites $x^* + 1$ and j so that the total size of S_1 and S_2 is equal to S which is the maximum segmentation tagged by T SNPs between sites i and j. In general case, there will be many pairs of t^* and x^* solutions which fit our request. For the purpose of time efficiency, we want to make x^* and t^* to approach the half of n and t as far as possible. In order to find the appropriate values of t^* and x^*, we can examine t^* in t_0 continuous possible values, $\lfloor \frac{t}{2} \rfloor - \lfloor \frac{t_0}{2} \rfloor \le t^* \le \lfloor \frac{t}{2} \rfloor + \lceil \frac{t_0}{2} \rceil$, and examine x^* in all SNPs loci for each selection of t^*. In the case of $T > t_0$, we first compute $f(i, x, t)$'s and $f(x + 1, j, T - t)$'s, $i \le x \le j - 1$, $\lfloor \frac{t}{2} \rfloor - \lceil \frac{t_0}{2} \rceil \le t \le \lfloor \frac{t}{2} \rfloor + \lceil \frac{t_0}{2} \rceil$, and put the result into two two dimensional arrays A and B. Note that the computation of $f(x+1, j, T-t)$'s uses the similar idea with opposite direction as the computation of $f(i, x, t)$'s; we use the right good partners R_i, $R_i = \{x | [i, x]$ is a feasible haplotype block$\}$, to compute $f(x + 1, j, T - t)$'s from $x = j - 1$ down to $x = i$. Then we can find x^* and t^* such that the total length of blocks tagged by t^* tag SNPs between sites i and x^* and blocks tagged by $T - t^*$ tag SNPs between sites $x^* + 1$ and j is maximized. That is, we can find x^* and t^* such that $f(i, x^*, t^*) + f(x^* + 1, j, T - t^*)$ is maximized. Next steps we use recursive algorithm ListTag(i, x^*, t^*) and ListTag$(x^* + 1, j, T - t^*)$ to list blocks tagged by t^* SNPs in $[i, x^*]$ and blocks tagged by $T - t^*$ SNPs in $[x^* + 1, j]$.

In the algorithm, we use five global data structures involving arrays E, F, S, A, and B. Arrays E and F are used to store the good partner points L_i and R_i for each SNP marker i, and array S is used to store the tag SNPs required for each feasible blocks. The data in E, F and S arrays were calculated in preprocessing, and the space of each array is L, the number of all feasible blocks. In addition, we use two dimensional array A for computing $f(i, x, 0..\lfloor \frac{T}{2} \rfloor + \lceil \frac{t_0}{2} \rceil)$'s and B for computing $f(x + 1, j, 0..\lfloor \frac{T}{2} \rfloor + \lceil \frac{t_0}{2} \rceil)$'s. Note that the computation of $f(i, j, t)$ will compare the values of $f(i, k - 1, t - tag(k, j))$'s, $k \in L_j$, and $f(i, j - 1, t)$.

```
LISTAG($i, j, T$)        ▷ List blocks covered by $T$ tag SNPs in $[i, j]$ with maximized total length.
Input: Interval $[i, j]$ and number of tag SNPs $T$.
Output: The boundaries of blocks covered by $T$ tag SNPs.
Global variable: $E$, $F$, $S$, $A$, $B$.     ▷ $E$ and $F$ are used to store the good partner pointers $L_i$ and
                 $R_i$ which have been preprocessed dependent on diversity constraints $D$,
                 $S$ is used to store the tag SNPs required for each feasible blocks,
                 two dimensional arrays $A$ and $B$ are global temporary working storages.
1 if $T \le t_0$ then
2   for $t \leftarrow 0$ to $T$ do
3       for $x \leftarrow i$ to $j$ do
4           Directly compute $A[t, x] = f(i, x, t)$ according to recurrence relation 7.
5   Trace back on $A$ array to output the boundaries of blocks covered by $T$ tag SNPs.
6   return
7 for $t \leftarrow 0$ to $\lfloor \frac{T}{2} \rfloor + \lceil \frac{t_0}{2} \rceil$ do
    ▷ Compute $A[t \; mod \; (t_0+1), x] = f(i, x, t)$, $\forall x \in [i..j-1]$, $t \in [\lfloor \frac{T}{2} \rfloor - \lfloor \frac{t_0}{2} \rfloor .. \lfloor \frac{T}{2} \rfloor + \lceil \frac{t_0}{2} \rceil]$.
8     for $x \leftarrow i$ to $j-1$ do
9         Compute $A[t \; mod \; (t_0+1), x] = f(i, x, t)$ by formula 7.
10 for $t \leftarrow 0$ to $\lfloor \frac{T}{2} \rfloor + \lceil \frac{t_0}{2} \rceil$ do
    ▷ Compute $B[t \; mod \; (t_0+1), x] = f(x+1, j, t)$, $\forall x \in [i..j-1]$, $t \in [\lfloor \frac{T}{2} \rfloor - \lfloor \frac{t_0}{2} \rfloor .. \lfloor \frac{T}{2} \rfloor + \lceil \frac{t_0}{2} \rceil]$.
11     for $x \leftarrow j$ down to $i+1$ do
12         Compute $B[t \; mod \; (t_0+1), x] = f(x, j, t)$ by formula 7 with opposite direction.
13 Find $(x^*, t^*) = \arg\max_{i \le x \le j, \lfloor \frac{T}{2} \rfloor - \lfloor \frac{t_0}{2} \rfloor \le t \le \lfloor \frac{T}{2} \rfloor + \lceil \frac{t_0}{2} \rceil} \{A[t, x] + B[T-t, x]\}$.
    ▷ Pointer back tracking to find the $x^*$, and $t^*$
14 LISTAG($i, x^*, t^*$)              ▷ recursive call to report blocks in interval $[i, x^*]$ with $t^*$ tags.
15 LISTAG($x^*+1, j, T-t^*$)        ▷ recursive call to report blocks in interval $[x^*, j]$ with $T-t^*$ tags.
```

Figure 5: The $O(nlt)$ time and linear space algorithm for haplotype blocking with constraints on diversity and the number of tag SNPs.

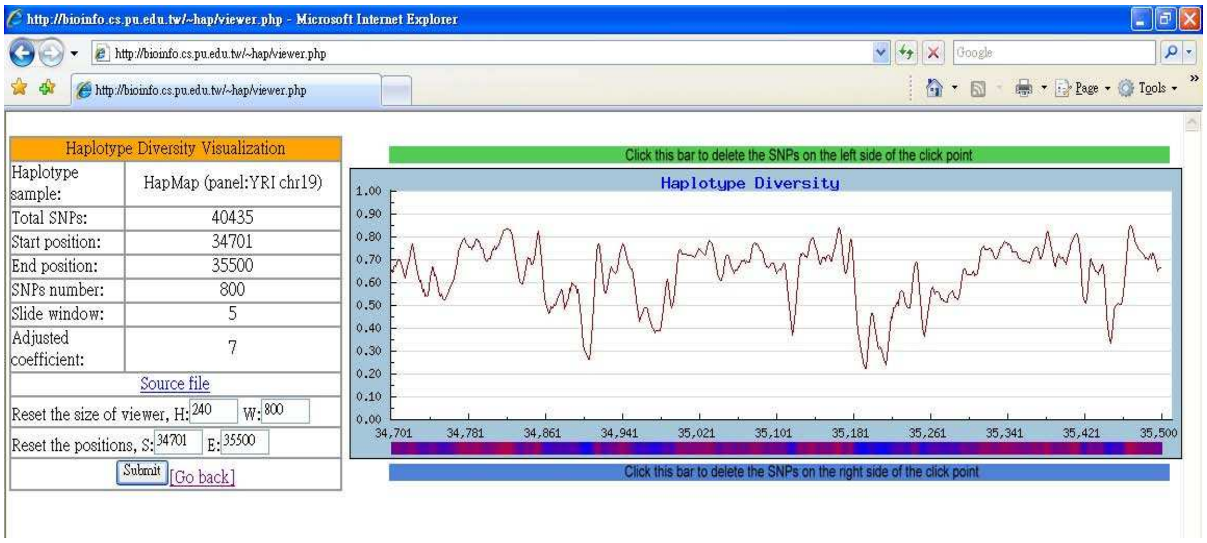
Therefore, if t_0 denotes the maximum $tag(k, j)$, the maximum number of tag SNPs required among all feasible blocks, we at most need to store the values of $f(\cdot, \cdot, (t-t_0)..(t-1))$ and $f(i, j-1, t)$ while compute the value of $f(i, j, t)$. In our experience, we know that the t_0 will be equal to 8 at most as an example of Patil's haplotype data. It means that the space of two dimensional arrays A and B is $t_0 \times n$, so the space complexity for the algorithm is $O(L + t_0 n)$. Since t_0 is generally a constant and $L > n$ in most practical cases, we can prove the space used by the algorithm is $O(L + n)$.

The time complexity of the algorithm is $O(nlt)$ as shown in the following by induction. Let $T(n, t)$ denotes the time needed for LISTAG$(1, n, t)$. Assume that $T(n', t') \le c_2 n' l t'$ for all $n' < n$, $t' < t$. According to the algorithm, we have:

$$T(n,t) = \underbrace{c_1 nlt}_{\text{line 7-13}} + \underbrace{T(n_1, t^*)}_{\text{line 14}} + \underbrace{T(n-n_1, t-t^*)}_{\text{line 15}}$$

By induction,
$T(n, t)$
$\le c_1 nlt + c_2 n_1 l t^* + c_2 (n - n_1) l (t - t^*)$
$\le l(c_2 nt + c_1 nt + 2c_2 n_1 t^* - c_2 n_1 t - c_2 n t^*)$
$\le l[c_2 nt + (\frac{5}{3} c_2 n_1 t^* - c_2 n_1 t)$
$\quad +(c_1 nt + \frac{1}{3} c_2 n_1 t^* - c_2 n t^*)]$
$\le l[inc_2 nt + c_2 n_1 (\frac{5}{3} t^* - t) + (c_1 nt + \frac{1}{3} c_2 nt - c_2 n t^*)]$
$\le l\{c_2 nt + c_2 n_1 [\frac{5}{3}(\lfloor \frac{t}{2} \rfloor + \lceil \frac{t_0}{2} \rceil) - t]$
$\quad +(c_1 + \frac{1}{3} c_2) nt - c_2 n (\lfloor \frac{t}{2} \rfloor - \lceil \frac{t_0}{2} \rceil)\}$
$\le l\{c_2 nt + c_2 n_1 [\frac{5}{3}(\frac{t}{2} + \frac{t_0}{2} + 1) - t]$
$\quad +(c_1 + \frac{1}{3} c_2) nt - c_2 n (\frac{t}{2} - \frac{t_0}{2} - 1)\}$
$\le l[c_2 nt + c_2 n_1 (\frac{5}{6} t_0 + \frac{5}{3} - \frac{1}{6} t) + (c_1 + \frac{1}{3} c_2) nt$
$\quad -\frac{1}{2} c_2 nt + c_2 n (\frac{t_0}{2} + 1)]$
$\le l[c_2 nt + c_2 n_1 (\frac{5}{6} t_0 + \frac{5}{3} - \frac{1}{6} t) + \frac{4}{10} c_2 nt - \frac{1}{2} c_2 nt$
$\quad +c_2 n (\frac{t_0}{2} + 1)]$(Let $c_1 = \frac{1}{15} c_2$)
$\le l[c_2 nt + c_2 n_1 (\frac{5}{6} t_0 + \frac{5}{3} - \frac{1}{6} t) + c_2 n (\frac{t_0}{2} + 1 - \frac{t}{10})]$
$\le c_2 nlt$

Let $t \ge 5t_0 + 10$, the above inequality will come into existence, so we can prove the time complexity of the algorithm is $O(nlt)$. □


http://bioinfo.cs.pu.edu.tw/~hap/viewer.php - Microsoft Internet Explorer
Click this bar to delete the SNPs on the left side of the click point
Haplotype Diversity
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
34,701
34,781
34,861
34,941
35,021
35,101
35,181
35,261
35,341
35,421
35,500
Click this bar to delete the SNPs on the right side of the click point

Haplotype Diversity Visualization	
Haplotype sample:	HapMap (panel:YRI chr19)
Total SNPs:	40435
Start position:	34701
End position:	35500
SNPs number:	800
Slide window:	5
Adjusted coefficient:	7
Source file	
Reset the size of viewer, H: 240 W: 800	
Reset the positions, S: 34701 E: 35500	
Submit [Go back]	

Figure 6: A view of the haplotype sample – HapMap phase II haplotypes, chr19 in the YRI panel – by diversity visualization tool.

3 Experimental Results and Diversity Visualizations

In response to needs for analysis and observation of human genome variation, we have established a website and apply the algorithms described in this paper to provide several analysis tools for bioinformatic and genetic researchers. The web-based system consists of a list of PHP and PERL CGI-scripts together with several C programs for selecting haplotype blocks and analyzing the haplotype diversity. We also collect the haplotype data of human chromosome 21 from Patil et al. [18] and download haplotypes for all the autosomes from phase II data of HapMap [8] so that the bioinformatic researchers can use the data to evaluate the performance of the tools. On the other hand, researchers also can input their own haplotype data by pasting the haplotype sample or uploading the file of haplotype sample, or input the address of other websites' haplotype data.

The website provides tools to examine the diversity of haplotypes and partition haplotypes into blocks by using different diversity functions. By using the diversity visualization tool, researchers can observe the diversity of haplotypes in the form of the diagram of curves. The tool uses the diversity function (1), δ_D, to calculate the diversities of all intervals. Figure 6 shows an example of the diversity visualization of the haplotype sample downloaded from HapMap.

The website also provides the tools for researchers to partition haplotypes into blocks with constraints on diversity and tag SNP number; by using the tool, researchers can find the longest segmentation consists of non-overlapping blocks with limited number of tag SNPs. Researchers can upload a haplotype sample or select a sample from our database, and then input a number of tag SNPs to partition the sample. As an example; we pick the shortest contig of Patil's haplotype sample (contig number: NT_001035, 69 SNPs) and paste to the system; applying the diversity function (3), δ_C, and requiring the diversity must not be greater than 0.2 (at least 80% of common haplotype) in each block; using 10 tag SNPs, the sample can be partitioned into 19 haplotype blocks, and there are total of 61 SNPs in these blocks.

In order to test our program, we also apply our block partition tools which can be used to find the longest segmentation covered by the minimum number of tag SNPs to the haplotype data of human chromosome 21 from Patil et al. [18]. Using the tool with the diversity function (3), δ_C, and the same criteria as in Patil (80% of common haplotype coverage), when the haplotype sample is partitioned into blocks fully, a total of 3,260 tag SNPs and a total of 2,266 haplotype blocks are identified. In contrast, Patil et al. [18] identified a total of 4,563 tag SNPs and a total of 4,135 blocks, and Zhang et al. [24] identified a total of 3,582 tag SNPs and a total of 2,575 blocks. Our program reduces the number of tag SNPs and blocks by 28.6% and 45.2% comparing to Patil et al.. We

Method	Common SNPs/block	No. of blocks	No. of blocks requiring ≥ 1 SNPs	Average no. common haplotype/block	All blocks(%)	Common SNP(%)
Ours	> 10	736	733	4.32	32.5	77.8
	$3 - 10$	751	686	3.16	33.1	18.0
	< 3	779	216	2.12	34.4	4.2
	Total	2,266	1,635	3.18	100.0	100.00
Zhang's	> 10	742	738	4.23	28.8	75.5
	$3 - 10$	909	842	3.03	35.3	19.5
	< 3	924	274	2.12	35.9	5.0
	Total	2,575	1,854	3.05	100.0	100.0
Patil's	> 10	589	589	3.75	14.2	56.8
	$3 - 10$	1,408	1,396	2.92	34.1	30.7
	< 3	2,138	1,776	2.30	51.7	12.4
	Total	4,135	3,761	2.72	100.0	100.0

Table 1: The comparison of properties of haplotype blocks defined by Zhang, Patil and us with 80% of common haplotype coverage.

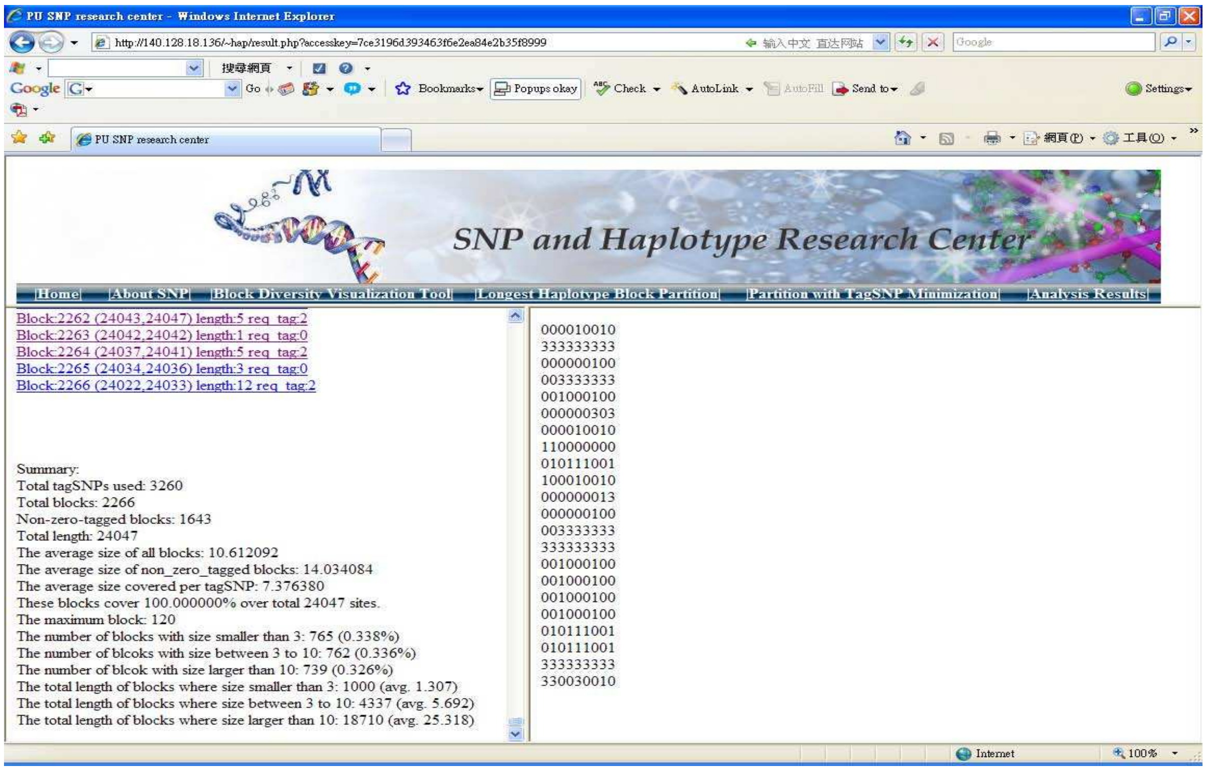
also demonstrate that the results of Zhang et al. are not optimum.

Table 1 shows the comparison of properties of haplotype blocks defined by Zhang, Patil and us with 80% coverage of common haplotype. Our program discovers a total of 736 blocks containing more than 10 SNPs per block. The blocks with more than 10 SNPs account for 32.48% of all of blocks. The average number of SNPs for all of the blocks is 10.61. The largest block contains 128 common SNPs, which is longer than the largest block (containing 114 SNPs) identified by Patil et al. and the same as in Zhang et al.. Figure 7 shows the partition results of the Patil's haplotype sample; each block boundaries and the summaries of the analysis are shown in the frame on the left, and the contents of each block are shown in the frame on the right.

Furthermore, Tables 2 and 3 show more analysis data of our experimental results. According to the results, we can partition 38.55% of genome region into blocks where does not require any tag SNPs. This is because that most of these blocks just contain few common SNPs, and there are 80% of unambiguous haplotypes have the same haplotype pattern (compatible) in these blocks. We term these SNP loci as *non-informative* markers because they are the same among most (80%) of population. These data also show that as the genome region covered increases, we need to increase more and more extra tag SNPs to capture the haplotype information of the blocks, and the number of zero-tagged blocks becomes fewer. Note that, although the average length of non-zero-tagged blocks becomes shorter as the chromosome region covered increase, the average length of total blocks becomes longer.

Both Zhang et al. and we have proposed a dynamic programming algorithm to solve the problem of finding the longest segmentation with limited tag SNPs. However, it is observed that our algorithm obtains better results than theirs on the same haplotype sample. One of the main reasons is that their algorithm presumes that the common haplotypes evaluation function proposed by Patil et al. satisfy the monotonic property. However, when the haplotype sample has missing data, the diversity function does not satisfy the monotonic property. For example, Figure 8 shows the analysis results of Zhang's and our algorithms on the same haplotype sample; this sample just has 69 SNPs, which is a small part of Pail's haplotype data, its contig number is NT_001035. Using the sample criteria (80% of common haplotype), our method can partition the sample into 20 blocks and identify 18 tag SNPs, on the other hand, Zhang's algorithm partitions the sample into 23 blocks and used 22 tag SNPs. The results are similar in interval [21840,21875], but in interval [21876,21899], our method discovers 3 blocks and 3 tag SNPs, which is better than Zhang's (6 blocks and 6 tag SNPs). In interval [21900,21908], both Zhang's and our methods find 2 blocks, but our method just needs 3 tag SNPs rather than 4 that is found by Zhang's method. These cases demonstrate that Zhang's algorithm does not find the optimal solution due to the non-monotonic property of common haplotype evaluation function.

Our web system is freely accessible at `http://bioinfo.cs.pu.edu.tw/∼hap/index.php`.


PU SNP research center - Windows Internet Explorer
http://140.128.18.136/~hap/result.php?accesskey=7ce3196d393463f6e2ea84e2b35f8999
SNP and Haplotype Research Center
|Home| |About SNP| |Block Diversity Visualization Tool| |Longest Haplotype Block Partition| |Partition with TagSNP Minimization| |Analysis Results|
Internet
100%

```
Block:2262 (24043,24047) length:5 req  tag:2
Block:2263 (24042,24042) length:1 req  tag:0
Block:2264 (24037,24041) length:5 req  tag:2
Block:2265 (24034,24036) length:3 req  tag:0
Block:2266 (24022,24033) length:12 req  tag:2


Summary:
Total tagSNPs used: 3260
Total blocks: 2266
Non-zero-tagged blocks: 1643
Total length: 24047
The average size of all blocks: 10.612092
The average size of non_zero_tagged blocks: 14.034084
The average size covered per tagSNP: 7.376380
These blocks cover 100.000000% over total 24047 sites.
The maximum block: 120
The number of blocks with size smaller than 3: 765 (0.338%)
The number of blcoks with size between 3 to 10: 762 (0.336%)
The number of blcok with size larger than 10: 739 (0.326%)
The total length of blocks where size smaller than 3: 1000 (avg. 1.307)
The total length of blocks where size between 3 to 10: 4337 (avg. 5.692)
The total length of blocks where size larger than 10: 18710 (avg. 25.318)
```

```
000010010
333333333
000000100
003333333
001000100
000000303
000010010
110000000
010111001
100010010
000000013
000000100
003333333
333333333
001000100
001000100
001000100
001000100
010111001
010111001
333333333
330030010
```

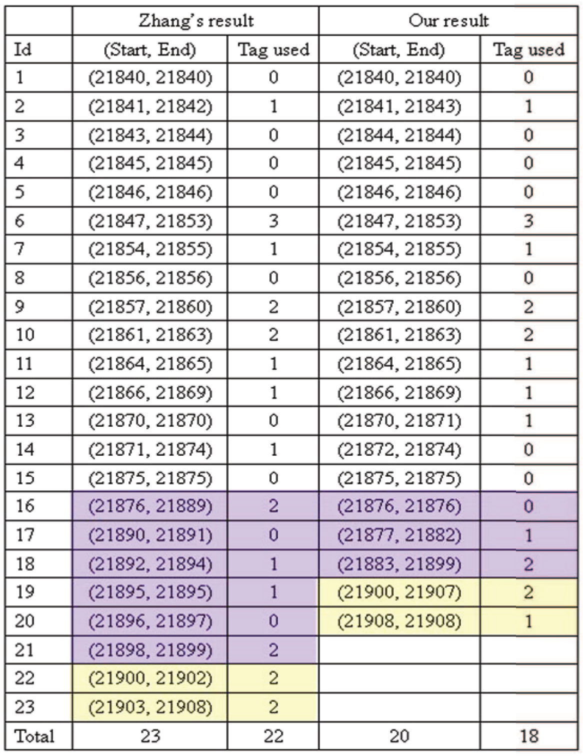
Figure 7: Use diversity function δ_C and require the diversity of blocks must not be greater than 0.2 to partition Patil's haplotype data into blocks fully.

tag SNPs used	genome region covered (%)	extra genome region increased (%)	0-tagged blocks	blocks with tags number > 0	avg. length of non-0-tagged blocks
0% (0)	38.55	38.55	6136	0	1.51(0-tagged blocks)
10% (326)	59.99	21.44	4991	192	35.52
20% (652)	70.85	10.86	4145	367	29.37
30% (978)	78.62	7.77	3387	516	26.79
40% (1304)	84.61	5.99	2897	712	22.38
50% (1630)	89.02	4.41	2250	844	21.29
60% (1956)	92.59	3.57	1814	1002	19.41
70% (2282)	95.30	2.71	1478	1159	17.79
80% (2608)	97.29	1.99	1014	1289	16.90
90% (2934)	98.64	1.35	719	1421	15.89
100% (3260)	100.00	1.36	631	1635	14.10

Table 2: The analysis data based on the number of tag SNPs required.

genome region covered (%)	tag SNPs required	extra tag SNPs required	0-tagged blocks number	blocks with tags number > 0	avg. length of non-0-tagged blocks
38.55	0	0	6136	0	1.51(0-tagged blocks)
40	8	8	6111	6	67.17
50	127	119	5630	80	43.75
60	327	200	4991	193	35.39
70	623	296	4213	347	30.22
80	1045	422	3307	567	25.14
90	1709	664	2208	888	20.58
100	3260	1551	631	1635	14.10

Table 3: The analysis data based on the percentage of genome region covered.



	Zhang's result		Our result	
Id	(Start, End)	Tag used	(Start, End)	Tag used
1	(21840, 21840)	0	(21840, 21840)	0
2	(21841, 21842)	1	(21841, 21843)	1
3	(21843, 21844)	0	(21844, 21844)	0
4	(21845, 21845)	0	(21845, 21845)	0
5	(21846, 21846)	0	(21846, 21846)	0
6	(21847, 21853)	3	(21847, 21853)	3
7	(21854, 21855)	1	(21854, 21855)	1
8	(21856, 21856)	0	(21856, 21856)	0
9	(21857, 21860)	2	(21857, 21860)	2
10	(21861, 21863)	2	(21861, 21863)	2
11	(21864, 21865)	1	(21864, 21865)	1
12	(21866, 21869)	1	(21866, 21869)	1
13	(21870, 21870)	0	(21870, 21871)	1
14	(21871, 21874)	1	(21872, 21874)	0
15	(21875, 21875)	0	(21875, 21875)	0
16	(21876, 21889)	2	(21876, 21876)	0
17	(21890, 21891)	0	(21877, 21882)	1
18	(21892, 21894)	1	(21883, 21899)	2
19	(21895, 21895)	1	(21900, 21907)	2
20	(21896, 21897)	0	(21908, 21908)	1
21	(21898, 21899)	2		
22	(21900, 21902)	2		
23	(21903, 21908)	2		
Total	23	22	20	18

Figure 8: The experimental results of Zhang's and our algorithms on a small part of Patil's data.

Some preliminary results, including the selection of different diversity functions as well as choosing meaningful diversity constraints, in using our tools can also be found in the web system.

4 Conclusion

In this paper, we examine several haplotype diversity evaluation functions; by using appropriate diversity functions, the block selection problem can be viewed as finding a segmentation of given haplotype matrix such that the diversities of chosen blocks satisfy certain value constraint. Tag SNPs can capture most of the haplotype diversity in the blocks, and therefore could potentially capture most of the information for association between a trait and the SNP marker loci. Instead of genotyping all of the SNP markers on the chromosome, one may wish to use only the genotype information on the tag SNP. We can figure out the haplotype features of most population by just checking a few SNP markers. Thus, studying the tag SNPs can dramatically reduce the time and effort for genotyping, without losing much haplotype information.

We present dynamic programming algorithms for haplotype blocks partitioning with constraints on diversity and tag SNP number. In Theorem 1, we show that finding a maximum segmentation with limited tag SNPs number can be done in $O(tnl)$ time; furthermore, we reduce the space complexity into $O(L+n)$ by using the algorithm $\textsc{ListTag}(i,j,T)$. We need to point out that these efficiency results of our algorithms can be applied in many different definition of diversity functions only if we can pre-compute boundaries of all of feasible blocks and tag SNPs required for these blocks.

We also show that the results discovered by our method is superior to Zhang et al.'s [24]. Due to the non-monotonic property of common haplotype evaluation function, we demonstrate that Zhang's algorithm will not find the optimal solution when the haplotype samples have missing data.

References

[1] Eric C. Anderson and John Novembre. Finding Haplotype Block Boundaries by Using the Minimum-Description-Length Principle. *Am. J. of Human Genetics*, 73:336–354, 2003.

[2] D. Clayton. Choosing a set of haplotype tagging SNPs from a larger set of diallelic loci. *Nature Genetics*, 29(2), 2001.

[3] M. J. Daly, J. D. Rioux, S. F. Schafiner, T. J. Hudson, and E. S. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29:229–232, 2001.

[4] S. B. Gabriel, S. F. Schaffner, H. Nguyen, et al. The structure of haplotype blocks in the human genome. *Science*, 296(5576):2225–2229, 2002.

[5] M.R. Garey and D.S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[6] G. Greenspan and D. Geiger. Model-based inference of haplotype block variation. In *Seventh Annual International Conference on Computational Molecular Biology (RECOMB)*, 2003.

[7] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[8] International HapMap Project. http://www.hapmap.org/index.html.en.

[9] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.

[10] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 18(6):341–343, 1975.

[11] Yao-Ting Huang, Kui Zhang, Ting Chen, and Kun-Mao Chao. Selecting additional tag SNPs for tolerating missing data in genotyping . *BMC Bioinformatics*, 6(263), 2005.

[12] R. R. Hudson and N. L. Kaplan. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. *Genetics*, 111:147–164, 1985.

[13] G. C. Johnson, L. Esposito, B. J. Barratt, et al. Haplotype tagging for the identification of common disease genes. *Nat Genet.*, 29(2):233 – 7, Oct 2001.

[14] M. Koivisto, M. Perola, R. Varilo, W. Hennah, J. Ekelund, M. Lukk, L. Peltonen, E. Ukkonen, and H. Mannila. An MDL method for finding haplotype blocks and for estimating the strength of haplotype block boundaries. In *8th Pacific Symposium on Biocomputing (PSB)*, pages 502–513, 2003.

[15] W.H. Li and D. Graur. *Fundamentals of Molecular Evolution*. Sinauer Associates, Inc, 1991.

[16] Yaw-Ling Lin, Tso-Ching Lee, and Wen-Pei Chen. Dynamic programming algorithms for haplotype blocks partitioning with tagSNPs minimization. In *Proceedings of the 24rd Workshop on Combinatorial Mathematics and Computation Theory*, pages 148–157, Nantou, Taiwan, April 28-29, 2007.

[17] Yaw-Ling Lin and Wei-Shun Su. Identifying long haplotype blocks with low diversity. In *Proceedings of the 23rd Workshop on Combinatorial Mathematics and Computation Theory*, pages 151–159, Changhua,Taiwan, Apr 28-29, 2006.

[18] N. Patil, A. J. Berno, D. A. Hinds, et al. Blocks of limited haplotype diversity revealed by high resolution scanning of human chromosome 21. *Science*, 294:1719–1723, 2001.

[19] J. D. Rioux, M. J. Daly, M. S. Silverberg, K. Lindblad, H. Steinhart, Z. Cohen, et al. Genetic variation in the 5q31 cytokine gene cluster confers susceptibility to Crohn disease. *Nature Genetics*, 29:223–228, 2001.

[20] Wei-Shun Su. A Study on SNP Haplotype Blocks. Master's thesis, Providence University, Jun 2006.

[21] Esko Ukkonen. On-Line Construction of Suffix Trees. *Algorithmica*, 14(3):249–260, 1995.

[22] J.D. Wall and J.K Pritchard. Haplotype blocks and linkage disequilibrium in the human genome. *Nature Reviews Genetics*, 4(8):587–597, 2003.

[23] N. Wang, J.M. Akey, K. Zhang, R. Chakraborty, and L. Jin. Distribution of recombination crossovers and the origin of haplotype blocks: the interplay of population history, recombination, and mutation. *Am. J. Human Genetics*, 71:1227–1234, 2002.

[24] K. Zhang, M. Deng, T. Chen, M.S. Waterman, and F. Sun. A dynamic programming algorithm for haplotype block partitioning. In *The National Academy of Sciences*, volume 99, pages 7335–7339, 2002.

[25] K. Zhang, Z.S. Qin, J.S. Liu, T. Chen T, M.S. Waterman, and F. Sun. Haplotype block partitioning and tag SNP selection using genotype data and their applications to association studies. *Genome Res.*, 14(5):908–916, 2004.

[26] Kui Zhang, Zhaohui Qin, Ting Chen, Jun S. Liu, Michael S. Waterman, and Fengzhu Sun. HapBlock: haplotype block partitioning and tag SNP selection software using a set of dynamic programming algorithms. *Bioinformatics*, 21(1):131–134, 2005.

[27] Kui Zhang, Fengzhu Sun, Michael S. Waterman, and Ting Chen. Dynamic programming algorithms for haplotype block partitioning: applications to human chromosome 21 haplotype data. In *RECOMB '03: Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 332–340, New York, NY, USA, 2003. ACM Press.

[28] Kun Zhang and Li Jin. HaploBlockFinder: haplotype block analyses. *Bioinformatics*, 19(10):1300–1301, 2003.

[29] Peisen Zhang, Huitao Sheng, and Ryuhei Uehara. A double classification tree search algorithm for index SNP selection. *BMC Bioinformatics*, 5(89), Jul 2004.

[30] W. Zhao, M. L. Fanning, and T. Lane. Efficient RNAi-based gene family knockdown via set cover optimization. *Artif. Intell. Med.*, 35(1):61–73, Sep 2005.