Mutation Region Detection for Closely Related Individuals without a Known Pedigree

Wenji Ma, Yong Yang, Zhi-Zhong Chen, and Lusheng Wang Member, IEEE

Abstract—Linkage analysis serves as a way of finding locations of genes that cause genetic diseases. Linkage studies have facilitated the identification of several hundreds of human genes that can harbor mutations which by themselves lead to a disease phenotype. The fundamental problem in linkage analysis is to identify regions whose allele is shared by all or almost all affected members but by none or few unaffected members. Almost all the existing methods for linkage analysis are for families with clearly given pedigrees. Little work has been done for the case where the sampled individuals are closely related, but their pedigree is not known. This situation occurs very often when the individuals share a common ancestor at least six generations ago. Solving this case will tremendously extend the use of linkage analysis for finding genes that cause genetic diseases. In this paper, we propose a mathematical model (the shared center problem) for inferring the allele-sharing status of a given set of individuals using a database of confirmed haplotypes as reference. We show the NP-completeness of the shared center problem and present a ratio-2 polynomial-time approximation algorithm for its minimization version (called the closest shared center problem). We then convert the approximation algorithm into a heuristic algorithm for the shared center problem. Based on this heuristic, we finally design a heuristic algorithm for mutation region detection. We further implement the algorithms to obtain a software package. Our experimental data shows that the software is both fast and accurate. The package is available at http://www.cs.cityu.edu.hk/~lwang/software/LDWP/ for non-commercial use.

Index Terms—Haplotype inference, linkage analysis, pedigree, allele-sharing status, and approximation algorithm.

1 INTRODUCTION

Linkage is the tendency for genes and other genetic markers to be inherited together because of their mutually close locations on the same chromosome. Linkage analysis aims at establishing linkage between mutated genes and genetic markers. Today linkage analysis serves as a way of identifying disease causal mutations. Linkage studies have facilitated the identification of several hundreds of human genes that can harbor mutations which by themselves lead to a disease phenotype. The fundamental problem in linkage analysis is to identify regions whose allele is shared by all or most affected members but by none or few unaffected family members.

Traditional approaches to linkage analysis have usually been based on sparse microsatellite markers when the recombination fraction between markers has to be considered. With the new development of microarray techniques, high-density SNP genotype data can be used for large-scale and cost-effective linkage analysis [11]. With high-density SNP genotype data, there exist a sufficient number of informative markers between every pair of recombination points, and the allele-sharing status among the family members can be unambiguously determined. Analysis tools designed for analyzing microsatellite genotype data may not work optimally with high-density SNP genotype data despite vigorous modifications. Lots of new computer programs have been developed for dealing with high-density SNP genotype data.

Almost all the existing methods for linkage analysis are for families with clearly given pedigrees. Existing approaches to linkage analysis can be classified into two categories, namely, probabilistic approaches and deterministic approaches. In probabilistic approaches, recombinant rates are estimated in a way to maximize the likelihood of the observed data [1], [7], [8], [9]. The well-known software tools based on such approaches include GeneHunter [8], LINKAGE [10], Allegro [7], Merlin [1], etc. According to [11], these tools have different performances and efficiencies. Some of them (such as those based on the Elston-Steward algorithm [5]) do not work well when the number of markers is large, while the others (such as those based on the Lander-Green algorithm [9]) do not work well when the number of family members is large. This still remains true even after tremendous improvement has been made to them through subsequent modifications [1], [7]. On the other hand, these tools can give very accurate outputs when the size of the pedigree is small.

Recently some deterministic approaches have been developed. The main idea is to minimize the total number of recombinants to infer the input genotype data so that all/most of diseased individuals share a segment that is shared by none of the normal individuals [3], [14]. The algorithm in [14] can give very accurate outputs when the number of family members is large enough and for
each nuclear family the genotype data for both parents are available. Subsequently, a new software package (called LIden) has been developed in [18]. LIden focuses on handling the case where the genotype data for the whole chromosome of one of the parents in a nuclear family is missing. It also uses the minimum recombinant model for haplotype inference in pedigrees. The main idea behind LIden is a heuristic that combines several local optimization algorithms to first infer the haplotype of each individual and then use the inferred haplotype data to determine the linked regions.

A closely related problem is the haplotype inference problem with a given pedigree. The purpose here is to infer the haplotype accurately. Many haplotype inference algorithms and programs have been developed. Qian and Beckmann [15] and Tapader et al. [17] proposed to minimize the number of recombinants when the pedigree is given. Zhang et al. [21] develop a program without recombinant for general pedigrees. Doi et al. [4] designed two algorithms for haplotype inference with a given pedigree. One of their algorithms works well when the number of marker loci is a fixed constant, while the other works well when the number of family members is bounded by a small constant. Li and Jiang [12], [13] proposed to use an integer linear programming approach for minimum recombinant configuration. Xiao et al. [19] designed a faster algorithm for the case where there is no recombinant. All the aforementioned algorithms heavily depend on the given pedigree and do not work at all without a given pedigree.

To our knowledge, no algorithm can give good output when the sampled individuals are closely related but the real relationship is hidden (most of the times because of remote relationship). This situation occurs very often when the individuals share a common ancestor at least six generations ago. With the new development of microarray techniques, high-density SNP genotype data can be used for large-scale and cost-effective linkage analysis. Recently, the international HapMap project has produced enormous amount of haplotype data for individuals in some major populations. This motivates us to propose a mathematical model that makes use of the existing haplotype databases for individuals in major populations.

Throughout this paper, a region on a chromosome, denoted by $[a, b]$, is a set of consecutive SNP sites (positions) starting at position $a$ and ending at position $b$. The general problem (referred to as the Mutation Region Detection Problem) is as follows: We are given three sets $D = \{g_1, g_2, \ldots, g_k\}$, $N = \{g_{k+1}, \ldots, g_n\}$, and $H = \{h_1, h_2, \ldots, h_m\}$, where $D$ consists of diseased individuals represented by their genotype data on a whole chromosome $C$, $N$ consists of normal individuals represented by their genotype data on $C$, and $H$ consists of confirmed haplotype data on $C$ of some individuals in the same (or similar) population. For convenience, we call $H$ the reference database. We remark that $H$ can be obtained from the database of HapMap project. The objective here is to find the true mutation regions of $C$. Here, a true mutation region of $C$ means a consecutive portion of $C$ where all the diseased individuals share a common haplotype segment that is shared by none of the normal individuals. The true mutation regions defined here are based on the haplotype segments of all individuals. If we know the haplotype segments of all the individuals, the true mutation regions can be easily computed.

The strategy to solve this problem is to first infer the haplotype of each given individual. After knowing the allele-sharing status of all the individuals, we can identify the regions of $C$ where all the diseased individuals share a common haplotype segment that is shared by none of the normal individuals. Those identified regions are candidate mutation regions. In order to get the allele-sharing status of all the input individuals, we divide the whole chromosome $C$ into a set $R$ of regions of a fixed length $L$. For each region $R \in R$, we first obtain $D_R$, $N_R$, and $H_R$, where $D_R$ (respectively, $N_R$ or $H_R$) is the set of (genotype or haplotype) strings in $D$ (respectively, $N$ or $H$) with their letters at positions outside $R$ removed. We then check if we can infer the haplotypes of the individuals over $R$ so that the following conditions hold:

(I) All the diseased individuals share a common haplotype segment $s$ that is shared by none of the normal individuals. That is, for each haplotype strand $h$ (as a string) of a normal
individual, there is at least one position (depending on s and h) where h and s differ.

(II) Each inferred haplotype is close to some haplotype in \( H_R \).

Consider a genotype segment \( g \) in \( D_R \cup N_R \), where the letter of \( g \) at each position can be 0, 1, or 2. A position of \( g \) with a letter 0 indicates that the inferred haplotypes of \( g \) both must have a 0 at the position, while a position of \( g \) with a letter 1 indicates that the inferred haplotypes of \( g \) both must have a 1 at the position. On the other hand, a position of \( g \) with a letter 2 indicates that one of the inferred haplotypes of \( g \) must have a 0 at the position while the other must have a 1 at the position. For convenience, we say that a position of \( g \) is decided if the letter of \( g \) at the position is 0 or 1, and is undecided otherwise. A haplotype pair for \( g \) is a pair \((h, h')\) of haplotypes satisfying the following conditions:

1) The letter of \( g \) at each decided position is the same as the letters of both \( h \) and \( h' \) at the same position.
2) For each undecided position of \( g \), one of \( h \) and \( h' \) has a 0 at the position while the other has a 1 at the position.

For \( D_R \), we define three sets as follows:

- The set of decided positions associated with \( D_R \) consists of all positions \( q \) in \( R \) such that \( q \) is a decided position of at least one string in \( D_R \).
- The set of undecided positions associated with \( D_R \) consists of all positions \( q \) in \( R \) such that \( q \) is an undecided position for all strings in \( D_R \).
- The set of conflicting positions associated with \( D_R \) consists of all positions \( q \) in \( R \) such that \( q \) is a decided position of two distinct \( g_i \in D_R \) and \( g_j \in D_R \) but the letters of \( g_i \) and \( g_j \) at position \( q \) differ.

Given \( D_R, N_R \), and \( H_R \), we want to decide if it is possible to find a haplotype pair for each genotype string in \( D_R \cup N_R \) such that Conditions (I) and (II) hold. If we can successfully find such a haplotype pair for each genotype string in \( D_R \cup N_R \), then \( R \) should be a portion of a true mutation region of the chromosome. In other words, to test whether \( R \) belongs to a true mutation region of the chromosome, we need to solve the following computational problem:

The shared center (SC) problem: We are given a quadruple \((D_R, N_R, H_R, \delta)\), where \( D_R = \{g_1, g_2, \ldots, g_k\} \) and \( N_R = \{g_{k+1}, g_{k+2}, \ldots, g_n\} \) are sets consisting of genotype segments of the same length \( L \), \( H_R = \{h_1, h_2, \ldots, h_m\} \) is a set consisting of haplotype segments of length \( L \), and \( \delta \) (referred to as the radius) is a nonnegative integer. The segments in \( D_R \) are from diseased individuals while those in \( N_R \) are from normal individuals. For convenience, for two binary strings \( s \) and \( t \), we denote their Hamming distance by \( \text{dist}(s, t) \). Moreover, for a string \( t \) and a set \( P \) of positions of \( t \), let \( t_P \) denote the string obtained from \( t \) by deleting the letters at the positions not in \( P \). A solution to \((D_R, N_R, H_R, \delta)\) consists of a center haplotype segment \( s \), a center index \( p \in \{1, 2, \ldots, m\} \), and a haplotype pair \((h_{i,1}, h_{i,2})\) for each \( g_i \in D_R \cup N_R \) such that the following conditions hold:

- C1. \( \text{dist}(s, h_{i,p}) \leq \delta \).
- C2. For each \( i \in \{1, 2, \ldots, k\} \), \( h_{i,1} = s \) and there is an integer \( \ell_i \in \{1, 2, \ldots, m\} \) such that \( \text{dist}(h_{i,2}, h_{i,\ell_i}) \leq \delta \).\)
- C3. For each \( i \in \{k+1, k+2, \ldots, n\} \) and for each \( j \in \{1, 2\} \), the following hold:
  - C3a. There is an integer \( \ell_{i,j} \in \{1, 2, \ldots, m\} \) with \( \text{dist}(h_{i,j}, h_{i,\ell_{i,j}}) \leq \delta \).
  - C3b. \( h_{i,j} \neq s \) or \( s \neq h_{i,j+2} \), i.e., there is at least one position \( q \) in \( U \) at which the letters of \( h_{i,j} \) and \( s \) differ, where \( U \) is the set of decided positions associated with \( D_R \).

Note that the position \( q \) in Condition C3b depends not only on \( i \) and \( j \) but also on \( h_{i,j} \), i.e., different \( i, j \), or \( h_{i,j} \) may yield different \( q \). Moreover, if \( U \) is empty, then there is no solution to \((D_R, N_R, H_R, \delta)\). Given \((D_R, N_R, H_R, \delta)\), an algorithm solving the SC problem is required to check whether there is a solution to \((D_R, N_R, H_R, d)\). If there is one, the algorithm outputs “yes”; otherwise, it outputs “no”. Roughly speaking, we want to compute a haplotype pair \((h_{i,1}, h_{i,2})\) for each \( g_i \in D_R \cup N_R \) such that all the diseased individuals “share” a center haplotype segment \( s \) that is shared by none of the normal individuals. We also want \( s \) and all the haplotypes \( h_{i,1} \) and \( h_{i,2} \) to be similar to some segments in \( H_R \).

Intuitively speaking, for each position \( q \in U \), there is a diseased individual whose haplotype at position \( q \) is already known and hence all the diseased individuals must share this haplotype at position \( q \) (cf. Condition C2). On the other hand, we need to compute a pair \((h_{i,1}, h_{i,2})\) of hypototypes to explain the genotype \( g_i \) of the normal individual so that neither \( h_{i,1} \) nor \( h_{i,2} \) is identical to the center haplotype segment \( s \) on \( R \). However, the condition that \( h_{i,1} \neq s \) (respectively, \( h_{i,2} \neq s \)) on \( R \) can be easily satisfied as long as we can make sure that \( R \) contains a position at which \( h_{i,1} \) (respectively, \( h_{i,2} \)) and \( s \) disagree. Note that unlike the letters of \( s \) at the positions in \( U \), the letters of \( s \) at the positions in \( R \setminus U \) are not fixed in advance. Thus, there is more freedom to find a position in \( R \setminus U \) at which \( h_{i,1} \) (respectively, \( h_{i,2} \)) and \( s \) disagree. Hence, the positions in \( R \setminus U \) are less reliable than the positions in \( U \) for distinguishing the diseased individuals from the normal individuals. In this sense, the condition that \( h_{i,1} \neq s \) and \( h_{i,2} \neq s \) looks too weak. This is why we maintain Condition C3b instead.

If a solution to \((D_R, N_R, H_R, \delta)\) exists for a region \( R \) of the target chromosome \( C \), then we call \( R \) a valid region. Suppose that \( R \) is a true mutation region of \( C \). Then, there is a real haplotype pair \((h_{i,1}, h_{i,2})\) for each \( g_i \in D_R \cup N_R \). If \( H_R \) contains both \( h_{i,1} \) and \( h_{i,2} \), for all \( g_i \in D_R \cup N_R \), then by setting \( d = 0 \), the SC problem always has a solution over \( R \). Intuitively speaking, a valid region

1. In our experiments, each string in \( D_R \cup N_R \cup H_R \) is of length 500 and we have never found a case where \( U = \emptyset \).
that the BCS problem is NP-complete [6].

To divide the true mutation regions of $C$ in a sophisticated way (such as merging adjacent valid regions into longer regions). Each length-$L$ region $R$ in $\mathcal{R}$ has to satisfy the inequality in Condition C3b. Consequently, a (long) true mutation region consisting of multiple valid length-$L$ regions actually has to satisfy multiple such inequalities.

We next show that the SC problem is NP-hard.

**Theorem 1:** The SC problem is NP-hard.

**Proof:** We reduce the binary closest-string (BCS) problem to the special case of the SC problem where all the individuals are diseased. Recall that an instance of the BCS problem is a tuple $(s_1, \ldots, s_n, d)$, where $s_1, \ldots, s_n$ are binary strings of the same length $m$ and $d$ is a nonnegative integer. Given $(s_1, \ldots, s_n, d)$, the BCS problem asks if there is a binary string $t$ of length $m$ such that $\text{dist}(t, s_i) \leq d$ for all $1 \leq i \leq n$. It is known that the BCS problem is NP-complete [6].

Let $(s_1, \ldots, s_n, d)$ be an instance of the binary closest-string problem. Let $m$ be the common length of the strings $s_1, \ldots, s_n$. For convenience, we denote by $\ell \in \{0, 1, 2\}$ and a nonnegative integer $i$, $I$ and $D$ a valid radius for $s_i$ if for each $\ell \in \{\ell \in \{0, 1, 2\} \}$ and an integer $d$, $I$ is a solution to the SC problem as follows.

1. $h_0 = s_1(\ell + 1)n$.
2. For each $i \in \{1, \ldots, n\}$, $h_i = s_1(\ell + 1)n - 1$.

We further obtain $n$ strings $g_1, \ldots, g_n$ as follows:

- For each $i \in \{1, \ldots, n\}$, $g_i = s_1(\ell + 1)n + 1$. $g_i$ contains both $s_i$ and a nonnegative integer $j$.

Suppose that $(s_1, \ldots, s_n, d)$ has a solution $t$ in the binary closest-string problem. Then, we construct a solution for the instance $(\{g_1, \ldots, g_n\}, \emptyset, \{h_0, \ldots, h_n\}; d)$ of the SC problem as follows.

1. $s = t(\ell + 1)n$. Note that $\text{dist}(s, h_0) \leq d$ because $t$ is a solution to $(s_1, \ldots, s_n, d)$ in the binary closest-string problem and hence $\text{dist}(t, s_i) \leq d$.
2. For each $i \in \{1, \ldots, n\}$, construct a haplotype pair $(h_{i,1}, h_{i,2})$ for $g_i$ by setting $h_{i,1} = s$ and $h_{i,2} = g_i(\ell + 1)n - 1$. Note that for each $1 \leq i \leq n$, $\text{dist}(h_{i,2}, h_i) = \text{dist}(t, \pi_i) = \text{dist}(t, s_i) \leq d$ because $t$ is a solution to $(s_1, \ldots, s_n, d)$ in the binary closest-string problem.

Conversely, suppose that the instance $(\{g_1, \ldots, g_n\}, \emptyset, \{h_0, \ldots, h_n\}; d)$ of the SC problem has a solution. Let $s$ be the haplotype segment in the solution. Let $t$ be the prefix of $s$ with $|t| = m$. We claim that $t$ is a solution to $(s_1, \ldots, s_n, d)$ in the binary closest-string problem. To see this, first note that for each $1 \leq i \leq (d + 1)n$, there is a $j \in \{1, \ldots, n\}$ such that the $i$th rightmost letter of $g_j$ is a 0. This implies that the last $(d + 1)n$ bits of $s$ are 0s. So, the string $h_i$ with $\text{dist}(s, h_i) \leq d$ has to be $h_0$ because there are $d + 1$ 1s in the last $(d + 1)n$ bits of each $h_j$ with $1 \leq j \leq n$. Thus, $\text{dist}(t, s_i) \leq d$. Moreover, for each $1 \leq i \leq n$, if we decompose $g_i$ into two strings $h_{i,1}$ and $h_{i,2}$ with $h_{i,1} = s$, then $h_{i,2} = g_i(\ell + 1)n - 1$. Hence, for each $1 \leq i \leq n$, the $h_j$ with $0 \leq j \leq n$ and $\text{dist}(h_j, h_{i,2}) \leq d$ has to be $h_i$ because of the different locations of the $d + 1$ 1s in the $(d + 1)n$ bits of $h_1, \ldots, h_n$. Therefore, $\text{dist}(t, s_i) = \text{dist}(t, \pi_i) \leq d$. This completes the proof of the claim and hence of the theorem.

In the minimization version of the SC problem, we are given a triple $(D_R, N_R, H_R)$, where $D_R, N_R,$ and $H_R$ are as in the SC problem. The objective is as follows. If there is an integer $d$ such that the instance $(D_R, N_R, H_R, d)$ to the SC problem has a solution, then we find the smallest such integer $d$ together with a solution to $(D_R, N_R, H_R, d)$. Otherwise, we report that no such integer $d$ exists. For convenience, we call the minimization version the closest shared center (CSC) problem.

### 3 An Approximation Algorithm for the CSC Problem

Throughout this section, let $I = (D_R, N_R, H_R)$ be an instance of the CSC problem, where $D_R = \{g_1, g_2, \ldots, g_n\}$, $N_R = \{h_{k+1}, h_{k+2}, \ldots, h_n\}$, and $H_R = \{h_1, h_2, \ldots, h_m\}$. Let $L$ be the common length of the strings in $D_R \cup N_R \cup H_R$.

First, we want to decide if there is an integer $d$ such that the instance $(D_R, N_R, H_R, d)$ to the SC problem has a solution. For convenience, we refer to such an integer $d$ as a valid radius for $I$. Section 3.1 is devoted to testing if valid radii exist for $I$. Note that if $d$ is a valid radius for $I$, then so are integers larger than $d$. We say that an integer $d$ is a semi-optimal radius for $I$ if $d$ is a valid radius for $I$ and is at most twice the smallest valid radius for $I$. After knowing the existence of a valid radius for $I$, we want to find a semi-optimal radius $d$ for $I$ together with a solution to $(D_R, H_R, H_R, d)$. Section 3.2 is devoted to this purpose.

#### 3.1 Testing If Valid Radii Exist

Obviously, there is a valid radius for $I$ if and only if $L$ is a valid radius for $I$. So, we consider how to test if $L$ is a valid radius for $I$. For convenience, we say that a string $s$ is a center haplotype segment shared by the strings in $D_R$ if for each $g_i \in D_R$, there is a haplotype pair $(h_{i,1}, h_{i,2})$ with $h_{i,1} = s$. Obviously, if the set of conflicting positions associated with $D_R$ is not empty, then there is no center haplotype segment shared by the strings in $D_R$. Moreover, if there is no center haplotype segment shared by the strings in $D_R$, then $L$ is not a valid radius for $I$. Hence, we hereafter assume that the following condition holds:

A1. The set of conflicting positions associated with $D_R$ is empty.
Let $U$ (respectively, $\overline{U}$) be the set of undecided (respectively, decided) positions associated with $D_R$. If there is a center haplotype segment $s$ shared by the strings in $D_R$, then the letter of $s$ at each position $q \in U$ can be uniquely fixed according to the following rules:

Rule 1. If some segment in $D_R$ is 0 at position $q$ and each of the other segments in $D_R$ is 0 or 2 at position $q$, then the letter of $s$ at position $q$ is 0.

Rule 2. If some segment in $D_R$ is 1 at position $q$ and each of the other segments in $D_R$ is 1 or 2 at position $q$, then the letter of $s$ at position $q$ is 1.

For convenience, we refer to the letter of $s$ at each position $q \in U$ as the center letter at position $q$. Because we only care if $L$ is a valid radius for $U$ or not, the letters of $s$ at the positions in $U$ are not important and neither is the center index $p$.

Now, consider each $g_i \in N_R$. Let $U_i$ (respectively, $\overline{U}_i$) denote the set of undecided (respectively, decided) positions of $g_i$. We say that $g_i$ is free if there is a position in $\overline{U}_i \cap U$ at which the center letter is different from the letter of $g_i$. On the other hand, we say that $g_i$ is dead if (1) $|\overline{U}_i \cap U| \leq 1$ and (2) at every position $q$ in $U_i \cap \overline{U}$, the center letter is the same as the letter of $g_i$.

We claim that if at least one $g_i \in N_R$ is dead, then $L$ is not a valid radius for $U$. Towards a contradiction, assume that this claim does not hold. Then, some $g_i \in N_R$ is dead but there is a solution $S$ to $(D_R, N_R, H_R, L)$. Let $s$ be the center haplotype segment in $S$, and $(h_{i,1}, h_{i,2})$ be the haplotype pair for $g_i$ in $S$. Since $g_i$ is dead, $|U_i \cap U| \leq 1$ and $g_i|_{U \setminus U_i} = s|_{U \setminus U_i}$. So, if $|U \setminus U_i| = 0$, then $h_{i,1} |_{\overline{U}_i} = h_{i,2} |_{\overline{U}_i} = s |_{\overline{U}_i}$, a contradiction against Condition C3b in Section 2. Thus, we may assume that $|U \setminus U_i| = 1$. Let $q$ be the unique position in $U \setminus U_i$. Obviously, either the letters of $h_{i,1}$ and $s$ at position $q$ are the same or the letters of $h_{i,2}$ and $s$ at position $q$ are the same. In the former case, $h_{i,1} |_{\overline{U}_i} = s |_{\overline{U}_i}$, while in the latter case, $h_{i,2} |_{\overline{U}_i} = s |_{\overline{U}_i}$. Thus, we always have a contradiction against Condition C3b in Section 2. This completes the proof of the claim.

So, we hereafter assume that the following condition holds:

A2. No string $g_i \in N_R$ is dead.

Under Condition A2, if a string $g_i \in N_R$ is not free, then $|U_i \cap U| \leq |U| - 2$.

Under Conditions A1 and A2, $L$ is a valid radius for $U$. Indeed, we can construct a solution to $(D_R, N_R, H_R, L)$ as follows. We let the center haplotype segment $s$ in the solution be any binary string such that the letter of $s$ at each position $q \in U$ is the center letter at position $q$. We let the center index $p$ in the solution be any integer in $\{1, \ldots, m\}$. For each $g_i \in D_R$, we obtain the unique haplotype pair $(h_{i,1}, h_{i,2})$ for $g_i$ with $h|_i = s$. For each free $g_i \in N_R$, we obtain an arbitrary haplotype pair $(h_{i,1}, h_{i,2})$. For each $g_i \in N_R$ that is not free, we first obtain an arbitrary haplotype pair $(h_{i,1}, h_{i,2})$ for $g_i$, then select two arbitrary positions $q_1$ and $q_2$ in $U \setminus U_i$, and further make some necessary modifications on the letters of $h_{i,1}$ and $h_{i,2}$ at positions $q_1$ and $q_2$ so that the letter of $h_{i,1}$ at position $q_1$ is different from the center letter at position $q_1$ and the letter of $h_{i,2}$ at position $q_2$ is different from the center letter at position $q_2$.

Note that it is easy to decide if Conditions A1 and A2 hold. So, it is easy to decide if $L$ is a valid radius for $U$.

### 3.2 Computing a Semi-Optimal Radius and a Solution

Throughout this subsection, we assume that $L$ is a valid radius for $U$. So, Conditions A1 and A2 hold. Let $d$ be the smallest valid radius for $U$. It is not hard to decide if $d = 0$. So, we hereafter assume that $d \geq 1$.

Our goal is to compute a valid radius $b$ for $U$ together with a solution $S$ to $(D_R, N_R, H_R, b)$ such that $b \leq 2d$. To find the center haplotype segment in $S$, our idea is to look at the strings $s_1, s_2, \ldots, s_m$ defined as follows:

- For each $p \in \{1, 2, \ldots, m\}$, let $s_p$ be the haplotype segment of length $L$ such that $s_p|_U = h_p |_U$ and the letter of $s_p$ at each position $q \in U$ is the center letter at position $q$.

Basically, our algorithm will select an appropriate $s_p$ among $s_1, s_2, \ldots, s_m$ and include it in $S$ as its center haplotype segment. After this, a haplotype pair for each $g_i \in D_R$ can be easily computed from $s_p$ as shown in the next lemma:

**Lemma 2:** For every $p \in \{1, 2, \ldots, m\}$ and every $i \in \{1, 2, \ldots, k\}$, there is a unique haplotype pair $(h_{p,i,1}, h_{p,i,2})$ for $g_i$ with $h_{p,i,1} = s_p$. Moreover, if $p$ is the center index in a solution to $(D_R, N_R, H_R, d)$, then $d_{p,i} \leq 2d$ for every $i \in \{1, 2, \ldots, k\}$, where $d_{p,i} = \min_{1 \leq j \leq m} \text{dist}(h_{p,i+2}, h_{i,j}).$

**Proof:** The first assertion in the lemma is obvious. To prove the second assertion, consider a solution $S^*$ to $(D_R, N_R, H_R, d)$. Recall that $S^*$ consists of a center haplotype segment $s$, a center index $p \in \{1, \ldots, m\}$, and a haplotype pair $(h_{p,i,1}, h_{p,i,2})$ for each $g_i \in D_R \cup N_R$ satisfying Conditions C1 through C3 in Section 2. Obviously, $\text{dist}(s, h_p) \leq \text{dist}(s, h_p) \leq \text{d}$. Moreover, for every $g_i \in D_R$, $\text{dist}(h_{p,i+2}, h_{i,1}) = \text{dist}(s, h_{i,1}) \leq \text{d}$. Therefore, $\text{dist}(h_{p,i+2}, h_{i,j}) \leq \text{dist}(h_{p,i+2}, h_{i,1}) + \text{dist}(h_{i,1}, h_{i,j}) \leq 2d$, where $\ell_i$ is the integer specified in Condition 2 in Section 2. Now, since $d_{p,i} \leq \text{dist}(h_{p,i+2}, h_{i,j})$, $d_{p,i} \leq 2d.$

By Lemma 2, to obtain $S$, it remains to obtain a haplotype pair for each $g_i \in N_R$. The following definitions will be useful:

- For each $i \in \{k + 1, k + 2, \ldots, n\}$ and each $j \in \{1, 2, \ldots, m\}$, let $d_{i,j}$ be the number of decided positions $q$ of $g_i$ such that the letters of $g_i$ and $h_j$ at position $q$ differ.

- For each triple $(i, j, j')$ with $i \in \{k + 1, k + 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$, and $j' \in \{1, 2, \ldots, m\}$, let $S_{i,j,j'}$ be the set of undecided positions $q$ of $g_i$ such that the letters of $h_j$ and $h_{j'}$ at position $q$ coincide, and let $d_{i,j,j'} = 1 + \max \{d_{i,j} + d_{i,j'}, 0.5(d_{i,j} + d_{i,j'} + |S_{i,j,j'}|)\}.$
For each \( p \in \{1, 2, \ldots, m\} \) and each \( i \in \{k + 1, k + 2, \ldots, n\} \), let \( d_{p,i} = \min_{(j,j')} d_{i,j,j'} \), where \( j \) and \( j' \) range over all integers in \( \{1, 2, \ldots, m\} \setminus \{p\} \).

Based on the above definitions, the following lemma shows how to compute a haplotype pair for each \( g_i \in N_R \):

**Lemma 3:** For each triple \((i, j, j')\) with \( i \in \{k + 1, k + 2, \ldots, n\} \), \( j \in \{1, 2, \ldots, m\} \), and \( j' \in \{1, 2, \ldots, m\} \), we can construct a haplotype pair \((h_{i,j,j'}, h_{i,j,j'})\) for \( g_i \) in \( O(L) \) time such that \( \text{dist}(h_{i,j,j'}, h_{i,j,j'}) \leq d_{i,j,j'} \). Unfortunately, such a pair is not necessarily what we need, because it might be the case that (i) the letter of \( h_{i,j,j'} \) at every position \( q \in \mathcal{U} \) is the center letter at position \( q \). So, we then show that when this bad case occurs, then it suffices to modify \( h_{i,j,j'} \) and \( h_{i,j,j'} \) by first selecting a suitable position \( q \in \mathcal{F} \) and further switching the letters of \( h_{i,j,j'} \) and \( h_{i,j,j'} \) at position \( q \). Note that this modification can increase \( \text{dist}(h_{i,j,j'}, h_{i,j,j'}) \) and \( \text{dist}(h_{i,j,j'}, h_{i,j,j'}) \) by at most 1, implying that we now have \( \text{dist}(h_{i,j,j'}, h_{i,j,j'}) \leq d_{i,j,j'} \) and \( \text{dist}(h_{i,j,j'}, h_{i,j,j'}) \leq d_{i,j,j'} \). Thus, after this modification, \((h_{i,j,j'}, h_{i,j,j'})\) becomes a required haplotype pair for \( g_i \).

We next detail the proof. By definition, \( d_{i,j,j'}^r \leq d_{i,j,j'}^t \), \( d_{i,j,j'}^t \leq d_{i,j,j'} \), and \( d_{i,j,j'}^t + d_{i,j,j'}^t + |S_{i,j,j'}^r| \leq 2d_{i,j,j'} \). So, we can easily partition \( S_{i,j,j'} \) into two subsets \( S_{i,j,j'}^r \) and \( S_{i,j,j'}^t \) such that \( d_{i,j,j'}^t + |S_{i,j,j'}^r| \leq d_{i,j,j'}^t + |S_{i,j,j'}^t| \). Thus, we can obtain a required haplotype pair \((h_{i,j,j'}, h_{i,j,j'})\) for \( g_i \) by performing the following steps in turn:

1. For each decided position \( q \) of \( g_i \), set the letters of \( h_{i,j,j'} \) and \( h_{i,j,j'} \) at position \( q \) to be the letter of \( g_i \) at position \( q \).
2. For each undecided position \( q \) of \( g_i \), set the letters of \( h_{i,j,j'} \) and \( h_{i,j,j'} \) at position \( q \) to be the letter of \( h_j \) and \( h_{j'} \) at position \( q \), respectively.
3. For each undecided position \( q \) in \( S_{i,j,j'}^r \), set the letter of \( h_{i,j,j'} \) at position \( q \) to be the letter of \( h_j \) at position \( q \) and set the letter of \( h_{i,j,j'} \) at position \( q \) to be the letter in \( \{0, 1\} \) different from the letter of \( h_j \) at position \( q \). (Note: After this step, \( \text{dist}(h_{i,j,j'}, h_{i,j,j'}) = d_{i,j,j'}^t + |S_{i,j,j'}^r| \leq d_{i,j,j'}^t \).)
4. For each undecided position \( q \) in \( S_{i,j,j'}^t \), set the letter of \( h_{i,j,j'} \) at position \( q \) to be the letter of \( h_j \) at position \( q \) and set the letter of \( h_{i,j,j'} \) at position \( q \) to be the letter in \( \{0, 1\} \) different from the letter of \( h_j \) at position \( q \). (Note: After this step, \( \text{dist}(h_{i,j,j'}, h_{i,j,j'}) = d_{i,j,j'}^t + |S_{i,j,j'}^t| \leq d_{i,j,j'}^t \).)
of $h_{i,j}$ and $h_{j'}$ at position $q$ differ. Thus, by Condition C3a, there must be a way to partition $S_{i,j,j'}$ into two subsets $S_{i,j,j'}$ and $S_{i,j,j'}$ such that $d_{i,j} + |S_{i,j,j'}| \leq d$ and $d_{i,j} + |S_{i,j,j'}| \leq d$. Hence, $d_{i,j} + |S_{i,j,j'}| \leq 2d$.

By the discussion in the last paragraph, we have the following three inequalities: $d_{i,j'} \leq d$, $d_{i,j} \leq d$, and $d_{i,j} + d_{i,j'} + |S_{i,j,j'}| \leq 2d$. So, $\max\{d_{i,j}, d_{i,j'}, [0.5(d_{i,j} + d_{i,j'} + |S_{i,j,j'}|)]\} \leq d$ because $d$ is an integer. Thus, by the definition of $d_{i,j}, d_{i,j'}$ and $d_{i,j'}, -1 \leq d$. Consequently, by the definition of $d_{p,i}$, $d_{p,i} \leq d + 1$. Finally, $d_{p,i} \leq 2d$ for $d \geq 1$.

We need two more definitions:

- For each $p \in \{1, 2, \ldots, m\}$, let $d_p = \max_{1 \leq i \leq n} d_{p,i}$.
- $b = \min_{1 \leq p \leq m} d_p$.

Corollary 5: If $p$ is the center index in a solution to $(D_R, N_R, H_R, d)$, then $d_p \leq 2d$. Consequently, $b \leq 2d$ and there is a solution to $(D_R, N_R, H_R, b)$.

Proof: The corollary follows from Lemmas 2, 3, and 4 immediately.

Based on Corollary 5, we design an approximation algorithm for the CSC problem. It is shown in Figure 1.

Theorem 6: The algorithm in Figure 1 achieves an approximation ratio of 2 and runs in $O(nLm^2 + nm^3)$ time.

Proof: By Corollary 5, the algorithm achieves an approximation ratio of 2. We next estimate its time complexity. It is easy to show that Step 1 can be done in $O(nLm^2)$ time. Clearly, Step 2 can be done in $(n-k)Lm^2$ time. Step 3 can be done in $O(kLm^2 + (n-k)m^3)$ time, because Steps 3.1, 3.2, 3.3, and 3.4 can be done in $O(L), O(kLm), O((n-k)m^3)$, and $O(n)$ time, respectively. Step 4 can be done in $O(m)$ time. So, the total time complexity is as claimed in the theorem.

Fig. 1. An approximation algorithm for the CSC problem.

**Input:** $(D_R, N_R, H_R)$ for which the common length of the strings in $D_R \cup N_R \cup H_R$ is a valid radius, where $D_R = \{g_1, g_2, \ldots, g_k\}$, $N_R = \{g_{k+1}, g_{k+2}, \ldots, g_n\}$, and $H_R = \{h_1, h_2, \ldots, h_m\}$.

**Output:** A valid radius $b$ for $(D_R, N_R, H_R)$ together with a solution to $(D_R, N_R, H_R, b)$.

1. Check if $b$ is a valid radius for $(D_R, N_R, H_R)$, and if so, output 0 together with a solution to $(D_R, N_R, H_R, 0)$ and then halt.
2. For each triple $(i, j, j')$ with $i \in \{k + 1, k + 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$, and $j' \in \{1, 2, \ldots, m\}$, perform the following steps:
   2.1. Compute $d_{i,j,j'}$ and construct a haplotype pair $(h_{i,j',j'}, h_{i,j,j'})$ for $g_i$ as in Lemma 3.
   2.2. For each integer $p \in \{1, 2, \ldots, m\}$, perform the following steps:
      2.2.1. Construct $s_p$.
      2.2.2. For each $i \in \{1, 2, \ldots, k\}$, perform the following steps:
         2.2.2.1. Construct the haplotype pair $(h_{p,i,j'}, h_{p,i,j})$ for $g_i$ with $h_{p,i,1} = s_p$.
         2.2.2.2. Compute $d_{p,i} = \min_{1 \leq j \leq m} \text{dist}(h_{p,i,j'}, h_{j'})$.
      2.2.3. For each $i \in \{k + 1, k + 2, \ldots, n\}$, perform the following steps:
         2.2.3.1. Compute $d_{p,i} = \min_{j,j'} d_{i,j,j'}$ where $j$ and $j'$ range over all integers in $\{1, 2, \ldots, m\}$ in an arbitrary order until at least one of $d_{i,j,j'}$.
         2.2.3.2. Find a pair $(j_p,i,j_p')$ such that $j_p,i \in \{1, 2, \ldots, m\} \backslash \{p\}$, $j_p' \in \{1, 2, \ldots, m\} \backslash \{p\}$, and $d_{i,j_p,j_p'} = d_{p,i}$.
      2.3. Compute $d_p = \max_{1 \leq i \leq n} d_{p,i}$.
   3. Compute $b = \min_{1 \leq p \leq m} d_p$ and find a $p \in \{1, 2, \ldots, m\}$ such that $d_p = b$.
5. Output $b$ and the solution to $(D_R, N_R, H_R, b)$ consisting of $s_p$, $p$, and the pairs
   $\{(h_{p,i,1}, h_{p,i,2}) | 1 \leq i \leq k\} \cup\{(h_{p,i,j_p,j_p'}, h_{p,i,j_p,j_p'}) | k + 1 \leq i \leq n\}$.

**Fig. 1.** An approximation algorithm for the CSC problem.

**k + 1 \leq i \leq n, 1 \leq \ell \leq L\}$ in $O(kLm^2)$ total time in advance. The following definition is for this purpose:

- For each pair $(i, \ell)$ with $k + 1 \leq i \leq n$ and $1 \leq \ell \leq L$, let $Q_{i,\ell}$ be the set of all pairs $(j, j')$ of integers in $\{1, 2, \ldots, m\}$ with $d_{i,j,j'} \leq \ell$.

Obviously, after performing Step 2 of the algorithm (in Figure 1), we can compute all the sets $Q_{i,\ell}$ with $k + 1 \leq i \leq n$ and $1 \leq \ell \leq L$ in $O((n-k)Lm^2)$ total time in advance. The crucial point is that with $Q_{i,\ell}$ known, we can compute $P_{p,i,\ell}$ in $O(m)$ time when $p$, $i$, and $\ell$ are given. The idea for computing $P_{p,i,\ell}$ is to scan the pairs in $Q_{i,\ell}$ in an arbitrary order until at least one of the following conditions holds:

- A pair $(j, j')$ with $j \neq p$ and $j' \neq p$ is found.
- Already $2m$ pairs in $Q_{i,\ell}$ or all the pairs in $Q_{i,\ell}$ have been scanned.

If Condition (i) holds, then we can let $P_{p,i,\ell} = \{(j, j')\}$.
Otherwise, we can let $P_{pi,t} = \emptyset$ because among any subset of $2m$ pairs in $Q_{i,t}$, there is at least one pair $(j, j')$ with $j \neq p$ and $j' \neq p$.

In summary, to speed up the algorithm in Figure 1, it suffices to replace Steps 3, 4, and 5 of the algorithm by the four steps in Figure 2.

3. For each integer $p \in \{1, 2, \ldots, m\}$, compute $d_p$, construct $s_p$, and construct the haplotype pair $(h_{p,i,1}, h_{p,i,2})$ for each $g_i \in D_R$ with $h_{p,i,1} = s_p$.
4. For each pair $(i, \ell)$ with $k + 1 \leq i \leq n$ and $1 \leq \ell \leq L$, compute $Q_{i,\ell}$.
5. For each triple $(p, i, \ell)$ with $1 \leq p \leq m$, $k + 1 \leq i \leq n$, and $1 \leq \ell \leq L$, compute $P_{pi,\ell}$.
6. For $b = 1, 2, \ldots, L$ (in this order), perform the following step:
   6.1. For each integer $p$ with $1 \leq p \leq m$, perform the following step:
     6.1.1. If $d_{pi} \leq b$ and $P_{pi,b} \neq \emptyset$ for all $i \in \{k+1, k+2, \ldots, n\}$, then output $b$ together with the solution to $(D_R, N_R, H_R, b)$ that consists of $s_p$, $p$, and the pairs in $(h_{p,i,1}, h_{p,i,2}) : 1 \leq i \leq k \cup \{(h_{p,j,i,1}, h_{p,j,i,2}) : k+1 \leq i \leq n \}$ and $(j, j')$ is the pair in $P_{pi,b}$.

Fig. 2. Replacing Steps 3, 4, and 5 of the algorithm in Figure 1.

Now, we are ready to state the main theorem of this section.

**Theorem 7**: The modified algorithm achieves an approximation ratio of 2 and runs in $O(nLm^2)$ time.

### 4 A Decision Algorithm

We can directly use the modified algorithm in Section 3 to approximately test if a given region $R$ in a chromosome belongs to a true mutation region. We first obtain the instance $(D_R, N_R, H_R)$ of the CSC problem in region $R$ and see if the modified algorithm can return an approximate solution with radius $2d$. If the algorithm cannot return an approximate solution with radius $2d$ for a user defined value of $d$, we can conclude that there is no solution to the instance $(D_R, N_R, H_R, d)$ of the SC problem and rule out the possibility that $R$ is part of a true mutation region. Otherwise, we should consider $R$ as part of a candidate mutation region for further processing.

In order to get better results in practice, we transform the modified algorithm in Section 3 into a decision algorithm which is shown in Figure 3. In the decision algorithm, we have a user defined value $d$ as part of the input and the algorithm returns either “yes” or “no”. The main difference is that instead of trying to find a small radius $b$ together with a solution to $(D_R, N_R, H_R, b)$ as in the modified algorithm, we test Conditions (a) and (b) in Step 6.1. Note that the inequality $\text{dist}(s_p, h_\ell) + \text{dist}(h_{p,i,2}, h_{\ell}) \leq 2d$ is much stronger than the inequality $\text{dist}(h_{p,i,2}, h_{\ell}) \leq 2d$ which holds in the modified algorithm. Thus, Conditions (a) and (b) together are much stronger than the existence of a solution to $(D_R, N_R, H_R, 2d)$. So, if the decision algorithm returns “yes”, then we can always get a solution to $(D_R, N_R, H_R, 2d)$. However, it is possible that Condition (b) does not hold but a solution to $(D_R, N_R, H_R, d + 1)$ exists. For example, when $\text{dist}(s_p, h_\ell) = \text{dist}(h_{p,i,2}, h_{\ell}) = d$ and $|\{ U - \text{dist}(h_{\ell}, h_{p,\ell}) \}) = 1$, the inequality $\text{dist}(s_p, h_\ell) + \text{dist}(h_{p,i,2}, h_{\ell}) \leq 2d$ does not hold but it is still possible to have a solution to $(D_R, N_R, H_R, d + 1)$.

### 5 Heuristics for Mutation Region Detection

In this section, we use the decision algorithm in Section 4 to design heuristics for the general mutation region detection problem.

In order to find the true mutation regions of a target chromosome $C$, we divide $C$ into a set $R$ of length-$L$ regions by cutting $C$ at the positions $L, 2 \cdot L, \ldots, \lfloor c/L \rfloor \cdot L$, where $c$ is the number of SNPs in $C$. In our experiments, we always fix $L = 500$. For each region $R \in R$, we first obtain the instance $(D_R, N_R, H_R, d)$ of the SC problem in region $R$ by setting $d = \lfloor L/10 \rfloor$. If the decision algorithm outputs “yes”, then we view $R$ as a valid region. Otherwise, we view $R$ as an invalid region.

Let $\mathcal{V}$ be the valid length-$L$ regions obtained as above. We then keep modifying $\mathcal{V}$ as follows. Whenever $\mathcal{V}$ contains two regions $R_1$ and $R_2$ that are at most $3L$ SNP sites apart on the chromosome $C$, we modify $\mathcal{V}$ by replacing $R_1$ and $R_2$ with the smallest region of $C$ that contains both $R_1$ and $R_2$. For example, if $L = 500$ and $[1, 1500]$ and $[1001, 1500]$ are two regions in $\mathcal{V}$, then we replace them by the larger region $[1, 1500]$. Finally, we
output the first few (say, 3 or 4) largest regions in $V$ as the
mutation regions of $C$. This completes the description of
our first heuristic for mutation region detection. For
convenience, we call it Heuristic 1.

We have tested the performance of Heuristic 1 on some
simulated data. Our experimental data shows that
Heuristic 1 often outputs several disjoint mutation re-
gions that indeed belong to a single long true mutation
region of the target chromosome $C$. If we just report
one of them, a big portion of the true mutation region
will be missing. Therefore, we further keep modifying
the set $V_1$ of mutation regions found by Heuristic 1 as
follows. Whenever $V_1$ contains two regions $R_1$ and $R_2$
that are at most $7L$ SNP sites apart on the chromosome
$C$, we modify $V_1$ by replacing $R_1$ and $R_2$ with the
smallest region of $C$ that contains both $R_1$ and $R_2$. After
modifying $V_1$ in this way, we output the regions in it.
For convenience, we call the new heuristic Heuristic 2.

We have tested the performance of Heuristic 2 on some
simulated data. Our experimental data shows that
Heuristic 2 often outputs a mutation region that can be
obtained from a true mutation region of the chromosome
$C$ by deleting a number of SNPs in its left or right end.
In other words, by extending a mutation region found by
Heuristic 2 in both (left and right) directions, we obtain
a true mutation region of $C$. This motivates us to modify
the set $V_2$ of mutation regions found by Heuristic 2 as
follows. For each region $R \in V_2$, we try to extend $R$
along $C$ in both directions each up to $4L$ SNP sites. More
precisely, if there are at least $4L$ SNP sites to the left
(respectively, right) of $R$ on $C$, then we divide the $4L$
SNP sites immediately to the left (respectively, right) of $R$
into a set $R'$ of 4 regions each of length $L$; otherwise,
we divide all SNP sites to the left (respectively, right) of $R$ on
$C$ into a set $R'$ of at most 4 regions, all of them except
one are of length $L$. For each $R' \in R'$, we ignore the
normal individuals to obtain the instance $(D_{R'}, \emptyset, H_{R'}, d)$
of the SC problem in region $R'$ and call the decision
algorithm to approximately solve the SC problem on
input $(D_{R'}, \emptyset, H_{R'}, d)$. If the algorithm outputs “yes” on
input $(D_{R'}, \emptyset, H_{R'}, d)$ and $R'$ is within a distance of $L$
SNP sites to $R$ on $C$, then we extend $R$ to include $R'$.
After modifying each region $R$ in the set $V_2$ in this way,
we output the regions in the set as the mutation regions
of $C$. For convenience, we call this heuristic Heuristic 3.

6 IMPLEMENTATION

We have implemented the algorithms in C++ to obtain a
software package that can run on a Windows machine.
It has two versions: one of them provides a graphical
user interface while the other does not. To run the pack-
age, one has to prepare three input files: children.ped,
genotype.txt, and haplotype.phased. Here, children.ped
contains the basic information about the input individu-
als such as their names, genders, and diseased statuses.
File genotype.txt corresponds to the union of $D$ and $N$
in Section 2 and hence contains the genotype data of
the input (diseased or normal) individuals. File haplo-
type.phased corresponds to $H$ in Section 2 and hence
contains the confirmed haplotype data of some individu-
als in the same population as the input individuals.
For the reader’s convenience, we provide an example
haplotype.phased which contains the haplotype data for
chromosome 1 of 170 unrelated Japanese in Tokyo and
Han Chinese in Beijing. These data were downloaded
from HapMap (http://hapmap.org). Given the three
files children.ped, genotype.txt, and haplotype.phased,
our package outputs the predicted mutation regions for
them. Each output region is shown by the indexes of its
starting SNP and ending SNP sites on the chromosome.

7 EXPERIMENTS

In order to evaluate the performance of the heuristics
and the feasibility of the mathematical model proposed
in this paper, we have written a program in C++ to
produce simulated data. The program takes a pedigree
and the haplotype data for the whole chromosome of
each founder in the pedigree as input. It generates the
haplotype data for the remaining individuals in the
pedigree using the standard $\chi^2$ model for recombination
with the parameter (the degree of freedom divided by 2)
equal to 4 [2] and according to the male/female aver-
gaged genetic map for chromosome 1 downloaded from
HapMap (http://hapmap.org). The haplotype data of a
non-founder in the pedigree are generated to randomly
inherit one strand of the four-strand chromatid bundle
from each parent of the non-founder. A mutation point
is selected uniformly at random from the SNP sites of
the chromosome and it appears on one strand of the
haplotype pair in the diseased founder. (Each pedigree
has one diseased founder.) Each diseased offspring is
forced to inherit (from each of its parent) the strand with
the mutation point and the normal offsprings are forced
to inherit the strand without the mutation point. In this
way, we can guarantee that there is at least one true
mutation region. Moreover, since we know the haplotype
data of all the individuals in the simulations, we can
easily find the true mutation regions. By definition (see
Section 2), there may exist more than one true mutation
region. In our experiments, we find that the chance to
have more than one true mutation region is less than 1%.
Thus, from now on we just use the unique true mutation
region containing the mutation point in the rest of the
paper.

In our experiments, we use the haplotype data for
chromosome 1 of 170 unrelated Japanese in Tokyo and
Han Chinese in Beijing as our reference database. The
founders of an input pedigree (and hence their haplotype
data) are randomly chosen from this database. We note
that there are 116415 SNPs in chromosome 1. When we
run our programs (to evaluate their performance) on
the simulated data, we delete the haplotype data of the
founders from the reference database $H$ (to avoid trivial
solutions).
To evaluate our programs, we use six different pedigrees. They are shown in Figures 4 through 9 and are denoted by $P_1$, $P_2$, ..., $P_6$, respectively. Pedigree $P_1$ is generated manually. The rest of pedigrees are modified from $P_1$. No couple is allowed to share any common ancestor in all the six pedigrees. Pedigrees $P_1$ through $P_4$ have 5 generations and each of them has 2, 3, 4, and 5 diseased individuals as part of the input for our program in the latest generation, respectively. Pedigrees $P_5$ and $P_6$ have 6 and 7 generations, respectively. In each figure, a square represents a male, while a circle represents a female. Moreover, a filled square (respectively, circle) represents a diseased male (respectively, female), while an unfilled square (respectively, circle) represents a normal male (respectively, female). Furthermore, if two circles (respectively, squares) enclose the same number in the figure, then they correspond to the same male (respectively, female) and their circumferences (respectively, sides) are dashed. This makes the figure more readable. Note that our programs only take the individuals in the youngest generation of the pedigree as input. So, to emphasize the youngest generation, we use a dotted rectangle to enclose them at the bottom of the pedigree. In this way, one can easily find out that $P_1$, $P_2$, $P_3$, and $P_4$ have 10, 10, 12, and 15 individuals in their youngest generation, respectively. Moreover, one can see that the four pedigrees have different structures, because they have 2, 3, 4, and 5 diseased individuals in the youngest generation, respectively.

We have done 150 experiments for each pedigree and calculated the average performance of our programs. We use precision and recall to evaluate the performance of our programs. The correctly detected mutation regions are the intersection of the regions output by the computer program and the true mutation regions. Here, precision is defined as the number of SNPs in the correctly detected mutation regions divided by the total number of SNPs in the regions output by the program. The value of recall is defined as the number of SNPs in the correctly detected mutation regions divided by the total number of SNPs in the true mutation regions. So, if the value of recall is 1, then all the SNPs in the true mutation regions have been output by the program. Similarly, if precision is 1, then all the SNPs reported by the program are in the true mutation regions.

The columns “$P_1$” through “$P_4$” in Table 1 show our experimental results for $P_1$ through $P_4$, respectively. The table consists of three parts separated by two consecutive horizontal lines. The first (respectively, second or third) part shows the result of Heuristic 1 (respectively, Heuristic 2 or Heuristic 3). Note that, in terms of recall, Heuristic 3 always gives the best results. Each part has three rows: “longest”, “first 2 longest”, and “first 3 longest”. The row “longest” is the result that our program just outputs the longest detected region. The row “first 2 longest” is the result that our program outputs the first two longest detected regions as the output. The row “first 3 longest” is the result that our program outputs
the first three longest detected regions as the output. In any case, if the output detected regions have no overlap with the true mutation regions, both precision and recall are treated as 0.

By the columns "P1" and "P2" in Table 1, the average recall values for the 150 tests are 90.3% and 86.3%, respectively. This implies that the reported regions for P1 and P2 cover 90.3% and 86.3% of the true mutation regions, respectively. The precision values 48.3% and 58.7% at the bottom of the columns "P1" and "P2" in Table 1 indicate that the sizes of the reported mutation regions are about twice of that for the true mutation regions. This shows that it is still very useful for narrowing the region for searching the mutation gene(s), which is the main purpose here. From the columns "P1" through "P4" in Table 1, we can see that the values of both precision and recall decrease when the number of diseased individuals increases. One of the possible reasons might be that the average length of the true mutation regions becomes shorter when the number of diseased individuals increases. Table 2 lists the average length of the true mutation regions for P1 through P6. For pedigrees P1 to P4, where all the four pedigrees have 5 generations, the length of the true mutation regions decreases from 7107 to 4128 with the increment of the number of diseased individuals. By the definition of the true mutation regions, when there are more diseased individuals, the length of the true mutation regions shared by all the diseased individuals will certainly decrease. Comparing pedigrees P5, P6, and P6, where they all have two diseased individuals in the input set and have 5, 6, and 7 generations, respectively, the length of the true mutation regions also decreases when the number of generations increases.

P5 and P6 demonstrate the situations where there are 6 and 7 generations, respectively. When there are two diseased individuals, the average recall of the 150 tests are very similar to that of 5 generations. The results are shown in the columns "P5" and "P6" in Table 1.

We have run the programs on a computer with Intel(R) Core(TM) 2 CPU 2.40GHz and 4GB memory. The running times of the heuristics range from several minutes to dozens of minutes for the six pedigrees. We find that

<table>
<thead>
<tr>
<th>Pedigree</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
</tr>
</thead>
<tbody>
<tr>
<td>real region</td>
<td>7107</td>
<td>6611</td>
<td>5088</td>
<td>4128</td>
<td>5508</td>
<td>4446</td>
</tr>
<tr>
<td>longest</td>
<td>5440</td>
<td>5667</td>
<td>2539</td>
<td>1278</td>
<td>5997</td>
<td>4711</td>
</tr>
<tr>
<td>first 2 longest</td>
<td>7349</td>
<td>7073</td>
<td>3223</td>
<td>1416</td>
<td>8635</td>
<td>6624</td>
</tr>
<tr>
<td>first 3 longest</td>
<td>8117</td>
<td>7628</td>
<td>3371</td>
<td>1437</td>
<td>10333</td>
<td>7470</td>
</tr>
<tr>
<td>longest</td>
<td>8441</td>
<td>7253</td>
<td>3877</td>
<td>1738</td>
<td>8590</td>
<td>6539</td>
</tr>
<tr>
<td>first 2 longest</td>
<td>10298</td>
<td>8564</td>
<td>4331</td>
<td>1795</td>
<td>12031</td>
<td>8809</td>
</tr>
<tr>
<td>first 3 longest</td>
<td>10918</td>
<td>9895</td>
<td>4394</td>
<td>1767</td>
<td>13548</td>
<td>9628</td>
</tr>
<tr>
<td>longest</td>
<td>9634</td>
<td>8149</td>
<td>4935</td>
<td>2610</td>
<td>10193</td>
<td>7738</td>
</tr>
<tr>
<td>first 2 longest</td>
<td>13325</td>
<td>10471</td>
<td>5995</td>
<td>2798</td>
<td>15513</td>
<td>11774</td>
</tr>
<tr>
<td>first 3 longest</td>
<td>14713</td>
<td>11301</td>
<td>6219</td>
<td>2818</td>
<td>18597</td>
<td>13738</td>
</tr>
</tbody>
</table>

**Table 2**
The average length of the true mutation regions and the regions computed by the heuristics for P1 through P6.

8 CONCLUDING REMARKS

We have proposed a mathematical model for inferring the allele-sharing status of a given set of individuals using a database of confirmed haplotypes as reference. Our experimental data shows that the method can report about 50% to 90% SNPs in the true mutation regions in different cases.

Based on the mathematical model, we know that if \( H_R \) contains all the real haplotype pairs \((h_{i,1}, h_{i,2})\) for all \(g_i \in \{D_R \cup N_R\}\), then by setting \(d = 0\), the SC problem

<table>
<thead>
<tr>
<th>Pedigree</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristic 1</td>
<td>15.47</td>
<td>13.62</td>
<td>9.23</td>
<td>2.3</td>
<td>27.78</td>
<td>34.08</td>
</tr>
<tr>
<td>Heuristic 3</td>
<td>17.59</td>
<td>15.30</td>
<td>10.51</td>
<td>2.92</td>
<td>29.72</td>
<td>35.38</td>
</tr>
</tbody>
</table>

**Table 3**
The average running times (in minutes) of Heuristics 1 through 3 for P1 through P6.
always has a solution over $R$. Note that, when we do the experiments, we delete the haplotype data of all the founders from the reference database. Thus, for some cases, the recall value is not very big. With the increasing of the size of the database, we can expect that the value of recall can be improved.

Acknowledgments We thank the referees for very helpful comments. This work is supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 121608]. Lusheng Wang is the corresponding author.

**REFERENCES**


Wenji Ma received the Bachelor’s degree in software engineering from Shandong University, Jinan, Shandong, China in 2009. She is now a PhD student in the Department of Computer Science, City University of Hong Kong. Her research interests include algorithms and bioinformatics.

Yong Yang received his Master of Philosophy degree in Computer Science from City University of Hong Kong, in 2010. His research interests include algorithms and bioinformatics.

Zhi-Zhong Chen received the PhD degree from the University of Electro-Communications, Tokyo, Japan in 1992. Currently, he is a professor in the Division of Information System Design, Tokyo Denki University. His research interests include algorithms, computational biology, and graph theory.

Lusheng Wang received the PhD degree from McMaster University, Hamilton, Ontario, Canada, in 1995. Currently, he is a professor in the Department of Computer Science, City University of Hong Kong. His research interests include algorithms, bioinformatics, and computational biology. He is a member of the IEEE.