Algorithmic approaches for the single individual haplotyping problem

RAIRO Operations Research
Will be set by the publisher

ALGORITHMIC APPROACHES FOR THE SINGLE INDIVIDUAL HAPLOTYPING PROBLEM

Giuseppe Lancia[1]

Abstract. Since its introduction in 2001, the Single Individual Haplotyping problem has received an ever-increasing attention from the scientific community. In this paper we survey, in the form of an annotated bibliography, the developments in the study of the problem from its origin until our days.

1. Introduction

The collection of a large amount of genomic data over the past few years has shown that our genetic makeup is remarkably well-conserved. Generally speaking, the differences at genomic level between any two individuals amount to less than 5% of their DNA sequences. This fact implies that the differences at the phenotype level (i.e., in the way the individuals look) must be caused by small regions of differences in the genomes. The smallest possible region consists of a single nucleotide and therefore it is called *Single Nucleotide Polymorphism* or SNP (pronounced "snip"). SNPs are the most common form of human genetic variation and their importance can hardly be overestimated. They are used, for example, in medical, drug-design, diagnostic, and forensic applications.

Generally speaking, a *polymorphism* is a trait common to everybody which can take different forms, ranging in a limited set of possibilities, called *alleles* (for example, the color of the eyes is a polymorphism, whose alleles include blue, black, brown, green). As far as SNPs are concerned, each SNP is a specific nucleotide site at which a statistically significant variability can be observed within a population. A SNP is almost always a polymorphism with only two alleles (out of the four possible, i.e., `A`, `C`, `G` and `T`). For a nucleotide site to be considered a SNP, it must

[1] DIMI, University of Udine, Via delle Scienze 206, 33100 Udine, Italy.

Individual 1, paternal:	caggtcc**C**tatttC**c**aggcgc**C**gtatacttcgacggg**T**ctata
Individual 1, maternal:	caggtcc**G**tatttA**c**aggcgc**G**gtatacttcgacggg**T**ctata
Individual 2, paternal:	caggtcc**C**tatttA**c**aggcgc**G**gtatacttcgacggg**T**ctata
Individual 2, maternal:	caggtcc**G**tatttC**c**aggcgc**G**gtatacttcgacggg**C**ctata
Individual 3, paternal:	caggtcc**C**tatttA**c**aggcgc**G**gtatacttcgacggg**T**ctata
Individual 3, maternal:	caggtcc**G**tatttA**c**aggcgc**C**gtatacttcgacggg**C**ctata

FIGURE 1. A chromosome in 3 individuals. There are 4 SNPs.

be the case that the less frequent allele is found in the population with some significant frequency.

Humans can be of two sexes and each individual has two parents, one for each sex. The human genome is then organized in pairs of chromosomes. In each chromosome pair, one of the chromosome copies is inherited from the father while the other is inherited from the mother. In general, the organisms whose genome is organized in pairs of homologous chromosomes are called *diploid* organisms. For a diploid organism, at each SNP an individual can either be *homozygous* (i.e., possess the same allele on both chromosomes) or *heterozygous* (i.e., possess two different alleles). The values of a sequence of SNPs on a particular chromosome copy define a *haplotype*.

In Fig. 1, we illustrate a simple example, showing a chromosome in three individuals. For each individual, the pair of his chromosome copies are reported. In this example there are four SNPs. The alleles for SNP 1 are `C` and `G`, while for SNP 4 they are `T` and `C`. Individual 1 is heterozygous for SNPs 1, 2 and 3, and homozygous for SNP 4. His haplotypes are `CCCT` and `GAGT`. The haplotypes of individual 3 are `CAGT` and `GACC`.

Haplotyping an individual consists in determining his two haplotypes for a given chromosome. With the larger availability in SNP genomic data, the recent years have seen the birth of many new computational problems related to haplotyping. Most of these problems are motivated by the fact that it can be difficult and/or very expensive to determine the haplotypes experimentally, so that *ad hoc* algorithms must be used to correct data errors or to infer missing data.

In this survey we will address the haplotyping problem for a single individual. For haplotyping problems defined over sets of individuals (also called *population haplotyping* problems), the reader is referred to [1]. The single individual haplotyping problem arises when haplotype data coming from experiments is inconsistent with the existence of exactly two parents for an individual. This inconsistency can be due to experimental errors and/or to missing data.

2. Single Individual Haplotyping

The process of turning the sequence of nucleotides in a DNA molecule into a string over the DNA alphabet is called *sequencing*. A sequencer is a machine that,

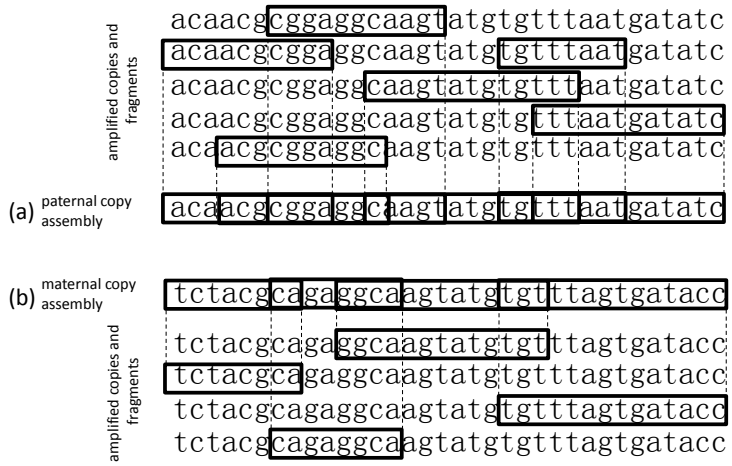


FIGURE 2. Sequence reads and assembly of the two chromosome copies.

given a DNA molecule, outputs the string of `A`'s, `T`'s, `C`'s, and `G`'s corresponding to the ordered nucleotides comprising the sequence. To each letter, the sequencer attaches a value (called the confidence level) which expresses the probability that the corresponding nucleotide has been correctly identified.

The main problem with sequencing is that, due technological limitations, it is impossible to sequence a long DNA molecule at once. What can be done, however, is to sequence short DNA fragments (also called *reads*), of length of about 1,000 nucleotides each, which can be seen at small "windows" showing a substring of the target molecule. In order to sequence a long DNA molecule, the molecule must first *amplified*, i.e., many copies of the molecule must be created. One possible way to achieve this goal is by using an experiment called *Polymerase Chain Reaction* (PCR, cite). The copies are then broken, at random, into several small fragments, which are fed to a sequencer that will sequence those of the right size. The amplification phase is necessary so that the reads can have non-empty overlap. From the overlap of two reads, one may infer (through a process called *fragment assembly*) a longer fragment, and so on, until eventually the original DNA sequence is reconstructed. The overall procedure, known as *shotgun sequencing*, was validated in the late 90's by Celera Genomics, a private biotech company trying to achieve the completion of the sequencing of the human genome faster than with other experimental techniques of the time [2,3]. At Celera the fragments were read by proprietary sequencers and then assembled back into the original sequence with the use of sophisticated algorithms and powerful computers.

In Figure 2 we show an example in which the two chromosome copies (a) and (b) have been amplified, and then a set of fragments (denoted by rectangular boxes) have been sequenced. The objective is to retrieve (a) and (b) given as input the set of sequenced fragments. The major difficulty obstructing this goal is that, during the amplification phase, the paternal and the maternal chromosome copies are amplified together, so that it is not known for each random read if it belongs to paternal or to the maternal original copy. One major problem in reconstructing the haplotypes consists therefore in segregating the paternal fragments from the maternal ones. In addition to the individual fragments described so far, there may be cases in which some extra information relates some fragments to some others. In particular, there exists a sequencing technique that allows to sequence the ends of a long fragment, i.e., to read a few hundred nucleotides at each of the two extremes. This experiment yields therefore pairs of reads, called *mate pairs*, which can then be part of the input data. Although even with this technique only about one thousand nucleotides are read at each end of the target, the result is stronger than reading two individual fragments (one for each end) since in this case it is known that the two reads must come from the same original chromosome copy. Furthermore, the experiment returns a fairly precise estimate of the distance, expressed in number of nucleotides, between the two reads of a mate pair.

Even with the best possible technology, sequencing errors are unavoidable. The main errors are due to bases that have been miscalled or skipped altogether, or to the presence of *contaminants*, i.e., DNA coming from organisms other than the one that had to be sequenced. Due to these experimental errors, the reconstruction of the haplotypes, given the the reads coming from sequencing, is not always straightforward and may require the correction of the input data. In a general way, the *haplotyping problem for an individual* can then be informally stated as follows:

> *Given inconsistent haplotype data coming from the sequencing of an individual's chromosome, find and correct the errors in the data so as to retrieve a consistent pair of haplotypes.*

Depending on what type of errors one is after, there can be many versions of this problem. Historically, the formalization of the first haplotyping problems for an individual was given by Lancia *et al.* in [4] Within that work, the *minimum fragment removal* (MFR) and *minimum SNP removal* (MSR) problems were introduced, which we briefly discuss in the next section.

3. Minimum fragment and minimum SNP removal

Given the fact that at each SNP only two alleles are possible, the alleles can be encoded by a binary alphabet. Hence, in the sequel, the two values that a SNP can take will be denoted by `0` and `1`. Under this encoding, a haplotype is simply a string over the alphabet $\{0, 1\}$.

The basic framework for the single individual haplotyping problems is as follows. The data consists of a set $\mathcal{F} = \{f_1, \ldots, f_m\}$ of fragments (i.e., reads coming from sequencing) all taken from a target region containing a set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of SNPs. Each SNP is covered by some of the fragments, and can take the values `0` or `1`. Since there is a natural ordering of the SNPs, given by their physical location on the chromosome, the data can be represented by an $m \times n$ matrix M over the alphabet $\{\texttt{0}, \texttt{1}, \texttt{-}\}$, called *SNP matrix*. Each column of the matrix corresponds to a SNP and each row corresponds to a fragment. If fragment f_i covers the SNP s_j, then $M[i, j]$ is the value of the allele for SNP s_j appearing in fragment f_i. The symbol "`-`" is used to represent a SNP not covered by a fragment (see Figure 3 (a) and (b) for an example of a SNP matrix).

A *gapless* fragment is one covering a set of consecutive SNPs (i.e., the `0`'s and `1`'s appear consecutively in that row). We say that a fragment has k gaps if it covers $k+1$ blocks of consecutive SNPs (for example, the fragment `00--101---01-`has two gaps). Although theoretically each fragment should be gapless, since it covers a sequence of consecutive SNPs, in practice the data may contain gaps. In particular, the gaps can be mainly due to two reasons:

(1) Thresholding of low-quality reads. When the sequencer cannot call a SNP `0` or `1` with enough confidence, the call is not made, and the SNP is marked with a "`-`" . For instance, the sequencer could have read the first fragment of Figure 3 (a) as `acgaac`, but with a high degree of uncertainty on the `g`. Then the fragment would be turned into `ac-aac`.
(2) Mate-pairing in shotgun sequencing. Consider the following simplistic example. If only 4 letters were read at each end of `aggctaccgatggtg`, then the resulting fragment could be described as `aggc-------ggtg`. Hence, in the case of mate pairs, there would be fragments with $k = 1$ gap.

A pair of fragments f_i and f_j are said to be *in conflict* if there exists a SNP s_k such that $M[i, k] \in \{\texttt{0}, \texttt{1}\}$, $M[j, k] \in \{\texttt{0}, \texttt{1}\}$ and $M[i, k] \neq M[j, k]$. The conflict of two fragments implies that the fragments are not coming from the same chromosome copy or, otherwise, there would be errors in the data. Given a SNP matrix M, the *fragment conflict graph* is the graph $G_{\mathcal{F}}(M) = (\mathcal{F}, E_{\mathcal{F}})$ with an edge for each pair of fragments in conflict (see figure 3(c)). A (less intuitive) notion of conflict is defined also for the pair of SNPs. Two SNPs s_i and s_j are said to be *in conflict* if there exist two fragments f_u and f_v such that the 2×2 submatrix defined by rows u and v and columns i and j contains three `0`'s and one `1`, or three `1`'s and one `0`. Given a SNP matrix M, the *SNP conflict graph* is the graph $G_{\mathcal{S}}(M) = (\mathcal{S}, E_{\mathcal{S}})$, with an edge for each pair of SNPs in conflict (see figure 3(d)).

When $G_{\mathcal{F}}(M)$ is a bipartite graph, the set of fragments $\mathcal{F}$ can be segregated into two subsets H_1 and H_2 of pairwise compatible fragments. From each set one can infer one haplotype by fragment overlap. Let h_1 and h_2 be the haplotypes thus obtained. Since h_1 and h_2 are obtained by the assembly of the input fragments, the single individual haplotyping problems are sometimes also referred to as *fragment assembly haplotyping* problems.

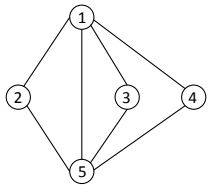
```
ac-aac
tt--t-
-tgatg
-tg-tg
t-aca-
```

(a) The reads

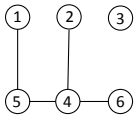
SNPs

fragments	1	2	3	4	5	6
1	0	1	-	0	0	1
2	1	0	-	-	1	-
3	-	0	1	0	1	0
4	-	0	1	-	1	0
5	1	-	0	1	0	-

(b) The SNP matrix M



(c) $G_{\mathcal{F}}(M)$



(d) $G_{\mathcal{S}}(M)$

FIGURE 3. SNP matrix and conflict graphs

Let us call a SNP matrix M *feasible* if $G_{\mathcal{F}}(M)$ is bipartite and *infeasible* otherwise. It is immediate to see that a SNP matrix for error-free data must be feasible. When the SNP matrix is not feasible the input data must contain errors that we want to detect and correct. All the single individual haplotyping problems proposed in the literature are therefore aimed at correcting an infeasible SNP matrix so as it becomes feasible.

The very first paper on single individual haplotyping appeared in 2001 and described two problems arisen at Celera Genomics in the context of sequencing the human genome [4] . These are optimization problems called *Minimum Fragment Removal* (MFR) and *Minimum SNP Removal* (MSR):

- **MFR:** Given a SNP matrix, remove the minimum number of fragments (rows) so that the resulting matrix is feasible.
- **MSR:** Given a SNP matrix, remove the minimum number of SNPs (columns) so that the resulting matrix is feasible.

The first problem is mainly suited for a situation in which, more than sequencing errors, one is worried about the presence of contaminants that should be removed from the input data. The second problem is more suited in the presence of sequencing errors only, i.e., when all the fragments are to be retained. Both objectives are driven by a general parsimony rule, in the same spirit as Occam's razor, i.e., look for the simplest explanation of why the matrix is not feasible.

Both MFR and MSR were shown to be NP-hard [4] so that exact algorithms for their solution are expected to be exponential branch-and-bound procedures. In [5], Lippert *et al.* described a combinatorial branch-and-bound algorithm for MFR. They also described an *Integer Linear Programming* (ILP) formulation of the problem, based on a reduction of MFR to the *Maximum Node-Induced Bipartite Subgraph* problem [6]. This problem requires to remove a smallest-size set of nodes from a graph G so as G becomes bipartite. In particular, at least one node must be removed from each odd cycle of G. The ILP model in [5] has an exponential number of constraints (one for each odd cycle), but it is shown that they can be separated in polynomial time. Good heuristic solutions for MFR were then obtained from the optimal solution of the LP relaxation by randomized rounding.

While the general case for MFR and MSR is NP-hard, both problems become tractable when the input fragments have no gaps. Let us call M a *gapless matrix* if each row of M is a gapless fragment. The main connection between MFR and MSR for gapless data is given by the following theorem.

Theorem 3.1. *[4] Let M be a gapless SNP matrix. Then, $G_{\mathcal{F}}(M)$ is a bipartite graph if and only if $G_{\mathcal{S}}(M)$ is a stable set.*

In [4] it was shown that both MSR and MFR are polynomial for a gapless SNP matrix. In particular, it was proved that, when M is gapless, $G_{\mathcal{S}}(M)$ is a perfect graph so that MSR boils down to finding the largest independent set in a perfect graph, i.e., a polynomial problem. Furthermore, MFR was shown to be polynomial by reducing it to a minimum cost flow problem on a suitably defined network. Note that all results for gapless data can be extended to matrices with gaps, as long as they possess the *consecutive ones property* (C1P), i.e., it is possible to sort the columns in such a way that each row becomes gapless. Since testing if a matrix M is C1P (and finding the corresponding sorting which makes M gapless) can be done in polynomial time [7], it follows that MFR and MSR are polynomial problems for matrices with C1P.

Later theoretical improvements extended these results from gapless fragments to gapped fragments in which the gaps length is bounded by a constant. In particular, there are $O(2^{2l}m^2n + 2^{3l}n^3)$ dynamic programming algorithms for MFR and $O(mn^{2l+2})$ for MSR for instances with gaps of total length l (Rizzi *et al.* [8], Bafna *et al.* [9]). These algorithms are hardly practical, however, on instances for which the gaps can be rather large. To overcome this problem, Xie and Wang [10] (see also Xie *el al.* [11]) proposed an algorithm for MFR which takes into account two new parameters: k, the maximum number of SNPs covered by any fragment and c, the maximum number of fragments covering any SNP site (also called *coverage*). Their solution has complexity $O(nc3^c + m \log m + mk)$. Since $k \leq n$ and c is generally at most 10, this method should be more suitable for mate-paired data, where l can be quite large.

4. Minimum letter flip

The very reason why single individual haplotyping problems exist is that no sequencing process can be error-free and the data can always contain errors, mostly due to miscalling or missing a base in a fragment. In order to model the correction of bases that were mistakenly read, a particular version of the single individual haplotyping problem was proposed in [5] (see also [12]) This version is called the *Minimum Letter Flip* (MLF) problem (also known as the *Minimum Error Correction* (MEC) problem), and is defined as follows:

> **MLF:** *Given a SNP matrix, swap the minimum number of entries (*`0` *into* `1`*, or* `1` *into* `0`*) so that the resulting matrix is feasible.*

In another version of MLF, each entry is associated to a non-negative weight, representing the confidence level with which the base corresponding to the entry had been called. The objective is to flip a set of entries with minimum total weight so as to make the matrix feasible. This problem is denoted by WMLF (Weighted Minimum Letter Flip).

The MLF problem was shown to be NP-hard, both for general [5] than for gapless matrices [13]. The literature contains several approaches for the solution of MLF. The reader is referred to Geraci [14] and Geraci and Pellegrini [15], for a comprehensive study in which most of these procedures are described and compared to each other.

One of the first heuristics for MLF was `FastHare`, by Panconesi and Suozo [16]. `FastHare` starts by first sorting the fragments according to their position of the chromosome (left to right), and then it reconstruct the two final haplotypes by correcting the sorted fragments in a greedy way. This heuristic is very fast and, despite its simplicity, it yields quite accurate solutions in general, especially when the error rate in the data in not too high. For high error-rate data, Genovese *et al.* proposed `SpeedHap` [17], an effective heuristic procedure organized in phases. For each phase the procedure performs three main steps: 1) it detects the likely positions of errors 2) it allocates the fragments to the two partially built final haplotypes, and 3) it decides the final alleles in the two haplotypes via a majority rule on ambiguous SNPs. Another similar greedy heuristic procedure is `FAHR` (Fast and Accurate Haplotype Reconstruction, by Wu and Liang [18]), which builds the final haplotypes by partitioning the fragments at each SNP in a greedy way. In computational experiments run over data coming from chromosome one of 60 individuals from the international HapMap Project [19], `FAHR` outperformed `FastHare` in both running time and correctness of the solution.

Bayzid et al. in [20] proposed still another heuristic for MLF, called `HMEC`. This is a steepest-descent local search procedure, which generally reaches good-quality solutions, but has the drawback of being somewhat slow. Specifically, each iteration of `HMEC` takes time $O(m^3n)$. Because of this time complexity, the use of `HMEC` may be impractical for instances with a large number of fragments.

The weighted version of MLF was considered by Zhao *et al.* in [21]. In this work, the authors proposed the use of a dynamic clustering heuristic, and introduced an

extended version of the problem defined in order to deal with the presence of contaminants. The new problem, denoted by CWMLF (*Complete Weighted Minimum Letter Flip*), requires to make the SNP matrix feasible not only by flipping some of its entries but also by possibly removing some of the rows. In [22], Wu *et al.* proposed a heuristic procedure for WMLF called `AHHAP`. The algorithm starts with a preprocessing phase aimed at reducing the size of the instance, and then it proceeds by a greedy assignment of the fragments to the haplotypes, according to a measure of similarity between the fragments and the partial haplotypes built so far. Computational experiments show that `AHHAP` performs in a similar way as the best previous methods.

Among the various types of approaches employed for the solution of MLF, there is the use of evolutionary algorithms such as *Genetic Algorithms* and *Particle Swarm Optimization* (PSO). In [23], Wang *et al.* proposed both a combinatorial branch-and-bound algorithm and a genetic algorithm for MLF. In the genetic algorithm, each candidate solution is represented by a binary vector v of length m, in which v_i specifies the haplotype copy (0 or 1) to which fragment f_i is assigned. Being exponential, the branch-and-bound is only applicable to instances of very small size, but the genetic algorithm can be used for large instances (e.g., more than 50 fragments over more than 50 SNPs).

A PSO heuristic for MLF was proposed by Kargar *et al.* [24]. The PSO is an evolutionary metaheuristic in which the flow of the algorithm is already defined and problem-independent, while the user must only specifying the solution format and the fitness function (i.e., the objective function) for his specific problem. In particular, similarly to the above genetic algorithm, a candidate solution is represented by a binary vector of length m, whose components specify the haplotype copy to which each fragment is assigned. Computational experiments show that the PSO turns out to be fast and appropriate for instances with a low error-rate in the data.

Among the most successful heuristics for MLF we recall `HapCUT`, proposed by Bansal and Bafna [25]. This algorithm makes use of an auxiliary graph defined from the fragments, where each SNP corresponds to a node in the graph and two nodes (i.e., two SNPs) are joined by an edge if there exists a fragment that covers both SNPs. In order to minimize the MLF cost of the reconstructed haplotypes, the algorithm proceeds by iteratively finding max-cuts in a sequence of auxiliary graphs. Computational experiments run on real-life data showed that haplotypes inferred by `HapCUT` were significantly more accurate (20-25% lower maximum error correction scores for all chromosomes) than those obtained by the previously published methods.

In [26] , He *et al.* proposed a dynamic programming algorithm for MLF with time complexity $O(2^L\, m\, n)$ where L is maximum length of a fragment. The algorithm can be used for values of L up to about 15, but it becomes impractical for larger values. When the input fragments are too long for the dynamic programming procedure, the authors proceed by modeling the problem as a satisfiability problem, and solve it via a MaxSAT solver. The quality of the solutions obtained

through this approach is shown to be quite high, but the solving process remains slow in many cases.

Another dynamic programming algorithm was proposed by Deng *et al.* in [27] and is parametrized by the maximum coverage c of each SNP (where the coverage of a SNP is defined to be the number of fragments to which that SNP belongs). The complexity of the dynamic program is $O(nc2^c)$, which can be quite high when c is large. For large values of c the authors propose a heuristic procedure derived from the dynamic programming algorithm. Experiments showed that the heuristic returns very accurate solutions on average.

Chen *et al.* [28] proposed an algorithm for MLF based on an ILP formulation of the problem. Since computing the ILP solution for the whole input SNP matrix would require an enormous amount of time, the algorithm starts by decomposing the matrix into small, independent, blocks. For each individual block, the ILP model is then computed and solved by the ILP solver. The program was tested on a popular benchmark, i.e., the filtered HuRef data set [29], and it took a total time of 31h (26h of which were spent on the most difficult block of the 15th chromosome). This was the first time that MLF optimal solutions were obtained for the filtered HuRef dataset.

5. Probabilistic models

Some models of single haplotyping problems are of probabilistic nature, i.e., loosely speaking, they seek a solution which has the greatest probability of being the correct one.

In [30], Li *et al.* proposed a probabilistic model for the haplotype inference problem. Their method is based on a statistical model of sequencing errors, compositional information, and haplotype memberships. In their model, they studied the conditional probability $P(h_1, h_2|M)$ of h_1 and h_2 being the correct haplotypes given that M was the SNP matrix measured. They pursued the objective of determining the most likely pair of correct haplotypes, i.e. a pair $\{h_1^*, h_2^*\}$ maximizing the above conditional probability. Due to computational complexity, the method proceeds in steps. The first step considers only consecutive SNPs and decides, for each pair of consecutive SNPs, which allele should be assigned to the first haplotype and which to the second. In a second step these assignments of alleles are combined in order to reconstruct the actual haplotypes. The solving procedure is based on Gibbs sampling and an *Expectation Maximization* (EM) algorithm. The procedure is also able to compute the accuracy, or confidence, of the reconstructed haplotypes.

Wang *et al.* in [31] formulated the haplotype assembly problem into a statistical framework and modeled it by a Markov Chain (MC) model which can account for various types of data errors. This Markov Chain approach presents several main advantages: (i) it is suitable for those data sets on which we have no previous information about the type of errors that we might expect in the data; (ii) the model can be solved by a polynomial time algorithm, and therefore it is particularly

effective for dealing with large data sets; (iii) haplotypes have very a strong linkage disequilibrium property [32], and the MC model is suitable for characterizing this kind of linkage property. The objective pursued by Wang *et al.* is the same as in Li *et al.*, i.e., determining the most probable pair of haplotypes given the observed SNP matrix. In the MC approach, the assignment of fragments to haplotypes is regarded as a Markov process in which successive allele pairs are generated based on the value of a small number of the preceding SNP sites in the sequence. In order to find the most probable haplotypes for the set of fragments, the authors proposed a Viterbi-like dynamic programming procedure. Computational tests run on HapMap data showed that the MC approach retrieves solutions with a higher probability of being correct than those obtained by the MLF procedures, and in a much shorter running time.

Probabilistic arguments were not only used in the definition of new models for haplotyping, but also in deriving probabilistic algorithms for the existing objective functions. For example, `HASH` (*Haplotype Assembly for Single Human*), is a probabilistic algorithm proposed by Bansal *et al.* [33] for assembling haplotype fragments under the MLF objective function. `HASH` is a *Markov Chain Monte Carlo* (MCMC) algorithm in which the transitions of the Markov chain are generated using min-cut computations on auxiliary graphs derived from the input reads. The method was applied to infer the haplotypes from real data consisting of whole-genome shotgun sequence fragments of a human individual. The results showed that the haplotypes inferred by using `HASH` were significantly more accurate than the haplotypes estimated by using the best heuristics available at the time.

Chen *et al.* [34] described a probabilistic algorithm for MLF in which they considered three error parameters, called α_1, α_2 and β. The parameter α_1 is the error-rate with which entries of the SNP matrix have been miscalled (i.e., a `0` in place of a `1`, or a `1` in place of a `0`). The parameter α_2 is the error-rate with which a "`-`" in the SNP matrix appears where, in fact, a SNP is covered by a fragment and therefore an allele `0` or `1` should be present. Finally, β is a measure of the expected dissimilarity between the haplotypes to be reconstructed. The authors gave a linear-time (in the size of M) probabilistic algorithm that reconstructs the correct haplotypes with high probability when all the parameters are known. Even for the case when some of the parameters are unknown, they provided a probabilistic algorithm that outputs the correct haplotypes with probability depending on the actual values of α_1, α_2 and β.

6. Generalizations and variants of haplotyping

Among the works proposed for the solution of the MLF model, there are some papers which introduced variants to the original framework in order to deal with some specific data problems. For instance, Zhang *et al.* [35] proposed a model called *Minimum Conflict Individual Haplotyping* (MCIH), suitable for data sets with particularly high error-rate. The problem was shown to be NP-hard. For

the solution of MCIH, the authors described a dynamic programming procedure, suitable for moderate-sized data sets, and a feed-forward neural network algorithm for larger instances.

Duitama *et al.* [36] proposed a model called *Maximum Fragments Cut* (MCF) whose objective is to identify a set of fragments (i.e., rows of the SNP matrix) maximizing a score proportional to their conflict with respect to the remaining rows. This set of fragments can be interpreted as the shore of a cut in a suitable graph G, so that the model requires the solution of a Max-Cut problem. In order to solve MCF, the authors utilized a local optimization heuristic, called `ReFHap`. The procedure starts by finding a good cut C in G via a classical local-search heuristic for max-cut. Then, after having found the cut, the algorithm uses the input SNP matrix to find the haplotype h that minimizes the number of entries to be corrected assuming that the rows in C belong to h and the other rows do not belong to h. Computational experiments were run showing the effectiveness of the heuristic procedure.

Xie *et al.* introduced in [37] a model called *Balanced Optimal Partition* (BOP), which generalizes both MLF and MCF. The model is, in a way, a weighted combination of MLF and MCF, and by setting some model parameters to proper extreme values, BOP degenerates into pure MLF or pure MCF respectively. To solve the model, the authors proposed a dynamic programming algorithm called `H-BOP`, which was made effective by limiting the number of intermediate solutions to a suitable small integer k. Extensive computational experiments showed that, for $k = 8$, `H-BOP` was generally faster and more effective than state-of-the-art procedures of the time, such as `ReFHap`. Furthermore, the experiments showed that `H-BOP` scales rather well to large-size input data.

Aguiar and Istrail proposed in [38] `HapCompass`, an algorithm for haplotype assembly of densely sequenced human genome data. `HapCompass` operates on the *compass graph*, i.e., a graph where each SNP is a node, and the edges are associated to the fragments. The edges are weighted, and the weight of an edge (s_i, s_j) measures the best coupling of the alleles for s_i and s_j on the two final haplotypes. In the model, the reconstruction of the final haplotypes corresponds to a minimum-weight edge removal problem until a special type of subgraph is obtained. `HapCompass` is a heuristic with a local reoptimization step. Computational results showed the effectiveness of the procedure and the good quality of its solution also with respect to the MLF objective. Later on, Aguiar and Istrail described in [39] some generalizations of compass graphs and of the HapCompass framework. They presented graph theory-based algorithms for (i) novel implementations of haplotype assembly optimizations (minimum error correction), (ii) assembly of a pair of individuals sharing a haplotype tract identical by descent and (iii) assembly of polyploid genomes (i.e., genomes having more than two sets of homologous chromosomes). The method were evaluated on the 1000 Genomes Project, Pacific Biosciences and simulated sequence data.

References

[1] D. Gusfield and S. H. Orzack, "Haplotype inference," in *Handbook of Computational Molecular Biology*, pp. 1–28, Champman and Hall/CRC-press, 2005.

[2] J. Venter *et al.*, "The sequence of the human genome," *Science*, vol. 291, pp. 1304–1351, 2001.

[3] F. S. Collins, M. Morgan, and A. Patrinos, "The human genome project: Lessons from large-scale biology," *Science*, vol. 300, no. 5617, pp. 286–290, 2003.

[4] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz, "SNPs problems, complexity and algorithms," in *Proceedings of the Annual European Symposium on Algorithms (ESA)*, vol. 2161 of *Lecture Notes in Computer Science*, pp. 182–193, Springer, 2001.

[5] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail, "Algorithmic strategies for the SNPs haplotype assembly problem," *Briefings in Bioinformatics*, vol. 3, no. 1, pp. 23–31, 2002.

[6] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] K. Booth and G. Lueker, "Testing for the consecutive ones property, interval graphs and graph planarity using pq-tree algorithms," *Journal of Computer System Science*, vol. 13, no. 3, pp. 335–379, 1976.

[8] R. Rizzi, V. Bafna, S. Istrail, and G. Lancia, "Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem," in *Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI)* (R. Guigo and D. Gusfield, eds.), vol. 2452 of *Lecture Notes in Computer Science*, pp. 29–43, Springer, 2002.

[9] V. Bafna, S. Istrail, G. Lancia, and R. Rizzi, "Polynomial and APX-hard cases of the individual haplotyping problem," *Theoretical Computer Science*, vol. 335, pp. 109–125, 2005.

[10] M. Xie and J. Wang, "An improved (and practical) parametrized algorithm for the individual haplotyping problem MFR with mate pairs," *Algorithmica*, vol. 52, pp. 250–266, 2008.

[11] M. Xie, J. Wang, and J. Chen, "A model of higher accuracy for the individual haplotyping problem based on weighted SNP fragments and genotype with errors," *Bioinformatics*, vol. 24, no. 13, pp. i105–i113, 2008.

[12] H. Greenberg, W. Hart, and G. Lancia, "Opportunities for combinatorial optimization in computational biology," *INFORMS Journal on Computing*, vol. 16, no. 3, pp. 1–22, 2004.

[13] R. Cilibrasi, L. V. Iersel, S. Kelk, and J. Tromp, "On the complexity of several haplotyping problems," in *Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI)*, vol. 3692 of *Lecture Notes in Computer Science*, pp. 128–139, Springer, 2005.

[14] F. Geraci, "A comparison of several algorithms for the single individual SNP haplotyping reconstruction problem," *Bioinformatics*, vol. 26, no. 18, pp. 2217–2225, 2010.

[15] F. Geraci and M. Pellegrini, "Rehap: an integrated system for the haplotype assembly problem from shotgun sequencing data," in *BIOINFORMATICS 2010 - Proceedings of the First International Conference on Bioinformatics* (A. L. N. Fred, J. Filipe, and H. Gamboa, eds.), pp. 15–25, INSTICC Press, 2010.

[16] A. Panconesi and M. Sozio, "Fast hare: A fast heuristic for single individual SNP haplotype reconstruction," in *Proceedings of Annual Workshop on Algorithms in Bioinformatics (WABI)*, vol. 3240 of *Algorithms in Bioinformatics*, pp. 266–277, Springer, 2004.

[17] L. Genovese, F. Geraci, and M. Pellegrini, "Speedhap: an accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate and low coverage," *IEEE/ACM Trans Comput Biol Bioinform*, vol. 5, no. 4, pp. 492–502, 2008.

[18] S. Bayzid, M. Alam, A. Mueen, and S. Rahman, "A fast and accurate algorithm for diploid individual haplotype reconstruction," *J. Bioinform. Comput. Biol.*, vol. 11, pp. 1–12, 2013.

[19] I. H. Consortium, "The international hapmap project," *Nature*, vol. 426, no. 6968, pp. 789–796, 2003.

[20] S. Bayzid, M. Alam, A. Mueen, and S. Rahman, "Hmec: A heuristic algorithm for individual haplotyping with minimum error correction," *ISRN Bioinformatics*, vol. 2013, no. Article ID.291741, pp. 1–10, 2013.

[21] Y. Zhao, L. Wu, J. Zhang, R. Wang, and X. Zhang, "Haplotype assembly from aligned weighted SNP fragments," *Comput Biol Chem*, vol. 29, no. 4, pp. 281–287, 2005.

[22] J. Wu, J. Wang, and J. Chen, "A heuristic algorithm for haplotype reconstruction from aligned weighted SNP fragments.," *Int J Bioinform Res Appl.*, vol. 9, no. 1, pp. 13–24, 2013.

[23] R. Wang, L. Wu, Z. Li, and X. Zhang, "Haplotype reconstruction from SNP fragments by minimum error correction," *Bioinformatics*, vol. 21, no. 10, pp. 2456–2462, 2005.

[24] M. Kargar, H. Poormohammadi, L.Pirhaji, M. Sadeghi, H. Pezeshk, and C. Eslahchi, "Enhanced evolutionary and heuristic algorithms for haplotype reconstruction problem using minimum error correction model," *MATCH Commun. Math. Comput. Chem.*, vol. 62, pp. 261–274, 2009.

[25] V. Bansal and V. Bafna, "HapCUT: an efficient and accurate algorithm for the haplotype assembly problem," *Bioinformatics*, vol. 24, pp. i153–i159, 2008.

[26] D. He, A. Choi, K. Pipatsrisawat, A. Darwiche, and E. Eskin, "Optimal algorithms for haplotype assembly from whole-genome sequence data," *Bioinformatics*, vol. 26, no. 12, pp. i83–i190, 2010.

[27] F. Deng, W. Cui, and L. Wang, "A highy accurate heuristic algorithm for the haplotype assembly problem," *BMC Genomics*, vol. 14, pp. 1–10, 2013.
[28] Z. Chen, F. Deng, and L. Wang, "Exact algorithms for haplotype assembly from whole-genome sequence data," *Bioinformatics*, vol. 29, no. 16, pp. 1938–1945, 2013.
[29] S. Levy, G. Sutton, P. Ng, L. Feuk, A. Halpern, B. Walenz, N. Axelrod, J. Huang, E. Kirkness, G. Denisov, Y. Lin, J. MacDonald, A. Pang, M. Shago, T. Stockwell, A. Tsiamouri, V. Bafna, V. Bansal, S. Kravitz, D. Busam, K. Beeson, T. McIntosh, K. Remington, J. Abril, J. Gill, J. Borman, Y.-H. Rogers, M. Frazier, S. Scherer, R. Strausber, and C. Venter, "The diploid genome sequence of an individual human," *PLoS Biol*, vol. 5, no. 10, p. e254, 2007.
[30] L. Li, J. Kim, and M. Waterman, "Haplotype reconstruction from SNP alignment," *Journal of Computationa Biology*, vol. 11, pp. 507–518, 2004.
[31] R. Wang, L. Wu, X. Zhang, and L. Chen, "A markov chain model for haplotype assembly from SNP fragments," *Genome Inform*, vol. 17, no. 2, pp. 162–171, 2006.
[32] J. Douglas, M. Boehnke, E. Gillanders, J. Trent, and S. Gruber, "Experimentally-derived haplotypes substantially increase the efficiency of linkage disequilibrium studies," *Nature Genetics*, vol. 28, no. 4, pp. 361–364, 2001.
[33] V. Bansal, A. Halpern, N. Axelrod, and V. Bafna, "An MCMC algorithm for haplotype assembly from whole-genome sequence data," *Genome Res.*, vol. 18, no. 8, pp. 1336–1346, 2008.
[34] Z. Chen, B. Fu, R. Schweller, B. Yang, Z. Zhao, and B. Zhu, "Linear time probabilistic algorithms for the singular haplotype reconstruction problem from SNP fragments," *Journal of Computationa Biology*, vol. 15, no. 5, pp. 535–546, 2008.
[35] X. Zhang, R. Wang, A. Wu, and W. Zhang, "Minimum conflict individual haplotyping from SNP fragments and related genotype," *Evolutionary Bioinformatics Online*, vol. 2, pp. 271–280, 2006.
[36] J. Duitama, T. Huebsch, G. McEwen, E. Suk, and M. Hoehe, "Refhap: a reliable and fast algorithm for single individual haplotyping," in *Proceedings of the 1st ACM International conference on Bioinformatics and Computational Biology*, DMTCS'03, (New York, NY), pp. 160–169, ACM, 2010.
[37] M. Xie, J. Wang, and T. Jiang, "A fast and accurate algorithm for single individual haplotyping," *BMC Systems Biology*, vol. 6, no. (Suppl 2), pp. 1–10, 2012.
[38] D. Aguiar and S. Istrail, "HapCompass: A fast cycle basis algorithm for accurate haplotype assembly of sequence data," *J Comput Biol.*, vol. 19, no. 6, pp. 577–590, 2012.
[39] D. Aguiar and S. Istrail, "Haplotype assembly in polyploid genomes and identical by descent shared tracts," *Bioinformatics*, vol. 29, no. 13, pp. 352–360, 2013.