

Haplotype Inference via Hierarchical Genotype Parsing

Pasi Rastas & Esko Ukkonen *

Department of Computer Science and
Helsinki Institute for Information Technology HIIT
P.O Box 68, FIN-00014 University of Helsinki, Finland
`Firstname.Lastname@cs.helsinki.fi`

Abstract. The within-species genetic variation due to recombinations leads to a mosaic-like structure of DNA. This structure can be modeled, e.g. by parsing sample sequences of current DNA with respect to a small number of founders. The founders represent the ancestral sequence material from which the sample was created in a sequence of recombination steps. This scenario has recently been successfully applied on developing probabilistic Hidden Markov Methods for haplotyping genotypic data. In this paper we introduce a combinatorial method for haplotyping that is based on a similar parsing idea. We formulate a polynomial-time parsing algorithm that finds minimum cross-over parse in a simplified ‘flat’ parsing model that ignores the historical hierarchy of recombinations. The problem of constructing optimal founders that would give minimum possible parse for given genotypic sequences is shown NP-hard. A heuristic locally-optimal algorithm is given for founder construction. Combined with flat parsing this already gives quite good haplotyping results. Improved haplotyping is obtained by using a hierarchical parsing that properly models the natural recombination process. For finding short hierarchical parses a greedy polynomial-time algorithm is given. Empirical haplotyping results on HapMap data are reported.

1 Introduction

Recombination is a major factor causing genetic variation between individuals of a population by combining mutations. In this paper we generalize and improve the combinatorial founder model [19] of recombinations. This model assumes that the current population is evolved from a small number of ‘founder’ individuals, thus the current sequences are recombinations of these founder sequences. If visualized by giving each founder sequence a different color, the founders define a coloring of the current sequences, thus uncovering a mosaic-like structure. Figure 1 shows an example mosaic obtained by parsing 20 sequences (‘Recombinants’) with respect to four founder sequences. The term *mosaic* is also used in this sense in [13, 20]. A key-question with the founder model is how to find

* Supported by the Academy of Finland under grant 211496 (From Data to Knowledge).

good founders. A natural parsimony criterion was used in [19]: find ancestral sequences that explain the given data with fewest recombinations. This model is also studied in [22].

Our contribution is three-fold. First we generalize the model from haplotypes to genotypes, and give a parsing algorithm that parses given genotype sequences into fragments that are taken from the haplotypic founders. This immediately suggests a phasing for the genotypes as the parse is composed of two haplotype sequences, i.e. we have a haplotyping algorithm.¹ This type of parsing is flat in the sense that it ignores the historical order of recombinations. Then we provide some computational complexity results on finding a founder set for which the flat parse has smallest possible number of recombinations: Finding founders that are optimal in this sense is NP-hard in general but polynomial-time if the number of founders is restricted to two. As finding optimal founders is hard, we develop a locally-optimal method for finding a good founder set.

Finally we improve the parsing model to include the hierarchical structure of recombinations. This leads to a novel parsing problem of finding a shortest hierarchical parse with respect to given founders. We propose a greedy polynomial-time parsing algorithm. The paper is concluded by reporting some haplotyping experiments on the HapMap data. It turns out that already the flat parsing with respect to the locally optimal set of founders gives reasonably good haplotyping results as compared to the state-of-the-art probabilistic methods. The performance improves if the hierarchical parsing is applied.

Our hierarchical parsing can be seen as a method for constructing a variant of the so-called ancestral recombination graph (ARG) that connects the founders to the given genotypes. An ARG is the most accurate model of the genealogy of sequences [3]. It is a directed graph, whose nodes correspond to sequences, single edges correspond to mutations, and edges connecting nodes a and b to a single node stand for a recombination between a and b . All sequences in the ARG model evolve from a single root sequence. A natural parsimonious problem is to find an ARG with minimum number of recombinations assuming each allele mutates only once. The problem is known to be NP-hard in the case of a known root sequence [21]. Current state-of-the-art methods can produce ARGs for about 20 SNPs and 40 haploid sequences [12].

2 Genotypes, Haplotypes, and Recombination

A Single Nucleotide Polymorphism (SNP) is a single base-pair position in genomic DNA where different nucleotides, called alleles, occur in some population. In most SNPs only two alleles out of A, C, G and T are present. Most of the genetic variance between individuals is due to SNPs. Thus, when studying genetic diseases or factors, one often studies variations in certain SNPs.

In humans and other diploid organisms, most cells contain two almost identical copies of each chromosome, one inherited from the organism's mother and the

¹ Essentially the same observation was independently made in [9].

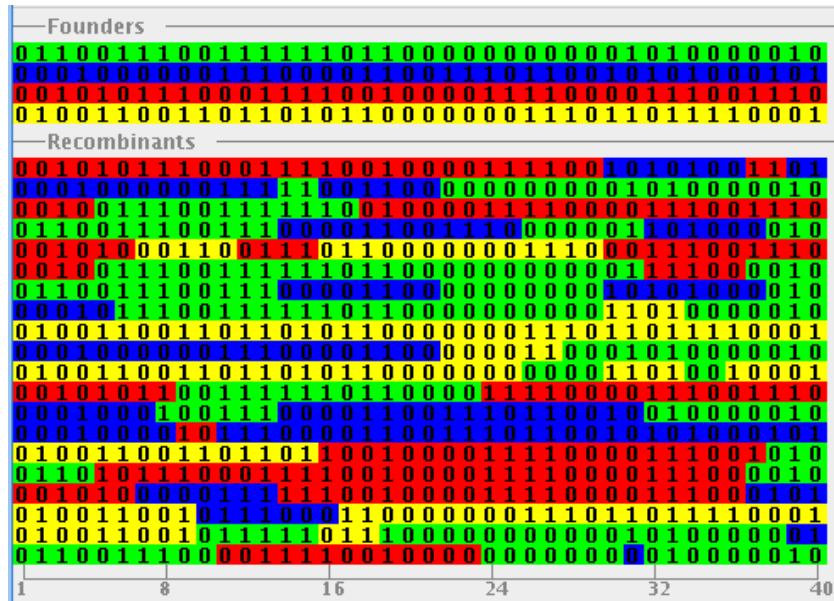


Fig. 1. Screenshot from HaploVisual program that implements the parsing algorithms of [19]. The program is available on www.cs.helsinki.fi/u/prastas/haplovisual/.

other from the father. These copies are called *haplotypes*. Thus, each individual has two haplotypes: maternal and paternal.

Current practical laboratory methods determine for each SNP of an individual only the alleles, but no information on which copy they are from. In this context, these sequences without copy information are called *genotypes*. Thus, a genotype is a sequence that gives at each SNP the alleles of the two haplotypes. For example, consider the case that alleles at three SNPs are A-A, C-T and A-T. The possible haplotypes are therefore ACA and ATT, or ATA and ACT. If the alleles at an SNP are different we call this site heterozygous (the second and the third SNP in the example), and otherwise homozygous (the first SNP). If a genotype has k heterozygous sites then there are 2^{k-1} possible pairs of haplotypes for that genotype. Without assumptions about the haplotypes or about the population, all of these possibilities are equally likely.

For measuring haplotyping performance we use a commonly used metric called the *switch distance* [11]. We use the unnormalized version of this distance, i.e. the number of switches. It equals the number of phase changes in the inferred haplotypes, that are needed to get the correct haplotypes. For example, assume that the correct haplotypes are AAAAA and TTTTT. Then a phasing solution ATTTT, TAAAA would score one switch, and a solution ATATA, TATAT four switches.

Figure 2 shows how recombination in a meiosis combines maternal and paternal sequences. Child's haplotype inherited from one parent contains fragments

from both haplotypes of that parent. An underlying assumption in this paper is that recombinations happen in an equal crossing over fashion, i.e. in such way that sequence fragments retain their locations in the resulting sequence [19].

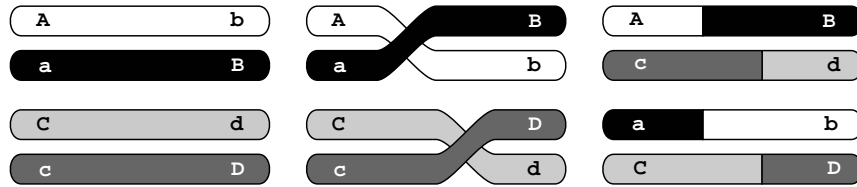


Fig. 2. On the left hand side there are the haplotypes of the parents and on the right hand side are the outcomes for the child's haplotypes from the two recombinations illustrated in the middle.

3 Combinatorial Mosaic Model

In this section the founder models presented in [19, 17, 8, 6] are generalized for genotypes and point mutations.

We assume that our sequences of interest are from a finite alphabet Σ , for example $\Sigma = \{A, C, G, T\}$ for DNA or $\Sigma = \{0, 1\}$ for SNP haplotypes. If our sequences are from n markers, then a sequence can be described as a string of length n from alphabet $\Sigma \cup \{-\}$, where symbol $-$ is used to indicate missing values.

We want to analyze a set D of current sequences with respect to some small set of founder sequences F . The sequences in D and F are haplotypes over the same n markers. We denote $|D| = m$ and $|F| = K$. The differences between D and F are due to point mutations and recombinations. Recombination is modeled as a process, that builds a new haploid sequence by combining a prefix of one sequence with a suffix of another one [8]. Mutation is simply an event that changes one symbol of a certain sequence.

A haplotype $h \in D$ has a *simple parse* (with no mutations) of cost k with respect to founder sequences F , if we can write h as concatenation $f_{i_0} \cdot f_{i_1} \cdots f_{i_k}$ of nonempty substrings f_{i_j} , where each f_{i_j} occurs in some $f \in F$ at the same position as it is used in the parse. Let c be a parameter that gives a relative weight for mutations as compared to recombinations. Then a haplotype h has a *parse* of cost $k + k'c$ with respect to F , if there is a simple parse of cost k of some h' and the Hamming distance between h and h' , $d(h, h')$, is k' . We say that a parse is optimal, if it has the lowest possible cost.

The score that we want to minimize is the sum of optimal parse costs for the sequences in D . This score can be computed for each sequence independently, and it depends on the number of recombination and mutation events. Each recombination adds amount of 1 to the score and each point mutation adds

amount of $c > 0$. By setting a high value for c , the parse is forced to use mutations only rarely, and by setting c to a small positive value, the parse is forced to use recombinations rarely.

Let $F = (F_1, \dots, F_K) \subset \Sigma^n$, $|F| = K$, be a fixed founder set where each $F_a = F_{a1} \dots F_{an}$. Then we can compute the minimum cost of a parse of a haplotype $h = h_1 \dots h_n$ with respect to F by dynamic programming using the following formulas:

$$\begin{cases} S(0, a) = 0 \\ S(i, a) = p_c(h_i, F_{ai}) + \min_{a'} (S(i-1, a') + I_{a' \neq a}) \end{cases} \quad (1)$$

for $a = 1, \dots, K$, and $i = 1, \dots, n$. Here I_A is the indicator function of predicate A , and $p_c(S, S')$ is the cost of mutating symbol S to symbol S' , i.e.,

$$p_c(S, S') = \begin{cases} 0 & , \text{ if } S = S' \text{ or } S = - \\ c & , \text{ otherwise.} \end{cases}$$

The minimum score, $score_F^c(h)$, is $\min_a S(n, a)$. The corresponding parse can be found using standard trace-back after each $S(i, a)$ has been computed. Direct evaluation of formula (1) would take space $O(nK)$ and time $O(nK^2)$. A more efficient evaluation is possible, however, as follows. By writing the minimization in the latter equation $S(i, a) = p_c(h_i, F_{ai}) + \min_{a'} (S(i-1, a') + I_{a' \neq a})$ as $\min (S(i-1, a), 1 + \min_{a'} S(i-1, a'))$, we can see that by maintaining the value $\min_{a'} S(i-1, a')$ during the computation one can reduce the running time to $O(nK)$. Moreover, the space requirement can be reduced trivially to $O(K)$, if we do not need the corresponding parse. However, space $O(K)$ and time $O(nK)$ is also enough to get the parse. This achieved by using a divide and conquer algorithm similar to the Hirshberg's algorithm [5].

Theorem 1. *The optimal parse and parsing score of a haplotype with respect to a given founder set F can be found in time $O(n|F|)$ and in space $O(|F|)$.*

The score of a set of m haplotypes H is defined as a sum of individual scores $h \in H$, i.e. $score_F^c(H) = \sum_{h \in H} score_F^c(h)$. The running time to compute the score for a set H of m haplotypes is $O(mn|F|)$.

Let us next consider unphased sequences, i.e. our dataset D consists of genotypes instead of haplotypes. We show that with some modifications to (1) we can parse genotype sequences as well. A similar algorithm appears in [9] while our formulation is from [14].

For reasons of clarity, we assume that our alphabet Σ is $\{0, 1\}$. All results apply for general alphabet too, but the notation would get too complicated. A *genotype* g is a string in alphabet $\{0, 1, 2, -\}$, where 0 and 1 denote the two homozygote alleles and value 2 denotes heterozygous alleles. Two haplotypes $h = h_1 \dots h_n \in \{0, 1\}^n$ and $h' = h'_1 \dots h'_n \in \{0, 1\}^n$ are *compatible* with a genotype

$g = g_1 \dots g_n \in \{0, 1, 2, -\}^n$, denoted $g = \gamma(h, h')$, if either $(g_i = h_i = h'_i)$ or $(g_i = 2 \text{ and } h_i \neq h'_i)$ or $(g_i = -)$ for each $i, 1 \leq i \leq n$.

Given a founder set F and a genotype g , we define the score of g , $score_F^c(g)$, as $\min_{h, h': \gamma(h, h')=g} (score_F^c(h) + score_F^c(h'))$. Score of a set of genotypes G is defined, as with haplotypes, as $\sum_{g \in G} score_F^c(g)$. Score of g can be computed efficiently using dynamic programming as follows:

$$\begin{cases} S(0, a, b) = 0 \\ S(i, a, b) = p_c(g_i, F_{ai}, F_{bi}) + \min_{a', b'} (S(i-1, a', b') + I_{a' \neq a} + I_{b' \neq b}) \end{cases} \quad (2)$$

for $a, b = 1, \dots, K$, and $i = 1, \dots, n$. Here $p_c(T, S, S')$ is the cost of mutating genotype T to $\gamma(S, S')$, i.e.

$$p_c(T, S, S') = \begin{cases} 0 & , \text{ if } T = \gamma(S, S') \text{ or } T = - \\ 2c & , \text{ if } T \neq \gamma(S, S') \text{ and } T \neq \gamma(S'', S') \text{ for all } S'' \in \Sigma \\ c & , \text{ otherwise.} \end{cases}$$

Minimum score, $score_F^c(g)$, is $\min_{a, b} S(n, a, b)$ and the parse can be uncovered by trace-back.

Direct evaluation of (2) would take $O(nK^4)$ time. Using similar trick as earlier we can write the minimization as $\min(C_{00}, C_{01}, C_{10}, C_{11})$, where $C_{00} = S(i-1, a, b)$, $C_{01} = 1 + \min_{a'} S(i-1, a', b)$, $C_{10} = 1 + \min_{b'} S(i-1, a, b')$ and $C_{11} = 2 + \min_{a', b'} S(i-1, a', b')$. Now we can maintain values $\min_{a'} S(i-1, a', b)$ and $\min_{b'} S(i-1, a, b')$ by using two arrays (actually one is enough) of size $O(K)$ (indexes a and b). These arrays can be computed in time $O(K^2)$ for column $i-1$. Further on, we need to keep track of a single value $\min_{a', b'} S(i-1, a', b')$. By computing $S(i, a, b)$ in this way we get time complexity of $O(nK^2)$. The space complexity can be reduced to $O(K^2)$ similarly as in the case of haplotypes, to $O(K)$.

The parse of a genotype also fixes its haplotypes, i.e. we can use this parse to infer haplotypes based on a founder set F .

Theorem 2. *The optimal parse and parsing score of a genotype with respect to a given founder set F can be found in time $O(n|F|^2)$ and space $O(|F|^2)$. The parse suggests a phasing of the genotype by giving two haplotypes that are compatible with the genotype.*

4 Hardness of Finding Founders

In this section we consider the complexity of the problem of finding a set F of K founder sequences that minimizes $score_F^c(D)$. The decision version of the problem can be proven to be NP-complete.

Problem 1. Given haplotype or genotype data D , parameters K, c and T , is there a set F of K founder sequences such that the score of data D , $score_F^c(D)$, is at most T ?

Theorem 3. *Problem 1 is NP-complete when the data consists of haplotypes, i.e., $D \subset \{0, 1, -\}^n$.*

Proof. Problem is in NP, because if we are given a founder set F we can check if it gives score $\leq T$ using the polynomial algorithm derived from (1).

Problem is NP-hard because we can reduce the graph coloring problem to it. The graph coloring problem asks one to color the vertices of a graph with K colors such that there are no edges between vertices with the same color. The problem is NP-hard even for $K = 3$ [2]. Let $G = (V, E)$ be a graph with $n = |V|$ vertices. Let $H = \{h_1, \dots, h_n\}$ be the corresponding set of haplotypes represented as an $n \times n$ matrix (H_{ij}) where $H_{ij} = h_{ij}$. Graph G is coded into H by setting

$$H_{ij} = \begin{cases} 1 & , \text{ if } i = j \\ 0 & , \text{ if } (i, j) \in E \\ - & , \text{ otherwise.} \end{cases}$$

In this coding, the vertex i of the graph G corresponds to the haplotype h_i .

Now there is a graph coloring of G with K colors if and only if there is founder set of size K giving for H total parsing score 0.

“if”: Let a founder set F give score 0, $|F| = K$. Then we can construct a parse where each $h \in H$ is parsed using exactly one $f \in F$. Now let $V_f \subset V$ be the set of all vertices i corresponding haplotypes $h_i \in H$ that are parsed using f . Since each h_i has 1 at position i and no other haplotype have one at that position, all haplotypes corresponding to vertices $V_f \setminus \{i\}$ must have symbol $-$ at position $i \in V_f$. Thus, there cannot be any edges between $i \in V_f$ and V_f and therefore we have a coloring of G with K colors by coloring each V_f using the same, unique color.

“only if”: Assume that G has a valid K coloring. Let us define V_j as the vertices that have been colored using the color j . Now we can construct a founder set F as a set of haplotypes f_j with ones at positions V_j and zeros otherwise. Because vertices V_j can be colored using same colors there cannot be any edges between them. So the corresponding haplotypes will have only values 1 and $-$ at positions $i \in V_j$ and therefore we can parse all haplotypes H with score 0. In such a parse, haplotype $h_i \in V_j$ could be parsed using founder f_j . \square

Theorem 4. *Problem 1 is NP-complete when the data consists of genotypes, i.e., $D \subset \{0, 1, 2\}^n$.*

Proof. The problem of finding founder set with score 0 for genotype data (without missing values) is the Pure Parsimony problem from [4]. In pure parsimony problem one asks whether there is a set of haplotypes F , such that $|F| \leq K$ and F generates input genotypes D . This problem is NP-complete [10]. We note that for maximum parsimony to have a solution, parameter K must be greater than $\sqrt{2|G|}$ [10]. \square

Theorem 5. *The optimization version of Problem 1 cannot be approximated in polynomial time within any factor, unless $NP=P$.*

Proof. Had we a polynomial time approximation algorithm with some fixed approximation factor α , we could solve Problem 1 in the case of $T = 0$, $D \subset \{0, 1, 2\}$ (genotypes) and $D \subset \{0, 1, -\}^n$ (haplotypes). But by Theorems 3 and 4, these problems are NP-hard. \square

Theorem 6. *Problem 1 is NP-complete when data $D \subset \{0, 1\}$ consists of haplotypes and $c = \frac{1}{n|D|}$.*

Proof. When c is $\frac{1}{n|D|}$ Problem 1 becomes a clustering problem. The problem can be stated: is there a set of K binary vectors F , such that $\sum_{h \in D} \min_{f \in F} d(f, h) \leq T$ where $d(f, h)$ is the Hamming distance of f and h . This problem is the complementary (Hamming distance instead of Hamming overlap) problem of the Hypercube segmentation problem, which is NP-complete even for $K = 2$ [7]. Thus, Problem 1 is also NP-complete. \square

5 Heuristic Algorithm for Founder Construction

The simplest algorithm to find the optimal set of founders is to enumerate all founder sets F of a given size K , compute $score_F^c(D)$ for each of them, and choose the best solution. Time complexity of this algorithm is proportional to $\binom{\Sigma^n}{K}$. This algorithm is not very practical, but could be improved by clever enumeration of the sets F (Branch-and-Bound).

In some cases finding founder sets for haplotypes is easy. If we set parameter c to ∞ , we try to find parses with minimum number of recombinations. Then, if $K = 2$ there is a polynomial $O(mn)$ time algorithm for finding optimal founder set [19], without missing values in the haplotypes. It is based on the fact that we can consider only haplotype columns on which both alleles are present and each such column infers the correct partition into two classes. The optimal founder set can be obtained in this case as follows:

Without loss of generality we can remove all columns that contain only single value. We process the haplotypes from left to right and consider adjacent columns. The alleles of the first founder column can be set arbitrarily to 0 and 1. Now let us assume that the founders have been fixed for column i and we proceed to the column $i + 1$. We count how many times substrings 00, 01, 10 and 11 occur at position i in the haplotypes. We pick either 01 and 10 or 00 and 11, depending which combination is more common. The founder column $i + 1$ is determined from the picked substrings; founders are set in a unique way such that the picked substrings occur at founder position i . Proceeding this way until column n we get a solution for $K = 2$ whose optimality is easily shown as between any successive columns this procedure uses a minimum number of recombinations on two founders.

Since the more common of substrings of 01 and 10 or 00 and 11 occurs at least in $\frac{1}{2}m$ haplotypes, we can have at most as many recombinations. Thus, an optimal solution can have at most $\frac{1}{2}m(n - 1)$ recombinations (this is a tight bound). Note that the above method finds optimal founder set of size 2 for genotypes as well

[22], as we can compute maximum number of occurrences of 00, 01, 10 and 11 in the case of genotype input. The possible substrings in genotypes are 00, 01, 10 and 11 as in haplotypes and 20, 21, 02, 12 and 22. In the latter substrings first four correspond uniquely to two of 00, 01, 10 and 11. Substring 22 can be resolved as 00 and 11 or 01 and 10. From these two possibilities we can choose the one minimizing the number of recombinations. However, this does not work when sequences contain missing values.

The algorithm just described works from left to right by assigning founder columns in a greedy fashion. Next we generalize this idea for arbitrary number of K founders.

The algorithm is the following. From left to right we construct the columns of F in a greedy manner. For column i we enumerate all $|\Sigma|^K$ possibilities and choose the one that minimizes the parsing score of the prefixes of the sequences in D up to column i . After the first pass we make repeated left-to-right passes until we have found a local optimum. In each pass the content of column i is reselected from the $|\Sigma|^K$ possibilities such that it minimizes the parsing score of the entire data D when F is kept fixed for all columns other than i . A single pass of this algorithm can be implemented in time $O(mnK|\Sigma|^K)$ (in time $O(mnK^2|\Sigma|^K)$) for dataset of m haplotypes (genotypes) of length n . Algorithm finds the optimum when $K = 2$, $c = \infty$, and sequences have no missing values. Similar algorithm was used in [15].

6 Hierarchical Parsing

The parsing scheme of Section 3 applies recombinations independently on each sequence to be parsed. There is no attempt to utilize the same recombination several times. The *hierarchical parsing scheme* aims at finding recombinations that are common to several sequences in the data. The structure of the hierarchical parsing is simple: we start from some given founder set and add to it recombinants of the founders (including the recombinants added in earlier steps) such that the resulting process finally generates data D .

This process forms a tree like history for the sequences D . Again, finding a shortest possible hierarchical parse is of interest but it seems very difficult. We note that if we start from a single sequence ($|F| = 1$) and add mutations and recombinations in this fashion, we would construct an ARG.

Instead of exact algorithm, we use following greedy heuristic algorithm: We try in all, at most $(n - 1)|F|(|F| - 1)$ ways to add new founder f that is recombinant of founders F , and add to the founder set the one that minimizes $score_{F \cup \{f\}}^c(D)$. By assigning the parameter c properly we would model errors or mutations in the sequences. If c is set to a low value we should stop the greedy algorithm when there are no recombinations. We take as the initial set of founders the ones that minimize $score_F^c(D)$.

Algorithm is the following: Find out founder set F_0 minimizing $score_{F_0}^c(D)$. Set $F_{i+1} = F_i \cup \{f\}$, where f is the recombinant of $f_1 \in F_i$ and $f_2 \in F_i$ that minimizes $score_{F_{i+1}}^c(D)$. Repeat until $score_{F_{i+1}}^c(D) = 0$.

As each greedy step decreases score by at least one, the total number of greedy steps cannot exceed $score_{F_0}^c(D)$. On the other hand, there must be a haplotype with at least $score_{F_0}^c(D)/m$ recombinations, where m is the number of haplotypes ($2|D|$ in case of genotypes). Thus we must take at least as many greedy steps. By taking starting founders F_0 as the ones minimizing $score_{F_0}^c(D)$, we minimize both the upper and lower bound on the number of greedy steps.

Trivial implementation of this greedy step would take time $O(mn^2|F_i|^3)$ in case of haplotypes and $O(mn^2|F_i|^4)$ in case of genotypes. Our implementation takes time $O(mn|F_i|^2)$ for haplotypes and $O(mn|F_i|^3)$ for genotypes. In practice, this algorithm becomes quite slow, as $|F_i|$ increases by one in every step.

7 Experimental Results

We used 220 datasets obtained from the HapMap data [18]. We selected data from two groups, abbreviated YRI (Yoruba) and CEU (Utah). For both these groups unphased genotypes are available for 30 trios, resulting in the total of 120 known haplotypes. From each of the 22 chromosomes we chose 5 fragments covering 100 consecutive SNPs starting from SNPs 1001, 2001, ..., 5001. We haplotyped 60 genotypes taken from the trios. We used $c = 1000$.

We compared our phasing results against fastPHASE [16] with standard settings (fastPHASE). Unlike our method, fastPHASE builds its final solution by combining several haplotype predictions. Therefore we also generated with fastPHASE a single-run solution based on 10 founders (clusters). We call this solution fastPHASE-10. With HIT [15] we generated a solution using 10 founders. We started the hierarchical algorithm with $K = 3, 7, 10$ initial founders, constructed by the algorithm of Section 5, and applied the greedy hierarchical parsing algorithm of Section 6 until the flat parsing score of the data decreased by one. So we can stop here as the recombinants added from now on would participate in only one parse and there is no hierarchy. Our Java implementation took a couple of minutes to run on a single HapMap dataset on a standard desktop PC.

Table 1 gives the number of switches averaged over CEU and YRI datasets. “Flat” is the parsing algorithm (2), and “Hierarchical” is the greedy hierarchical algorithm. The hierarchical method gives comparable results to fastPHASE-10, but fastPHASE and HIT are somewhat better. Figure 3 shows how the number of switches develops after every hierarchical step. The figure also shows the switch distances achieved by fastPHASE, fastPHASE-10, and HIT.

Our implementation selects between equally good alternatives (columns of initial founders and added recombinants of the greedy parsing step) with equal probabilities. Sometimes the actual choices had a significant effect on the result.

	Flat($K=3/7/10$)	Hierarchical($K=3/7/10$)	fastPHASE-10	fastPHASE	HIT
CEU	225/138/136	119/110/113	136	85	89
YRI	406/248/230	203/181/178	207	134	143

Table 1. Average values of switch distances for different algorithms and datasets.

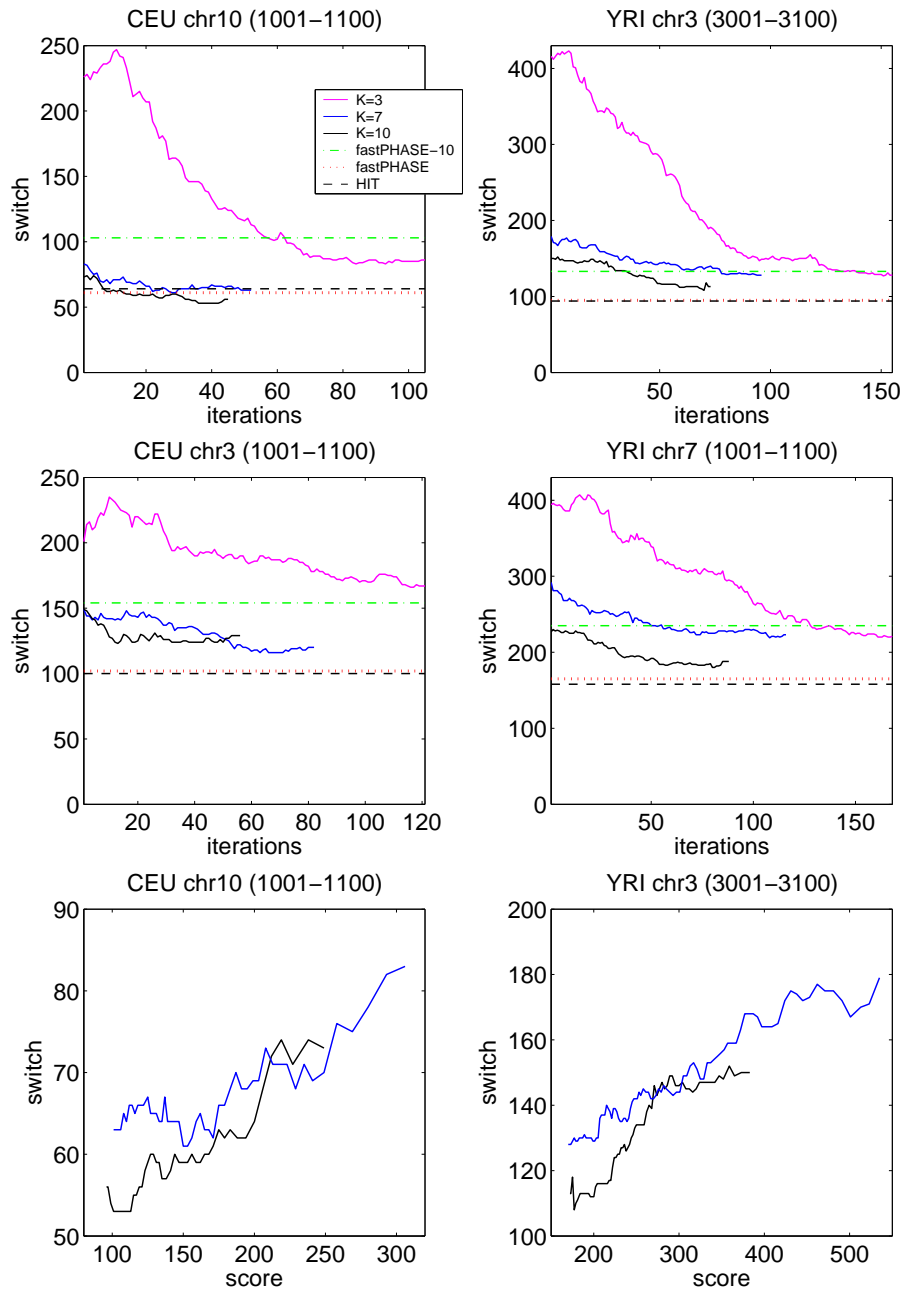


Fig. 3. Four upper panels illustrate on four typical datasets the behaviour of the switch distance after each greedy hierarchical step that adds a new recombinant to the founders. We used $K = 3, 7, 10$ initial founders (magenta, blue and black regular lines). The straight red dotted line shows fastPHASE's, straight dash dotted green line fastPHASE-10's, and straight dashed black line HIT's performance. The two lower diagrams visualize the correlation between the flat parsing score and the switch distance during the greedy hierarchical steps.

References

1. Daly, M., Rioux, J., Schaffner, S., Hudson, T., Lander, E.: High-resolution haplotype structure in the human genome. *Nature Genetics* **29** (2001) 229–232
2. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory on NP-Completeness*. W. H. Freeman and Company (1979)
3. Griffiths, R., Marjoram, P.: Ancestral inference from samples of DNA sequences with recombination. *Journal of Computational Biology* **3** (1996) 479–502
4. Gusfield, D.: Haplotype inference by pure parsimony. Technical Report CSE-2003-2, Department of Computer Science, University of California (2003)
5. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. *Comm. ACM* **18** (1975) 341–343
6. Kececioglu, J., Gusfield, D.: Reconstructing a history of recombinations from a set of sequences. *Discrete Applied Mathematics* **88** (1998) 239–260
7. In practice several alternative columns of initial founders as well as several recombinants to be added in the greedy parsing step have equally good effect. Kleinberg, J., Papadimitriou, C., Raghavan, P.: Segmentation problems. In: *Proc. STOC '98*, New York, USA, pp. 473–482. ACM Press 1998
8. Koivisto, M., Rastas, P., Ukkonen, E.: Recombination systems. In: *Theory Is Forever*, LNCS 3113, pp. 159–169. Springer 2004
9. Lajoie, M., El-Mabrouk, N.: Recovering haplotype structure through recombination and gene conversion. *Bioinformatics* vol. 21, suppl. 2, pp. ii173–ii179, 2005
10. Lancia, G., Pinotti, C., Rizzi, R.: Haplotyping populations: Complexity and approximations. Technical Report DIT-02-0080, Department of Information and Communication Technology, University of Trento 2002
11. Lin, S., Cutler, D.J., Zwick, M.E., Chakravarti, A.: Haplotype inference in random population samples. *American Journal of Human Genetics* **71** (2002) 1129–37
12. Lyngsø, R., Song, Y., Hein, J.: Minimum recombination histories by branch and bound. In: *Proc. WABI 2005*, LNCS 3692, pp. 239–250. Springer 2005
13. Pääbo, S.: The mosaic in our genome. *Nature* **421** (2003) 409–412
14. Rastas, P.: Haplotyyppien määrittäminen (Haplotype inference). Report C-2004-69 (M.Sc. thesis), Department of Computer Science, University of Helsinki 2004
15. Rastas, P., Koivisto, M., Mannila, H., Ukkonen, E.: A hidden markov technique for haplotype reconstruction. In: *WABI 2005*, LNCS 3692, pp. 140–151. Springer 2005
16. Scheet, P., Stephens, M.: A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *American Journal of Human Genetics* **78** (2006) 629–44
17. Schwartz, R., Clark, A., Istrail, S.: Methods for inferring block-wise ancestral history from haploid sequences. In: *Proc. WABI 2002* LNCS 2452, pp. 44–59. Springer 2002
18. The International HapMap Consortium: A haplotype map of the human genome. *Nature* **437** (2005) 1299–1320
19. Ukkonen, E.: Finding founder sequences from a set of recombinants. In: *Proc WABI 2002*, LNCS 2452, pp. 277–286. Springer 2002
20. Wade, C., Kulbokas, E., Kirby, A., Zody, M., Mullikin, J., Lander, E., Daly, M.: The mosaic structure of variation in the laboratory mouse genome. *Nature* **420** (2002) 574–578
21. Wang, L., Zhang, K., Zhang, L.: Perfect phylogenetic networks with recombination. *Journal of Computational Biology* **8** (2001) 69–78
22. Wu, Y., Gusfield, D.: Improved algorithms for inferring the minimum mosaic of a set of recombinants. To appear in: *Proc. CPM 2007*. Springer 2007