BIOINFORMATICS REVIEW Vol. 26 no. 18 2010, pages 2217–2225
doi:10.1093/bioinformatics/btq411

Sequence analysis Advance Access publication July 11, 2010

A comparison of several algorithms for the single individual SNP haplotyping reconstruction problem

Filippo Geraci
Istituto di Informatica e Telemetica, CNR, V. Moruzzi 1, Pisa, Italy

Associate Editor: Jonathan Wren

ABSTRACT

Motivation: Single nucleotide polymorphisms are the most common form of variation in human DNA, and are involved in many research fields, from molecular biology to medical therapy. The technological opportunity to deal with long DNA sequences using shotgun sequencing has raised the problem of fragment recombination. In this regard, Single Individual Haplotyping (SIH) problem has received considerable attention over the past few years.

Results: In this article, we survey seven recent approaches to the SIH problem and evaluate them extensively using real human haplotype data from the HapMap project. We also implemented a data generator tailored to the current shotgun sequencing technology that uses haplotypes from the HapMap project.

Availability: The data we used to compare the algorithms are available on demand, since we think they represent an important benchmark that can be used to easily compare novel algorithmic ideas with the state of the art. Moreover, we had to re-implement six of the algorithms surveyed because the original code was not available to us. Five of these algorithms and the data generator used in this article endowed with a Web interface are available at http://bioalgo.iit.cnr.it/rehap

Contact: filippo.geraci@iit.cnr.it

Received on December 10, 2009; revised on June 14, 2010; accepted on July 6, 2010

1 INTRODUCTION

Recently, many researchers' focus has shifted from what individuals of a certain species have in common to their differences, and thus to DNA mutations. The single nucleotide polymorphism (*SNP* pronounced 'snip') is the most widespread form of variation in human DNA, and consists in the variation of the base present in a single fixed position of the DNA strand. The sequence of all SNPs in a given chromosome is called a *haplotype*. Humans are diploid organisms, this means that except for the sexual chromosomes of males, the chromosomes come in two copies: one inherited from the mother and one from the father. As a consequence, the haplotypes of a chromosome can be fully described by two sequences of SNPs: the mother's haplotype and the father's. Since haplotypes contain all the information about DNA variations, they play a crucial role in many studies about variations in gene expression and prediction of diseases. For this reason, several sequencing projects have been launched with the ultimate goal of building a complete map of the SNPs present in the human DNA (Frazer *et al.*, 2007; Levy *et al.*, 2007; Via *et al.*, 2010). At the moment, the HapMap Consortium has produced the most complete map of SNPs in human DNA, consisting of over 3.1 million SNPs (Frazer *et al.*, 2007) and has estimated that the overall number of SNPs in the human DNA is about 9–10 million. Single Individual Haplotype (SIH) reconstruction problem is one of the core problems in the reconstruction of whole genomes (Zhao *et al.*, 2007). It consists in rebuilding the two haplotypes from a set of fragments obtained by the shotgun sequencing of the chromosomes. Current shotgun technology produces a very large set of fragments with lengths in the range of 200–900 bases with a certain degree of overlap between them (Morozova and Marra, 2008). This technology does not allow keeping track of the association of a fragment with its haplotype. An important characteristic of this problem is that, unlike the fragment assembly problem, the position and orientation of the fragments is known a priori; this means that fragments can be arranged in a matrix called *SNP Matrix*. In the absence of errors, it is easy to find a bipartition of the SNP matrix such that the fragments belonging to each partition do not conflict (two fragments are said to be conflicting if for a certain position not gap they have different values). In the real-world application, this is not the case and errors affect the SNP matrix. Errors can originate from various sources. Reading errors are typically due to chemical/optical errors in reading the SNPs and as a result lead to the insertion of a wrong base in a certain position. Ambiguous readings occur when the signal strength of a SNP in not enough to establish its correct value with a high degree of confidence. The effect of ambiguous readings is typically the insertion of a gap in the sequence.

The SIH problem has attracted considerable attention in the past few years and a large number of models and algorithms have been introduced in the literature. Many models have NP-hard solutions, thus heuristics are often used. However, a common framework to compare all those algorithms with each other is still lacking. We tested several algorithms for the SIH problem and from them, we selected the seven most effective heuristic algorithms. Moreover, we defined a common framework in which all the algorithms were evaluated. An exhaustive comparison of all the algorithms for the SIH problem is not feasible for many reasons. For example, many algorithms are designed for simplified models of the SNP matrix and are inadequate for the matrices generated according to the current shotgun sequencing technology. In this class of algorithms, there are those (like *KMec*; Xie *et al.*, 2008) that do not allow the insertion of gaps inside the fragments or the insertion of mate pair sequences. Another class of algorithms (as in Wang *et al.*, 2006) uses genotype information to solve the SIH problem. This information can improve the accuracy of the reconstructed haplotypes, but is not often available in practice.

We restrict our analysis to those algorithms designed for a model of the SNP matrix compatible with the current shotgun technology. For all the algorithms considered, we requested the software from the corresponding authors. In the cases in which the software was not provided, we checked whether the corresponding papers contain a description of the algorithms enough accurate, allowing us to carefully re-implement them. (We were unable to include in this review all the algorithms for which we had insufficient details). Re-implementing software raises the issue of validating its correctness and running time performance. To validate our implementations we tested each algorithm. For each test, we used exactly the same parameters used by the authors (i.e. the same error rate, coverage and haplotype length). Wherever possible we also used the same datasets. For example, this is the case of Wang *et al.* (2007) and Zhao *et al.* (2005) who generate their haplotypes from the freely available dataset described in Daly *et al.* (2001). Our tests reveal that all our implementations are comparable with the original algorithms both in accuracy and running time. We also excluded some algorithms for which we received the software due to their characteristics. For example, the Branch and Bound algorithm described in Wang *et al.* (2005) is much slower than its competitors, and its running time makes it unsuitable in practice even for small datasets. The same problem is present in *HASH* (Bansal *et al.*, 2008). We also tested the PM-MFR algorithm described in Xie and Wang (2008) (in this case as well we received the software from the authors). The running time of this algorithm depends on many parameters. One of these parameters is the number of gaps in mate pair sequences. Our tests show that even for small values of this parameter, the running time of this algorithm becomes high.

Our test data are available on request, which means that they can be used in the near future to compare new ideas with the actual state of the art.

Despite the high number of real human haplotypes freely available on the Web, there are no real SNP matrices. In this article, we made the effort to generate realistic data, writing a software program that gets in input a real haplotype and simulates the actual technologies of shotgun sequencing to produce realistic SNP matrices. The SNP matrices simulator is described in Section 4.1.

The article is organized as follows: in Section 2, we formally define the SIH reconstruction problem and its most common computational models. In Section 3, we describe all the compared algorithms. Section 4 reports quality and running time evaluation. We conclude in Section 5.

2 PROBLEM FORMULATION

Due to the diploid nature of humans cells, except for male sex chromosomes, each chromosome comes in two nearly identical copies, one inherited from the mother and one from the father. Current technology of shotgun sequencing is unable to keep track of the association between a fragment and its own chromosome. Thus, from the biological point of view, the Single Individual Haplotyping (SIH) problem consists in the reassignment of each fragment to the original haplotype.

From the computational point of view, the problem was first formalized in Lancia *et al.* (2001). A fragment is represented as a string of length m such that each character corresponds to a base in the alphabet $\Sigma = \{a, c, g, t\}$ or a $-$ in case of gap. The natural way to store all n fragments is a matrix M with n rows and m columns, such that, to each row corresponds a fragment $f_i = M[i]$. The matrix M is called SNP matrix. The element $M[i][j]$ stored in the j-th entry of the i-th row of M represents the j-th SNP of the haplotype for fragment f_i. We will denote this element also as $f_i[j]$. In case f_i does not cover the j-th position of the haplotype, we have $M[i][j] = -$.

We say that a fragment f contains a gap (or is gapped) if for $i, j, k \in [1, m]$ such that $i < j < k$ we have $f[i] \neq -, f[j] = -$ and $f[k] \neq -$. If no fragments in M are gapped, then the SNP matrix is said to be gapless otherwise it is gapped.

In absence of errors in the SNP matrix, each column of M can contain one or two distinct elements. The presence of only one element indicates that the SNP in the corresponding site is homozygous otherwise the SNP is heterozygous. We say that two fragments f_i and f_j are in conflict if the following condition is true: $\exists k \in [1, n]$ such that $f_i[k] \neq f_j[k] \wedge f_i[k] \neq - \wedge f_j[k] \neq -$. According to the definition of conflict between pairs of fragments, in Lancia *et al.* (2001) the conflict graph is defined. Let $G = \{V, E\}$ be a graph such that each vertex corresponds to a fragment and there is an edge between two fragments if there is a conflict between them. If M does not contain errors, then G is bipartite (Fig. 1a). The bipartition is not necessarily unique, if the graph has several connected components. In practice, due to errors in the SNP matrix, the conflict graph is never bipartite (Fig. 1b). Thus, in this case, the SIH reconstruction problem can be formalized as the problem of removing a certain number of edges from G until the resulting graph becomes bipartite. The problem of reducing a graph to a bipartite graph is well studied in the literature where many models were proposed. Among them, in the context of the SIH problem the following formalizations are often used:

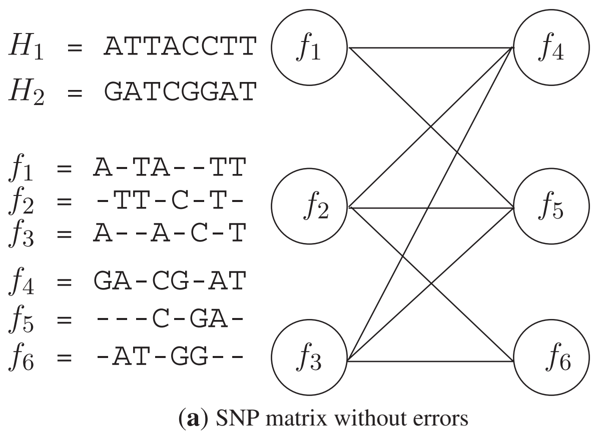
MEC (minimum error correction): determine a minimal set of entries of the matrix M whose correction to a different value induces a bipartite graph

MFR (minimum fragment removal): determine a minimal number of fragments whose removal from the input set induces a bipartite graph

MSR (minimum SNP removal): determine a minimal number of SNPs whose removal from the input set induces a bipartite graph

LHR (longest haplotype reconstruction): determine set of fragments whose removal from the input set induces a bipartite graph and the length of the induced haplotype is maximized.

In the presence of gaps, all the above problem formalizations are NP-hard. More details about the complexity of some of these problems can be found in Cilibrasi *et al.* (2007). Even if MEC is the most complex of the above models, it is the most commonly used in practice. Four of the algorithms described in this article approach the SIH problem using the MEC model (2d-mec, SHR, HapCUT) or a model derived from it (MLF). The other algorithms we considered do not follow the above problem formulations. Note that there is no proven relationship among the above problem formulations and the SIH problem (i.e. a more accurate solution of the above problems does not necessarily correspond to a better solution of the SIH problem). Since the goal of this review is to evaluate the accuracy of the algorithms with respect to the SIH problem, we derived the SNP matrices from pairs of real haplotypes from the HapMap project and used them as gold standard for evaluating the consensus haplotypes.



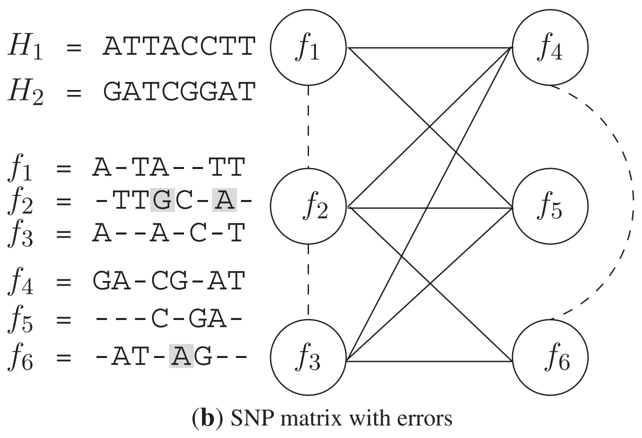


Fig. 1. (**a**) Two haplotype strings, six fragments (without errors) and the corresponding bipartite conflict graph. (**b**) The same two haplotypes, six fragments (with errors in gray) and the corresponding conflict graph (not bipartite).

We now introduce some definitions and notations that we will use later in the algorithms description. Let C be a set of rows of M and let $x_C[i] \in \Sigma$ be the character that appears most frequently at position i among the fragments in C (or $x_C[i] = -$ if all the fragments of C have a gap in position i), we define the haplotype consensus $H(C)$ deduced by C as the string of m characters such that the character at position i is $x_C[i]$.

We define the generalized Hamming distance between two fragments f_i and f_j as:

$$\mathrm{Ham}(f_i,f_j) = \sum_{k=1}^{m} \mathrm{Ham}(f_i[k],f_j[k]) \qquad (1)$$

where

$$\mathrm{Ham}(f_i[k],f_j[k]) = \begin{cases} 1 \text{ if } f_i[k] \neq f_j[k] \neq - \\ 0 \text{ otherwise} \end{cases}$$

If the generalized Hamming distance between two fragments is different from 0, we say that the two fragments are in conflict.

3 ALGORITHMS

3.1 2d-mec: a clustering algorithm for the MEC model

In Wang *et al.* (2007), a clustering algorithm is used to split the rows of M in two sets. The main contribution of the article consists in the combination of the two distance functions used by the clustering algorithm. As first distance the authors used the generalized Hamming distance *Ham* as defined in Equation (1). This distance takes into account only the number of mismatches between two fragments. The second distance is defined as follows: let f_i and f_j be two fragments

$$D'(f_i,f_j) = \sum_{k=1}^{m} d'(f_i[k],f_j[k]) \qquad (2)$$

where

$$d'(f_i[k],f_j[k]) = \begin{cases} -1 \text{ if } f_i[k] = f_j[k] \neq - \\ 1 \text{ if } f_i[k] \neq f_j[k] \neq - \\ 0 \text{ otherwise} \end{cases}$$

The definition of distance in Equation (2) also takes into account the number of matches between the two fragments. This means that given a certain fixed number of mismatches between two fragments, the more they overlap the closer they are according to D'. Note that D' is not a distance in a strict sense: in fact it can be negative and has values in the range $[-m,m]$. Moreover, the triangular inequality does not hold.

Using the above distance functions, the authors proposed a simple iterative clustering procedure. To compare two fragments, the functions *Ham* and D' are evaluated in cascade.

The algorithm proceeds as follows:

(1) for each possible pair of fragments in the SNP matrix the generalized Hamming distance is computed. Let f_i and f_j be the two furthest fragments according to *Ham*, we initialize the two sets $C_1 = f_i$ and $C_2 = f_j$.
(2) Let $H_1 = H(C_1)$ and $H_2 = H(C_2)$ be the two consensus strings derived from C_1 and C_2: all the fragments are compared with H_1 and H_2 and assigned to the corresponding closer set. If a fragment is equidistant from the two consensus strings, the distance D' is used to decide to which set assign the fragment.
(3) Once all fragments are assigned, the consensus strings H_1 and H_2 are updated and the algorithm restarts from (2). The procedure loops until a stable haplotype pair is found (i.e. when the consensus haplotypes are the same before and after the update).

3.2 Clustering algorithm for the MLF problem

In Zhao *et al.* (2005), the authors raised a weighted variant of the MEC problem called Weighted Minimum Letter Flip (WMLF). Assuming we have a matrix W, such that each entry is a number in the range $[0,1]$ representing the degree of confidence of the SNP in the same position in the matrix M. We can define a weighted version of the generalized Hamming distance between two fragments f_i and f_j as:

$$WHam(f_i,f_j) = \sum_{k=1}^{m} wh(f_i[k],f_j[k])$$

where

$$wh(f_i[k],f_j[k])=\begin{cases}\min(W[i][k],W[j][k]) \text{ if } f_i[k]\neq f_j[k]\neq -\\ 0 \text{ otherwise}\end{cases}$$

The distance *WHam* is extended to deal with haplotypes, assuming that the weight associated with each character of the consensus string is 1.

The proposed algorithm is based on the well-known one-pass k-means clustering algorithm (McQueen, 1967). The procedure initialization consists in randomly partitioning the rows of M in two sets C_1 and C_2 deriving from them two consensus strings: $H_1=H(C_1)$ and $H_2=H(C_2)$. In the main procedure loop, at each iteration C_1 and C_2 are reinitialized and a new partition of the rows in M is done. For each fragment f_i, we compare its distance from H_1 and H_2. If $WHam(f_i,H_1)>WHam(f_i,H_2)$ then f_i is assigned to C_1, otherwise the fragment is assigned to C_2. The procedure terminates when two stable haplotypes are detected. More details regarding the convergence of this method can be found in McQueen (1967). Due to the random initialization of sets C_1 and C_2, every run of this algorithm on a certain dataset can return a different haplotype consensus. To mitigate the effect of randomness, the authors run the algorithm 100 times and return as final result the consensus pair that minimizes the following target function:

$$F(C_1,C_2)=\sum_{k=1}^{2}\sum_{f\in C_k} WHam(f,H(C_k)) \qquad (3)$$

The main drawback of this algorithm is that with actual shotgun sequencing technology, the information about the confidence level of each fragment is typically not available. In this case, the algorithm has to assume the same level of confidence for each fragment: thus *WHam* reduces to *Ham* and the target function to minimize in Equation (3) reduces to the target function of the MEC model. Even in Zhao *et al.* (2005), experiments are made assuming each entry of W as equal to 1.

3.3 Fast Hare

Fast Hare (Panconesi and Sozio, 2004) is one of the first heuristics for the SIH problem. Although it was designed to work in a gapless environment, our experiments and those reported in Genovese *et al.* (2008) confirm that this method still works nicely also in the more general case in which gaps are allowed. As a preliminary step, Fast Hare removes from the SNP matrix M all those columns where there is a character (not $-$) that is over-represented with respect to the others. These columns are considered as homozygous sites. We call this reduced matrix $\hat{M}$. More formally: let $|x_c|$ be the number of occurrences of the character $x\in\Sigma$ in the column c of M. The probability to find x in c is

$$P_c(x)=\frac{|x_c|}{\sum_{\sigma\in\Sigma}|\sigma_c|}$$

If a character x in c such that $P_c(x)\geq t$ exists, the column c is removed from the SNP matrix and the character x will be inserted in the final solution. In Panconesi and Sozio (2004) and in our experiments, the threshold t is set to 0.8. The intuition behind this filtering comes from the fact that when $P_c(x)\geq t$ column c represents a homozygous SNP (thus the column does not help in the reconstruction procedure) and the positions (not gaps) not containing x are errors. As the first step, Fast Hare sorts the fragments of $\hat{M}$ according to the following ordering criterion: let k_i and k_j be, respectively, the position of the first character not gap in f_i and f_j, thus $k_i\leq k_j \Rightarrow f_i \preceq f_j$. After sorting the fragments of $\hat{M}$, Fast Hare initialize two sets $C_1=C_2=\emptyset$. At this point, according to their order, the fragments of $\hat{M}$ are scanned one at a time. The first fragment is assigned to C_1. Considering the fragment f_i, Fast Hare computes its similarity with the two partial consensus strings $H(C_1)$ and $H(C_2)$. As similarity score, Fast Hare uses $-D'$ where D' is the distance we defined in Equation (2). The fragment is assigned to the set whose consensus shows higher similarity. Note that the values of $-D'$ stand in the range $[-1,1]$ and for all fragments holds $-D'(f,H(\emptyset))=0$.

3.4 SpeedHap

SpeedHap (Genovese *et al.*, 2007, 2008) approaches the haplotype assembly problem differently from previously described algorithms. Instead of considering each fragment as a whole, it attempts to solve n instances of the haplotyping problem on 1-base long fragments and combines results.

SpeedHap is a greedy heuristic. It builds its solution in a pre-processing phase and three main phases. The goal of each phase is to exploit the outcome solution of the preceding phase and improve it by relaxing some constraints.

Pre-processing: in this phase, SpeedHap performs a statistical analysis of the columns of the SNP matrix attempting to locate detectable errors and, if possible, correct them. In this phase, the heuristic also set up some data structures.

First phase: in this phase, the heuristic selects an initial set of columns such that they are likely to contain as few errors as possible. For each column, SpeedHap builds a set G_i (called *profile*) in which each element is the set of all the indices of the fragments containing the same character in position i. It is easy to observe that only profiles having two elements (i.e. columns of the SNP matrix containing exactly two distinct characters) are of interest since an empty profile corresponds to a hole in the haplotype, a profile with just one element corresponds to a homozygous site, a profile with more than two characters must contains errors. Let $P_i=(P_i(1),P_i(2))$ be the profile of column i such that it corresponds to a heterozygous site. Given two columns i and j, we can define the *error matrix* as

$$E_{i,j}=\begin{pmatrix} P_i(1)\cap P_j(1) & P_i(1)\cap P_j(2)\\ P_i(2)\cap P_j(1) & P_i(2)\cap P_j(2)\end{pmatrix}$$

When the error matrix has positive values only in one diagonal and it is of full rank, there are no detectable conflicts between the two involved columns. Now, consider a graph such that it has a vertex for each column of M corresponding to a heterozygous site and there is an edge between two vertices if the corresponding error matrix does not reveal inconsistencies (i.e. the matrix is diagonal and of full rank). Using a DFS search, we can partition the graph in connected components. Each component corresponds to a set of columns not conflicting among themselves. The initial partitioning of the fragments of M is extracted from the largest set.

Second phase: in this phase, the algorithm works in a similar manner. The partitioning obtained from the previous step acts as a special profile (pivot). All the columns not involved in the previous phase are compared with the pivot and the error matrix is computed. If the error matrix does not show inconsistencies, the column is included in the solution. This procedure is repeated iteratively until it is no longer possible to add new columns to the pivot.

Third phase: in this phase, some constraints are relaxed: the insertion of the columns in the final solution no longer requires an error matrix of full rank.

Another important contribution of SpeedHap is the use of the context for resolving ambiguities in the final haplotype reconstruction. In Genovese *et al.* (2008), the authors parsed a large database of human haplotypes measuring the empirical entropy of order up to 2. As a result, they show that there seems to exist a statistical correlation between the base in a certain SNP site, its preceding SNP site and its succeeding one.

When the coverage (i.e. the number of fragments that cover a certain position) is low and the error rate is relatively high, it is not infrequent the case in which, building a haplotype, exactly half of the fragments in a certain position have a certain value and half of them have another value. In this case, the choice of which character should appear in the haplotype is arbitrary. To break ties, SpeedHap exploits the statistical correlation among contiguous sites. Consider the case in which the procedure has to choose for the site in position i whether select A or B. Let x be the character in position $i-1$ and z the character in position $i+1$. The procedure will decide for: A if the empirical entropy of the string xAz is lower than those of the string xBz, otherwise it will decide for B.

3.5 HapCUT

HapCUT (Bansal and Bafna, 2008) approaches the haplotype assembly as a MAX-CUT problem. The HapCUT algorithm considers the submatrix of the SNP matrix in which we remove all the columns corresponding to Homozygous SNPs and all the columns in which there are present more than two distinct bases (i.e. there must be at least a mistaken base). Let us call the resulting matrix X. Due to the fact that each column of X contains exactly two possible SNP values, it can be represented using the restricted alphabet $0,1,-$. The haplotype pair H associated with X is composed by a binary string h and its bit-wise complement $\hat{h}$.

Given a certain haplotype pair H, the authors defined a graph $G_X(H)$ such that there is a vertex for each column of the matrix X and there is an edge between two vertices of $G_X(H)$ if the corresponding columns in X are linked by at least one fragment. Consider the fragment X_i such that it covers both positions j and k. Let $X_i[j,k]$ and $H[j,k]$ represent the restriction of X_i and H to loci j and k. There are two cases: $X_i[j,k]$ matches one of the two haplotype strings of $H[j,k]$, or $X_i[j,k]$ does not match any. The weight $w_H(j,k)$ associated with the edge between node j and k in the graph $G_X(H)$ is given by the number of fragments such that $X_i[j,k]$ does not match any string in $H[j,k]$ minus the number of fragments such that the match exists. The higher $w_H(j,k)$, the weaker is the correlation between the haplotype pair H and the SNP matrix restricted to columns j and k. Let $(S,X-S)$ be a cut of G, the weight of the cut is defined as follows:

$$w_H(S) = \sum_{j\in S,k\in X-S} w_H(j,k)$$

Consider the haplotype pair H_S derived from H by flipping all the elements involved in S. The authors showed that if $w_H(S)$ is positive for the graph $G_X(H)$ then the following holds:

$$\text{MEC}(H_S) = \text{MEC}(H) - w_H(S) > \text{MEC}(H)$$

As a consequence of the above result, the problem of finding a haplotype pair minimizing the MEC score is reduced to the problem of finding a max-cut in $G_X(H)$. This problem is well known to be NP-complete (Karp, 1972), thus heuristic methods are often used.

The HapCUT procedure exploits the connection between the MEC optimization and the max-cut problem. Starting from a random haplotype pair, HapCUT iteratively attempts to refine the haplotype pair to reduce the MEC score. At each iteration, the algorithm proceeds as follows: (1) compute the graph $G_X(H)$, (2) compute a max-cut S using a greedy heuristic like that in Sahni and Gonzalez (1974), (3) if the MEC score of the pair H_S is smaller than the score of H, keeps as new haplotype pair H_S.

The procedure loops until it is no longer possible to reduce the MEC score.

3.6 DGS

In Levy *et al.* (2007), the authors described a large study on genome sequencing. In the paper they also described a good algorithm for the SIH problem. For lack of a better name, we call this algorithm *DGS*. As in HapCUT, this algorithm works with a submatrix of M in which we remove all the columns corresponding to homozygous sites and those with more than two distinct values. Let us again call this matrix X. Even in this case, X can be represented using the restricted alphabet $0,1,-$. The haplotype pair H associated with X is composed by a binary string h and its bit-wise complement $\hat{h}$. The DGS procedure works in two phases: an initialization in which we build a pair of initial haplotypes and a refinement step in which haplotypes are iteratively refined. The initialization works as follows:

(1) the fragment with the minimal number of gaps is used to initialize a haplotype. The other haplotype is initialized with the complementary string;

(2) until no more fragments share non-missing information with a haplotype, select the fragment such that the number of columns it has in common with one haplotype minus number of columns indicating the other haplotype is maximal and assign it to the corresponding haplotype. The other haplotype is updated with the complementary string.

The second phase iteratively refines the haplotype consensus strings and stops when, at the end of an iteration, the solution no longer changes. At each iteration, the haplotype consensus strings are determined by majority rule, then each fragment is associated with the closest haplotype.

3.7 SHR-three

In Chen *et al.* (2008), the authors proposed a probabilistic framework to approach the SIH problem. According to the proposed model, the authors designed a novel randomized algorithm [i.e. an algorithm that receives, in addition to its input data, a stream of random bits that it can use for the purpose of making random choices (Karp, 1991)] and generalize it to handle reading errors and gaps. The most general variant of this algorithm is called *SHR-three*. The algorithm requires as input the SNP matrix and a parameter u that controls the number of iterations made by the main loop. As proposed by the authors in their experiments, we set $u=10$.

The *SHR-three* main loop is as follows:

(1) select at random two fragments f_1 and f_2 from M and assign to each of them an empty set (C_1 to f_1 and C_2 to f_2);

(2) each fragment of M is compared through the generalized Hamming distance with f_1 and f_2 and inserted in the set related to the closest fragment;

(3) for each of the two sets compute the MEC score (i.e. the sum of the distances among each fragment in C_i and f_i for $i=\{1,2\}$) and get as score the highest value; and

(4) if the computed score is lower than the previous computed ones, than $\hat{C_1}=C_1$ and $\hat{C_2}=C_2$.

As consensus strings *SHR-three* returns $H(\hat{C_1})$ and $H(\hat{C_2})$.

4 RESULTS

4.1 Input data and fragment generation

The research project *HapMap* (HapMap, 2005) has produced a certain number of maps of the human haplotypes that are publicly available. In the first two phases of the project, the consortium used PHASE software for estimating haplotypes from population genotype data. Recently, in the third phase, the consortium has adopted sequencing for estimating haplotypes. For our experiments, we used the *Phase I HapMap* data[1]. A detailed description of the dataset characteristics can be found in HapMap (2005).

The Phase I HapMap dataset consists of all 22 chromosomes (for females, the haplotypes of the X chromosome are also available) of 269 different individuals coming from four different populations

- **CEU**: Utah residents with ancestry from northern and western Europe (90 individuals)
- **YRI**: Yoruba in Ibadan, Nigeria (90 individuals)
- **HCB**: Han Chinese in Beijing, China (45 individuals)
- **JPT**: Japanese in Tokyo, Japan (44 individuals).

Individuals of the HCB and JPT populations are unrelated among each other. Individuals of CEU and YRI are parent-offspring trios. To prevent possible undesired effects due to the parental relationship, in our experiments, we consider only the haplotypes of the parents. This reduces the population to 209 unrelated individuals.

Thus, we were able to generate the fragments and the SNP matrices from real data instead of using synthetic haplotypes as input. Using real haplotypes, the Hamming distance between them is no longer a free parameter. We observed that haplotype pairs show a great variability in the Hamming distance. Typical values of Hamming distance are in the range $[0.4m, m]$. To evaluate whether Hamming distance produces effects in the outcome of the tested algorithms, we performed two sets of experiments: one in which we select haplotype pairs having Hamming distance $<0.7\,m$ and one in which the Hamming distance between the haplotypes considered is $>0.7\,m$. Our tests show that Hamming distance does not affect the performance (both in terms of reconstruction rate and running time) of the algorithms considered.

Distilling realistic SNP matrices from the haplotypes is crucial for designing meaningful experiments. In this sense one should pay attention to three main aspects: the simulation of the shotgun sequencing process, the estimation of some technological parameters of sequencers and some intrinsic characteristics of human DNA. To simulate the shotgun sequencing process, we used the widely accepted algorithm described in Myers (1999). According to Li *et al.* (2003) and Metzker (2005), current shotgun sequencers are able to manage DNA fragments of hundreds of bases and the average distance in bp of two SNPs in human DNA is quantified in 300 bp on average. Thus, each DNA fragment covers roughly a number of SNPs in the range $[3,7]$. The distribution of errors in the SNP matrices depends on the characteristics of sequencers. A hint about this distribution for different manufacturers can be found in Metzker (2005).

According to the above considerations, the generation of the SNP matrix is as follows: given a pair of haplotypes of length l, each haplotype is replicated c times (the parameter c is called coverage), then each copy is broken into non-overlapping fragments whose length is in the range $[3,7]$. According to a certain probability some fragments are merged again in order to simulate matepair sequences (In our experiments, at the end of this phase, globally 50% of the fragments are 1-gapped). Once created, we arrange the fragments in a matrix and insert errors according to a uniform distribution. Note that the number of fragments is not determined a priori but depends on the length l, on the coverage c and on the distribution of the fragment lengths.

4.2 Quality evaluation

To evaluate the quality of the algorithms tested, we use a slightly modified version of the well-known error rate. Let $H=(h_1,h_2)$ be the pair of correct haplotypes each of which has length m. Let $\hat{H}=(\hat{h}_1,\hat{h}_2)$ be the pair of consensus haplotypes returned by an algorithm. According to the standard definition, the reconstruction rate is

$$R_{\hat{H},H}=1-\frac{\min(D(h_1,\hat{h}_1)+D(h_2,\hat{h}_2),D(h_1,\hat{h}_2)+D(h_2,\hat{h}_1))}{2m}$$

where D is the generalized Hamming distance and m the haplotype length. The main disadvantage of using the above formula is that the generalized Hamming distance assigns the same score to two matching characters and to characters matching with gaps. As a consequence of this, a pair of empty haplotypes (i.e. haplotypes in which each position contains a gap) receives the same positive evaluation of the correct solution. To remove the bias introduced dealing with gaps, we used in the computation of the error rate a variant of the Hamming distance in which gaps receive the same penalty of errors. We defined D in greater detail as follows:

$$D(h_i,\hat{h}_j)=\sum_{k=1}^{m} d(h_i[k],\hat{h}_j[k])$$

where

$$d(h_i[k],\hat{h}_j[k])=\begin{cases}0 \text{ if } h_i[k]=\hat{h}_j[k]\\ 1 \text{ otherwise}\end{cases}$$

For each parameter assignment, Table 1 reports the average reconstruction rate (over 100 runs on distinct instances of SNP matrices) of all the algorithms for the case in which the average Hamming distance between the input haplotypes is $>0.7\,m$. We do not report results for the case in which the Hamming distance is

[1]Original HapMap haplotypes can be downloaded from: http://hapmap.ncbi.nlm.nih.gov/downloads/phasing/2005-03_phaseI/full/

Table 1. Each entry in the table represents the average, over 100 randomly selected HapMap strings, of the Reconstruction Rate when the Hamming distance is in the range $[0.7m, m]$

	Algo	$e=0.0$				$e=0.1$				$e=0.2$				$e=0.3$			
		$c=3$	$c=5$	$c=8$	$c=10$	$c=3$	$c=5$	$c=8$	$c=10$	$c=3$	$c=5$	$c=8$	$c=10$	$c=3$	$c=5$	$c=8$	$c=10$
l = 100	Baseline	1.000	1.000	1.000	1.000	0.971	0.992	0.997	0.999	0.898	0.944	0.967	0.980	0.787	0.840	0.878	0.903
	SpeedHap	**0.999**	**1.000**	**1.000**	**1.000**	0.895	0.967	**0.989**	0.990	0.623	0.799	0.852	0.865	0.480	0.637	0.667	0.676
	Fast Hare	**0.999**	**0.999**	**1.000**	**1.000**	0.919	0.965	**0.993**	**0.998**	0.715	0.797	0.881	0.915	0.617	0.639	0.661	0.675
	2d-mec	0.990	**0.997**	**1.000**	**1.000**	0.912	0.951	0.983	0.988	0.738	0.793	0.873	0.894	**0.623**	0.640	0.675	0.678
	HapCUT	**1.000**	**1.000**	**1.000**	**1.000**	**0.929**	0.920	0.901	0.892	**0.782**	**0.838**	0.864	0.871	0.602	0.629	0.673	0.709
	MLF	0.973	0.992	**0.997**	**0.998**	0.889	0.970	0.985	**0.995**	0.725	**0.836**	**0.918**	**0.938**	0.618	**0.653**	**0.697**	**0.715**
	SHR-three	0.816	0.861	0.912	0.944	0.696	0.738	0.758	0.762	0.615	0.655	0.681	0.699	0.557	0.599	0.632	0.632
	DGS	**1.000**	**1.000**	**1.000**	**1.000**	**0.930**	**0.985**	**0.989**	**0.997**	0.725	0.813	0.878	0.917	0.611	0.647	0.663	0.688
l = 350	Baseline	1.000	1.000	1.000	1.000	0.978	0.990	0.997	0.999	0.896	0.943	0.968	0.981	0.783	0.840	0.873	0.903
	SpeedHap	**0.999**	**1.000**	**1.000**	**1.000**	0.819	0.959	0.984	0.984	0.439	0.729	0.825	0.855	0.251	0.578	**0.629**	0.638
	Fast Hare	0.990	**0.999**	**1.000**	**0.999**	0.871	0.945	0.985	**0.995**	0.684	0.746	0.853	**0.877**	**0.590**	0.602	0.626	0.644
	2d-mec	0.965	0.993	**0.998**	**0.999**	0.837	0.913	0.964	0.978	0.675	0.729	0.791	0.817	**0.593**	**0.606**	0.623	0.634
	HapCUT	**1.000**	**1.000**	**1.000**	**1.000**	**0.930**	0.913	0.896	0.888	**0.771**	**0.831**	**0.862**	0.867	0.565	0.582	0.621	**0.664**
	MLF	0.864	0.929	0.969	0.981	0.752	0.858	0.933	0.962	0.642	0.728	0.798	0.831	0.581	**0.606**	**0.634**	0.641
	SHR-three	0.830	0.829	0.895	0.878	0.682	0.724	0.742	0.728	0.591	0.632	0.670	0.668	0.548	0.557	0.604	0.619
	DGS	**0.999**	**1.000**	**1.000**	**1.000**	**0.926**	**0.978**	**0.996**	**0.998**	0.691	0.769	0.842	**0.878**	0.578	**0.609**	0.628	0.641
l = 700	Baseline	1.000	1.000	1.000	1.000	0.971	0.991	0.997	0.999	0.898	0.942	0.966	0.980	0.786	0.838	0.875	0.902
	SpeedHap	**0.999**	**1.000**	**1.000**	**1.000**	0.705	0.947	**0.985**	0.986	0.199	0.681	0.801	0.813	0.095	0.523	**0.616**	**0.627**
	Fast Hare	**0.988**	**0.999**	**1.000**	**0.999**	**0.829**	0.949	**0.986**	**0.995**	0.652	0.712	0.808	**0.872**	**0.581**	0.591	0.615	0.616
	2d-mec	0.946	0.976	0.992	**0.997**	0.786	0.880	0.948	0.965	0.647	0.697	0.751	0.778	**0.583**	**0.596**	**0.613**	0.622
	HapCUT	**1.000**	**1.000**	**1.000**	**1.000**	**0.927**	0.916	0.896	0.889	**0.753**	0.825	**0.856**	0.861	0.552	0.555	0.597	0.645
	MLF	0.787	0.854	0.919	0.933	0.698	0.809	0.863	0.884	0.624	0.682	0.747	0.765	0.570	**0.594**	**0.614**	**0.625**
	SHR-three	0.781	0.832	0.868	0.898	0.668	0.716	0.743	0.726	0.591	0.617	0.653	0.675	0.536	0.562	0.611	**0.625**
	DGS	**0.999**	**1.000**	**1.000**	**1.000**	**0.931**	**0.977**	**0.987**	**0.997**	0.669	**0.741**	0.818	0.861	0.573	0.595	**0.614**	0.622

The free parameters are: (i) the haplotype length $l=100, 350, 700$; (ii) the coverage $c=3, 5, 8, 10$; and (iii) the error rate $e=0\%, 10\%, 20\%, 30\%$. In bold the algorithms with highest performance, in gray the algorithms with the second-best performance. We consider as equal to the performance of two algorithms when the difference between their Reconstruction Rate is in the range $[0, 005]$.

lower since the performances (both in terms of reconstruction rate and running time) of the algorithms are similar.

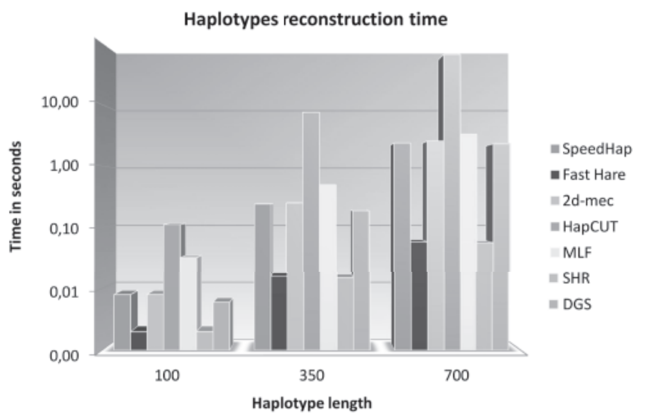
To compare all the algorithms with the optimal haplotype reconstruction, Table 1 also reports the reconstruction rate for the naive baseline algorithm that can access the true fragment bipartition and simply reconstruct haplotypes by majority.

As shown in Table 1, when the error rate is low (up to 0.1) the DGS algorithm performs permanently better than the others. For higher error rate, there is no algorithm that works clearly better than the others. For small fragments (with $l=100$) and coverage higher than three, MLF outperforms the others. For the other cases, there is no strong winner. If we consider the best and the second-best result (highlighted values in Table 1), we observe that Fast Hare should be considered reliable for an error rate up to 0.2. In the case in which the haplotype length is set to 100, MLF can be considered the most reliable algorithm. When the haplotype length is set to 350 bp, the most reliable algorithm is DGS followed by Fast Hare. It is possible to observe that for a low error rate DGS is always among the best algorithms. For low error rate SpeedHap performs quite well, while for high error rate MLF becomes reliable. The case in which the haplotype length is set to 700 is similar to the previous case. The DGS algorithm is reliable in all settings and outperforms the other algorithms for low error rate. Even in this case, Fast Hare is the second-best algorithm.

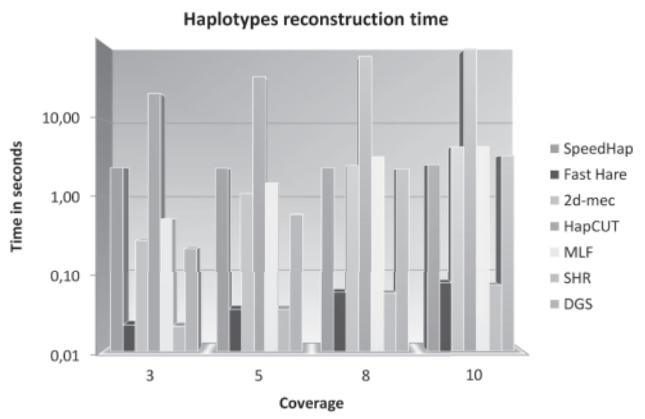
The MLF performances highlight that the higher the error rate, the better the strategy of computing many independent haplotype pairs (and return the solution that minimizes the MEC score) works.

It is surprising that even in the cases in which the error rate is set to 0, the 2d-mec, MLF and SHR-three algorithms can introduce errors in their final solution. This can be explained by the fact that their initializations involve random choices that can heavily affect the final result. The other algorithms are almost always able to rebuild the haplotypes entirely without errors (sometimes introducing some gaps).

It should be observed that the SHR-three algorithm consistently has a reconstruction rate lower than that of the other algorithms. This can be imputed to the initialization step of the algorithm in which the two sets C_1 and C_2 are initialized with two random fragments that have high probability of being mates in the correct bipartition.



(a) Variable haplotype length, $c = 8$



(b) Variable coverage, $l = 700$

Fig. 2. Average running time expressed in seconds over 100 instances for different settings of the haplotype length (**a**) and coverage (**b**). The error rate is set to 0.2.

Another important observation is that when the error rate is 0.3 and the coverage is 3, the SpeedHap algorithm performs much worse than in the other cases. This is due to the fact that the high error rate and low coverage makes the first phase of SpeedHap to be unable to select the set of columns that are likely to contain few errors. The effect of this phenomenon is that SpeedHap is unable to assign most of the rows, hence a large part of the haplotypes is filled with gaps.

Looking at the algorithms that compute a certain number of solutions and return the one that explicitly minimize the MEC score (MLF and SHR-three), we can observe that their performances are not among the best ones. According to this observation, it seems that the target function of minimizing the MEC score performs worse than other approaches. We do not want to claim that this observation is necessarily true, in fact (even if not explicitly) DGS minimize the MEC score and attains good results.

4.3 Running time evaluation

In this section, we show the running time of the algorithms compared. Except for HapCUT whose implementation is freely provided by the authors, we reimplemented all the other algorithms using Python v2.6. For each parameter assignment, we run the algorithms over 100 different instances and collect the average running time. For our tests, we used a Pentium D 3.2 GHz endowed with 3 Gb of RAM.

Figure 2 shows a comparison of the haplotype reconstruction times. The slowest algorithm is HapCUT, while the two fastest algorithms are Fast Hare and SHR-three. Except for HapCUT, it is possible to observe that all the algorithms are able to solve all the instances of the reconstruction problem for each parameter assignment in <5 s in the worst case, making all of them suitable for real applications. Instead, HapCUT running time does not scale and requires tens of seconds for large instances of the reconstruction problem.

Figure 2 also shows that both haplotype length and coverage affect the final running time of all the algorithms in different ways. Coverage involves a linear increase of the running time for all algorithms, while haplotype length involves a quadratic increase. Due to lack of space, in this article we do not report results in the case in which we vary the error rate or the Hamming distance because we observed that they have no effect on the running time of all the algorithms.

5 CONCLUSIONS

The SIH problem is one of the core problems in the whole genome sequencing. Due to the presence of various types of errors and missing data in the fragments, the problem is very hard to solve. In recent years, many algorithms and heuristics were proposed in the literature, but a systematic comparison between them is still missing. In this article, we survey seven algorithms that are among the most commonly used for the SIH problem. We also developed a common framework to compare them. Our framework simulates the actual technology for shotgun sequencing to generate realistic SNP matrices from real human haplotypes collected from the HapMap project. The web-based interface of the framework allows to control all the generation parameters. The Hamming distance and fragment size can be specified only as a range. Due to the fact that we use real haplotypes, it is possible (for too-small ranges) that the selected chromosome does not contain a portion of haplotype with the desired Hamming distance. In this case an error is raised. The choice of the generation parameters is heavily influenced by the sequencing hardware, and it can have considerable impact on the final results. To facilitate the choice of parameters, our framework suggests default settings compatible with actual standards. Our experiments show that the DGS algorithm can be considered the best choice, especially in those experiments in which some parameters cannot be estimated in advance. The data we used for the comparison are available upon request, and thus can be used in the near future to compare novel algorithmic ideas with the actual state of the art.

ACKNOWLEDGEMENTS

We would like to thank Rui-Sheng Wang, Jianxin Wang, Minzhu Xie, Zhixiang Chen and Zhiyu Zhao who provided us with their software and allowed us to test them.

Funding: Italian Registry of .it ccTLD (partially); the EU funded 7FP Virtual Physiological Human Network of Excellence (VPH NoE) (contract number 223920) (partially).

Conflict of Interest: none declared.

REFERENCES

Bansal,V. and Bafna,V. (2008) HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. In *European Conference on Computational Biology*, Cagliari, Italy, pp. 153–159.

Bansal,V. *et al.* (2008) An MCMC algorithm for haplotype assembly from whole-genome sequence data. *Genome Res.*, **18**, 1336–1346.

Chen,Z. *et al.* (2008) Linear time probabilistic algorithms for the singular haplotype reconstruction problem from SNP fragments. *J. Comput. Biol.*, **15**, 535–546.

Cilibrasi,R. *et al.* (2007) On the complexity of the single individual SNP haplotyping problem. *Algorithmica*, **49**, 13–36.

Daly,M.J. *et al.* (2001) High-resolution haplotype structure in the human genome. *Nat. Genet.*, **29**, 229–232.

Frazer,K. *et al.* (2007) A second generation human haplotype map of over 3.1 million SNPs. *Nature*, **449**, 851–861.

Genovese,L.M. *et al.* (2007) A fast and accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate and low coverage. In *Workshop on Algorithms in Bioinformatics, Philadelphia, PA*, *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, pp. 49–60.

Genovese,L.M. *et al.* (2008) SpeedHap: an accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate and low coverage. *EEE/ACM Trans. Comput. Biol. Bioinform.*, **5**, 492–502.

HapMap (2005) A haplotype map of the human genome. *Nature*, **437**, 1299–1320.

Karp,R.M. (1972) Reducibility among combinatorial problems. *Complex. Comput. Comput.*, 85–103.

Karp,R.M. (1991) An introduction to randomized algorithms. *Discrete Appl. Math.*, **34**, 165–201.

Lancia,G. *et al.* (2001) SNPs problems, complexity, and algorithms. In *Proceedings of the Ninth European Symposium on Algorithms, Aarhus, Denmark*, *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, pp. 182–193.

Levy,S. *et al.* (2007) The diploid genome sequence of an individual human. *PLoS Biol.*, **5**, 2113–2144.

Li,L. *et al.* (2003) Haplotype reconstruction from SNP alignment. In *Proceedings of the Seventh International Conference on Computational Molecular Biology, Lisbon, Portugal*, *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, pp. 207–216.

McQueen,J. (1967) Some methods for classification and analysis of multivariate observations. In *Fifth Berkeley Symposium on Mathematics, Statistics, and Probability, Statistical Laboratory of the University of California, Berkeley*. University of California Press, Berkeley, CA, pp. 281–298.

Metzker,M.L. (2005) Emerging technologies in DNA sequencing. *Genome Res.*, **15**, 1767–1776.

Morozova,O. and Marra,M.A. (2008) Applications of next-generation sequencing technologies in functional genomics. *J. Genomics*, **5**, 255–264.

Myers,G. (1999) A dataset generator for whole genome shotgun sequencing. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, AAAI, Heidelberg, Germany, pp. 202–210.

Panconesi,A. and Sozio,M. (2004) Fast hare: a fast heuristic for single individual SNP haplotype reconstruction. In *Workshop on Algorithms in Bioinformatics, Bergen, Norway*, *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, pp. 266–277.

Sahni,S. and Gonzalez,T. (1974) P-complete problems and approximate solutions. *Annu. Symp. Switching Automata Theory*, **23**, 28–32.

Via,M. *et al.* (2010) The 1000 genomes project: new opportunities for research and social challenges. *Genome Med.*, **2**, 3.

Wang,R. *et al.* (2005) Haplotype reconstruction from SNP fragments by minimum error correction. *Bioinformatics*, **21**, 2456–2462.

Wang,R. *et al.* (2006) A markov chain model for haplotype assembly from SNP fragments. *Genome Inform.*, **17**, 162–171.

Wang,Y. *et al.* (2007) A clustering algorithm based on two distance functions for MEC model. *J. Comput. Biol. Chem.*, **31**, 148–150.

Xie,M. and Wang,J. (2008) An improved (and practical) parameterized algorithm for the individual haplotyping problem MFR with mate-pairs. *Algorithmica*, **52**, 250–266.

Xie,M. *et al.* (2008) A practical exact algorithm for the individual haplotyping problem MEC. In *BMEI : Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics*, IEEE, Sanya, Hainan, China, pp. 72–76.

Zhao,Y. *et al.* (2005) Haplotype assembly from aligned weighted SNP fragments. *J. Comput. Biol. Chem.*, **29**, 281–287.

Zhao,Y. *et al.* (2007) An overview of the haplotype problems and algorithms. *Front. Comput. Sci. China*, **1**, 272–282.