An Efficient Algorithm for Haplotype Inference on Pedigrees with Recombinations and Mutations

Yuri Pirola, Paola Bonizzoni, and Tao Jiang

Abstract—Haplotype Inference (HI) is a computational challenge of crucial importance in a range of genetic studies. Pedigrees allow to infer haplotypes from genotypes more accurately than population data, since Mendelian inheritance restricts the set of possible solutions. In this work, we define a new HI problem on pedigrees, called Minimum-Change Haplotype Configuration (MCHC) problem, that allows two types of genetic variation events: recombinations and mutations. Our new formulation extends the Minimum-Recombinant Haplotype Configuration (MRHC) problem, that has been proposed in the literature to overcome the limitations of classic statistical haplotyping methods. Our contribution is twofold. First, we prove that the MCHC problem is APX-hard under several restrictions. Secondly, we propose an efficient and accurate heuristic algorithm for MCHC based on an L-reduction to a well-known coding problem. Our heuristic can also be used to solve the original MRHC problem and can take advantage of additional knowledge about the input genotypes. Moreover, the L-reduction proves for the first time that MCHC and MRHC are $O\left(\frac{m^2}{\log m}\right)$-approximable on general pedigrees, where $n$ is the pedigree size and $m$ is the genotype length. Finally, we present an extensive experimental evaluation and comparison of our heuristic algorithm with several other state-of-the-art methods for HI on pedigrees.

Index Terms—algorithms, haplotyping, haplotype inference, pedigree, recombinations, mutations

1 Motivations

After the first draft of the human genome was published in 2000, a lot of research efforts have been devoted to the discovery of genetic differences among same-species individuals and to the characterization of their impact to the expression of different phenotypic traits such as disease susceptibility or drug resistance. Most of these efforts are driven by the International HapMap Project [1], which discovered, investigated and characterized millions of genomic positions (called loci or sites) where different individuals carry different genetic subsequences (called alleles). In practice, unordered pairs of alleles coming from both parents of each individual studied are routinely collected, since determining the parental source of each allele is too time-consuming and expensive to be performed on large studies [2]. The pairs of alleles located at a given set of loci of an individual are called the (multi-locus) genotype of the individual, while the sequence of alleles that were inherited from a single parent is called a haplotype. The advance of high-throughput and high-density genotyping technologies, combined with a consistent reduction of genotyping costs, had led to a great abundance of genotypic data. Such genotypes (also called SNP genotypes) are generally biallelic (i.e., at each locus only two distinct alleles are observed in the population) and they will be the focus of this work. A number of association studies based on SNP genotypes have been carried out but, since haplotypes substantially increase the power of genetic variation studies [3], accurate and efficient computational prediction of haplotypes from genotypes is highly desirable. Mendelian inheritance laws, which govern the transmission of genetic material from parents to children, have been effectively used to improve the accuracy of haplotyping methods. However, the increasing density and length of SNP genotypes challenge classic statistics-based methods (such as Lander-Green [4] and Elson-Stewart [5] methods) because they do not scale well on large datasets and they do not take directly into account the presence of Linkage Disequilibrium among loci in the founder population. Combinatorial formulations have been proposed to overcome such limitations. Among them, the most popular formulation is represented by the Minimum-Recombinant Haplotype Configuration (MRHC) problem [6], [7]. The aim of this formulation is the computation of a haplotype configuration which is consistent with an input genotyped pedigree and induces the minimum number of recombinations. The formulation naturally arises since recombinations are the most common form of variation events. However, with the progressive increase of the size of genetic variation studies, the incidence of other types of variation events (such as mutations) will inevitably become noticeable.

The above observation motivates the work in this paper, where the Haplotype Inference (HI) problem on pedigrees admitting recombination and mutation events, called Minimum-Change Haplotype Configuration (MCHC), is studied. Polynomial-time exact algorithms for MCHC are unlikely to exist since it is...
possible to prove that MCHC is APX-hard even on simple instances. The main contribution of this paper is an efficient and accurate heuristic algorithm for MCHC. Our algorithm is based on an approximation-preserving reduction of MCHC (namely an L-reduction [8, Def. 8.4]) to a fundamental problem of coding theory: the Nearest Codeword Problem (NCP) [8, probl. MS3]. Although NCP is theoretically hard to approximate [9], there exists several heuristics that compute near-optimal solutions of NCP in practice [10]. Our idea is to transform the instance of MCHC to an instance of NCP, to solve it with a custom-tailored version of a heuristic for NCP, and, finally, to reconstruct a solution of the original instance of MCHC from the solution of NCP. Our L-reduction guarantees that the transformation of the instance and the reconstruction of the solution are performed in polynomial-time while preserving the solution cost.

The work is structured as follows. First, in Section 2, we formalize the Minimum-Change Haplotype Configuration problem and define the related basic terminology. In Section 3, we prove that MCHC is APX-hard under several tight restrictions, and that the problem remains APX-hard even if we admit only mutation events. Then, in Section 4, we present a heuristic algorithm based on an L-reduction from MCHC to NCP. Based on such a reduction, we also prove the first approximation bound for MCHC and MRHC on unconstrained pedigrees. Finally, in Section 5, we discuss the results of an experimental evaluation of our algorithm under several simulated scenarios and compare the accuracy and efficiency of our algorithm with those of several state-of-the-art methods for HI on pedigrees in the literature.

2 The Computational Problem

In this section we define the basic concepts and formalize the computational problem that will be studied in the rest of the work.

A pedigree graph is an oriented acyclic graph \( P = (V, E) \) such that (i) vertices correspond to individuals and are partitioned into male and female vertices (i.e., \( V = M \cup F \), with \( M \) and \( F \) disjoint), (ii) each vertex has indegree 0 or 2, and (iii) if a vertex has indegree 2, then one edge must come from a male node and the other from a female node. For each edge \((p, c) \in E\), we say that \( p \) is a parent of \( c \) and \( c \) is an offspring (or child) of \( p \). More precisely, we say that \( p \) is the father (mother, resp.) of \( c \) if \( p \) is male (female, resp.). A trio is a triple \((f, c, m)\) where \( f \) is the father and \( m \) is the mother of \( c \). Individual \( f \) and individual \( m \) are said to be mates in such a trio.

A pedigree graph contains a mating loop if there exists two nodes \( a \) and \( d \) such that they are connected by two distinct paths. A pedigree graph is a tree pedigree if it does not contain mating loops. A pedigree graph is a binary tree pedigree if each individual has at most one offspring.

Let \( \Sigma \) be an ordered set \( \{l_1, \ldots, l_m\} \) of \( m \) loci and \( c \) an individual of the pedigree \( P \). A haplotype of individual \( c \) is an \( m \)-dimensional vector over the set \( \{0, 1\} \). The genotype \( g_c \) of individual \( c \) is an \( m \)-dimensional vector over the set \( \{0, 1\} \), where the \( i \)-th element (denoted with \( g_{c,i} \)) represents the pair of alleles that individual \( c \) possesses at locus \( l_i \). We follow the convention of encoding pair \( \{0, 0\} \) as 0, \( \{1, 1\} \) as 1, and \( \{0, 1\} \) as 2. An individual \( c \) is said to be heterozygous in a given locus \( i \) if \( g_{c,i} = 2 \), homozygous otherwise.

A genotyped (haplotyped, respectively) pedigree is a pedigree such that every individual has been associated with a genotype (an ordered pair of haplotypes, respectively). We use \( g_c \) to denote the genotype associated with an individual \( c \) of a genotyped pedigree and \( \langle h_{c,0}^0, h_{c,1}^1 \rangle \) the haplotypes associated with an individual \( c \) of a haplotyped pedigree. Moreover, we say that \( h_{c,0}^0 \) is the paternal haplotype of \( c \) and \( h_{c,1}^1 \) is the maternal haplotype of \( c \). A genotyped pedigree is a \( m \)-locus pedigree if its members are associated with a genotype defined on a set of \( m \) loci.

A haplotyped pedigree \( P_h \) is consistent with a genotyped pedigree \( P_g \) of the same set of individuals if for each individual \( c \), the genotype \( g_c \) is resolved by the pair of haplotypes \( \langle h_{c,0}^0, h_{c,1}^1 \rangle \). An individual is called a founder if its indegree is 0. Otherwise it is called a non-founder. The grandparental source vector of a non-founder individual \( c \) w.r.t. one of its parents \( p \) is an \( m \)-long binary vector \( s_{p,c} \) defined as follows. Let \( l_i \) be a locus of \( \Sigma \). If \( p \) is the father (mother, resp.) of \( c \), then \( s_{p,c,i} = 0 \) if the allele of the paternal (maternal, resp.) haplotype of \( c \) at locus \( l_i \) has been inherited from the paternal haplotype of \( p \). On the other hand, \( s_{p,c,i} = 1 \) if the allele has been inherited from the maternal haplotype of \( p \). Given a genotyped pedigree \( P_g \), a (consistent) haplotype configuration of \( P_g \) is a pair \((P_h, S)\) where \( P_h \) is a (consistent) haplotyped pedigree of \( P_g \) and \( S \) an assignment of two grandparental source vectors to each individual of \( P \).

The Haplotype Inference (HI) problem on pedigrees asks for a haplotype configuration (or the set of haplotype configurations) consistent with a given genotyped pedigree. However, since there can exist an exponential number of consistent haplotype configurations, additional constraints are generally imposed. A particularly successful approach is the formulation that attempts to minimize the number of genetic variation events that are induced in the resulting haplotype pedigree [6], [7].

Two types of variation events will be considered, recombinations and mutations, defined as follows. Let \((P_h, S)\) be a consistent haplotype configuration of a genotyped pedigree \( P_g \). The haplotype configuration induces (or contains) a recombination at locus \( l_i \) between an individual \( c \) and one of its parents \( p \) if \( s_{p,c,i} \neq s_{p,c,i+1} \). The haplotype configuration induces (or contains) a mutation at locus \( l_i \) between \( c \) and its parent \( p \) if \( h_{c,i}^j \neq h_{p,i}^j \) where \( s = s_{p,c,i} \) and \( j = 0 \) \((j = 1, \text{resp.})\) if \( p \) is the father (mother, resp.) of \( c \). If a mutation occurs between \( c \) and its father (mother, resp.), then we say that \( c \) has a mutation in the paternal (maternal, resp.) haplotype.

In this work we are interested in the computational problem of computing a haplotype configuration that is
consistent with a given genotyped pedigree and that induces the minimum number of variation events. We call such a problem **MINIMUM-CHANGE HAPLOTYPE CONFIGURATION (MCHC)** problem. The MCHC problem is a generalization of two problems proposed in the literature: the **MINIMUM-RECOMBINANT HAPLOTYPE CONFIGURATION (MRHC)** problem (where only recombinations are allowed [7]), and the **MINIMUM-MUTATION HAPLOTYPE CONFIGURATION (MMHC)** problem (where only mutations are allowed [11]). Differently from [11], in the following we do not restrict the number of mutations at each locus (among all individuals) to be at most one.

### 3 Computational Complexity

The computational complexity and the approximation hardness of various cases of the **MINIMUM-RECOMBINANT HAPLOTYPE CONFIGURATION (MRHC)** problem have been extensively studied by Liu et al. [12]. They essentially present three results: (i) MRHC is NP-hard even for simple pedigrees (namely, for binary-tree pedigrees), (ii) MRHC with missing data (i.e., loci where the genotype of an individual is not known) cannot be approximated, and (iii) MRHC is APX-hard even for simple instances (2-locus pedigrees and tree pedigrees).

In this section we extend their work by studying the computational complexity of the MMHC problem and the MCHC problem. In particular we designed reductions that use simple instances of MCHC and MMHC in order to highlight that the problems are hard to solve (and approximate) even on tight restrictions. We succeed in our aim: In fact we prove (Sect. 3.1) that MMHC on binary-tree pedigrees (binary-tree-MMHC) is APX-hard and, based on the work of Liu et al., we also show (Sect. 3.2) that MMHC and MCHC are APX-hard even on 2-locus pedigrees. These results obviously imply the APX-hardness of MCHC and MMHC on general pedigrees.

#### 3.1 binary-tree-MMHC is APX-hard

Here we present an L-reduction [8, Def. 8.4] from the **MINIMUM EDGE-BIPARTIZATION** problem on cubic graphs (MIN EDGE-BIPARTIZATION-R3) to the MMHC problem. As a consequence we obtain that MMHC is APX-hard.

Before describing the reduction, we define the **MINIMUM EDGE-BIPARTIZATION** problem.

**Problem 1** (optimization version of GT25 [13]).

**MINIMUM EDGE-BIPARTIZATION**.

*Input*: An unoriented graph $G = (V, E)$.

*Output*: A minimum subset $E'$ of $E$ such that $G' = (V, E \setminus E')$ is bipartite.

**MINIMUM EDGE-BIPARTIZATION** is NP-hard even if we restrict the input graph to be cubic (a graph which the degree of each vertex is exactly 3) [14]. From the APX-hardness of the Maximum Cut problem [8] on cubic graphs, it is also possible to derive the APX-hardness of **MINIMUM EDGE-BIPARTIZATION** (as in [15]).

In the following we will start by describing the gadget used in the reduction from **MINIMUM EDGE-BIPARTIZATION**-R3 to MMHC and then we will formally prove that the reduction is an L-reduction.

#### 3.1.1 Description of the gadget

Let $G = (V, E)$ be a cubic graph, and let $n$ and $m$ be the cardinality of $V$ and $E$, respectively. Define the set of loci as the set $L := \{v_i \mid v_i \in V\}$. First of all we need a set of gadgets to encode the edge set. An edge $e_k = (v_i, v_j)$ of $G$ is represented by a trio $g_{e_k} = (F_k, E_k, M_k)$ where $F_k$ and $M_k$ are founders. The genotype of individual $F_k$ is set to 1 in locus $v_i$, 0 in locus $v_j$, and 2 in all the other positions. The genotype of individual $M_k$, instead, is set to 0 in locus $v_i$, 1 in locus $v_j$, and 2 in all the other positions. Finally, individual $E_k$ is heterozygous in every locus. For simplicity, in the remainder we say that a locus $i$ is 0-homozygous (1-homozygous, respectively) in a given individual if the genotype of the individual at locus $i$ is 0 (1, respectively).

The edge-gadgets $g_{e_k}$ are connected to a sort of “backbone” that represents the entire graph. The “backbone” is composed by $m + 1$ individuals $I_k$ with $k = 0 \ldots m$ such that $I_k$ and $E_k$ are the parents of $I_k$ for all $1 \leq k \leq m$. (By convention we assume that $I_0$ are male nodes, while $E_k$ are female.) Finally we have to specify the genotypes of individuals $I_k$. For all $k = 0 \ldots m$, individuals $I_k$ are heterozygous at every locus.

Figure 1 illustrates the complete gadget. Given a cubic graph $G$, we indicate with $P_G$ the complete gadget (i.e., the pedigree graph) that encodes $G$.

![Figure 1. The gadget used in the reduction. The part in the gray circle is the edge-gadget $g_{e_k}$ representing edge $e_k = (v_i, v_j)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS
3.1.2 The L-reduction

The reduction from Min Edge-Bipartization-R3 to MMHC is composed by the following two lemmas.

**Lemma 1.** Let $G = (V, E)$ be a cubic graph and let $E' \subseteq E$ be a solution of Min Edge-Bipartization-R3 on $G$. Then, a haplotype configuration $H$ for $P_G$ with at most $|E'|$ mutations can be computed in polynomial time.

**Lemma 2.** Let $G = (V, E)$ be a cubic graph and let $H$ be a haplotype configuration for $P_G$ that requires $l$ mutations. Then, a solution $E'$ of Min Edge-Bipartization-R3 on $G$ such that $|E'| \leq l$ can be computed in polynomial time.

Clearly the following corollary can be easily derived from the previous lemmas and from the APX-hardness of Min Edge-Bipartization-R3.

**Corollary 3.** Min Edge-Bipartization-R3 is L-reducible to MMHC, thus MMHC is APX-hard.

Lemma 1 can be easily proved by constructing a haplotype configuration with at most $|E'|$ mutations based on the given solution. The second lemma (Lemma 2), instead, is a little bit trickier than the first one, and requires the transformation of the given haplotype configuration to another haplotype configuration where the mutations are located in specific haplotypes. After that, a Min Edge-Bipartization-R3 solution of the given cardinality can be easily reconstructed. We start to prove these lemmas by demonstrating two key properties about the gadget.

**Property 4.** Every haplotype configuration of the whole pedigree can be transformed in polynomial time in a haplotype configuration such that each trio $g_{E_k} = (F_k, E_k, M_k)$ contains no mutations. Such a haplotype configuration is called basic.

**Proof:** Suppose that the gadget $g_{E_k}$ represents edge $e_k = (v_i, v_j)$. Notice that $F_k$ and $M_k$ are founders of the pedigree and they are heterozygous at every locus but $v_i$ and $v_j$. Given a haplotype configuration that contains (at least) a mutation in trio $g_{E_k}$, we can show that such a mutation can be avoided or “moved” to a different place.

Two cases must be studied separately: either (1) the mutation does not involve locus $v_i$ or $v_j$, or (2) the mutation involves one of such loci.

In the first case, let $v$ be the locus where the haplotype of $E_k$ inherited from parent $P_k$ has been mutated. Then swap the paternal and the maternal alleles of individual $P_k$ at locus $v$. Clearly this change does not violate the genotype consistency of the individual and does not violate the Mendelian inheritance since (i) $P_k$ is a founder (thus its haplotypes are not inherited from other individuals), and (ii) it does not have other children apart from $E_k$ (thus its haplotypes are not transmitted to other individuals). Then, the two alleles at every locus except $v_i$ and $v_j$ that has been mutated can be swapped until the haplotype configuration of the trio does not contain mutations in any loci but $v_i$ or $v_j$.

In the second case, suppose, without loss of generality, that a mutation has occurred in a haplotype of $E_k$ at locus $v$. Clearly, since both parents of $E_k$ are homozygous at locus $v$, we cannot proceed as in the previous case. Instead we show how the mutation can be “pushed down” from individual $E_k$ to individual $I_k$ of the “backbone” (or, in some case, it can be completely removed). By genotype consistency, notice that if a mutation has occurred in locus $v_i$ between individual $F_k$ and $E_k$, then a mutation in locus $v_i$ has occurred also between individual $M_k$ and $E_k$ and vice versa. In fact, individuals $F_k$ and $M_k$ are homozygous at locus $v_i$ and have different alleles. Since the same locus is heterozygous in $E_k$, a mutation in the maternal haplotype implies a mutation in the paternal one, and vice versa. Now, we claim that if a third mutation has occurred in the maternal haplotype of $I_k$ at locus $v_{ij}$ that is between $E_k$ and its child $I_k$, then this mutation and the two others at the same locus of $E_k$ can be removed. The removal is obtained by swapping the paternal and the maternal alleles of $E_k$ at locus $v_{ij}$, as this site is heterozygous and its alleles are both mutated, while the two haplotypes of $I_k$ do not change. Thus removing the three mutations does not affect genotype consistency and, furthermore, also Mendelian consistency is preserved. Finally, we claim that if locus $v_i$ has not been mutated in the maternal haplotype of $I_k$, then the two mutations between $E_k$ and its parents can be replaced with a single mutation in the paternal haplotype of $I_k$. Indeed, if we swap the paternal and maternal alleles of $E_k$ at locus $v_i$ without changing the haplotypes of $I_k$, we remove the two mutations at locus $v_i$ of $E_k$ while introducing a single mutation between $E_k$ and its child $I_k$. Also in this case, genotype and Mendelian consistency are preserved since $I_k$ is the only child of $E_k$.

In all cases, we were able to compute (in polynomial time) a new haplotype configuration without adding new mutations and such that no mutations occur between two individuals of any trio $(F_k, E_k, M_k)$.

From the definition of basic haplotype configuration, we can easily derive the following statement.

**Property 5.** In a basic haplotype configuration, for all individuals $E_k$ of the gadget which represents edge $(v_i, v_j)$, the alleles on the same haplotype at loci $v_i$ and $v_j$ differ.

**Proof:** The property is a direct consequence of the fact that $F_k$ and $M_k$ have different alleles at loci $v_i$ and $v_j$, since the haplotypes of each individual $E_k$ do not contain mutations in a basic haplotype configuration.

We are now ready to prove the first lemma used in the reduction.

**Proof of Lemma 1:** Let $(V_1, V_2)$ be a bipartition of $V$ induced by the removal of $E'$. Set the paternal haplotype of every individual $I_k$ such that every locus $v$ has allele 0 if $v \in V_1$, otherwise set it to allele 1. Set the maternal haplotype as the bit-wise complement of the paternal haplotype. Finally set the haplotypes of each trio $(F_k, E_k, M_k)$ in such a way that the haplotype con-
single mutation occurs in the transmission of a haplotype $E$ to $I$. In fact, no matter which haplotypes individual $E_k$ inherits from its parents, a single mutation occurs in the transmission of a haplotype from $E_k$ to $I_k$. By Property 5, the alleles on the same haplotype at loci $v_i$ and $v_j$ are different in individual $E_k$, while they are equal in individual $I_k$ because $e_k$ belongs to $E'$. Thus, exactly one mutation is contained in the haplotype configuration for each edge $e_k \in E'$. Instead, the alleles of the loci that represent the endpoints of an edge in the graph to make it bipartite, the alleles of the loci that represent the endpoints of an edge in the graph to make it bipartite, we then obtain a solution $E'$ for MIN EDGE-BIPARTIZATION-R3. We claim that $E'$ is a solution of the proposed problem. By Property 4, we can safely remove from the graph to make it bipartite, the alleles of the loci that represent the endpoints of an edge in the graph to make it bipartite, we then obtain a solution $E'$ for MIN EDGE-BIPARTIZATION-R3. We claim that $E'$ is a solution of the proposed problem. By Property 4, we can safely remove from the graph to make it bipartite, the alleles of the loci that represent the endpoints of an edge $e_k$ which does not belong to $E'$ are different, so no mutations from $E_k$ to $I_k$ occur, which concludes the proof.

The basic idea which the proof of Lemma 2 relies on is that the haplotype configuration of individual $I_0$ naturally encodes a partition of the vertex set. By removing the edges that do not cross the bipartition, we then obtain a solution $E'$ for MIN EDGE-BIPARTIZATION-R3. We claim in the following that the haplotype configuration can be transformed to another haplotype configuration which induces mutations only on the maternal haplotype of individuals $I_k$. Each mutation of this haplotype configuration corresponds to an edge of $E'$.

Proof of Lemma 2: By Property 4, we can safely assume that the haplotype configuration $H$ is basic. In the first part of the proof, we show that mutations of the haplotype configuration $H$ can be moved to specific loci, while in the second part we show that the resulting mutations are in correspondence with edges that are removed from the graph to make it bipartite.

In the first part, we consider 4 different cases and we move (or remove) the mutations until one of the cases applies. Termination of this procedure is guaranteed by the fact that a mutation is either removed or moved from an individual $I_k$ to its child $I_{k+1}$. For the presentation of the cases, let $e_k$ be a generic edge $(v_i, v_j)$ of the cubic graph $G$.

Case 1: If the maternal haplotype of individual $I_k$ contains a mutation in a locus $v_i$ different from $v_i$ and $v_j$, then we remove the mutation by swapping the paternal and the maternal alleles at locus $v_i$ in all the individuals of the trio $g_{e_k}$ without affecting genotype or Mendelian consistency. Indeed haplotypes of $E_k$, $F_k$, and $M_k$ are heterozygous at all loci except $v_i$ and $v_j$ (Fig. 2(a)).

Case 2: If the paternal haplotype of $I_k$ contains a mutation at a locus $v_i$ different from $v_i$ and $v_j$, then we move the mutation to the paternal haplotype of $I_{k+1}$ by swapping the two alleles at locus $v_i$ of individual $I_k$ and of its maternal ancestors $E_k$, $F_k$, and $M_k$ (Fig. 2(b)).

Case 3: If both loci $v_i$ and $v_j$ are mutated in the maternal haplotype of $I_k$, then such mutations can be avoided by setting that $I_k$ inherits from the other haplotype of $E_k$ (such that loci $v_i$ and $v_j$ are the only loci which do not contain a mutation) and then removing the “newly”-created mutations (which are located in the maternal haplotype at loci different from $v_i$ and $v_j$) as in Case 1 (Fig. 2(c)).

Case 4: If the paternal haplotype of $I_k$ contains two mutations at loci $v_i$ and $v_j$, then we can move them to the paternal haplotype of $I_{k+1}$ by swapping the paternal and the maternal alleles at such loci, and then continuing as in Case 3 (Fig. 2(d)).

Let $H'$ be a haplotype configuration and let $l'$ be the number of mutations of $H'$. (Clearly $l' \leq l$, where $l$ is the number of mutations of the original haplotype configuration $H$.) Construct a bipartition $\{V_1, V_2\}$ of the vertex set $V$ as follows: $v_i \in V_1$ if the locus $v_i$ of the paternal haplotype of $I_0$ contains allele 0, otherwise $v_i \in V_2$. Let us show that bipartition $\{V_1, V_2\}$ induces a MIN EDGE-BIPARTIZATION-R3 solution $E'$ such that $|E'| = l'$.

By construction of $H'$, a mutation can occur (i) in a paternal haplotype of $I_k$, or (ii) in a maternal haplotype of $I_k$. (Suppose that $e_k = (v_i, v_j)$.) Moreover, either $v_i$ or $v_j$ has been mutated (suppose w.l.o.g. $v_i$), and the alleles at the other loci have been inherited without mutations. We now analyse separately the two cases. In case (i), the paternal haplotype of $I_k$ contains a mutation at locus $v_i$. Therefore, if the haplotype inherited from $I_{k-1}$ had different alleles at loci $v_i$ and $v_j$, then the paternal haplotype of $I_k$ would have the same allele at such loci. Since the haplotype configuration is basic, this also means that the paternal haplotype of $I_k$ contains a mutation either at locus $v_i$ or at locus $v_j$. The two mutations (the one of the paternal haplotype and the one of the maternal haplotype of $I_k$) can be safely removed by swapping two alleles at locus $v_i$ of individual $I_k$ and propagating the change to the individuals of the trio $g_{e_k}$. As a consequence, we can assume that the alleles at loci $v_i$ and $v_j$ in the paternal haplotype of $I_k$ are different and that are equal in the one haplotype of $I_{k-1}$. Please remember that the input graph is cubic, thus only three edges are incident to $v_i$. Let us denote $e_{k, v_i}$, $e_{v_i}$, and $e_{v_i'}$ the three edges incident to $v_i$. Clearly three different subcases may arise: (a) $k < t' < t''$, (b) $t' < k < t''$, and (c) $t' < t'' < k$. In the first subcase, no parental ancestor of $I_k$ is homozygous at locus $v_j$, thus we can remove the mutation by swapping the two alleles at locus $v_i$ of individual $I_k$ and propagating the change to its parental ancestors. In the second subcase, set the paternal allele at locus $v_i$ in $I_k$ equal to the paternal allele at the same locus in $I_k$ and propagate such a change to descendants and ancestors. Clearly, this operation moves the mutation on the maternal haplotype of $I_k$ and it does not create any other mutations. In the last subcase, the mutation can be moved (but not removed) to the maternal haplotype and a new mutation is created in the paternal haplotype of individual $I_{k+1}$. However, since no edge $e_t = (v_t, v_t')$ with $t > k$ exists, this new mutation can be removed by iteratively applying the procedure of Case 2 described in the first part of the proof.
In case (ii) the alleles of the maternal (and thus of the paternal) haplotype of $I_k$ at loci $v_i$ and $v_j$ coincide. Assuming that the paternal haplotype of $I_k$ at loci $v_i$ and $v_j$ is equal to (or is the complement of) the paternal haplotype of $I_0$, vertices $v_i$ and $v_j$ belong to the same set of the bipartition $\{V_1, V_2\}$, thus edge $e_k$ belongs to $E'$.

In the discussion of case (i), we have shown how to move mutations in order to put them on the maternal haplotypes of individuals $I_k$ without creating new mutations, thus we can assume that the haplotype configuration contains mutations only on the parental haplotype of the individuals $I_k$. However, the analysis of case (ii) has shown that a mutation on a paternal haplotype is in correspondence with an edge of $E'$. Moreover, it is easy to see that for each edge $e_k \in E \setminus E'$, individual $I_k$ does not contain mutations. Since the procedure that moves and removes mutations does not increase their number, we have $|E'| = l' \leq l$, where $l$ is the number of mutations of the original haplotype configuration $H$, that concludes the proof.

It is interesting to notice that, in case the input graph of MIN EDGE-BIPARTIZATION-R3 is already bipartite, the optimum solution is the empty set. Then, there exists a haplotype configuration of the genotyped pedigree that we use to encode the graph that does not contain mutations. The problem to decide if a pedigree can be haplotyped without mutations is solvable in polynomial time (cfr. [16] and others). This fact does not contrast with the complexity of the MIN EDGE-BIPARTIZATION-R3 problem: indeed also recognizing bipartite graphs can be performed in polynomial time (by a simple visit of the graph).

### 3.2 2-locus-MCHC and 2-locus-MMHC are APX-hard

In this section we show that a previous result of Liu et al. [12] can be extended to prove the APX-hardness of MCHC on a 2-locus pedigree (i.e., a pedigree in which the genotypes of the individuals have 2 loci). We also point out that the same arguments can be used to show the APX-hardness of 2-locus-MMHC, completing the results presented in the previous section. However, differently from the previous section, we remark that in this case we are not constraining the pedigree structure that can be arbitrarily complex.

We achieve the APX-hardness of 2-locus-MCHC by observing that the effects of a recombination on a 2-locus
pedigree cannot be distinguished from the effects of a mutation on the same point.

**Lemma 6.** Let \( H \) be a haplotype configuration on a 2-locus pedigree \( P \). If a recombination is present between individual \( I \) and individual \( J \) starting at locus \( l \), then the recombination can be replaced by a mutation between \( I \) and \( J \) on locus \( l \).

**Proof:** First notice that, in a 2-locus pedigree, a recombination can only occur at the second locus. Moreover, individual \( I \) has to be heterozygous at locus 2 because otherwise the recombination would not have any effect. Let \( h_j = \langle a_1, a_2 \rangle \) be the haplotype inherited by \( J \) from its parent \( I \). Since individual \( I \) is heterozygous at locus 2 and since there are only two possible alleles (denoted 0 and 1), one haplotype of individual \( I \) is \( h_i = \langle a_1, 1 - a_2 \rangle \). Clearly a mutation between \( I \) and \( J \) would replace allele \( 1 - a_2 \) with \( a_2 \), having the same effect of the recombination.

Lemma 6 permits to derive the following result.

**Corollary 7.** MCHC on a 2-locus pedigree is APX-hard.

**Corollary 8.** MMHC on a 2-locus pedigree is APX-hard.

**Proof of Corollary 7 and Corollary 8:** Liu et al. proved that there exists an L-reduction from MIN EDGE-BIPARTIZATION (called MinUncut) to MRHC on a 2-locus pedigree (Lemma 9 of [12]). They rely on a pedigree where each individual is either a founder or a son of a founder. All founders are heterozygous at every locus, thus also the converse of Lemma 6 holds. In fact it is possible to prove that a mutation at locus \( l \) from an individual that is heterozygous at locus \( l \) can be replaced by a recombination starting at \( l \). Therefore the same gadget used in [12] can be applied to prove the existence of an L-reduction from MinEdge-Bipartization to MCHC or to MMHC. As a consequence, by the APX-hardness of MIN EDGE-BIPARTIZATION [17], we obtain the APX-hardness of MCHC and of MMHC on 2-locus pedigrees.

---

**4 A Heuristic Algorithm for MCHC**

The presentation of the heuristic algorithm that we propose is divided into three parts. First, we give an extension of the system of linear equations over the field \( \mathbb{Z}_2 \) proposed by Xiao et al. [16] for representing the set of haplotype configurations that are consistent with the input genotyped pedigree. In the extended system that we propose, recombinations and mutations are explicitly modeled as variables of the equations. In the second part, we establish an L-reduction from MCHC to the well-known NEAREST CODEWORD PROBLEM (NCP) by splitting the system into two parts where one part contains only variables needed for the haplotype reconstruction and the other contains only recombination and mutation variables. Finally, we present a tailored version of a well-known heuristic algorithm for NCP. Using this heuristic, we can guarantee that a feasible solution for NCP (and hence for MCHC) is found.

### 4.1 A System of Linear Equations for MCHC

In this part, we first illustrate the linear system over \( \mathbb{Z}_2 \) proposed in [16] for the HI problem where no recombinations or mutations are permitted (i.e., the zero-recombinant haplotype configuration problem orZRHC), and then we describe how it can be extended to accommodate recombinations and mutations events. For simplicity of presentation, we denote with the symbol + the addition over \( \mathbb{Z}_2 \) instead of using \( \oplus \).

**A Linear System for ZRHC**

Computing the paternal haplotypes of all individuals is sufficient to fully describe the haplotyped pedigree because the maternal haplotype can be reconstructed from the paternal haplotype and the genotype of the individual. Therefore, we introduce a variable \( h_i[l] \) for each individual \( i \) and locus \( l \) which represents the allele present at locus \( l \) of the paternal haplotype of \( i \). Secondly, we need to represent the grandparental source. Let \( i \) be an individual and \( p \) one of its parents. Since no recombinations are admitted, the grandparental source is denoted as a single variable \( s_{p,i} \). Variable \( s_{p,i} \) is equal to 0 if \( i \) has inherited from \( p \) the paternal haplotype of \( p \), or 1 otherwise. To express concisely the linear equations, we need two additional sets of constants: the \( w \)- and the \( d \)- constants. For each locus \( l \) and individual \( i \), constant \( w_i[l] \) is equal to 0 if \( i \) is homzygous at locus \( l \), and 1 otherwise. For each locus \( l \) and pair of individuals \( p \) and \( i \) such that \( p \) is a parent of \( i \), constant \( d_{p,i}[l] \) is equal to 0 if \( p \) is the father of \( i \) and equal to \( w_i[l] \) if \( p \) is the mother of \( i \). Finally, since the paternal haplotype (and hence the maternal haplotype) is known at homozygous loci, we set \( h_i[l] = g_i[l] \) for every individual \( i \) and locus \( l \) such that \( g_i[l] \neq 2 \).

A case-by-case analysis shows that any solution of the following linear system over \( \mathbb{Z}_2 \) is a zero-recombinant haplotype configuration consistent with the genotyped pedigree (and vice versa) [16]. For all loci \( l \) and individuals \( i \),

\[
\begin{align*}
\begin{cases}
 h_p[l] + s_{p,i} \cdot w_p[l] = h_i[l] + d_{p,i}[l] & \text{for each parent } p \text{ of } i \\
n_i[l] = g_i[l] & \text{if } g_i[l] \neq 2 \\
w_i[l] = 0 & \text{if } g_i[l] \neq 2 \\
w_i[l] = 1 & \text{if } g_i[l] = 2 \\
d_{p,i}[l] = 0 & \text{if } p \text{ is the father of } i \\
d_p[l] = w_p[l] & \text{if } p \text{ is the mother of } i 
\end{cases}
\end{align*}
\]

(1)

Notice that, if the pedigree has \( n \) members and the genotypes are defined over a set of \( m \) loci, then we have \( nm \) \( h \)-variables, at most \( 2n \) \( s \)-variables, and at most \( 2nm \) equations.

**A Linear System for MCHC**

We now show how the previous linear system can be modified for representing all the consistent haplotype configurations that may contain recombinations and mutations.
To accommodate recombinations, we introduce a set of \( \delta \)-variables defined as follows. For each locus \( l \), variable \( \delta_{p,i}[l] \) is equal to 1 if a recombination has occurred at locus \( l \) between an individual \( p \) and one of its children \( i \), and 0 otherwise. The grandparental source vector of a consistent haplotype configuration can be expressed as a (linear) function of an \( s \)-variable and a subset of \( \delta \)-variables. In particular, by induction on \( l \), it is easy to prove that the grandparental source of i w.r.t. \( p \) at locus \( l \), \( s_{p,i}[l] \), is equal to \( s_{p,i} + \sum_{j=1}^{l} \delta_{p,i}[j] \). Denote as \( \Delta_{p,i}[l] \) the sum \( \sum_{j=1}^{l} \delta_{p,i}[j] \). By replacing \( s_{p,i} \) with \( (s_{p,i} + \Delta_{p,i}[l]) \) in Eq. 1, we obtain a linear system that represents all the haplotype configurations consistent with the genotyped pedigree and allows recombinaction events. Since mutations are point events that replace an allele inherited during inheritance from \( p \) to \( i \), \( \mu_{p,i}[l] \) and \( \mu_{p,i}[l] = 0 \) otherwise. The following lemma is straightforward.

**Lemma 9.** Let \( P_p \) be a genotyped pedigree. Then each solution of the system

\[
\begin{align*}
   h_p[l] + (s_{p,i} + \Delta_{p,i}[l]) \cdot w_p[l] & = h_i[l] + d_{p,i}[l] + \mu_{p,i}[l] \\
   h_i[l] & = g_i[l] \\
   w_i[l] & = 0 \\
   w_l[l] & = 1 \\
   d_{p,i}[l] & = 0 \\
   d_{p,i}[l] & = w_i[l]
\end{align*}
\]

for each parent \( p \) of \( i \) and individuals \( i \) represents a haplotype configuration consistent with \( P_p \) that admits recombinaction and mutation events. Conversely, a haplotype configuration consistent with \( P_p \) that admits recombinaction and mutation events is represented by a solution of the linear system.

By construction, a haplotype configuration that induces \( k \) variation events is represented by a solution of the linear system where exactly \( k \) \( \delta \)- and \( \mu \)-variables are non-zero.

### 4.2 Reducing MCHC to NCP

The **Nearest Codeword Problem** is the problem of coding theory that reconstructs the original codeword of a given received message by minimizing the Hamming distance between them. More formally, given an \( r \times n \) matrix \( H \) over \( \mathbb{Z}_2 \), and a column vector \( q \in \mathbb{Z}_2^n \), the **Nearest Codeword Problem** [8, probl. MS3] asks for a vector \( e \in \mathbb{Z}_2^n \) with the minimum number of non-zero entries such that \( H \cdot e = q \). The number of non-zero entries of a vector \( v \) is called the **weight** of the vector and is denoted as \( ||v|| \).

The basic idea of our reduction is to split the linear system of Lemma 9 into two linear systems: one containing only \( h \)- and \( s \)-variables, and the other one containing only \( \delta \)- and \( \mu \)-variables. The second part of the system is, directly, an instance of NCP.

Since all \( w_i[l] \) and \( d_{p,i}[l] \) assume constant (predetermined) values, we can write the linear system of Eq. 2 as the following matricial equation:

\[
A_{h,s} \cdot x_{h,s} + A_{\delta,\mu} \cdot x_{\delta,\mu} = b
\]

where: \( x_{h,s} \) is the column vector of the \( h \)- and \( s \)-variables, \( x_{\delta,\mu} \) the column vector of the \( \delta \)- and \( \mu \)-variables, \( A_{h,s} \) the \( n \times m_1 \) matrix of the coefficients of the \( h \)- and \( s \)-variables, \( A_{\delta,\mu} \) the \( n \times m_2 \) matrix of the coefficients of the \( \delta \)- and \( \mu \)-variables, and \( b \) a column vector composed by constant entries.

Let \( k \) be the rank of the matrix \( A_{h,s} \) and \( A_{\delta,\mu}^T \) be its transpose. Suppose w.l.o.g. that the first \( k \) rows of \( A_{h,s} \) are linearly independent. Now, we construct the instance of NCP associated to an instance of MCHC as follows. Let \( B = \{ v_1, \ldots, v_r \mid v_i \in \mathbb{Z}_2^n \} \) be a basis of the vector space \( \ker(A_{\delta,\mu}^T) = \{ y \in \mathbb{Z}_2^n \mid A_{\delta,\mu}^T \cdot y = 0 \} \), where \( 0 \) denotes the all-zero column vector. Collate vectors \( v_i \) to form a \( r \times n \) matrix \( V \) such that the \( i \)-th row is equal to \( v_i^T \). Then, the instance \( I' \) of NCP associated with an instance \( I = (A_{h,s}, A_{\delta,\mu}, x_{h,s}, x_{\delta,\mu}, b) \) of MCHC is the pair \( I' = (H, q) \) where \( H = VA_{\delta,\mu} \) and \( q = Vb \). Clearly, the transformation of \( I \) into \( I' \) can be computed in polynomial-time via Gaussian elimination (to compute \( V \)) and two matrix multiplications (to compute \( H \) and \( q \)).

We complete the L-reduction from MCHC to NCP by proving the following two lemmas. Lemma 10 illustrates how to reconstruct in polynomial-time a solution of an MCHC instance given a solution for the associated NCP instance, and Lemma 11 shows how to compute (in polynomial-time) a solution for an instance \( I' \) of NCP associated with an instance \( I \) of MCHC given a solution for \( I \). Since both above transformations preserve the cost of solutions, the reduction is an L-reduction with parameters \( \beta = \gamma = 1 \). See [8, Def. 8.4] for the formal definition of L-reduction and an explanation of these parameters.

**Lemma 10.** Let \( I = (A_{h,s}, A_{\delta,\mu}, x_{h,s}, x_{\delta,\mu}, b) \) be an instance of MCHC and \( I' = (H, q) \) the NCP instance associated with \( I \). Then, given a solution \( e \) of NCP on \( I' \), it is possible to compute in polynomial-time a haplotype configuration \( \tilde{x}_{h,s}, \tilde{x}_{\delta,\mu} \) of \( I \) that induces \( ||e|| \) variation events.

**Proof:** Let \( n \) be the number of rows of \( A_{h,s} \) (or \( A_{\delta,\mu} \)), \( m_1 \) the number of columns of \( A_{h,s} \), and \( m_2 \) the number of columns of \( A_{\delta,\mu} \). We want to prove that \( x_{\delta,\mu} = e \) is a partial solution of the linear system \( A_{h,s} \cdot x_{h,s} + A_{\delta,\mu} \cdot x_{\delta,\mu} = b \), or, in other words, that the linear system \( A_{h,s} \cdot x_{h,s} + A_{\delta,\mu} \cdot x_{\delta,\mu} = b + A_{\delta,\mu} \cdot e \) is consistent (i.e., it admits at least one solution). For simplicity, denote as \( M \) the matrix \( A_{h,s} \) and as \( t \) the vector \( b + A_{\delta,\mu} \cdot e \). A well-known result in linear algebra (see, for example, [18, Sect. 2.3]) states that the linear system \( M \cdot x_{h,s} = t \) is consistent iff the rank of matrix \( M \) is equal to the rank of the augmented matrix \( (M|t) \) (i.e., the matrix obtained from \( M \) by adding one new
column equal to the vector $t$). Since the rank of matrix $M$ is the maximum number of its independent rows, to achieve the consistency of the system $Mx_{h,s} = t$ we have to prove that a subset of rows of $M$ is linearly dependent iff the same subset of rows of $(M|t)$ is linearly dependent. In other words, we have to prove that, for each row vector $d \in Z^n_2$ with $d \cdot M = 0^T$ we have $d \cdot (M|t) = 0^T$ and vice versa.

$(\Rightarrow)$ By construction, vector $d^T$ is precisely an element of the subspace $\ker(A^T_{h,s})$. Recall that the construction of the instance $I'$ uses an $n \times n$ matrix $V$ whose rows form a basis of $\ker(A^T_{h,s})$. Therefore there exists a row vector $\alpha_d \in Z^n_2$ such that $\alpha_d \cdot V = d$. Since $d \cdot M = 0^T$, we have $\alpha_d \cdot V \cdot M = 0^T$. Let us consider the row vector $z := d \cdot (M|t)$. We want to show that $z = 0^T$. Since $(M|t)$ is an augmented matrix of $M$, the first $m_1$ elements of $z$ are equal to $d \cdot M$, and hence equal to zeros. The last element, instead, is equal to $d \cdot t = d \cdot (b + A_{h,s} \cdot e) = \alpha_d \cdot V \cdot b + \alpha_d \cdot V \cdot A_{h,s} \cdot e$. Since $e$ is a solution of $I'$, we have that $V' \cdot A_{h,s} \cdot e = V' \cdot b$. Thus $d \cdot t = \alpha_d \cdot V \cdot b + \alpha_d \cdot V \cdot b = 0$, and $d \cdot (M|t) = 0^T$, completing this part of the proof.

$(\Leftarrow)$ If $d \cdot (M|t) = 0^T$, we have to prove that $d \cdot M = 0^T$. As before, the first $m_1$ elements of vector $d \cdot (M|t)$ are equal to $d \cdot M$. Since $d \cdot (M|t) = 0^T$, we have $d \cdot M = 0^T$, completing the proof.

As a consequence, the rank of $M$ is equal to the rank of $(M|t)$, implying that there exists a vector $\tilde{x}_{h,s}$ such that $A_{h,s} \tilde{x}_{h,s} + A_{h,s} e = b$. In other words, the pair of vectors $(\tilde{x}_{h,s}, \tilde{x}_{h,s})$ with $\tilde{x}_{h,s} = e$ is a consistent haplotype configuration, and induces exactly $\|e\|$ variation events.

Now we prove that a solution of the NCP instance $I'$ associated with a MCHC instance $I$ can be computed in polynomial-time starting from a solution of $I$ and preserving the cost.

**Lemma 11.** Let $S = (\tilde{x}_{h,s}, \tilde{x}_{h,s})$ be a solution of MCHC on the instance $I = (A_{h,s}, A_{h,s} \tilde{x}_{h,s}, x_{h,s}, x_{h,s})$ and $I' = (H, q)$ the NCP instance associated with $I$. Then, vector $e := \tilde{x}_{h,s}$ is a solution of NCP on $I'$.

**Proof:** Since $S$ is a solution for $I$, we have $A_{h,s} \tilde{x}_{h,s} + A_{h,s} e = b$. Recall that the construction of the instance $I'$ uses a matrix $V$ whose rows form a basis of $\ker(A^T_{h,s})$. Multiplying each side of the above equation by $V$, we obtain $V' \cdot A_{h,s} \tilde{x}_{h,s} + V' \cdot A_{h,s} e = V' \cdot b$. By construction, each row $v_i^T$ of $V$ is an element of $\ker(A^T_{h,s})$. Thus $v_i^T \cdot v_i = (v_i^T \cdot A_{h,s})^T = 0$. Therefore, $V' \cdot A_{h,s}$ is a null-zero matrix and, as a consequence, $V' \cdot A_{h,s} \tilde{x}_{h,s} + V' \cdot A_{h,s} e = V' \cdot b$. Since $H = V' \cdot A_{h,s}$ and $q = V' \cdot b$, we can conclude that $e = \tilde{x}_{h,s}$ is a solution of NCP on the instance $I'$. Clearly, the number of variation events induced by the haplotype configuration $S$ is equal to $\|e\|$. The next corollary easily follows from Lemma 10 and Lemma 11.

**Corollary 12.** MCHC is L-reducible to NCP with parameters $\beta = \gamma = 1$.

**Approximability of MCHC, MRHC and MMHC**

An immediate (positive) consequence of Corollary 12 is that MCHC, MRHC and MMHC are $O(\frac{nm}{\log nm})$-approximable on general pedigrees since there exists a polynomial-time $O(\frac{n}{\log n})$-approximation algorithm for NCP [19]. The best known bound for the MRHC problem is $O(\sqrt{\log n})$ [12] but it only holds for 2-locus pedigrees and not, as in this case, for unconstrained pedigrees and genotypes. This result, however, is mainly of theoretical interest because the approximation bound is too large. Indeed, on real data, only a small number of genetic variation events are expected to occur, much smaller than the number $O(nm)$ of all possible events. Unfortunately, no (significantly) better approximation algorithms are known and, moreover, if $P \neq NP$, the NCP problem cannot be approximated within any constant factor (i.e., NCP $\notin$ APX) [9].

**4.3 The Heuristic Algorithm**

In this section, we present an efficient heuristic algorithm that solves the MCHC problem. In addition, this heuristic can be also used to solve the MRHC and MMHC problems by restricting the types of variation events that are allowed. An implementation of the heuristic described below is released under the GNU General Public License version 3 or later and can be freely downloaded from the web page http://www.algodab.eu/Heu-MCHC/.

The algorithm is based on the above L-reduction from MCHC to NCP. Since NCP $\notin$ APX [9], there do not exist algorithms that can guarantee a good (i.e., constant) approximation ratio unless $P = NP$. Nevertheless, it has been shown that the sum-product (SP) algorithm [10] (independently proposed in Artificial Intelligence as the belief-propagation algorithm [20]) is an effective and efficient heuristic for NCP. The SP algorithm computes an approximation of the likelihood that each bit of the received message has been “flipped” during the transmission of the message. Such an approximation is computed by employing the set of parity constraints of the linear code and a vector $q$ (called syndrome) representing the constraints that are not satisfied by the received message.

Our idea is to consider the variation events (recombinations and mutations) as the “errors” that we have to reconstruct and, once the “errors” (variation events) have been determined, it is easy to reconstruct the haplotyped pedigree (by Gaussian elimination). The L-reduction in Corollary 12 formalizes this idea. The set of parity constraints and the syndrome $q$ are obtained from the genotyped pedigree (represented by the matrices $A_{h,s}$ and $A_{h,s}$) as illustrated in the previous section. The likelihoods computed by the SP algorithm on this instance represents the likelihoods that each $\delta$- or $\mu$-variable is equal to 1. In other words, for each possible
Algorithm 1: The heuristic algorithm for MCHC

Data: A genotyped pedigree \( P_{g'} \).
Result: A haplotype configuration consistent with \( P_{g'} \).

1. Let \( I = (A_{h,s}, A_{\delta,\mu}, x_{h,s}, x_{\delta,\mu}, b) \) be the linear system of Lemma 9 (in the form of Eq. 3) associated with \( P_{g'} \).
2. Compute the NCP instance \( I' = (H, q) \) associated with \( I \);
3. Let \( N = \{ e \mid e \) is a \( \delta \)- or \( \mu \)-variable \} and \( E = \emptyset \);
   /∗ Each column of \( H \) is associated with a variable of the set \( N \) since \( H = V \cdot A_{h,\mu} \) for some matrix \( V \) ∗/
4. while \( q \neq 0 \) do
   5. Let vector \( L \) contain the likelihood of each variable in \( N \) computed by the SP algorithm on \( H \) and \( q \);
   6. Let \( e^* \) be a variable of \( N \) such that \( L[e^*] \) is maximized;
   7. foreach row \( r \) such that \( H[r, e^*] = 1 \) do
      8. Change the value of \( q[r] \) to \( 1 - q[r] \);
   end
   9. Remove the column associated with \( e^* \) from \( H \);
   10. Move \( e^* \) from \( N \) to \( E \);
11. end
12. Set \( x_{\delta,\mu}[e] = 1 \) if \( e \in E \), or \( x_{\delta,\mu}[e] = 0 \) otherwise;
13. Solve the system \( A_{h,s} \cdot x_{h,s} = b + A_{\delta,\mu} \cdot x_{\delta,\mu} \) in the variables \( x_{h,s} \);
14. return \((x_{h,s}, x_{\delta,\mu})\);

variation event, it computes the likelihood that the event has occurred on the pedigree.

Our heuristic iteratively adds the most likely variation event (as computed by the SP algorithm) to a set \( E \) of imputed variation events until a haplotype configuration that induces exactly the imputed events can be found. Given a set of variation events \( E \), the reconstruction of the haplotype configuration that induces \( E \) can be performed efficiently. Indeed, it suffices to solve the linear system of Lemma 9 with the \( \delta \)- and \( \mu \)-variables assigned to 1 if the corresponding events (the mutations or the recombinations they represent) belong to \( E \), or 0 otherwise.

The details of the approach are given in Algorithm 1. In particular, lines 1–2, compute the NCP instance \( I' = (H, q) \) associated with the genotyped pedigree \( P_{g'} \). Initially (line 3), no variation events are imputed (thus \( E = \emptyset \)) while a set \( N \) contains all the possible variation events (represented by the corresponding \( \delta \)- and \( \mu \)-variables). For each binary linear code, the set of parity constraints is represented by a particular binary matrix \( H \), called the check matrix, such that \( H \cdot y = 0 \) if and only if \( y \) is a valid codeword. In our L-reduction, the check matrix associated with the MCHC instance is computed as \( H = V \cdot A_{\delta,\mu} \) for some matrix \( V \). As a consequence, matrix \( H \) has the same number of columns as \( A_{\delta,\mu} \), each of which is associated with a \( \delta \)- or \( \mu \)-variable. We associate each column \( i \) of \( H \) with the \( \delta \)- or \( \mu \)-variable that is associated with the \( i \)-th column of \( A_{\delta,\mu} \). For simplicity, we identify each column \( i \) of \( H \) with the associated variable.

The haplotype configuration is computed in two steps: first (lines 4–12) the set of variation events \( E \) that makes the reconstruction of a haplotype configuration possible is computed, then (lines 13–14) the haplotype configuration is recovered using the imputed events \( E \).

The first step iteratively constructs the set of variation events. Using the SP algorithm, it computes an event \( e^* \) that most likely is induced in a haplotype configuration consistent with the pedigree (lines 5–6). If more than one event have the maximum likelihood, one of them is chosen at random. Once \( e^* \) has been determined, the corresponding \( \delta \)- or \( \mu \)-variable is set to 1, and the syndrome is updated according to the check matrix \( H \) (lines 7–9). Then, the column of \( H \) associated with event \( e^* \) can be removed, and \( e^* \) can be moved from the set of possible events \( N \) to the set of imputed events \( E \). Based on the remaining parity constraints, we check if the presence (or absence) of other variation events is implied by \( e^* \) and the other events contained in \( E \). This check can be performed by the Gaussian elimination algorithm. This step ends if all the remaining parity constraints are satisfied.

The second step (lines 13–14) reconstructs the haplotype configuration consistent with the input genotyped pedigree by solving the linear system of Eq. 2 using, as a partial solution, the set \( E \) of imputed events.

To improve clarity of the presentation we omitted to describe one step in Algorithm 1. To guarantee that the algorithm finds a haplotype configuration consistent with the genotyped pedigree, we have to check, at each iteration, if matrix \( H \) and syndrome \( q \) imply the presence or the absence of some variation events in the haplotype configuration. We call such events determined events. Determined events can be easily recognized by the Gauss elimination algorithm. Indeed, they correspond to the variables that do not depend to free variables in the solution of the linear system \( H \cdot x = s \). Therefore by applying the Gauss elimination algorithm on \((H|s)\) and removing the determined events at the beginning of each iteration, we can guarantee that the algorithm finds a haplotype configuration consistent with the genotyped pedigree (if such a configuration exists).

An important remark is in order. The SP algorithm (used in line 5) considers as an additional input the prior probability that each variation event \( e \) has occurred. Although we have not incorporated this feature into the current algorithm, it could be extremely useful to model recombination hotspots (by increasing the prior probability of recombination events in regions where recombinations occurs more frequently), to differentiate the rates of recombinations and mutations (by increasing the prior probability of a recombination event with respect to a mutation event), and/or to model additional
knowledge about the input genotypes. This feature of the SP algorithm could also allow us to combine the combinatorial formulation of the problem presented here with some elements of statistics-based methods.

The time complexity of the heuristic depends on several parameters. Let $n$ be the number of individuals of the genotyped pedigree and $m$ the number of loci. The NCP instance $I'$ is calculated by the Gaussian elimination algorithm on $A_{r,s}$ and by two matrix multiplications, requiring $O(n^3m^3)$ time. The check-matrix $H$ has $O(nm)$ rows and at most $4nm$ columns (one for each variation event). Therefore the reduction from the pedigrees to the NCP instance is computed in $O(n^3m^3)$ time. The time required by each iteration is bounded by $O(n^3m^3)$ since the check of the existence of predetermined events (by Gaussian elimination) requires $O(n^3m^3)$ time, the SP algorithm requires linear time in the number of one-entries of matrix $H$, and the other operations that update parity constraints and the syndrome can be accomplished in $O(n^2m^2)$ time. The resolution of the final linear system can be performed in $O(n^2m^3)$ time by the Gaussian elimination algorithm. Let $k$ be the number of events that are imputed, then the overall time complexity of the heuristic is $O(kn^3m^3)$.

5 Experimental Results

Our approach has been experimentally analyzed under several simulated scenarios. The experimental analysis is divided into two parts. In the first part, we evaluate the accuracy and efficiency of our heuristic on randomly generated MCHC instances. In the second part, we compare the performance of our heuristic with that of three state-of-the-art approaches for MRHC and MMHC: PedPhase v2.1 [21], SimWalk2 [22], and MMPhase [11].

5.1 Evaluation on Random Instances

The first part of our experimentation involves randomly generated instances under several choices of 4 parameters: pedigree size ($n$), the number of loci ($m$), recombination probability ($\theta_r$), and mutation probability ($\mu_r$). For each choice of the parameters, we generated 30 haplotype configurations on 6 different random pedigree graphs. We applied a variation event at each locus with probability $\theta_r$ for recombinations and $\mu_r$ for mutations. For each instance, we ran our heuristic 10 times and we picked the solution with the minimum number of induced events.

We evaluated the quality of the results considering phase error (the ratio between the number of incorrectly predicted haplotype alleles and twice the number of heterozygous loci) and approximation ratio (the ratio between the number of predicted events and the number of generated events). The approximation ratio can be less than 1.0 because the generated haplotype configuration might be suboptimal. Finally, we also evaluated the total running time required by the heuristic on all 10 executions.

We chose a base set of values for the parameters $n$, $m$, $\theta_r$, and $\mu_r$ and we conducted three series of tests. In each series, we modified the value of one of these parameters: pedigree size in the first, genotype length in the second, and the two probabilities $\theta_r$ and $\mu_r$ in the third. The base values were: pedigree size $n = 40$, number of loci $m = 40$, recombination probability $\theta_r = 0.02$, and mutation probability $\mu_r = 0.004$. The detailed results of the three series of tests are summarized in Table 1.

In the first series of tests, we varied the pedigree size $n$ and analysed the cases $n = 40$, $n = 60$, and $n = 100$ on both tree pedigrees and “general” pedigrees (i.e., pedigree with mating loops). In all cases, the heuristic never required more than 6 minutes (169 seconds on average) on a standard PC with a 1.66GHz CPU and 2GB of main memory and it always found a haplotype configuration that induces fewer variation events than the generated one (i.e., pedigree with mating loops). Although this fact does not imply that the heuristic computed the optimal solution, it increases our confidence in the soundness of the approach. The values of the quality measures are similar in all cases and, on average, are equal to 0.02 and 0.97 for phase error and approximation ratio, respectively. In the second series of tests, we varied the number of loci $m$ and considered the following cases: $m = 40$, $m = 60$, and $m = 100$. Similarly to the previous series, we obtained 0.039 as the average phase error and 0.96 as the average approximation ratio, with an average running time of 182 seconds. In the third series of tests, we varied the probabilities of recombinations and mutations in the range of (0.02, 0.004) to (0.10, 0.02). In this case, the quality of the results decreases with the increase of the number of generated events. The worst results were obtained when recombination and mutation probabilities were at the maximum. Note that when this happens, the generated haplotype configuration significantly deviates from the parsimony principle that MCHC assumes. In fact, our heuristic reconstructs a solution with much fewer events than the generated haplotype configuration (in such a situation the average approximation ratio is 0.85, significantly less than 1 in comparison with the average approximation ratio obtained in the other series of tests).

5.2 Comparison with State-of-the-Art Methods

In the second part of the experimental evaluation, we compare the accuracy and efficiency of our heuristic with those of some state-of-the-art approaches for HI on pedigrees. Popular approaches to HI on pedigrees do not allow for both recombinations and mutations at the same time. Therefore, we separately considered two classes of algorithms. The first one consists of algorithms for MRHC (i.e., only recombinations are allowed) and the second class consists of algorithms for MMHC (i.e., only
Table 1
Summary of the results obtained by our heuristic on randomly generated instances. Each table reports the quality and performance measures of the heuristic on a subset of instances where a parameter has been varied. The default settings of the parameters are: pedigree size $n = 40$, number of loci $m = 40$, recombination probability $\theta_r = 0.02$, and mutation probability $\mu_r = 0.004$.

(a) Increasing pedigree size ($n$)

<table>
<thead>
<tr>
<th>Pedigree size $n =$</th>
<th>Tree pedigrees</th>
<th>General pedigrees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>Avg. no. of generated events</td>
<td>22.0</td>
<td>30.4</td>
</tr>
<tr>
<td>Avg. no. of predicted events</td>
<td>21.3</td>
<td>29.5</td>
</tr>
<tr>
<td>Avg. phase error</td>
<td>0.027</td>
<td>0.029</td>
</tr>
<tr>
<td>Avg. approximation ratio</td>
<td>0.968</td>
<td>0.975</td>
</tr>
<tr>
<td>Avg. time (in seconds)</td>
<td>36</td>
<td>73</td>
</tr>
</tbody>
</table>

(b) Increasing the number of loci ($m$)

<table>
<thead>
<tr>
<th>Number of loci $m =$</th>
<th>Tree pedigrees</th>
<th>General pedigrees</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>Avg. no. of generated events</td>
<td>24.0</td>
<td>34.7</td>
</tr>
<tr>
<td>Avg. no. of predicted events</td>
<td>23.1</td>
<td>33.0</td>
</tr>
<tr>
<td>Avg. phase error</td>
<td>0.035</td>
<td>0.057</td>
</tr>
<tr>
<td>Avg. approximation ratio</td>
<td>0.964</td>
<td>0.956</td>
</tr>
<tr>
<td>Avg. time (in seconds)</td>
<td>41</td>
<td>95</td>
</tr>
</tbody>
</table>

(c) Increasing recombination and mutation probabilities ($\theta_r$ and $\mu_r$)

<table>
<thead>
<tr>
<th>Recombination prob. $\theta_r =$</th>
<th>0.02</th>
<th>0.04</th>
<th>0.10</th>
<th>0.02</th>
<th>0.04</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutation probability $\mu_r =$</td>
<td>0.004</td>
<td>0.01</td>
<td>0.02</td>
<td>0.004</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Avg. no. of generated events</td>
<td>24.5</td>
<td>48.8</td>
<td>111.5</td>
<td>24.9</td>
<td>48.9</td>
<td>121.8</td>
</tr>
<tr>
<td>Avg. no. of predicted events</td>
<td>23.8</td>
<td>45.7</td>
<td>94.8</td>
<td>24.0</td>
<td>45.8</td>
<td>105.3</td>
</tr>
<tr>
<td>Avg. phase error</td>
<td>0.035</td>
<td>0.061</td>
<td>0.114</td>
<td>0.020</td>
<td>0.053</td>
<td>0.099</td>
</tr>
<tr>
<td>Avg. approximation ratio</td>
<td>0.973</td>
<td>0.937</td>
<td>0.848</td>
<td>0.963</td>
<td>0.939</td>
<td>0.866</td>
</tr>
<tr>
<td>Avg. time (in seconds)</td>
<td>45</td>
<td>74</td>
<td>164</td>
<td>63</td>
<td>86</td>
<td>248</td>
</tr>
</tbody>
</table>

mutations are allowed). We adapted our heuristic algorithm to the two problems by keeping only the variables associated with the type of events that are allowed ($\delta$-variables for MRHC and $\mu$-variables for MMHC).

Comparison with MRHC Algorithms

Several algorithms for MRHC have been proposed in the literature. For our comparison, we chose two popular approaches with different computational characteristics: PedPhase v2.1 [21] (an exact ILP-based algorithm) and SimWalk2 [22] (a popular statistical approach for HI). We generated 750 instances using SimPed [23], a simulation program for the generation of haplotyped pedigrees based on user-supplied biological information (such as intramarker distances and allele frequencies). The same biological information have then been used to correctly initialize the input parameters of SimWalk2. The instance sizes ranged from pedigrees with 8 members and 10 loci to pedigrees with 100 members and 100 loci. We limited the running time on each instance to 1 hour. Our heuristic was the only method that completed all the 750 instances within this time limit. PedPhase completed 565 instances and SimWalk2 only 495 of them. PedPhase took over 5 hours to solve the 565 instances that it was able to tackle, while our heuristic on the same instances took only 575 seconds. Our heuristic was able to compute a solution with the same number of recombinations as PedPhase (i.e., an optimal solution) in 560 of the 565 cases. SimWalk2 is a much slower approach; it took nearly 108 hours of computation while our method allows more than one mutation on the same locus. Moreover, while MMPhase was able to

Comparison with MMPhase

We compared our heuristic with MMPhase [11], the only other algorithm that has been explicitly proposed for MMHC in the literature (to the best of our knowledge). MMPhase is an ILP-based approach for MMHC with two restrictions: the model explicitly forbids two mutations at the same locus in different individuals (called the infinite-site assumption) and the current implementation is only able to handle tree pedigrees. Therefore, we generated 300 random instances of different sizes according to these restrictions. In particular we considered 4 different pedigree sizes (50, 75, 100, 150) and 3 different numbers of loci (50, 100, 150) and we generated 25 instances for each possible combination of the two parameters. The comparison revealed that MMPhase is noticeably faster than our heuristic (on average MMPhase required 167 seconds per instance vs. 483 seconds for our heuristic). However we observe that MMPhase exploits the infinite-site assumption in order to reduce the solution space, while our method allows more than one mutation on the same locus. Moreover, while MMPhase was able to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Table 4
Summary of the comparison with another method for MMHC. Each row represents 25 instances whose pedigree size and number of loci are specified in the first two columns. For each method we report the number of instances that have been solved within an hour (i.e., completed instances), the average running time, the average number of computed mutations, and the average phase error. We reported in brackets the performances of the two methods computed on the instances that have been completed by both within the time limit.

<table>
<thead>
<tr>
<th>Pedigree size</th>
<th>No. of loci</th>
<th>Compl. instances</th>
<th>Avg. running time (s)</th>
<th>Avg. no. of mutations</th>
<th>Avg. phase error</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>50</td>
<td>25</td>
<td>5.04</td>
<td>11.12</td>
<td>0.0035</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>25</td>
<td>8.61</td>
<td>22.84</td>
<td>0.0037</td>
</tr>
<tr>
<td>50</td>
<td>150</td>
<td>25</td>
<td>18.18</td>
<td>35.28</td>
<td>0.0031</td>
</tr>
<tr>
<td>75</td>
<td>50</td>
<td>25</td>
<td>15.04</td>
<td>15.28</td>
<td>0.0032</td>
</tr>
<tr>
<td>75</td>
<td>100</td>
<td>25</td>
<td>31.50</td>
<td>33.32</td>
<td>0.0035</td>
</tr>
<tr>
<td>75</td>
<td>150</td>
<td>25</td>
<td>116.43</td>
<td>50.68</td>
<td>0.0031</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>25</td>
<td>54.08</td>
<td>19.60</td>
<td>0.0029</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>25</td>
<td>106.29</td>
<td>41.76</td>
<td>0.0029</td>
</tr>
<tr>
<td>100</td>
<td>150</td>
<td>25</td>
<td>386.99</td>
<td>63.40</td>
<td>0.0032</td>
</tr>
<tr>
<td>150</td>
<td>50</td>
<td>25</td>
<td>201.31</td>
<td>27.04</td>
<td>0.0022</td>
</tr>
<tr>
<td>150</td>
<td>100</td>
<td>25</td>
<td>642.02</td>
<td>54.72</td>
<td>0.0025</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
<td>25</td>
<td>460.13</td>
<td>63.73</td>
<td>0.0025</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(20)</td>
<td>(83.80)</td>
<td>(0.0025)</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>297</td>
<td>167.54</td>
<td>37.77</td>
<td>0.0030</td>
</tr>
<tr>
<td></td>
<td></td>
<td>295</td>
<td>(167.02)</td>
<td>(37.46)</td>
<td>(0.0030)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compl. instances</th>
<th>Avg. running time (s)</th>
<th>Avg. no. of mutations</th>
<th>Avg. phase error</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>8.00</td>
<td>11.12</td>
<td>0.0029</td>
</tr>
<tr>
<td>25</td>
<td>36.98</td>
<td>23.24</td>
<td>0.0037</td>
</tr>
<tr>
<td></td>
<td>152.83</td>
<td>36.76</td>
<td>0.0034</td>
</tr>
<tr>
<td></td>
<td>15.90</td>
<td>15.28</td>
<td>0.0030</td>
</tr>
<tr>
<td></td>
<td>131.07</td>
<td>34.04</td>
<td>0.0034</td>
</tr>
<tr>
<td></td>
<td>471.61</td>
<td>51.96</td>
<td>0.0031</td>
</tr>
<tr>
<td></td>
<td>35.68</td>
<td>20.28</td>
<td>0.0032</td>
</tr>
<tr>
<td></td>
<td>288.69</td>
<td>42.16</td>
<td>0.0029</td>
</tr>
<tr>
<td></td>
<td>1028.84</td>
<td>65.16</td>
<td>0.0031</td>
</tr>
<tr>
<td></td>
<td>97.88</td>
<td>27.40</td>
<td>0.0022</td>
</tr>
<tr>
<td></td>
<td>886.75</td>
<td>56.20</td>
<td>0.0025</td>
</tr>
<tr>
<td></td>
<td>826.95</td>
<td>86.17</td>
<td>0.0025</td>
</tr>
<tr>
<td></td>
<td>(2892.79)</td>
<td>(86.65)</td>
<td>(0.0025)</td>
</tr>
<tr>
<td>Total</td>
<td>298</td>
<td>483.58</td>
<td>38.83</td>
</tr>
<tr>
<td></td>
<td>(295)</td>
<td>(463.43)</td>
<td>(38.38)</td>
</tr>
</tbody>
</table>

solve 297 of the 300 instances within an hour of time limit each, our method was able to solve 298 instances in the same time limit. Although our heuristic obtained solutions with slightly more mutations than MMPhase on 38 of the 298 instances, the average phase errors of the two methods are identical (0.0030). If we compare the performances of the two methods only on the instances that have been solved by both, the results change only slightly: the biggest change is represented by the average time required by our heuristic that decreases to 463 seconds. The results of the comparison are summarized in Table 4.

6 Conclusion
In this paper, we proposed a new formulation for the haplotype inference problem on pedigrees, called MINIMUM-CHANGE HAPLOTYPE CONFIGURATION problem, that extends previous formulations by allowing two types of variation events: recombinations and mutations. We showed that the problem is APX-hard under several tight restrictions and that is \( O(nm \log nm) \)-approximable on the general case, where \( n \) is the pedigree size and \( m \) is the genotype length. This approximability result holds also for the formulations that we extended, the MINIMUM-RECOMBINANT HAPLOTYPE CONFIGURATION and the MINIMUM-MUTATION HAPLOTYPE CONFIGURATION problems, for which no approximation algorithms were known. We also presented a heuristic algorithm that has been shown both accurate and efficient by an extensive experimental evaluation under several simulated scenarios. The heuristic also compares favorably with several other state-of-the-art methods. It is faster than (but as accurate as) the other methods that consider only recombinations. Moreover, the only method considered in this study that is faster than our heuristic (MMPhase, which allows only point mutations) requires and exploits more restrictive assumptions about the input data than our method. The heuristic algorithm could handle moderately large pedigrees very well (in some of our tests, it was able to process tree pedigrees with 50 individuals and 1000 loci in approximately 2.5 hours of computation time on a standard PC). However, it cannot be applied directly to genome-scale data with millions of loci. Fortunately, the haplotype block structure observed in the human genome [24] provides a straightforward way of partitioning long genotypes into short blocks which can be readily handled by our method.

Acknowledgment
This research was supported in part by FAR MIUR 60% grant “Metodi algorithmi per l’analisi di strutture combinatorie in Bioinformatica” (Univ. di Milano-Bicocca) to YP and PB, by “ProZoo” project (Regione Lombardia - D.G. Agricoltura, Fondazione Cariplo and Fondazione Banca Popolare di Lodi) to YP, and NIH grant 2R01LM008991 and NSF grant IIS-0711129 to TJ. Part of the work was done when YP was visiting at University of California, Riverside. We would like to thank Wei-Bung Wang for valuable discussions and sharing the MMPhase code with us, and the anonymous reviewers for their thoughtful and helpful comments.
Table 2
Summary of the comparison with other methods for MRHC. Each row represents 50 instances whose pedigree size and number of loci are specified in the first two columns. For each method we report the number of instances that have been solved within an hour of computation (i.e., completed instances), the average running time, the average number of predicted recombinations, and the average phase error.

<table>
<thead>
<tr>
<th>Pedigree size</th>
<th>Number of loci</th>
<th>Completed instances</th>
<th>PedPhase [21]</th>
<th>SimWalk2 [22]</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Avg. running time (s)</td>
<td>Avg. no. of recombinations</td>
<td>Avg. phase error</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>50</td>
<td>0.14</td>
<td>0.40</td>
<td>0.023</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>50</td>
<td>0.27</td>
<td>3.50</td>
<td>0.046</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>50</td>
<td>1.90</td>
<td>5.70</td>
<td>0.078</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>50</td>
<td>0.09</td>
<td>1.90</td>
<td>0.012</td>
</tr>
<tr>
<td>16</td>
<td>50</td>
<td>50</td>
<td>0.58</td>
<td>6.40</td>
<td>0.030</td>
</tr>
<tr>
<td>16</td>
<td>100</td>
<td>50</td>
<td>29.31</td>
<td>16.60</td>
<td>0.047</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>50</td>
<td>0.21</td>
<td>4.00</td>
<td>0.009</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>50</td>
<td>10.45</td>
<td>21.50</td>
<td>0.026</td>
</tr>
<tr>
<td>40</td>
<td>100</td>
<td>50</td>
<td>401.59</td>
<td>48.29</td>
<td>0.038</td>
</tr>
<tr>
<td>60</td>
<td>10</td>
<td>50</td>
<td>0.48</td>
<td>5.50</td>
<td>0.003</td>
</tr>
<tr>
<td>60</td>
<td>50</td>
<td>50</td>
<td>18.12</td>
<td>34.30</td>
<td>0.023</td>
</tr>
<tr>
<td>60</td>
<td>100</td>
<td>50</td>
<td>155.13</td>
<td>64.00</td>
<td>0.039</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>10</td>
<td>0.87</td>
<td>8.00</td>
<td>0.002</td>
</tr>
<tr>
<td>100</td>
<td>50</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>565</td>
<td>35.83</td>
<td>14.25</td>
</tr>
</tbody>
</table>

Table 3
Detailed comparison of our heuristic with other methods for MRHC on the subset of test instances that have been solved by all the methods within an hour of computation. The first three columns represent the pedigree size, the number of loci, and the number of instances (out of a maximum of 50) that have been solved by all the methods. For each method we report the average running time, the average number of predicted recombinations, and the average phase error.

<table>
<thead>
<tr>
<th>Pedigree size</th>
<th>Number of loci</th>
<th>Completed instances</th>
<th>PedPhase [21]</th>
<th>SimWalk2 [22]</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Avg. running time (s)</td>
<td>Avg. no. of recombinations</td>
<td>Avg. phase error</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>50</td>
<td>0.14</td>
<td>0.40</td>
<td>0.023</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>50</td>
<td>0.27</td>
<td>3.50</td>
<td>0.046</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>50</td>
<td>1.90</td>
<td>5.70</td>
<td>0.078</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>50</td>
<td>0.09</td>
<td>1.90</td>
<td>0.012</td>
</tr>
<tr>
<td>16</td>
<td>50</td>
<td>50</td>
<td>0.58</td>
<td>6.40</td>
<td>0.030</td>
</tr>
<tr>
<td>16</td>
<td>100</td>
<td>50</td>
<td>29.31</td>
<td>16.60</td>
<td>0.047</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>50</td>
<td>0.21</td>
<td>4.00</td>
<td>0.009</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>50</td>
<td>11.32</td>
<td>21.22</td>
<td>0.025</td>
</tr>
<tr>
<td>60</td>
<td>10</td>
<td>50</td>
<td>0.48</td>
<td>5.50</td>
<td>0.003</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>10</td>
<td>0.87</td>
<td>7.50</td>
<td>0.002</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>455</td>
<td>4.76</td>
<td>7.10</td>
</tr>
</tbody>
</table>
Yu Ri

IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS


REFERENCES


Yuri Pirola received the MSc and the PhD degrees in computer science from the Università

Paola Bonizzoni is full professor of Computer

Science at the Università di Milano-Bicocca in

15

The International HapMap Consortium, “A second generation


This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS


Tao Jiang received BS in computer science and
technology from the University of Science and


in computer science and bioinformatics journals and conferences, and	

Paola Bonizzoni is full professor of Computer

Paola Bonizzoni is full professor of Computer

Science at the Università di Milano-Bicocca in

Science at the Università di Milano-Bicocca in

in computer science and bioinformatics journals and conferences, and won several best paper awards. More information about his work can be found at http://www1.cs.ucr.edu/~jiang

Tao Jiang received BS in computer science and
technology from the University of Science and

Tao Jiang received BS in computer science and
technology from the University of Science and

Tao Jiang received BS in computer science and
technology from the University of Science and

Tao Jiang received BS in computer science and
technology from the University of Science and