Mutation Region Detection for Closely Related Individuals without a Known Pedigree

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

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 show that the software is both fast and accurate. The package is available at http://www.cs.cityu.edu.hk/~lwang/software/LDWP/ for noncommercial 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], [16]. 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

- *W. Ma, Y. Yang, and L. Wang are with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: wenjima2@student.cityu.edu.hk, cswangl@cityu.edu.hk.*
- *Z.-Z. Chen is with the Department of Information System Design, Tokyo Denki University, Tokyo, Japan. E-mail: zzchen@mail.dendai.ac.jp.*

Manuscript received 21 Apr. 2011; revised 2 Sept. 2011; accepted 10 Sept. 2011; published online 17 Oct. 2011.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2011-04-0103.
Digital Object Identifier no. 10.1109/TCBB.2011.134.

1545-5963/12/$31.00 © 2012 IEEE Published by the IEEE CS, CI, and EMB Societies & the ACM

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. For example, there are 340 haplotypes in the group "Japanese in Tokyo"+"Han Chinese in Beijing." These new developments make it possible for us to propose new mathematical models for finding genes causing genetic diseases when the sampled individuals are closely related but their pedigree is unknown.

In this paper, we propose a mathematical model (the *shared center (SC) problem*) for inferring the allele-sharing status of a given set of individuals using a database of confirmed haplotypes as reference. We show that the shared center problem is NP-complete. We then present a ratio-2 polynomial-time approximation algorithm for the *closest shared center* (CSC) problem which is the minimization version of the shared center problem. We further convert the approximation algorithm into a heuristic algorithm for the shared center problem. Using this heuristic algorithm as a subroutine, we finally design a heuristic algorithm for mutation region detection. We also implement the algorithms to obtain a software package for mutation region detection. Experiments show that the method can report about 50 to 90 percent SNPs in the true mutation regions in different cases.

2 Method

Our task here is to solve the problem where the given individuals are closely related but their pedigree is unknown. 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 $D = \{\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_k\}$, $N = \{\hat{g}_{k+1}, \ldots, \hat{g}_n\}$, and $H = \{\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_m\}$, 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 haplotypes 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 $\mathcal{R}$ of regions of a fixed length L. For each region $R \in \mathcal{R}$, 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:

1. 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.
2. 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 1 and 2 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 problem. We are given a quadruple (D_R, N_R, H_R, d), 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 d (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 $dist(s,t)$. 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, d) 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. $dist(s, h_p) \le d$.

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 $dist(h_{i,2}, h_{\ell_i}) \le d$.

C3. For each $i \in \{k+1, k+2, \ldots, n\}$ and for each $j \in \{1, 2\}$, the following hold:

a. There is an integer $\ell_{i,j} \in \{1, 2, \ldots, m\} \setminus \{p\}$ with $dist(h_{i,j}, h_{\ell_{i,j}}) \le d$.

b. $h_{i,j}|_{\overline{U}} \ne s|_{\overline{U}}$, i.e., there is at least one position q in $\overline{U}$ at which the letters of $h_{i,j}$ and s differ, where $\overline{U}$ 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 $\overline{U}$ is empty,[1] then there is no solution to (D_R, N_R, H_R, d).

Given (D_R, N_R, H_R, d), 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 $q \in \overline{U}$, 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 hyplotypes 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 $h_{i,1} \ne s$ (respectively, $h_{i,2} \ne s$) 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 $\overline{U}$, the letters of s at the positions in $R \setminus \overline{U}$ are not fixed in advance. Thus, there is more freedom to find a position in $R \setminus \overline{U}$ at which $h_{i,1}$ (respectively, $h_{i,2}$) and s disagree. Hence, the positions in $R \setminus \overline{U}$ are less reliable than the positions in $\overline{U}$ for distinguishing the diseased individuals from the normal individuals. In this sense, the condition that $h_{i,1} \ne s$ and $h_{i,2} \ne s$ looks too weak. This is why we maintain Condition C3b instead.

If a solution to (D_R, N_R, H_R, d) 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 $(\tilde{h}_{i,1}, \tilde{h}_{i,2})$ for each $g_i \in D_R \cup N_R$. If H_R contains both $\tilde{h}_{i,1}$ and $\tilde{h}_{i,2}$ 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 R should belong to (part of) a true mutation region of C if H_R contains both $\tilde{h}_{i,1}$ and $\tilde{h}_{i,2}$ for all $g_i \in D_R \cup N_R$.

To find the true mutation regions of C, our idea is to divide C into a set $\mathcal{R}$ of length-L regions and test whether each of them is a valid region. After finding all valid regions in $\mathcal{R}$, we then use them to construct 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 $dist(t, s_i) \le d$ for all $1 \le i \le n$. 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, for a letter $\ell \in \{0, 1, 2\}$ and a nonnegative integer i, let ℓ^i denote the string consisting of $i\ell$s. Note that ℓ^0 is the empty string.

[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 $\overline{U} = \emptyset$.

For a binary string s, let $\overline{s}$ denote the string obtained from s by flipping each bit. We obtain $n+1$ strings h_0, $h_1, \ldots, h_n$ as follows:

1. $h_0 = s_1 0^{(d+1)n}$.
2. For each
$$i \in \{1, \ldots, n\}, h_i = \overline{s_i} 0^{(d+1)(i-1)} 1^{d+1} 0^{(d+1)(n-i)}.$$

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

- For each
$$i \in \{1, \ldots, n\}, g_i = 2^m 0^{(d+1)(i-1)} 2^{d+1} 0^{(d+1)(n-i)}.$$

Suppose that $(s_1, \ldots, s_n, d)$ has a solution t in the binary closest-string problem. Then, we can 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. $s = t0^{(d+1)n}$. Note that $dist(s, h_0) \leq d$ because t is a solution to $(s_1, \ldots, s_n, d)$ in the binary closest-string problem and hence $dist(t, s_1) \leq d$.
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 $h_{i,2} = \overline{t}0^{(d+1)(i-1)}1^{d+1}0^{(d+1)(n-i)}$. Note that for each $1 \leq i \leq n$, $dist(h_{i,2}, h_i) = dist(\overline{t}, \overline{s_i}) = dist(t, s_i) \leq d$ 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 center 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 ith 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 $dist(s, h_i) \leq d$ 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, $dist(t, s_1) \leq d$. 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 $h_{i,2} = \overline{t}0^{(d+1)(i-2)}1^{d+1}0^{(d+1)(n-i)}$. Hence, for each $1 \leq i \leq n$, the h_j with $0 \leq j \leq n$ and $dist(h_j, h_{i,2}) \leq d$ has to be h_i because of the different locations of the $d+1$ 1s in the last $(d+1)n$ bits of $h_1, \ldots, h_n$. Therefore, $dist(t, s_i) = dist(\overline{t}, \overline{s_i}) \leq d$. This completes the proof of the claim and hence that 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, H_R, H_R, d). Otherwise, we report that no such integer d exists. For convenience, we call the minimization version the *closest shared center problem*.

3 An Approximation Algorithm for the CSC Problem

Throughout this section, let $\mathcal{I} = (D_R, N_R, H_R)$ be an instance of the CSC problem, where $D_R = \{g_1, g_2, \ldots, g_k\}$, $N_R = \{g_{k+1}, g_{k+2}, \ldots, g_n\}$, and $H_R = \{h_1, h_2, \ldots, h_m\}$. 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 $\mathcal{I}$. Section 3.1 is devoted to testing if valid radii exist for $\mathcal{I}$. Note that if d is a valid radius for $\mathcal{I}$, then so are integers larger than d. We say that an integer d is a *semioptimal radius* for $\mathcal{I}$ if d is a valid radius for $\mathcal{I}$ and is at most twice the smallest valid radius for $\mathcal{I}$. After knowing the existence of a valid radius for $\mathcal{I}$, we want to find a semioptimal radius d for $\mathcal{I}$ 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 $\mathcal{I}$ if and only if L is a valid radius for $\mathcal{I}$. So, we consider how to test if L is a valid radius for $\mathcal{I}$. 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 $\mathcal{I}$. 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 $q \in \overline{U}$ 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 $q \in \overline{U}$ as the *center letter* at position q. Because we only care if L is a valid radius for $\mathcal{I}$ 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 $\overline{U_i} \cap \overline{U}$ 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) $|\overline{U} \setminus \overline{U_i}| \leq 1$ and 2) at every position q in $\overline{U_i} \cap \overline{U}$, 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 $\mathcal{I}$. Toward 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, $|\overline{U} \setminus \overline{U_i}| \leq 1$ and $g_i|_{\overline{U_i} \cap \overline{U}} = s|_{\overline{U_i} \cap \overline{U}}$. So, if $|\overline{U} \setminus \overline{U_i}| = 0$, then $h_{i,1}|_{\overline{U}} = h_{i,2}|_{\overline{U}} = s|_{\overline{U}}$, a contradiction against Condition C3b in Section 2. Thus, we may assume that

$|\overline{U} \setminus \overline{U_i}| = 1$. Let q be the unique position in $\overline{U} \setminus \overline{U_i}$. 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, $h_{i,1}|_{\overline{U}} = s|_{\overline{U}}$, while in the latter case, $h_{i,2}|_{\overline{U}} = s|_{\overline{U}}$. 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 $|\overline{U_i} \cap \overline{U}| \leq |\overline{U}| - 2$.

Under Conditions A1 and A2, L is a valid radius for $\mathcal{I}$. 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 $q \in \overline{U}$ is the center letter at position q. We let the center index p in the solution be any integer in $\{1, \dots, m\}$. For each $g_i \in D_R$, we obtain the unique haplotype pair $(h_{i,1}, h_{i,2})$ for g_i with $h_{i,1} = s$. 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 $\overline{U} \setminus \overline{U_i}$, 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 $\mathcal{I}$.

3.2 Computing a Semioptimal Radius and a Solution

Throughout this section, we assume that L is a valid radius for $\mathcal{I}$. So, Conditions A1 and A2 hold. Let d be the smallest valid radius for $\mathcal{I}$. 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 $\mathcal{I}$ 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 $s_1, s_2, \dots, s_m$ defined as follows:

- For each $p \in \{1, 2, \dots, m\}$, 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 $q \in \overline{U}$ is the center letter at position q.

Basically, our algorithm will select an appropriate s_p among $s_1, s_2, \dots, s_m$ 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 $p \in \{1, 2, \dots, m\}$ and every $i \in \{1, 2, \dots, k\}$, 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 $i \in \{1, 2, \dots, k\}$, where $d_{p,i} = \min_{1 \leq j \leq m} dist(h_{p,i,2}, h_j)$.*

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 $p \in \{1, \dots, m\}$, and a haplotype pair $(h_{i,1}, h_{i,2})$ for each $g_i \in D_R \cup N_R$ satisfying Conditions C1 through C3 in Section 2. Obviously, $dist(s_p, h_p) \leq dist(s, h_p) \leq d$. Moreover, for every $g_i \in D_R$, $dist(h_{p,i,2}, h_{i,2}) = dist(s_p, s) \leq d$. Therefore,

$$dist(h_{p,i,2}, h_{\ell_i}) \leq dist(h_{p,i,2}, h_{i,2}) + dist(h_{i,2}, h_{\ell_i}) \leq 2d,$$

where ℓ_i is the integer specified in Condition 2 in Section 2. Now, since $d_{p,i} \leq dist(h_{p,i,2}, h_{\ell_i})$, $d_{p,i} \leq 2d$. □

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 $i \in \{k+1, k+2, \dots, n\}$ and each $j \in \{1, 2, \dots, m\}$, 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 $i \in \{k+1, k+2, \dots, n\}$, $j \in \{1, 2, \dots, m\}$, and $j' \in \{1, 2, \dots, m\}$, 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 $d_{i,j,j'} = 1 + \max\{d'_{i,j}, d'_{i,j'}, \lceil 0.5(d'_{i,j} + d'_{i,j'} + |S_{i,j,j'}|)\rceil\}$.
- For each $p \in \{1, 2, \dots, m\}$ and each $i \in \{k+1, k+2, \dots, n\}$, let $d_{p,i} = \min_{(j,j')} d_{i,j,j'}$, where j and j' range over all integers in $\{1, 2, \dots, m\} \setminus \{p\}$.

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 $i \in \{k+1, k+2, \dots, n\}$, $j \in \{1, 2, \dots, m\}$, and $j' \in \{1, 2, \dots, m\}$, we can construct a haplotype pair $(h_{i,j,j',1}, h_{i,j,j',2})$ for g_i in $O(L)$ time such that $dist(h_{i,j,j',1}, h_j) \leq d_{i,j,j'}$, $dist(h_{i,j,j',2}, h_{j'}) \leq d_{i,j,j'}$, there is at least one position $q_1 \in \overline{U}$ at which the letter of $h_{i,j,j',1}$ is not the center letter, and there is at least one position $q_2 \in \overline{U}$ at which the letter of $h_{i,j,j',2}$ is not the center letter.*

Proof. Fix a triple (i, j, j') with $i \in \{k+1, k+2, \dots, n\}$, $j \in \{1, 2, \dots, m\}$, and $j' \in \{1, 2, \dots, m\}$. For convenience, let $\bar{d}_{i,j,j'} = d_{i,j,j'} - 1$. The remainder of the proof can be sketched as follows: First, we show how to construct a haplotype pair $(h_{i,j,j',1}, h_{i,j,j',2})$ for each $g_i \in N_R$ such that $dist(h_{i,j,j',1}, h_j) \leq \bar{d}_{i,j,j'}$ and $dist(h_{i,j,j',2}, h_{j'}) \leq \bar{d}_{i,j,j'}$. Unfortunately, such a pair $(h_{i,j,j',1}, h_{i,j,j',2})$ is not necessarily what we need, because it might be the case that 1) the letter of $h_{i,j,j',1}$ at every position $q \in \overline{U}$ is the center letter at position q or 2) the letter of $h_{i,j,j',2}$ at every position $q \in \overline{U}$ is the center letter at position q. So, we then show that if this bad case occurs, then it suffices to modify $h_{i,j,j',1}$ and $h_{i,j,j',2}$ by first selecting a suitable position $q \in \overline{U}$ and further switching the letters of $h_{i,j,j',1}$ and $h_{i,j,j',2}$ at position q. Note that this modification can increase $dist(h_{i,j,j',1}, h_j)$ and $dist(h_{i,j,j',2}, h_{j'})$ each by at most 1, implying that we now have $dist(h_{i,j,j',1}, h_j) \leq d_{i,j,j'}$ and $dist(h_{i,j,j',2}, h_{j'}) \leq d_{i,j,j'}$. Thus, after this modification, $(h_{i,j,j',1}, h_{i,j,j',2})$ becomes a required haplotype pair for g_i.

We next detail the proof. By definition, $d'_{i,j} \leq \bar{d}_{i,j,j'}$, $d'_{i,j'} \leq \bar{d}_{i,j,j'}$, and $d'_{i,j} + d'_{i,j'} + |S_{i,j,j'}| \leq 2\bar{d}_{i,j,j'}$. So, we can easily partition $S_{i,j,j'}$ into two subsets $\hat{S}_{i,j,j'}$ and $\tilde{S}_{i,j,j'}$ such that $d'_{i,j} + |\hat{S}_{i,j,j'}| \leq \bar{d}_{i,j,j'}$ and $d'_{i,j'} + |\tilde{S}_{i,j,j'}| \leq \bar{d}_{i,j,j'}$. Thus, we can obtain a required haplotype pair $(h_{i,j,j',1}, h_{i,j,j',2})$ for g_i by performing the following steps in turn:

1. For each decided position q of g_i, set the letters of $h_{i,j,j',1}$ and $h_{i,j,j',2}$ at position q to be the letter of g_i at position q.
2. For each undecided position q of g_i not contained in $S_{i,j,j'}$, set the letters of $h_{i,j,j',1}$ and $h_{i,j,j',2}$ at position q to be the letters of h_j and $h_{j'}$ at position q, respectively.
3. For each undecided position q in $\hat{S}_{i,j,j'}$, set the letter of $h_{i,j,j',2}$ at position q to be the letter of $h_{j'}$ at position q and set the letter of $h_{i,j,j',1}$ at position q to be the letter in $\{0,1\}$ different from the letter of h_j at position q. (*Note:* After this step, $dist(h_{i,j,j',1}, h_j) = d'_{i,j} + |\hat{S}_{i,j,j'}| \leq \bar{d}_{i,j,j'}$.)
4. For each undecided position q in $\tilde{S}_{i,j,j'}$, set the letter of $h_{i,j,j',1}$ at position q to be the letter of h_j at position q and set the letter of $h_{i,j,j',2}$ at position q to be the letter in $\{0,1\}$ different from the letter of $h_{j'}$ at position q. (*Note:* After this step, $dist(h_{i,j,j',2}, h_{j'}) = d'_{i,j'} + |\tilde{S}_{i,j,j'}| \leq \bar{d}_{i,j,j'}$.)
5. If a) the letter of $h_{i,j,j',1}$ at every position $q \in \overline{U}$ is the center letter at position q, or b) the letter of $h_{i,j,j',2}$ at every position $q \in \overline{U}$ is the center letter at position q, then choose an arbitrary position $q \in \overline{U} \setminus \overline{U_i}$ and switch the letters of $h_{i,j,j',1}$ and $h_{i,j,j',2}$ at position q. (*Note:* After this step, $dist(h_{i,j,j',1}, h_j) \leq 1 + d'_{i,j} + |\hat{S}_{i,j,j'}| \leq 1 + \bar{d}_{i,j,j'} = d_{i,j,j'}$ and similarly $dist(h_{i,j,j',2}, h_j) \leq d_{i,j,j'}$. Moreover, for every position $q \in \overline{U} \setminus \overline{U_i}$, either the letter of $h_{i,j,j',1}$ at position q is not the center letter at position q, or the letter of $h_{i,j,j',2}$ at position q is not the center letter at position q. Thus, if $\overline{U} \setminus \overline{U_i} \neq \emptyset$, then it is impossible that both Conditions a) and b) in Step 5 hold.)

To finish the proof of the lemma, it suffices to show that after Step 5, the letter of $h_{i,j,j',1}$ at some position $q \in \overline{U}$ is not the center letter at position q and the letter of $h_{i,j,j',2}$ at some position $q' \in \overline{U}$ is not the center letter at position q'. To this end, suppose that Condition a) or b) in Step 5 holds. Then, g_i is not free. Moreover, by Condition A2, g_i is not dead. Hence, $|\overline{U} \setminus \overline{U_i}| \geq 2$. Consider two arbitrary positions q and q' in $\overline{U} \setminus \overline{U_i}$. Since q is an undecided position of g_i, either the letter of $h_{i,j,j',1}$ at position q is not the center letter at position q or the letter of $h_{i,j,j',2}$ at position q is not the center letter at position q. Similarly, either the letter of $h_{i,j,j',1}$ at position q' is not the center letter at position q' or the letter of $h_{i,j,j',2}$ at position q' is not the center letter at position q'. Consequently, because Conditions a) or b) holds, **either** the letters of $h_{i,j,j',1}$ at positions q and q' are the center letters at positions q and q' but the letters of $h_{i,j,j',2}$ at positions q and q' are not the center letters at positions q and q', **or** the letters of $h_{i,j,j',2}$ at positions q and q' are the center letters at positions q and q' but the letters of $h_{i,j,j',1}$ at positions q and q' are not the center letters at positions q and q'. In either case, after switching the letters of $h_{i,j,j',1}$ and $h_{i,j,j',2}$ at position q, the letter of $h_{i,j,j',1}$ at some position $r \in \{q, q'\}$ is not the center letter at position r and the letter of $h_{i,j,j',2}$ at the position $r' \in \{q, q'\} \setminus \{r\}$ is not the center letter at position r'. □

Lemma 4. *If p is the center index in a solution to (D_R, N_R, H_R, d), then $d_{p,i} \leq 1 + d \leq 2d$ for every $i \in \{k+1, k+2, \ldots, n\}$.*

Proof. Consider a solution S^* to (D_R, N_R, H_R, d) that consists of a center haplotype segment s, a center index $p \in \{1, \ldots, m\}$, and a haplotype pair $(h_{i,1}, h_{i,2})$ for each $g_i \in D_R \cup N_R$ satisfying Conditions C1 through C3 in Section 2. Fix an integer $i \in \{k+1, k+2, \ldots, n\}$. Let $j = \ell_{i,1}$ and $j' = \ell_{i,2}$ be the integers in $\{1, 2, \ldots, m\} \setminus \{p\}$ that are guaranteed to exist for i by Condition C3a. Then, $dist(h_{i,1}, h_j) \leq d$ and $dist(h_{i,2}, h_{j'}) \leq d$. Moreover, by the definition of $d'_{i,j}$ and $d'_{i,j'}$, $dist(h_{i,1}, h_j) \geq d'_{i,j}$, and $dist(h_{i,2}, h_{j'}) \geq d'_{i,j'}$. Furthermore, by the definition of $S_{i,j,j'}$, we know that for every position $q \in S_{i,j,j'}$, either the letters of $h_{i,1}$ and h_j at position q differ or the letters of $h_{i,2}$ 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 $\hat{S}_{i,j,j'}$ and $\tilde{S}_{i,j,j'}$ such that $d'_{i,j} + |\hat{S}_{i,j,j'}| \leq d$ and $d'_{i,j'} + |\tilde{S}_{i,j,j'}| \leq d$. Hence, $d'_{i,j} + d'_{i,j'} + |S_{i,j,j'}| \leq 2d$.

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, $\max\{d'_{i,j}, d'_{i,j'}, \lceil 0.5(d'_{i,j} + d'_{i,j'} + |S_{i,j,j'}|)\rceil\} \leq d$ because d is an integer. Thus, by the definition of $d_{i,j,j'}$, $d_{i,j,j'} - 1 \leq d$. 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 Fig. 1.

Theorem 6. *The algorithm in Fig. 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 $O(L)$, $O(kLm)$, $O((n-k)m^2)$, 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. □

We next show that we can modify the above algorithm so that it runs in $O(nLm^2)$ time. The point is how to compute b in $O(nLm^2)$ time when we know $b \geq 1$. For each $p \in \{1, 2, \ldots, m\}$, let $d'_p = \max_{1 \leq i \leq k} d_{p,i}$ and $d''_p = \max_{k+1 \leq i \leq n} d_{p,i}$. Then, $b = \min_{1 \leq p \leq m} \max\{d'_p, d''_p\}$. Moreover, it is obvious that $b \leq L$. So, b is the smallest integer $\ell \in \{1, 2, \ldots, L\}$ such that there is a $p \in \{1, 2, \ldots, m\}$ with $d'_p \leq \ell$ and $d''_p \leq \ell$. Thus, it suffices to consider how to check if $d'_p \leq \ell$ and $d''_p \leq \ell$. Obviously, we can compute all the integers in $\{d'_p | 1 \leq p \leq m\}$ in $O(kLm^2)$ total time in advance. With d'_p known, we

```
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 0 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 step:
  2.1. Compute $d_{i,j,j'}$ and construct a haplotype pair $(h_{i,j,j',1}, h_{i,j,j',2})$ for $g_i$ as in Lemma 3.
3. For each integer $p \in \{1, 2, \ldots, m\}$, perform the following steps:
  3.1. Construct $s_p$.
  3.2. For each $i \in \{1, 2, \ldots, k\}$, perform the following steps:
    3.2.1. Construct the haplotype pair $(h_{p,i,1}, h_{p,i,2})$ for $g_i$ with $h_{p,i,1} = s_p$.
    3.2.2. Compute $d_{p,i} = \min_{1 \le j \le m} dist(h_{p,i,2}, h_j)$.
  3.3. For each $i \in \{k+1, k+2, \ldots, n\}$, perform the following steps:
    3.3.1. Compute $d_{p,i} = \min_{(j,j')} d_{i,j,j'}$, where $j$ and $j'$ range over all integers in $\{1, 2, \ldots, m\} \setminus \{p\}$.
    3.3.2. Find a pair $(j_{p,i}, j'_{p,i})$ such that $j_{p,i} \in \{1, 2, \ldots, m\} \setminus \{p\}$, $j'_{p,i} \in \{1, 2, \ldots, m\} \setminus \{p\}$, and $d_{i,j_{p,i},j'_{p,i}} = d_{p,i}$.
  3.4. Compute $d_p = \max_{1 \le i \le n} d_{p,i}$.
4. Compute $b = \min_{1 \le p \le m} d_p$ 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 in $\{(h_{p,i,1}, h_{p,i,2}) \mid 1 \le i \le k\} \cup \{(h_{p,j_{p,i},j'_{p,i},1}, h_{p,j_{p,i},j'_{p,i},2}) \mid k+1 \le i \le n\}$.
```

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

can check if $d'_p \le \ell$ in $O(1)$ time when p and ℓ are given. However, we do not know how to compute $d''_1, d''_2, \ldots, d''_m$ in $O((n-k)Lm^2)$ total time. So, we want to decide if $d''_p \le \ell$ without actually knowing d''_p. The following definition is for this purpose:

- For each triple (p, i, ℓ) with $1 \le p \le m$, $k+1 \le i \le n$, and $1 \le \ell \le L$, let $P_{p,i,\ell}$ be a set consisting of an (arbitrary) pair (j, j') of integers in $\{1, 2, \ldots, m\} \setminus \{p\}$ with $d_{i,j,j'} \le \ell$ if such a pair exists, and let $P_{p,i,\ell}$ be the empty set otherwise.

The crucial point is that $d''_p \le \ell$ if and only if all the sets in $\{P_{p,i,\ell} | k+1 \le i \le n\}$ are nonempty. Thus, it remains to consider how to compute the sets in $\{P_{p,i,\ell} | 1 \le p \le m, k+1 \le i \le n, 1 \le \ell \le L\}$ in $O(kLm^2)$ total time in advance. The following definition is for this purpose:

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

```
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 $k+1 \le i \le n$ and $1 \le \ell \le L$, compute $\mathcal{Q}_{i,\ell}$.
5. For each triple $(p, i, \ell)$ with $1 \le p \le m$, $k+1 \le i \le n$, and $1 \le \ell \le L$, compute $P_{p,i,\ell}$.
6. For $b = 1, 2, \ldots, L$ (in this order), perform the following step:
  6.1. For each integer $p$ with $1 \le p \le m$, perform the following step:
    6.1.1. If $d'_p \le b$ and $P_{p,i,b} \ne \emptyset$ 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 $\{(h_{p,i,1}, h_{p,i,2}) \mid 1 \le i \le k\} \cup \{(h_{p,j_i,j'_i,1}, h_{p,j_i,j'_i,2}) \mid k+1 \le i \le n$ and $(j_i, j'_i)$ is the pair in $P_{p,i,b}\}$.
```

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

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

1. A pair (j, j') with $j \ne p$ and $j' \ne p$ is found.
2. Already $2m$ pairs in $\mathcal{Q}_{i,\ell}$ or all the pairs in $\mathcal{Q}_{i,\ell}$ have been scanned.

If Condition 1 holds, then we can let $P_{p,i,\ell} = \{(j, j')\}$. Otherwise, we can let $P_{p,i,\ell} = \emptyset$ because among any subset of $2m$ pairs in $\mathcal{Q}_{i,\ell}$, there is at least one pair (j, j') with $j \ne p$ and $j' \ne p$.

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

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 3.2 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 3.2 into a decision algorithm which is shown in Fig. 3. In the decision algorithm, we have a user defined value d as part of the input and the algorithm

```
Input:   An instance $(D_R, N_R, H_R, d)$ of the SC problem.
Output:  "Yes" if a solution to $(D_R, N_R, H_R, 2d)$ exists; "no" otherwise.
1 – 5.  Same as those of the modified algorithm in Section 3.
6.  For each integer $p \in \{1, 2, \dots, m\}$, perform the following step:
  6.1.  If the following conditions (a) and (b) hold, then return "yes".
        (a) For each $i \in \{k+1, k+2, \dots, n\}$, $P_{p,i,d} \neq \emptyset$.
        (b) For each $g_i \in D_R$, there is an integer $\ell_i \in \{1, \dots, m\}$ such that
            the haplotype pair $(s_p, h_{p,i,2})$ for $g_i$ satisfies that
            $dist(s_p|_{\overline{U}}, h_p|_{\overline{U}}) \leq d$,
            $dist(h_{p,i,2}|_{\overline{U}}, h_{\ell_i}|_{\overline{U}}) \leq d$, and
            $dist(s_p, h_p) + dist(h_{p,i,2}, h_{\ell_i})$
            $= dist(s_p|_{\overline{U}}, h_p|_{\overline{U}}) + dist(h_{p,i,2}|_{\overline{U}}, h_{\ell_i}|_{\overline{U}}) + (|U| - dist(h_{\ell_i}|_U, h_p|_U)) \leq 2d$.
7.  Return "no".
```

Fig. 3. A decision 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 $dist(s_p, h_p) + dist(h_{p,i,2}, h_{\ell_i}) \leq 2d$ is much stronger than the inequality $dist(h_{p,i,2}, h_{\ell_i}) \leq 2d$ 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 $dist(s_p|_{\overline{U}}, h_p|_{\overline{U}}) = dist(h_{p,i,2}|_{\overline{U}}, h_{\ell_i}|_{\overline{U}}) = d$ and $(|U| - dist(h_{\ell_i}|_U, h_p|_U)) = 1$, the inequality $dist(s_p, h_p) + dist(h_{p,i,2}, h_{\ell_i}) \leq 2d$ 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 $\mathcal{R}$ of length-L regions by cutting C at the positions $L, 2 \cdot L, \dots, \lfloor c/L \rfloor \cdot L$, where c is the number of SNPs in C. In our experiments, we always fix $L = 500$. For each region $R \in \mathcal{R}$, 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, 500] 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 $\mathcal{V}$ 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 show that Heuristic 1 often outputs several disjoint mutation regions 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 $\mathcal{V}_1$ of mutation regions found by Heuristic 1 as follows: Whenever $\mathcal{V}_1$ contains two regions R_1 and R_2 that are at most $7L$ SNP sites apart on the chromosome C, we modify $\mathcal{V}_1$ by replacing R_1 and R_2 with the smallest region of C that contains both R_1 and R_2. After modifying $\mathcal{V}_1$ 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 show 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 $\mathcal{V}_2$ of mutation regions found by Heuristic 2 as follows: For each region $R \in \mathcal{V}_2$, 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 $\mathcal{R}'$ of four regions each of length L; otherwise, we divide all SNP sites to the left (respectively, right) of R on C into a set $\mathcal{R}'$ of at most four regions, all of them except one are of length L. For each $R' \in \mathcal{R}'$, 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 $\mathcal{V}_2$ 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 package, one has to prepare three input files: children.ped, genotype.txt, and haplotype.phased. Here, children.ped contains the basic information about the input individuals 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 haplotype.phased corresponds to H in Section 2 and hence contains the confirmed haplotype data of some individuals in the same population as the input individuals. For the reader's convenience, we provide an example haplotype.phased which contains the haplotype

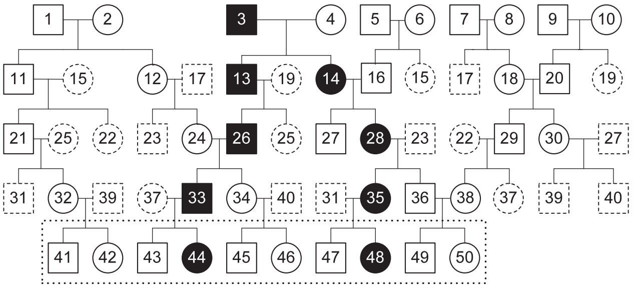

Fig. 4. A pedigree P_1 spanning five generations among which the youngest consists of two diseased individuals and eight normal individuals.

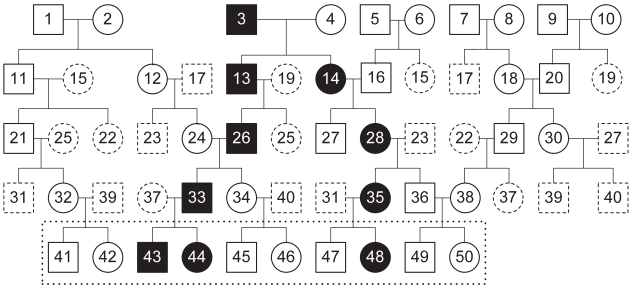

Fig. 5. A pedigree P_2 spanning five generations among which the youngest consists of three diseased individuals and seven normal individuals.

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 χ^2 model for recombination with the parameter (the degree of freedom divided by 2) equal to 4 [2] and according to the male/female averaged genetic map for chromosome 1 downloaded from HapMap (http://hapmap.org). The haplotype data of a nonfounder in the pedigree are generated to randomly inherit one strand of the four-strand chromatid bundle from each parent of the nonfounder. 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 percent. Thus, from now on we just use the unique true mutation region containing the mutation point in the rest of the paper.

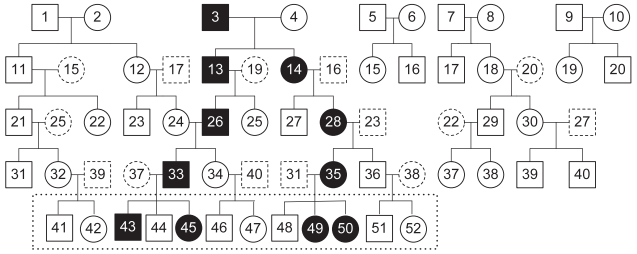

Fig. 6. A pedigree P_3 spanning five generations among which the youngest consists of four diseased individuals and eight normal individuals.

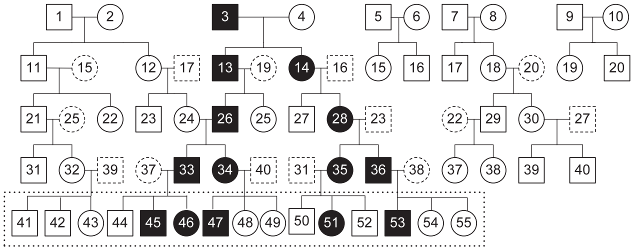

Fig. 7. A pedigree P_4 spanning five generations among which the youngest consists of five diseased individuals and 10 normal individuals.

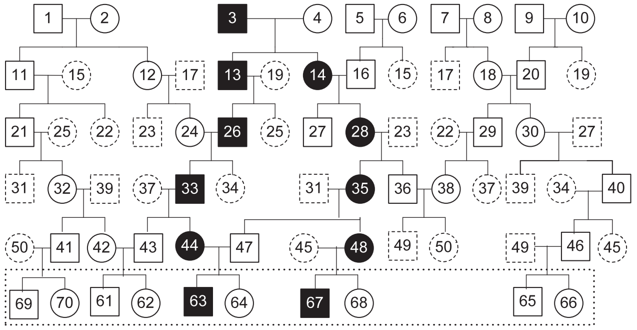

Fig. 8. A pedigree P_5 spanning six generations among which the youngest consists of two diseased individuals and eight normal individuals.

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 116,415 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 Figs. 4, 5, 6, 7, 8, and 9 and are denoted by $P_1, P_2, \ldots, P_6$, 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 five 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 six and seven generations, respectively. In each figure, a

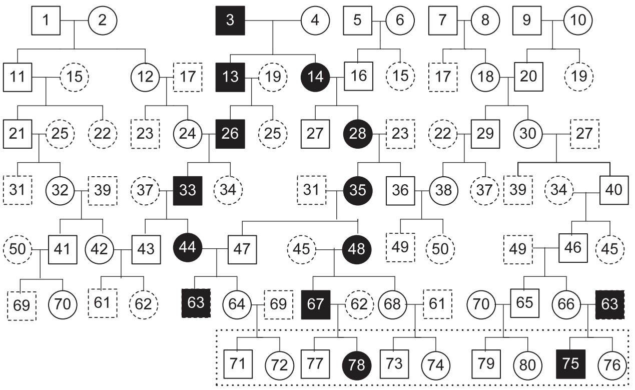

Fig. 9. A pedigree P_6 spanning seven generations among which the youngest consists of two diseased individuals and eight normal individuals.

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 "P_1" and "P_2" in Table 1, the average *recall* values for the 150 tests are 90.3 and 86.3 percent, respectively. This implies that the reported regions for P_1 and P_2 cover 90.3 and 86.3 percent of the true mutation regions, respectively. The precision values 48.3 and 58.7 percent at the bottom of the columns "P_1" and "P_2" in Table 1 indicate that the sizes of the reported mutation regions are about twice of that for the true mutation regions. This is still very useful for narrowing the region for searching the mutation gene(s), which is the main purpose here. From the columns "P_1" through "P_4" 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 P_1 through P_6. For pedigrees P_1 to P_4, where all the four pedigrees have five generations, the length of the true mutation regions decreases from 7,107 to 4,128 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 P_1, P_5, and P_6, where they all have two diseased individuals in the input set and have 5, 6,

TABLE 1
The Results for P_1 through P_6, Where Each Percentage Is Relative to 150 Tests

	Pedigrees											
	P_1		P_2		P_3		P_4		P_5		P_6	
Heuristics	*precision*	*recall*	*precision*	*recall*	*precision*	*recall*	*precision*	*recall*	*precision*	*recall*	*precision*	*recall*
longest	68.5%	51.5%	74.7%	59.3%	73.6%	37.4%	75.4%	28.7%	53.2%	50.1%	53.0%	50.9%
first 2 longest	62.5%	61.6%	69.2%	66.8%	69.0%	43.2%	74.8%	30.7%	46.7%	65.2%	46.6%	64.3%
first 3 longest	59.4%	64.6%	67.6%	69.4%	67.7%	43.9%	74.7%	30.9%	42.9%	69.6%	44.1%	67.9%
longest	64.7%	68.2%	74.3%	70.2%	69.9%	49.4%	74.5%	35.2%	48.6%	60.6%	47.6%	59.0%
first 2 longest	59.3%	77.5%	68.5%	76.8%	67.9%	53.4%	74.7%	36.1%	42.0%	74.9%	43.6%	73.2%
first 3 longest	57.2%	78.8%	67.1%	77.7%	67.4%	53.4%	74.7%	36.1%	39.4%	79.3%	41.5%	76.2%
longest	55.4%	68.7%	65.9%	72.9%	63.6%	59.1%	68.2%	50.4%	39.1%	60.5%	36.4%	50.6%
first 2 longest	50.5%	86.7%	60.0%	83.6%	62.2%	69.1%	67.8%	52.4%	33.5%	80.8%	32.3%	72.4%
first 3 longest	48.3%	90.3%	58.7%	86.3%	61.7%	69.4%	67.8%	52.5%	30.6%	87.4%	31.6%	83.7%

The table consists of three parts separated by two consecutive horizontal lines. The first (respectively, second or third) part shows the results of Heuristic 1 (respectively, Heuristic 2 or Heuristic 3).

TABLE 2
The Average Length of the True Mutation Regions and the Regions Computed by the Heuristics for P_1 through P_6

Pedigree	P_1	P_2	P_3	P_4	P_5	P_6
real region	7107	6611	5088	4128	5508	4446
longest	5440	5667	2539	1278	5997	4711
first 2 longest	7349	7073	3223	1416	8635	6624
first 3 longest	8117	7628	3371	1437	10033	7470
longest	8441	7253	3877	1738	8590	6537
first 2 longest	10298	8564	4331	1787	12031	8809
first 3 longest	10918	8905	4394	1787	13548	9628
longest	9654	8149	4905	2610	10193	7738
first 2 longest	13252	10471	5995	2798	15513	11774
first 3 longest	14713	11301	6219	2818	18597	13738

TABLE 3
The Average Running Times (in Minutes) of Heuristics 1 through 3 for P_1 through P_6

Pedigree	P_1	P_2	P_3	P_4	P_5	P_6
Heuristic 1	15.47	13.62	9.23	2.3	27.78	34.08
Heuristic 3	17.59	15.50	10.51	2.92	29.72	35.58

and 7 generations, respectively, the length of the true mutation regions also decreases when the number of generations increases.

P_5 and P_6 demonstrate the situations where there are six and seven generations, respectively. When there are two diseased individuals, the average *recall* of the 150 tests are very similar to that of five generations. The results are shown in the columns "P_5" and "P_6" in Table 1.

We have run the programs on a computer with Intel (R) Core (TM) 2 CPU 2.40 GHz and 4 GB memory. The running times of the heuristics range from several minutes to dozens of minutes for the six pedigrees. We find that Heuristics 1 and 2 have almost the same running time for all the six pedigrees, and hence we only list the average running times of Heuristics 1 and 3 in Table 3.

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 halpotypes as reference. Our experimental data show that the method can report about 50 to 90 percent 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 $(\tilde{h}_{i,1}, \tilde{h}_{i,2})$ for all $g_i \in D_R \cup N_R$, then by setting $d = 0$, the SC problem always has a solution over R. 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

The authors 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

[1] G. Abecasis, S. Cherny, W. Cookson, and L. Cardon, "Merlin-Rapid Analysis of Dense Genetic Maps Using Sparse Gene Flow Trees," *Nature Genetics,* vol. 30, pp. 97-101, 2002.

[2] K.W. Broman and J. Weber, "Characterization of Human Crossover Interference," *Am. J. Human Genetics,* vol. 66, pp. 1911-1926, 2000.

[3] Z. Cai, H. Sabaa, Y. Wang, R. Goebel, Z. Wang, J. Xu, P. Stothard, and G. Lin, "Most Parsimonious Haplotype Allele Sharing Determination," *BMC Bioinformatics,* vol. 10, article 115, 2009.

[4] K. Doi, J. Li, and T. Jiang, "Minimum Recombinant Haplotype Configuration on Tree Pedigrees," *Proc. Workshop Algorithms in Bioinformatics (WABI),* pp. 339-353, 2003.

[5] R.C. Elston and J. Stewart, "A General Model for the Analysis of Pedigree Data," *Human Heredity,* vol. 21, pp. 523-542, 1971.

[6] M. Frances and A. Litman, "On Covering Problems of Codes," *Theory of Computing Systems,* vol. 30, pp. 113-119, 1997.

[7] D.F. Gudbjartsson, K. Jonasson, M.L. Frigge, and A. Kong, "Allegro, a New Computer Program for Multipoint Linkage Analysis," *Nature Genetics,* vol. 25, pp. 12-13, 2000.

[8] L. Kruglyak, M.J. Daly, M.P. Reeve-Daly, and E.S. Lander, "Parametric and Nonparametric Linkage Analysis: A Unified Multipoint Approach," *Am. J. Human Genetics,* vol. 58, pp. 1347-1363, 1995.

[9] E. Lander and P. Green, "Construction of Multilocus Genetic Linkage Maps in Human," *Proc. Nat'l Academy of Sciences USA,* vol. 84, pp. 2363-2367, 1987.

[10] G.M. Lathrop, J.M. Lalouel, C. Julier, and J. Ott, "Strategies for Multilocus Linkage Analysis in Humans," *Proc. Nat'l Academy of Sciences USA,* vol. 81, pp. 3443-3446, 1984.

[11] I. Leykin, K. Hao, J. Cheng, N. Meyer, M.R. Pollak, R.J.H. Smith, W.H. Wong, C. Rosenow, and C. Li, "Comparative Linkage Analysis and Visualization of High-Density Oligonucleotide SNP Array Data," *BMC Genetics,* vol. 6, article 115, 2005.

[12] J. Li and T. Jiang, "Computing the Minimum Recombinant Haplotype Configuration from Incomplete Genotype Data on a Pedigree by Integer Linear Programming," *J. Computational Biology,* vol. 12, no. 6, pp. 719-739, 2005.

[13] J. Li and T. Jiang, "An Exact Solution for Finding Minimum Recombinant Haplotype Configurations on Pedigrees with Missing Data by Integer Linear Programming," *Proc. Symp. Computational Molecular Biology (RECOMB),* pp. 20-29, 2004.

[14] G. Lin, Z. Wang, L. Wang, Y.-L. Lau, W. Yang, "Identification of Linked Regions Using High-Density SNP Genotype Data in Linkage Analysis," *Bioinformatics,* vol. 12, no. 6, pp. 86-93, 2008.

[15] D. Qian and L. Beckmann, "Minimum Recombinant Haplotyping in Pedigrees," *Am. J. Human Genetics,* vol. 70, pp. 1434-1445, 2002.

[16] G. Sellick, C. Longman, J. Tolmie, R. Newbury-Ecob, L. Geenhalgh, S. Hughes, M. Whiteford, C. Carrett, and R. Houlston, "Genomewide Linkage Searches for Mendelian Disease Loci Can Be Efficiently Conducted Using High-Density snp Genotyping Arrays," *Nucleic Acids Research,* vol. 12, no. 6, pp. e164, 2004.

[17] P. Tapadar, S. Ghosh, and P.P. Majumder, "Haplotyping in Pedigrees via a Genetic Algorithm," *Human Heredity,* vol. 50, pp. 43-56, 2000.

[18] L. Wang, Z. Wang, and W. Yang, "Linked Region Detection Using High-Density SNP Genotype Data via the Minimum Recombinant Model of Pedigree Haplotype Inference," *BMC Bioinformatics,* vol. 10, article 115, 2009.

[19] J. Xiao, L. Liu, L. Xia, and T. Jiang, "Fast Elimination of Redundant Linear Equations and Reconstruction of Recombination-Free Mendelian Inheritance on a Pedigree," *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA),* pp. 655-664, 2007.

[20] W. Yang, Z. Wang, L. Wang, P.-C. Sham, P. Huang, and Y.L. Lau, "Predicting the Number and Sizes of IBD Regions among Family Members and Evaluating the Family Size Requirement for Linkage Studies," *European J. Human Genetics,* vol. 12, no. 6, pp. 1535-1543, 2008.

[21] K. Zhang, F. Sun, and H. Zhao, "Haplore: A Program for Haplotype Reconstruction in General Pedigrees without Recombination," *Bioinformatics,* vol. 21, pp. 90-103, 2005.

Wenji Ma received the bachelor's degree in software engineering from Shandong University, Jinan, Shandong, China in 2009 and is now working toward the PhD degree in the Department of Computer Science, City University of Hong Kong. Her research interests include algorithms and bioinformatics.

Yong Yang received the 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.