# Diploid Alignments and Haplotyping

Veli Mäkinen(✉) and Daniel Valenzuela

Helsinki Institute for Information Technology HIIT
Department of Computer Science
University of Helsinki, Helsinki, Finland
{vmakinen,dvalenzu}@cs.helsinki.fi

**Abstract.** Sequence alignments have been studied for decades under the simplified model of a consensus sequence representing a chromosome. A natural question is if there is some more accurate notion of alignment for diploid (and in general, polyploid) organisms. We have developed such a notion in our recent work, but unfortunately the computational complexity remains open for such a diploid pair-wise alignment; only a trivial exponential algorithm is known that goes over all possible diploid alignments. In this paper, we shed some light on the complexity of diploid alignments by showing that a haplotyping version, involving three diploid inputs, is polynomial time solvable.

## 1 Introduction and Related Work

There are myriads of variants of pair-wise sequence alignments trying to capture various biological sequence features, such as mutation biases, repeats (DNA), splicing (RNA), alternative codons (protein) [4,5], but the fundamental feature of a genome of a higher organism being diploid or even polyploid has remained largely unexplored in alignment literature. The closest come some fairly recent approaches in progressive multiple alignment that model a multiple alignment profile as a *labeled directed acyclic graph* (labeled DAG) [7,8]. These works define the alignment of two such labeled DAGs $A$ and $B$ as the problem of finding a path $P^A$ through $A$ and a path $P^B$ through $B$ such that the optimal alignment score of $P^A$ and $P^B$ is maximized. Since a pair-wise alignment models a diploid chromosome pair accurately, giving the synchronization of their haploid sequences, the labeled DAG alignment could be applied to model diploid alignment. However, the caveat is that this approach takes only partial information into account from the diploids, not their full content. It was shown in [9] how to modify the approach into a *covering* version that takes full content of diploids into account. Unfortunately, the computational complexity of this accurate model of diploid alignment remains open.

In this paper, we shed some light on the complexity of diploid alignment by showing that a haplotyping version, involving three diploid inputs, is polynomial time solvable. In addition to the theoretical interest, the haplotyping version may also be of practical value as a complementary technique to haplotype

Partially supported by Academy of Finland under grant 284598 (CoECGR).


R. Harrison et al. (Eds.): ISBRA 2015, LNBI 9096, pp. 235–246, 2015.
DOI: 10.1007/978-3-319-19048-8_20

assembly. We give some proof-of-concept simulation results that show excellent performance on realistic input scenarios on an implementation of the approach.

In what follows, we fix our mindset on the haplotyping problem to fix the terminology and to motivate our study also from the practical point of view. Then we formalize the notion of diploid alignments and show how this formalization can be extended to modeling the haplotyping problem.

### 1.1 Genotyping and Haplotyping

In *diploid organism* a pair of haplotype DNA sequences forms a chromosome pair, where one haplotype is inherited from the mother and one from the father. Each inherited haplotype sequence is a mixture (*recombination*) of the two haplotype sequences forming the corresponding diploid chromosome pair in the parent.

*Genotyping* consists in the discovery of variants in specific positions in the genome of an individual with respect to a *consensus genome* of the species. After genotyping of child, mother, and father, one can reason which variants came from the mother, which came from the father, and which are new. The inherited variants are called *germ-line variants* and the new *de novo variants*. A *homozygous variant* is inherited from both mother and father, and a *heterozygous variant* is inherited only from one of them.

*Haplotyping* consists in the assignment of heterozygous variants to the correct *phase*, that is, to a haplotype inherited from the mother or to a haplotype inherited from the father. The importance of this process is not just in revealing the inheritance pattern, but also in understanding the function of each haplotype; after all, genes and other functional units are residing in haplotypes, and the function always depends on the exact sequence content. Genotype information is just enough to argue about the effect of a single mutation, while haplotype information gives the full power of reasoning about the combined effect of a set of mutations.

The state of the art is that genotyping is nowadays a rather routinely conducted process, when studying e.g. human individuals in the hunt of disease causing mutations. It can be conducted by high-throughput sequencing of individual DNA, aligning the sequencing reads to the consensus genome, and analyzing the read alignments for variants supported by many reads. *Single-nucleotide polymorphisms (SNPs)* affecting a single genome position can be revealed with high accuracy, but larger indels and structural variants are much more hard to identify [11].

Given a set of predicted heterozygous variants, haplotyping is still a challenging task, and it is often solved using statistical methods [1,10,2]. Recent advances in pseudo-polynomial algorithms for *haplotype assembly* are however making large-scale haplotyping feasible [12]. In haplotype assembly the $j$-th read is reduced to a sequence $R^j$ from alphabet $\{*, 0, 1\}$ with $R^j[i] = *$ denoting that the read does not overlap $i$-th heterozygous variant, with $R^j[i] = 0$ denoting that the read overlaps the $i$-th heterozygous variant but does not support it, and with $R^j[i] = 1$ denoting that the read overlaps the $i$-th heterozygous variant and supports it. The task is to assign each read to one of the haplotypes such

that minimal flipping of bits inside reads is required to make them uniform with their chosen consensus haplotype pattern. The approach works if the read length is long enough to contain many variants.

The reads can be separated into small independent blocks (i.e. such that there are no shared variants among different blocks) and then solve the problem for each block independently. Some blocks have been identified as particularly difficult to phase [3].

There are also more tailored approaches to haplotyping that combine computational methods with problem-targeted sequencing technology [13].

In this paper, we propose a haplotyping algorithm that works directly at the DNA sequence level, thus differing from previous approaches. Our method is independent of the sequencing technology. We do not require any specific information about reads, so the variants might have been obtained using different methods.

The purpose of our algorithm is to haplotype *complex genome regions*, that may contain long and possibly overlapping variants that are difficult to capture by the haplotype assembly framework. We assume that one has identified complex regions of the child genome and predicted variants from those regions as well as from the same regions in the mother and father genomes. Such data can be produced e.g. by targeted high-coverage sequencing followed by variant prediction. Our haplotyping algorithm takes $O(n^3)$ time, where $n$ is the length of the genome region in question. This approach fits a haplotyping project, where a computationally light approach can be applied on easy-to-haplotype regions, and a more computational heavy approach can be applied on the identified complex regions.

### 1.2 Alignment of Diploid Individuals

Recently, new alignment models that are designed for diploid organisms, incorporating the possibility of recombination, were introduced [9]. In this section we briefly present these models and some basic definitions.

A *pair-wise alignment* (or simply an *alignment*, when it is clear from the context) of sequences $A$ and $B$ is a pair of sequences $(\mathsf{S}^A, \mathsf{S}^B)$ such that $\mathsf{S}^A$ is a supersequence of $A$, $\mathsf{S}^B$ is a supersequence of $B$, $|\mathsf{S}^A| = |\mathsf{S}^B| = n$ is the length of the alignment, and all positions which are not part of the subsequence $A$ (respectively B) in $\mathsf{S}^A$ (respectively $\mathsf{S}^B$), contain the *gap* symbol $'-'$, which is not present in the original sequences.

Given a similarity function $s(a, b)$ that assets the similarity between two characters, the similarity of a pair-wise alignment is simply defined as

$$S(\mathsf{S}^A, \mathsf{S}^B) = \sum_{i=1}^{n} s(\mathsf{S}^A[i], \mathsf{S}^B[i]).$$

The similarity of two sequences is then defined as

$$S(A, B) = max\{\sum_{1=1}^{n} s(\mathsf{S}^A[i], \mathsf{S}^B[i]) : (\mathsf{S}^A, \mathsf{S}^B) \text{ is an alignment of } A \text{ and } B\}.$$

An alignment that achieves that value is called an *optimal alignment*.

We say that $(\mathsf{S}^{A'}, \mathsf{S}^{B'})$ is a *recombination* of an alignment $(\mathsf{S}^{A}, \mathsf{S}^{B})$ if both alignments have the same length $n$ and there exists a binary string $P$ (for *phase*) such that $\mathsf{S}^{A'}[i] = \mathsf{S}^{A}[i]$ and $\mathsf{S}^{B'}[i] = \mathsf{S}^{B}[i]$ if $P[i] = 0$, and $\mathsf{S}^{A'}[i] = \mathsf{S}^{B}[i]$ and $\mathsf{S}^{B'}[i] = \mathsf{S}^{A}[i]$ if $P[i] = 1$. We say that the characters are *swapped* in the positions in which $P[i] = 1$.

We denote this *recombination relation* by $(\mathsf{S}^{A'}, \mathsf{S}^{B'})\Re(\mathsf{S}^{A}, \mathsf{S}^{B})$.

**Diploid to Diploid Similarity[9][1]:** Given two pair-wise alignments $(\mathsf{S}^{A}, \mathsf{S}^{B})$ and $(\mathsf{S}^{X}, \mathsf{S}^{Y})$ the diploid to diploid similarity is the sum of the optimal similarity scores by components, given by the best possible recombination of both individuals. More formally:

$$S_{d-d}((A,B),(X,Y)) = \max\{S(A',X') + S(B',Y') : (\mathsf{S}^{A'}, \mathsf{S}^{B'})\Re(\mathsf{S}^{A}, \mathsf{S}^{B}) \wedge (\mathsf{S}^{X'}, \mathsf{S}^{Y'})\Re(\mathsf{S}^{X}, \mathsf{S}^{Y}))\}$$

Neither algorithms nor complexity bounds were provided for the similarity above. However, a simpler version where one of the individuals is considered as a diploid and the other as a pair of haploids was developed:

**Pair of Haploids to Diploid Similarity[9]:** Given a pair-wise alignment $(\mathsf{S}^{A}, \mathsf{S}^{B})$ and two sequences $X$ and $Y$, the pair of haploids to diploid distance is defined as the sum of the optimal similarity scores by components, given by the best possible recombination of the diploid individual. More formally:

$$S_{d-hh}((A,B),(X,Y)) = \max\{S(A',X) + S(B',Y) : (\mathsf{S}^{A'}, \mathsf{S}^{B'})\Re(\mathsf{S}^{A}, \mathsf{S}^{B})\}$$

The best algorithm for this similarity measure runs in $O(n^3)$ time and requires $O(n^2)$ memory. A modified version of the above was also presented, which can be computed in $O(nk)$ time and $O(n)$ memory, where $k$ is the resulting edit distance (considering the analogous problem with min instead of max and costs instead of scores).

## 2 Haplotype Sequences via Alignment

The measures covered in the previous section intent to measure the similarity between individuals in which heterozygous and homozygous variations are known, but there is no knowledge about the correct phasing of the variations.

As such, they are not useful for haplotype phasing, as the evolutionary recombination pattern is unique to the individual. To take the evolutionary context into account, we need to extend the measures to mother-father-child trios, as it is considered next.

[1] The original paper considers the distance measure instead of the similarity, but these are computationally equivalent.

### 2.1 The Similarity Model

Let us consider three pair-wise alignments, $(\mathsf{S}^{M_1}, \mathsf{S}^{M_2})$,$(\mathsf{S}^{F_1}, \mathsf{S}^{F_2})$ and $(\mathsf{S}^{C_1}, \mathsf{S}^{C_2})$ of length $L_M$, $L_F$ and $L_C$ respectively. Those represent the diploid sequences of the mother, father and child, and we call the three of them a mother-father-child (m-f-c) trio.

We define the *haplotyping similarity of an m-f-c trio*, $\mathsf{H}$(m-f-c), as the maximum pair-wise similarity between one of the sequences of the mother and one of the sequences of the child, plus the pair-wise similarity between the other child sequence and one of the father sequences, *assuming that none of the diploids is phased correctly*. This means that we need to allow free recombination in each pair-wise alignment, to let the model discover the real phase. More formally:

$$\mathsf{H}(\text{m-f-c}) = \max\{S(M_1', C_1') + S(F_1', C_2') : (\mathsf{S}^{M_1'}, \mathsf{S}^{M_2'})\Re(\mathsf{S}^{M_1}, \mathsf{S}^{M_2}) \\ \wedge (\mathsf{S}^{F_1'}, \mathsf{S}^{F_2'})\Re(\mathsf{S}^{F_1}, \mathsf{S}^{F_2}) \wedge (\mathsf{S}^{C_1'}, \mathsf{S}^{C_2'})\Re(\mathsf{S}^{C_1}, \mathsf{S}^{C_2})\}$$

Figure 1 shows an optimal alignment of a m-f-c trio.

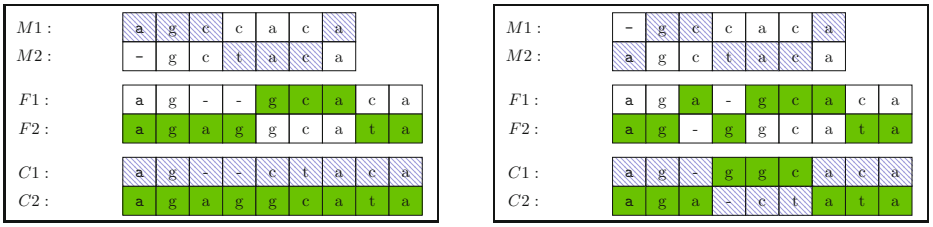



**Fig. 1.** On the left we show how the pair-wise alignments would be if we knew the correct phasing of the three individuals. On the right, the same individuals are presented, but the haplotype phasing is not known a priori. We assume a similarity function that scores 1 for equal characters, and $-1$ for indels and mismatches. The colored recombinations show the sequences $M_1'$ (hatched blue), $F_1'$ (green) from the recombination that gives the optimal alignment. The haplotyping similarity of the trio is $S(\texttt{AGCTACA}, \texttt{AGCTACA}) + S(\texttt{AGAGGCATA}, \texttt{AGAGGCATA}) = 7 + 9 = 16$. The binary strings associated to the recombinations are $P_M = \texttt{1001110}$, $P_F = \texttt{110100011}$ and $P_C = \texttt{000111000}$. Note that the latter corresponds to the predicted phase for the child genome. It is also important to note that none of the binary strings are signaling evolutionary recombinations, but they are signaling phasing errors in the input data.

Notice that this similarity measure is an extension of the diploid to pair of haploids similarity, but it is easier than the diploid to diploid similarity measure. The feasibility of the solution comes from the fact that only the child genome needs to be *covered* by the alignment: Sequences induced by $\mathsf{S}^{C_1'}$ and $\mathsf{S}^{C_2'}$ are both aligned in the $\mathsf{H}$(m-f-c) definition, whereas only sequences induced by $\mathsf{S}^{M_1'}$ and $\mathsf{S}^{F_1'}$ are aligned from the recombined father and mother sequences, respectively. The difficulty of the diploid to diploid measure lies in the requirement of covering both recombined inputs, which appears difficult to capture at least by dynamic

programming. For the m-f-c trio case, we can luckily extend the cubic solution of the diploid to pair of haploids similarity.

### 2.2 Dynamic Programming Algorithm

In this section we present our algorithm to compute the m-f-c similarity. We propose a dynamic programming formulation that computes values $\mathsf{H}_{i,j,k,m,f,c}$ with $i \in \{1, L_M\}$, $j \in \{1, L_F\}$, $k \in \{1, L_C\}$, $m \in \{1, 2\}$, $f \in \{1, 2\}$ and $c \in \{1, 2\}$. The value stored in $\mathsf{H}_{i,j,k,m,f,c}$ stands for the similarity score between $(\mathsf{S}^{M_1}[1, i], \mathsf{S}^{M_2}[1, i])$, $(\mathsf{S}^{F_1}[1, j], \mathsf{S}^{F_2}[1, j])$, and $(\mathsf{S}^{C_1}[1, k], \mathsf{S}^{C_2}[1, k])$, with the additional constrain that the last character of the mother alignment is swapped if and only if $m = 1$, the last character of the father alignment is swapped if and only if $f = 1$ and the last character of the child alignment is swapped if and only if $c = 1$.

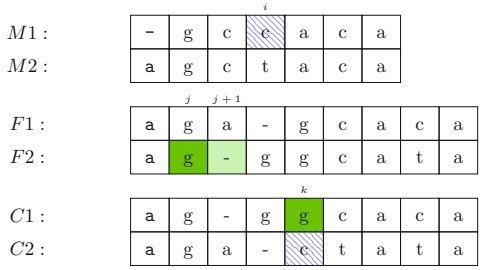
We first consider the particular case when the input alignments contain no gaps. That is, $\mathsf{S}^{C_1} = C1$, $\mathsf{S}^{C_2} = C_2$, $\mathsf{S}^{F_1} = F_1$, etc. It is possible to compute those values recursively as follows:

$$\mathsf{H}_{i,j,k,m,f,c} = \max \begin{cases} \mathsf{H}_{i-1,j,k,*,f,c} & + s('-', M_m[i]) & & \text{if } i > 1 \\ \mathsf{H}_{i-1,j,k-1,*,f,*} & + s(C_c[k], M_m[i]) & + s(C_{c\oplus 1}[k], '-') & \text{if } i, k > 1 \\ \mathsf{H}_{i,j-1,k,m,*,c} & & + s('-', F_f[j]) & \text{if } j > 1 \\ \mathsf{H}_{i,j-1,k-1,m,*,*} & + s(C_c[k], '-') & + s(C_{c\oplus 1}[k], F_f[j]) & \text{if } j, k > 1 \\ \mathsf{H}_{i-1,j-1,k-1,*,*,*} & + s(C_c[k], M_m[i]) & + s(C_{c\oplus 1}[k], F_f[j]) & \text{if } i, j, k > 1 \end{cases}$$

The recurrence uses several short-hand notations as follows. With $\mathsf{H}_{i,j,k,*,*,*}$ we mean $\max_{\{m,f,c\} \in \{1,2\}^3} \{\mathsf{H}_{i,j,k,m,f,c}\}$ in order to consider all the 8 valid subproblems where the previous last characters could have been swapped or not (and analogously when only one or two $*$ symbols are present). With $c \oplus 1$ we mean 2 if $c = 1$ and 1 otherwise.

The first and third cases correspond to the scenarios where the last character of the mother (respectively, of the father) is not aligned with any character of the child, and therefore a gap symbol is inserted. The second and fourth cases corresponds to the scenarios where one of the last characters of the child is aligned with one of the last characters of the mother (respectively, of the father), and the other character of the child is not aligned, therefore, a gap is inserted. The fifth case is the scenario where the last character of one of the child sequences is aligned with one of the last sequences of the mother, and the last character of the other child sequence is aligned with the last character of one of the sequences of the father.

The correctness of the algorithm can be seen as a generalization of the classic dynamic programming algorithm: Firstly, all the possibilities of alignment among the last characters of the input sequences are considered. For each of those case, it remains the subproblem of the m-f-c alignment where the characters that had just been aligned (with another character or with a gap) are removed. For the alignment to be optimal, it is required that the subproblem is solved optimally too, and therefore, the recursion holds true.




$$H_{i,j+1,k,1,2,2} = max\{H_{i,j,k,1,*,2}\} = H_{i,j,1,k,1,2,2}$$
$$H_{i,j,k,1,2,2} = max\{H_{i-1,j-1,k-1,*,*,*}\} + s(c,c) + s(g,g)$$

**Fig. 2.** Example showing two steps of the dynamic programming algorithm. First for the computation of $H_{i,j+1,k,1,2,2}$ we highlight the characters that need to be considered. As the $j+1$ character of the father sequence that is being considered is a gap, the recursion returns the previous value of $j$, keeping all the parameters constant, except for the sequence of the father that can be considered (line 10 of Algorithm 1.) For the computation of $H_{i,j,k,1,2,2}$ the previous values indicated by lines 6,7,11,12, and 14 needs to be considered. Those correspond to all possible combinations of alignments between the highlighted characters (allowing some of them to be ignored, but not all of them).

Notice that we are allowing free recombinations when a path change its value in either $m$,$f$, or $c$ indexes. It is straightforward to include a penalty in those changes of indexes, as it was proposed in [9], however, we decided to stay free of such penalties for reasons that are discussed in Section 3.

It remains to consider the scenarios where the input sequences do have gaps. Observe that the gaps in the input alignments need to be ignored without any cost in order to model the similarity measure correctly; the gaps in the input sequences are just required for keeping the positions of the two haplotypes of the diploid synchronized so that recombination can be modeled. Algorithm 1 shows our pseudo-code to handle this: If for a given configuration the last character of the mother is a gap, we can immediately resort to the value computed for the position $i-1$. This just ignores that gap character. The case for the father is handled analogously. When the gap character comes in one of the child sequences, it is handled implicitly by the recursion, given that $s('-','-') = 0$ allows the gap character from the child to be ignored through the second and fourth cases of the recurrence. Figure 2 simulates one step of the computation.

The straightforward implementation as in Algorithm 1 would require $O(n^3)$ time and memory, as we need to retrieve the phasing. We implemented the checkpoint method [14], a flexible variant of Hirschberg's algorithm [6] that allows our algorithm to run in $O(n^3)$ time and $O(n^2)$ memory. Still, in order to obtain a scalable method, it is possible to apply some heuristics [9].

### 2.3 Haplotype Phasing

Once all the values $\mathsf{H}_{i,j,k,m,f,c}$ are computed, it is enough to trace back the path that originated the optimal score to obtain the optimal alignment. We notice that if we collect the three last indexes $m$, $f$ and $c$ from the path we will obtain the recombination binary strings for the mother, father, and child. In particular, the latter gives us the phasing of the child diploid that maximizes the similarity of the m-f-c trio. In the example of Figure 1 the binary strings are $P_M$, $P_F$ and $P_C$; the last one being the phasing of the child diploid.

**Algorithm 1.** Haplotyping similarity of an m-f-c trio. The algorithm corresponds to the dynamic programming implementation of the recurrence presented in Section 2.2, modified to handle the gaps in the input sequences properly.

```
1:  function HaploidSimilarity(M_1, M_2, F_1, F_2, C_1, C_2)
2:
3:      H[0, 0, 0, *, *, *] ← 0
4:      SetGlobal(H,M1,M2,F1,F2,C1,C2)
5:      for i ← 0 to L_M do
6:          for j ← 0 to L_M do
7:              for k ← 0 to L_M do
8:                  for m ← 1 to 2 do
9:                      for f ← 1 to 2 do
10:                         for c ← 1 to 2 do
11:                             H[i, j, k, m, f, c] ← HValue(i,k,m,f,c)
12:     return max H[L_m, L_f, L_c, *, *, *]

1:  function HValue(i, j, k, m, f, c)
2:      value ← − inf
3:      if i > 0 then
4:          if M_m[i] =′ −′ then
5:              return max{H[i − 1, j, k, *, f, c]}
6:          value ← max{value, H[i − 1, j, k, *, f, c] + s(′−′, M_m[i])}
7:          value ← max{value, H[i − 1, j, k − 1, *, f, *] + s(C_c[k], M_m[i]) + s(C_{c⊕1}[k],′ −′)}
8:      if j > 1 then
9:          if F_f[i] =′ −′ then
10:             return max{H[i, j − 1, k, m, *, c]}
11:         value ← max{value, H[i, j − 1, k, m, *, c] + s(′−′, F_f[j])}
12:         value ← max{value, H[i, j − 1, k − 1, m, *, c] + s(C_c[k],′ −′) + s(C_{c⊕1}[k], F_f[j])}
13:     if j > 1 & > 1 then
14:         value ← max{value, H[i − 1, j − 1, k − 1, m, *, c] + s(C_c[k], M_m[i]) + s(C_{c⊕1}[k], F_f[j])}
15:     return value
```

## 3 From Variants to Unphased Diploid Genome

Our phasing algorithm assumes the inputs as pair-wise alignments representing unphased diploid genomes. These can be constructed as follows. After sequencing the target region (or whole genome) on each individual involved, one can align the reads and analyse the variants [11]. All the predicted homozygous variants on an individual can be *applied* to the consensus genome to produce a base $S$ of a pair-wise alignment; with applying we mean that the content of consensus is replaced by the variants. Then the heterozygous variants can be greedily applied

to $S$ from left to right such that if a variant overlaps a previously applied variant, it is not applied. This forms the non-gapped content $T$ of the top row of a pair-wise alignment. The remaining set of heterozygous variants are applied to $S$ to produce the non-gapped content $B$ of the bottom row of a pair-wise alignment. Finally, enough gaps are added such that $T$ and $B$ are synchronized according to their origin in $S$. This process produces an unphased representation of a diploid genome. It is important to notice that our algorithm is invariant to the phasing of the input variants: In Figure 1 it is shown two different inputs that correspond to the same variants; in the left the input is already correctly phased for the trio, and in the right the variants are incorrectly phased, and the result in both scenarios is the same. This is possible because we allow free recombination in each of the diploids, thus giving an equal opportunity for each variant to be phased either way.

We shall consider in Sect. 5 the case when the overlap depth is higher, so that some variants are left after constructing $B$.

## 4 Experimental Results

We implemented our method fully in $C++$, and the source code is freely available[2]. We ran our experiments in a computer node with 2 Intel Xeon E5540 2.53GHz processors, 32GB of RAM. The operating system was Ubuntu 12.04.4. Our code was compiled with gcc 4.6.4, optimization option $-O3$.

To study the difficult areas to phase we simulated our father-mother-child trios directly without adding the variant analysis step of Sect. 3. In this way we could control the amount the mutation ratio, the type of variations, and the measurement/predictions errors that are present in the input data.

To simulate the mother sequences, we started from two identical copies of a sample from human chromosome 21. We inserted different types of variations, and then we simulated a recombination of that pair-wise alignment to obtain the child chromosome that is inherited from the mother. For the recombination process we choose recombination points at random. We did analogously to simulate the father, and the chromosome that the child inherits from the father.

The variations planted on the parents were as follow: *point mutations* consisted of SNPs and single nucleotide deletions. Then we introduced *long indels* (larger than 50 base pairs). In the case of insertions, those consisted of random base pairs. We also inserted *short tandem repeats*[15] consisting of insertions of length between 20 and 60 repeating a sequence between 2 and 6 base pairs. After the recombination has been simulated to generate the child sequences, we introduced *de novo point mutations*. (We also considered *short indels* but as the results turned out to be analogous to long indels, we omitted them from the results reported here for the lack of space.)

In addition to all the previous parameters, we also introduced *random errors* over all the sequences at the end, to take into account errors during the

[2] http://www.cs.helsinki.fi/u/dvalenzu/code/haplotyping/

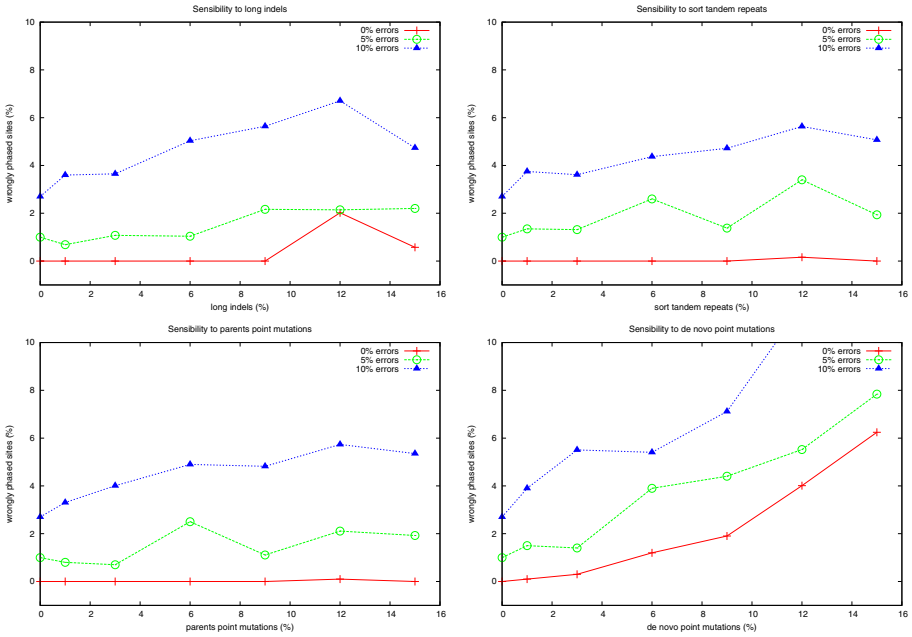



**Fig. 3.** We show the percentage of incorrectly phased positions versus the mutation ratio. For every graphic we include three levels of random noise. The two first plots (above) show the behavior against long indels and short tandem repeats. The third plot (below left) shows the behavior against point mutations in the parents and the fourth plot (below right) shows de novo variants.

sequencing of the sequences and during variant calling to discover the genotype patterns that constitute the inputs of our algorithm.

We studied the quality of our phasing algorithm by measuring how sensible it was with respect to each of the parameters. For that sake, we made our simulations from a 1000 bp sequence and we measured the percentage of positions that were incorrectly phased. For each different type of variation, the ratio ranged between 0 and 15%. Regarding to the error ratios we considered three scenarios: a very optimistic one, were the genotyping was done without errors (0%), a moderated scenario where error ratio is 5%, and a pessimistic scenario where the error ratio is 15%. The time used for haplotyping each simulated trio was less than a hour. The results are shown in Figure 3.

## 5 Discussion

Our case study and experimentation on the haplotyping problem show that our method can provide a complementary technique to perform haplotype phasing in complex genome sequences that are too difficult for haplotype assembly

modeling. Our experiments are so far showing the proof-of-concept, and several aspects need to be taken into account in order to apply the method on the real data setting of Sect. 3. As the motivation for our approach is complex genome regions, it should be observed that just the detection of variants in such areas is challenging. It can happen that variant predictions overlap such that at some positions a diploid genome is not enough to cover all variants [16]. For the process in Sect. 3, this means that after constructing $T$ and $B$ for the content of an unphased diploid genome, there are still some heterozygous variants to be applied. Consider continuing the process further to create a *multiple alignment* with some small number $c$ of sequences for child genome, $m$ for mother genome, and $f$ for father genome. Our dynamic programming approach can be extended to this scenario, by considering all $\binom{c}{2}$ pairs of rows from child multiple alignment at each column to be aligned to $m$ possible rows in mother multiple alignment at column $i$ and $f$ possible rows in father multiple alignment at column $j$. We plan to implement this scheme so as to compare our approach to other haplotyping methods.

Our liberal model of allowing crossover at every position can be made more restrictive by exploiting the connection to labeled DAGs we already discussed in Sect. 1: Consider the alignment visualization in Fig. 1. This can be viewed as a labeled DAG, by interpreting each cell as a vertex, and drawing an arc from bottom cell to its neighbor on the right and to its neighbor on its top right, and symmetrically for top cells. Each vertex has then two outgoing arcs except for the two last vertices. With some existing local haplotype information, some crossovers can be forbidden by removing non-horizontal arcs. The problem to be solved becomes that of finding two paths $C_1$ and $C_2$ through the child DAG, a path $M_1$ through mother DAG, and a path $F_1$ through father DAG, such that $S(C_1, M_1) + S(C_2, F_1)$ is maximized. Extending our dynamic programming approach to this generalization is left as future work, but we think this is feasible. On a more direct extension, it is straightforward to include a penalty cost for each recombination in our equations in order to avoid overfitting.

Finally, the main objective of this study is to illustrate that sequence alignments can be extended to take the full content of diploid chromosome representations into account, and that meaningful alignment problems under this model can be stated and solved in polynomial time. Given the unknown complexity of the very basic diploid alignment problem on two diploid inputs and the connection to covering problems on labeled DAGs, the current study is probably only scratching the surface of a prominent subarea of research.

## References

1. Browning, S.R., Browning, B.L.: Haplotype phasing: existing methods and new developments. Nature Reviews Genetics 12(10), 703–714 (2011)
2. Chen, W., Li, B., Zeng, Z., Sanna, S., Sidore, C., Busonero, F., Kang, H.M., Li, Y., Abecasis, G.R.: Genotype calling and haplotyping in parent-offspring trios. Genome Research 23(1), 142–151 (2013)

3. Zhi-Zhong Chen, Fei Deng, and Lusheng Wang. Exact algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics, btt349 (2013)
4. Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G.: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press (1998)
5. Gusfield, D.: Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. Cambridge University Press (1997)
6. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Communications of the ACM 18(6), 341–343 (1975)
7. Lee, C., Grasso, C., Sharlow, M.F.: Multiple sequence alignment using partial order graphs. Bioinformatics 18(3), 452–464 (2002)
8. Löytynoja, A., Vilella, A.J., Goldman, N.: Accurate extension of multiple sequence alignments using a phylogeny-aware graph algorithm. Bioinformatics 28(13), 1684–1691 (2012)
9. Mäkinen, V., Valenzuela, D.: Recombination-aware alignment of diploid individuals. BMC Genomics 15(suppl. 6), S15 (2014)
10. Marchini, J., Cutler, D., Patterson, N., Stephens, M., Eskin, E., Halperin, E., Lin, S., Qin, Z.S., Munro, H.M., Abecasis, G.R., et al.: A comparison of phasing algorithms for trios and unrelated individuals. The American Journal of Human Genetics 78(3), 437–450 (2006)
11. Pabinger, S., Dander, A., Fischer, M., Snajder, R., Sperk, M., Efremova, M., Krabichler, B., Speicher, M.R., Zschocke, J., Trajanoski, Z.: A survey of tools for variant analysis of next-generation genome sequencing data. Briefings in Bioinformatics 15(2), 256–278 (2014)
12. Patterson, M., Marschall, T., Pisanti, N., van Iersel, L., Stougie, L., Klau, G.W., Schönhuth, A.: WHATSHAP: Haplotype assembly for future-generation sequencing reads. In: Sharan, R. (ed.) RECOMB 2014. LNCS, vol. 8394, pp. 237–249. Springer, Heidelberg (2014)
13. Peters, B.A., Kermani, B.G., Sparks, A.B., Alferov, O., Hong, P., Alexeev, A., Jiang, Y., Dahl, F., Tang, T., Haas, J., et al.: Accurate whole-genome sequencing and haplotyping from 10 to 20 human cells. Nature 487(7406), 190–195 (2012)
14. Powell, D.R., Allison, L., Dix, T.I.: A versatile divide and conquer technique for optimal string alignment. Information Processing Letters 70(3), 127–139 (1999)
15. Weber, J.L., Wong, C.: Mutation of human short tandem repeats. Human molecular genetics 2(8), 1123–1128 (1993)
16. Wittler, R.: Unraveling overlapping deletions by agglomerative clustering. BMC Genomics 14(S-1), S12 (2013)