Abstract. Haplotype Inference (HI) is a computational challenge of crucial importance in a range of genetic studies, such as functional genomics, pharmacogenetics and population genetics. Pedigrees have been shown a valuable data that allows us to infer haplotypes from genotypes more accurately than population data, since Mendelian inheritance restricts the set of possible solutions. In order to overcome the limitations of classic statistical haplotyping methods, a combinatorial formulation of the HI problem on pedigrees has been proposed in the literature, called Minimum-Recombinant Haplotype Configuration (MRHC) problem, that allows a single type of genetic variation events, namely recombinations. In this work, we define a new problem, called Minimum-Change Haplotype Configuration (MCHC), that extends the MRHC formulation by allowing also a second type of natural variation events: mutations. 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 it can take advantage of additional knowledge about the input genotypes, such as the presence of recombination hotspots and different rates of recombinations and mutations. 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 under several simulated scenarios.

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 [14], 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 much time-consuming and expensive to be performed on large studies [3]. 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 [15], 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 Lande Green [7] and Elson-Stewart [4] 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. 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 [12,9]. 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 L-reduction [2] of MCHC to a fundamental problem of coding theory: the Nearest Codeword Problem (NCP) [2, probl. MS3]. Although NCP is theoretically hard to approximate [1], there exists several heuristics that compute near-optimal solutions of NCP in practice [6]. 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. Then, in Section 3, we present a heuristic algorithm based on an L-reduction from MCHC to NCP. Finally, in Section 4, 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 triplet $(f, c, m)$ 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.

Let $\Sigma$ be an ordered set $\langle l_1, \ldots, l_m \rangle$ 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 $\{0, 1, 2\}$, 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 $\langle 0, 0 \rangle$ as 0, $\langle 1, 1 \rangle$ as 1, and $\langle 0, 1 \rangle$ as 2.

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 $(h_c^0, h_c^1)$ the haplotypes associated with an individual $c$ of a haplotyped pedigree. Moreover, we say that $h_c^0$ is the paternal haplotype of $c$ and $h_c^1$ is the maternal haplotype of $c$. 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 $(h_c^0, h_c^1)$. 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 maternal haplotype of $p$. On the other hand, $s_{p,c}[i] = 1$ if the allele has been inherited from the paternal 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 haplotyped pedigree [12,9]. Two types of variation events will be considered, recombinations and mutations, defined as follows. Let \((P, S)\) 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 \(h^s_p[i] \neq h^s_c[i]\) where \(s = s_{p,c}[i]\) and \(j = 0 (j = 1, \text{resp.})\) if \(p\) is the father (mother, resp.) of \(c\).

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 [9]), and the Minimum-Mutation Haplotype Configuration (MMHC) problem (where only mutations are allowed [16]). Differently from [16], in the following we do not restrict the number of mutations at each locus (among all individuals) to be at most one. It is possible to prove that MCHC is APX-hard even on instances where genotypes are defined on only 2 loci or where each individual has at most one mate and one child. Due to the page limit, the proof is deferred to the full version of this paper.

3 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. [17] 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.

3.1 A System of Linear Equations for MCHC

In this part, we first illustrate the linear system over \(\mathbb{Z}_2\) proposed in [17] for the HI problem where no recombinations or mutations are permitted (i.e., the zero-recombinant haplotype configuration problem or ZRHC), 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 \( \oplus \).

### 3.2 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 homozygous 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) [17]. For all loci \( l \) and individuals \( i \),

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

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.

### 3.3 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
Haplotype Inference on Pedigrees with Recombinations and Mutations

δ-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 recombination events. Since mutations are point events that replace an allele inherited from the parent with another allele, it suffices to add a term in the first equation of the original linear system to model mutation events. We denote this term as \( \mu_{p,i}^{l} \) and set \( \mu_{p,i}^{l} = 1 \) if a mutation at locus \( l \) between \( p \) and \( i \) has occurred, and \( \mu_{p,i}^{l} = 0 \) otherwise. The following lemma is straightforward.

**Lemma 1.** Let \( P_g \) be a genotyped pedigree. Then each solution of the system

\[
\begin{align*}
    h_{p}^{l} + (s_{p,i} + \Delta_{p,i}^{l}) \cdot w_{p}^{l} &= h_{i}^{l} + d_{p,i}^{l} + \mu_{p,i}^{l} \quad \text{for each parent } p \text{ of } i \\
    h_{i}^{l} &= g_{i}^{l} \quad \text{if } g_{i}^{l} \neq 2 \\
    w_{i}^{l} &= 0 \quad \text{if } g_{i}^{l} \neq 2 \\
    w_{i}^{l} &= 1 \quad \text{if } g_{i}^{l} = 2 \\
    d_{p,i}^{l} &= 0 \quad \text{if } p \text{ is the father of } i \\
    d_{p,i}^{l} &= w_{i}^{l} \quad \text{if } p \text{ is the mother of } i
\end{align*}
\]

concerning all loci \( l \) and individuals \( i \) represents a haplotype configuration consistent with \( P_g \) that admits recombination and mutation events. Conversely, a haplotype configuration consistent with \( P_g \) that admits recombination 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 \( k \) δ- and \( \mu \)-variables are non-zero.

### 3.4 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 \( q \in \mathbb{Z}_2^r \), the Nearest Codeword Problem [2, 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 1 into two linear systems: one containing only \( h \)- and \( s \)-variables, and the other one containing only δ- 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 \(A_{h,s}^T\) 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 \(\ker(A_{h,s}^T) = \{y \in \mathbb{Z}_2^n \mid A_{h,s}^T \cdot y = 0\}\), 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 2 illustrates how to reconstruct in polynomial-time a solution of an MCHC instance given a solution for the associated NCP instance, and Lemma 3 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 [2, Def. 8.4] for the formal definition of L-reduction and an explanation of these parameters. Due to the page limit, the proofs of the lemmas are deferred to the full version of the paper.

**Lemma 2.** 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.

**Lemma 3.** Let \(S = (\tilde{x}_{h,s}, \tilde{x}_{\delta,\mu})\) be a solution of MCHC on the instance \(I = (A_{h,s}, A_{\delta,\mu}, x_{h,s}, x_{\delta,\mu}, b)\) and \(I' = (H, q)\) the NCP instance associated with \(I\). Then, vector \(e := \tilde{x}_{\delta,\mu}\) is a solution of NCP on \(I'\).

The following corollary easily follows from Lemma 2 and Lemma 3.

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

### 3.5 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 can be freely downloaded from the web page [http://www.algolab.eu/Heu-MCHC/](http://www.algolab.eu/Heu-MCHC/).

The algorithm is based on the above L-reduction from MCHC to NCP. Since NCP \(\not\in\) APX [1], 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 [6] (independently proposed in Artificial Intelligence as the belief-propagation algorithm [11]) 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 1 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 $A_{\delta,\mu}$) 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 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 1 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. Initially, 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 the set of variation events $E$ that makes the reconstruction of a haplotype configuration possible is computed, then 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. 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 $H$. 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 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.

An important remark is in order. The SP algorithm 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 occur 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 \( A_{h,s}^T \) 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 pedigree 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 \( O(n^3m^3) \) 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 \( O(kn^3m^3) \).

4 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 [10], SimWalk2 [13], and MMPhase [16].

4.1 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., the approximation ratio is always smaller than 1.0). 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).

4.2 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.
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>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>40</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>Avg. no. of generated events</td>
<td>22.0</td>
<td>30.4</td>
<td>55.2</td>
</tr>
<tr>
<td>Avg. no. of predicted events</td>
<td>21.2</td>
<td>29.5</td>
<td>54.2</td>
</tr>
<tr>
<td>Avg. phase error</td>
<td>0.027</td>
<td>0.029</td>
<td>0.028</td>
</tr>
<tr>
<td>Avg. approximation ratio</td>
<td>0.968</td>
<td>0.975</td>
<td>0.965</td>
</tr>
<tr>
<td>Avg. time (in seconds)</td>
<td>36</td>
<td>73</td>
<td>265</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>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>40</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>Avg. no. of generated events</td>
<td>24.0</td>
<td>34.7</td>
<td>53.2</td>
</tr>
<tr>
<td>Avg. no. of predicted events</td>
<td>23.1</td>
<td>33.0</td>
<td>51.2</td>
</tr>
<tr>
<td>Avg. phase error</td>
<td>0.035</td>
<td>0.057</td>
<td>0.042</td>
</tr>
<tr>
<td>Avg. approximation ratio</td>
<td>0.964</td>
<td>0.956</td>
<td>0.964</td>
</tr>
<tr>
<td>Avg. time (in seconds)</td>
<td>41</td>
<td>95</td>
<td>247</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>Tree pedigrees</th>
<th>General pedigrees</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>0.04</td>
<td>0.10</td>
<td>0.02</td>
</tr>
<tr>
<td>0.004</td>
<td>0.01</td>
<td>0.02</td>
<td>0.004</td>
</tr>
<tr>
<td>Avg. no. of generated events</td>
<td>24.5</td>
<td>48.8</td>
<td>111.5</td>
</tr>
<tr>
<td>Avg. no. of predicted events</td>
<td>23.8</td>
<td>45.7</td>
<td>94.8</td>
</tr>
<tr>
<td>Avg. phase error</td>
<td>0.035</td>
<td>0.061</td>
<td>0.114</td>
</tr>
<tr>
<td>Avg. approximation ratio</td>
<td>0.973</td>
<td>0.937</td>
<td>0.848</td>
</tr>
<tr>
<td>Avg. time (in seconds)</td>
<td>45</td>
<td>74</td>
<td>164</td>
</tr>
</tbody>
</table>

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 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 [10] (an exact ILP-based algorithm) and SimWalk2 [13] (a popular statistical approach for HI). We generated 750 instances using SimPed [8], a simulation program for the generation of haplotyped pedigrees based on user-supplied biological information.
Table 2. Summary of the comparison with other methods for MRHC on 750 instances whose sizes vary from pedigrees with 8 members and 10 loci to pedigrees with 100 members and 100 loci. For each method, we report the number of instances that have been solved within an hour of computation (i.e., completed instances), the average number of predicted recombinations, the average phase error, and the average running time. To facilitate the comparison, for each row, we report in brackets the same performance measure values obtained by our heuristic on the instances completed by the other two methods.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed instances</td>
<td>750</td>
<td>565</td>
</tr>
<tr>
<td>Avg. no. of recombinations</td>
<td>26.42</td>
<td>14.25 (14.27)</td>
</tr>
<tr>
<td>Avg. phase error</td>
<td>0.030</td>
<td>0.030 (0.031)</td>
</tr>
<tr>
<td>Avg. running time (s)</td>
<td>4.7</td>
<td>35.8 (1.0)</td>
</tr>
</tbody>
</table>

Table 3. Summary of the comparison with another method for MMHC on 300 instances whose sizes vary from pedigrees with 50 members and 50 loci to pedigrees with 150 members and 150 loci. For each method, we report the number of instances that have been solved within an hour (i.e., completed instances), the average number of computed mutations, the average phase error, and the average running time.

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>MMPhase [16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed instances</td>
<td>298</td>
</tr>
<tr>
<td>Avg. no. of mutations</td>
<td>38.83</td>
</tr>
<tr>
<td>Avg. phase error</td>
<td>0.0030</td>
</tr>
<tr>
<td>Avg. running time (s)</td>
<td>483.58</td>
</tr>
</tbody>
</table>

(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 heuristic completed the same set of instances in less than 90 seconds. Moreover, SimWalk2 never computed a solution with fewer recombinations than our heuristic. On the other hand, the average phase errors of the three approaches are almost identical (PedPhase 0.030, SimWalk2 0.037, and our heuristic 0.030), implying that the sets of recombinations computed by the three approaches, albeit different, are similar. A summary of the results is presented in Table 2.
Comparison with MMPhase. We compared our heuristic with MMPhase [16], 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 solve 297 of the 300 instances within an hour, 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). A summary of the comparison results is presented in Table 3.

5 Conclusion

In this paper, we presented a heuristic method for the haplotype inference problem on pedigrees allowing two types of variation events: recombinations and mutations. The experimental evaluation under several simulated scenarios showed that the heuristic is both accurate and efficient. 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 [5] provides a straightforward way of partitioning long genotypes into short blocks which can be readily handled by our method.

Acknowledgments. This research was supported in part by FAR MIUR 60% grant “Algorithmic methods and combinatorial structures in Bioinformatics” (Univ. di Milano-Bicocca) to YP and PB, 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.
References


