WhatsHap: Haplotype Assembly for Future-Generation Sequencing Reads

Murray Patterson[1,2,⋆], Tobias Marschall[1,⋆], Nadia Pisanti[3,4], Leo van Iersel[1], Leen Stougie[1,5], Gunnar W. Klau[1,5,⋆⋆], and Alexander Schönhuth[1,⋆⋆]

[1] Life Sciences, CWI Amsterdam, The Netherlands
{m.d.patterson,t.marschall,gunnar.klau,a.schoenhuth}@cwi.nl
[2] LBBE, CNRS and Université de Lyon 1, Villeurbanne, France
[3] Department of Computer Science, University of Pisa, Italy
[4] LIACS, Leiden University, The Netherlands
[5] VU University Amsterdam, The Netherlands

Abstract. The human genome is diploid, that is each of its chromosomes comes in two copies. This requires to *phase* the *single nucleotide polymorphisms (SNPs)*, that is, to assign them to the two copies, beyond just detecting them. The resulting haplotypes, lists of SNPs belonging to each copy, are crucial for downstream analyses in population genetics. Currently, statistical approaches, which avoid making use of direct read information, constitute the state-of-the-art. *Haplotype assembly*, which addresses phasing directly from sequencing reads, suffers from the fact that sequencing reads of the current generation are too short to serve the purposes of genome-wide phasing.

Future sequencing technologies, however, bear the promise to generate reads of lengths and error rates that allow to bridge all SNP positions in the genome at sufficient amounts of SNPs per read. Existing haplotype assembly approaches, however, profit precisely, in terms of computational complexity, from the limited length of current-generation reads, because their runtime is usually exponential in the number of SNPs per sequencing read. This implies that such approaches will not be able to exploit the benefits of long enough, future-generation reads.

Here, we suggest WhatsHap, a novel dynamic programming approach to haplotype assembly. It is the first approach that yields provably optimal solutions to the *weighted minimum error correction (wMEC)* problem in runtime linear in the number of SNPs per sequencing read, making it suitable for future-generation reads. WhatsHap is a *fixed parameter tractable (FPT)* approach with coverage as the parameter. We demonstrate that WhatsHap can handle datasets of coverage up to 20x, processing chromosomes on standard workstations in only 1-2 hours. Our simulation study shows that the quality of haplotypes assembled by WhatsHap significantly improves with increasing read length, both in terms of genome coverage as well as in terms of switch errors. The switch error rates we achieve in our simulations are superior to those obtained by state-of-the-art statistical phasers.

⋆ Joint first authorship.
⋆⋆ Joint last authorship.

R. Sharan (Ed.): RECOMB 2014, LNBI 8394, pp. 237–249, 2014.

1 Introduction

The human genome is *diploid*, that is, each of its chromosomes comes in two copies (except for sex chromosomes in males), one from the mother and one from the father. These parental copies are affected by different *single nucleotide polymorphisms (SNPs)*, and assigning the variants to the copies is an important step towards the full characterization of an individual genome. The corresponding assignment process is referred to as *phasing* and the resulting groups of SNPs are called *haplotypes*. Phasing SNPs in population studies allows to, for example, identify selective pressures and subpopulations, and to link possibly disease-causing SNPs with one another [13]. This explains that phasing SNPs has been an instrumental step in many human whole-genome projects [5,28]. In the meantime, globally concerted efforts have generated *reference panels* of haplotypes, for various populations, which may serve corresponding downstream analyses [29,30].

There are two major approaches to phasing variants. The first class of approaches relies on *genotypes* as input, which are lists of SNP alleles, together with their zygosity status. While *homozygous* alleles show on both chromosomal copies, and obviously apply for both haplotypes, *heterozygous* alleles show on only one of the copies, and have to be partitioned into two groups. If m is the number of heterozygous SNP positions, there are 2^m many possible haplotypes. This illustrates that directly phasing from genotype data is a hard computational problem. The corresponding approaches are usually statistical in nature, and they integrate existing reference panels. The underlying assumption is that the haplotypes to be computed are a mosaic of reference haplotype blocks that arises from recombination during meiosis. The output is the statistically most likely mosaic, given the observed genotypes. Most prevalent approaches are based on latent variable modeling [17,21,26]. Other approaches use Markov chain Monte Carlo techniques [23].

The other class of approaches makes direct usage of sequencing read data. Such approaches virtually assemble reads from identical chromosomal copies and are referred to as *haplotype assembly* approaches. Following the parsimony principle, the goal is to compute two haplotypes to which one can assign all reads with the least amount of sequencing errors to be corrected and/or erroneous reads to be removed. Among such formulations, the *minimum error correction (MEC)* problem has gained most of the recent attention. The MEC problem, which we will formally define in Section 2, consists of finding the minimum number of corrections to the SNP values to be made to the input in order to be able to arrange the reads into two haplotypes without conflicts. A major advantage of MEC is that it can be easily adapted to a weighted version (wMEC), in order to deal with phred-based error probabilities. Such error schemes are common in particular for *next-generation sequencing (NGS)* data. An optimal solution for the wMEC problem then translates to a maximum likelihood scenario relative to the errors to be corrected.

In tera-scale sequencing projects, e.g., [5,28], ever increasing read length and decreasing sequencing cost make it clearly desirable to phase directly from read

data. However, statistical approaches are still the methodology of choice because: (i) most NGS reads are still too short to bridge so-called *variant deserts*. Successful read-based phasing, however, requires that all pairs of neighboring heterozygous SNP alleles are covered; and (ii) the MEC problem is $\mathcal{NP}$-hard, and so are all other similar problem formulations.

Most advanced existing algorithmic solutions to MEC [6,16] take time exponential in the number of variants per read, and, ironically, often benefit precisely from variant deserts, because these allow to decompose a problem instance into independent parts. A major motivation behind read-based approaches, however, is to handle long reads that cover as many variants as possible, thereby bridging all variant deserts. Hence, the current perception of haplotype assembly is often that it underlies theoretical limitations that are too hard to overcome.

Here, we present a *fixed parameter tractable (FPT)* approach to wMEC where *coverage*, that is the number of fragments that cover a SNP position, is the only parameter. Hence, the runtime of our approach is, for the first time, *polynomial (in fact: linear) in the number of SNPs per read*, which addresses the future sequencing technologies that will generate reads of several tens of thousands of base pairs (bp) in length, and that the currently existing approaches are not suitable for processing such data. A carefully engineered implementation of our algorithm allows the treatment of whole-genome datasets of maximum coverage up to 20x on the order of hours on a standard workstation. For datasets of higher coverage, we provide a technique for choosing a reasonable selection of reads. We demonstrate that the resulting haplotypes suffer from only minor amounts of errors, even on high-coverage datasets, while we provide a provably optimal solution to the wMEC problem on bounded-coverage datasets. To do so, we test against a long-read benchmark dataset that we produced. Such a dataset will be useful for future tools that leverage long reads.

2 The Minimum Error Correction (MEC) Problem

The input to the MEC problem is a matrix $\mathcal{F}$ with entries in $\{0, 1, -\}$. Each row of $\mathcal{F}$ corresponds to a fragment/read. Each column of $\mathcal{F}$ corresponds to a SNP position. The "$-$" symbol, which is referred to as a *hole*, is used when a fragment does not contain any information at the corresponding SNP position. This can be either because the SNP position is not covered by the read, or because the read gives no accurate information at that position. Let n be the number of rows (or fragments) of $\mathcal{F}$ and m the number of columns (or SNP positions).

A *haplotype* can formally be defined as a string of length m consisting of 0's and 1's. If h is a haplotype, then the i-th row of $\mathcal{F}$ is said to *conflict* with h if there is some SNP position j for which $h(j) \neq \mathcal{F}(i, j)$ while $\mathcal{F}(i, j) \neq -$. We say that $\mathcal{F}$ is *conflict free* if there exist two haplotypes h_1, h_2 such that each row of $\mathcal{F}$ does not conflict with at least one of h_1 and h_2. Under the *all-heterozygous assumption*, where all columns correspond to heterozygous sites, h_1 must be the complement of h_2.

The goal of MEC is to make $\mathcal{F}$ conflict free by flipping a minimum number of entries of $\mathcal{F}$ from 0 to 1 or vice versa. The weighted variant of MEC, denoted

wMEC, has an additional weight function w as input. This weight function assigns a non-negative weight $w(i, j)$ to each entry $\mathcal{F}(i, j)$ of $\mathcal{F}$. This weight can reflect the relative confidence that the entry is correctly sequenced. The goal of wMEC is to make $\mathcal{F}$ conflict free by flipping entries in $\mathcal{F}$ with a minimum total weight.

The MEC problem, which is also called *minimum letter flip*, was introduced by Lippert *et al.* [22]. Cilibrasi *et al.* [7] showed that this problem is NP-hard even if each fragment is "gapless", i.e., if it consists of a consecutive sequence of 0's and 1's with holes to the left and to the right. Panconesi and Sozio [25] were the first to propose a practical heuristic for solving MEC. An exact branch and bound algorithm and a heuristic genetic algorithm were presented by Wang *et al.* [31]. Levy *et al.* [19] designed a greedy heuristic to assemble the haplotype of the genome of J. Craig Venter. Bansal *et al.* [4] developed an MCMC method to sample a set of likely haplotypes. In a follow-up, some of the authors proposed a much faster MAX-CUT-based heuristic algorithm called HapCUT [3], which they show to outperform [25,19], while showing similar accuracy to [4] in shorter running time. Very recently, Selvaraj *et al.* [27] combine the HapCUT [3] algorithm with proximity-ligation, which exploits information from "chromosome territories", to develop a method which reports good results on whole-genome haplotype reconstruction. In another recent paper, He *et al.* [16] proposed an exact dynamic programming algorithm. However, their algorithm depends exponentially on the length of the longest read, which means that for practical data this method has to ignore all long reads.

The weighted variant of MEC was first suggested by Greenberg *et al.* [12]. Zhao *et al.* [34] propose a heuristic for a special case of wMEC and present experiments showing that wMEC is more accurate than MEC.

More recently, in 2012, Aguiar and Istrail [2,1] propose a different heuristic approach for MEC which they show to perform well compared to previous methods. Exact *integer linear programming (ILP)* based approaches were also proposed very recently by Fouilhoux and Mahjoub [11] and Chen *et al.* [6]. Both methods have difficulties solving practical instances optimally. For this reason, Chen *et al.* also propose a heuristic for solving difficult subproblems.

3 A Dynamic Programming Algorithm for wMEC

We now present the WhatsHap algorithm for solving wMEC. WhatsHap is an exact dynamic programming approach that solves wMEC instances in linear time if we assume bounded coverage.

Consider the input matrix $\mathcal{F}$ of the wMEC problem. Each entry $\mathcal{F}(i, j) \neq -$ is associated with a confidence degree $w(i, j)$ telling how likely it is that $\mathcal{F}(i, j)$ is correctly sequenced and that its fragment i is correctly mapped to location j. We use such values as a weight for the correction we need to minimize in the wMEC model. When these weights are log-likelihoods, summing them up corresponds to multiplying probabilities and, thus, finding a minimum weight solution corresponds to finding a maximum likelihood bipartition of the reads/fragments.

Our *dynamic programming (DP)* formulation is based on the observation that, for each column, only *active* fragments need to be considered; a fragment i is said to be active in every column j that lies in between its leftmost non-hole entry and its rightmost non-hole entry. Thus, paired-end reads remain active in the "internal segment" between the two reads. Let $F(j)$ be the set of fragments that are active at SNP position j and let F be the set of all fragments. The aim is to find a bipartition (R^*, S^*) of F such that the changes in R^* and S^* to make $\mathcal{F}$ conflict free have minimum total weight.

Proceeding columnwise from SNP position 1 to m, our approach computes a DP table column $C(j, \cdot)$ for each $j \in \{1, \ldots, m\}$. We say that a bipartition $B' = (R', S')$ of all fragments F *extends* bipartition $B = (R, S)$ of $F(j)$, if $R \subseteq R'$ and $S \subseteq S'$. We define $\mathcal{B}(X)$ to be the set of all bipartitions of X. Given a bipartition (R, S), we denote $\mathcal{B}\big(X \,|\, (R, S)\big)$ the set of all bipartitions of X that extend (R, S), that is,

$$\mathcal{B}\big(X \,\big|\, (R, S)\big) := \big\{(R', S') \in \mathcal{B}(X) \,\big|\, R \subseteq R' \text{ and } S \subseteq S'\big\} \ .$$

The basic idea of our dynamic program is as follows: for every bipartition $B = (R, S)$ of $F(j)$, entry $C(j, B)$ gives the minimum cost of a bipartition of all fragments F that renders positions $1, \ldots, j$ conflict free *and* which extends B. By definition of $C(j, B)$, the cost of an optimal solution to the wMEC problem then equals $\min_{B \in F(m)} C(m, B)$. An optimal bipartition of the fragments can be obtained by backtracking along the columns of the DP table up to the first SNP position in $\mathcal{F}$.

To compute the contribution $\Delta_C(j, (R, S))$ of column j to the cost $C\big(j, (R, S)\big)$ of bipartition (R, S), we define the following quantities.

Definition 1. *For a position j and a set R of fragment indices in $F(j)$, let $W^0(j, R)$ (resp. $W^1(j, R)$) denote the cost of setting position j on all fragments of R to 0 (resp. 1), flipping if required: i.e.,*

$$W^0(j, R) = \sum_{\substack{i \in R \\ \mathcal{F}(i,j)=1}} w(i, j) \quad \text{and} \quad W^1(j, R) = \sum_{\substack{i \in R \\ \mathcal{F}(i,j)=0}} w(i, j) \ .$$

Hence, given a bipartition (R, S) of $F(j)$, the minimum cost to make position j conflict free is

$$\Delta_C\big(j, (R, S)\big) := \min\{W^0(j, R), W^1(j, R)\} + \min\{W^0(j, S), W^1(j, S)\} \ .$$

Notice that, under the *all heterozygous assumption*, where one wants to enforce all SNPs to be heterozygous, the equation becomes

$$\Delta_C\big(j, (R, S)\big) := \min\{W^0(j, R) + W^1(j, S), W^1(j, R) + W^0(j, S)\} \ .$$

In both cases, we only need the four values $W^0(j, R)$, $W^1(j, R)$, $W^0(j, S)$, and $W^1(j, S)$ to compute $\Delta_C\big(j, (R, S)\big)$. We now proceed to state in detail our DP formulation.

	1	2	
0	0 5	-	...
1	1 3	0 2	...
2	1 6	1 1	...
3	-	0 2	...

Fig. 1. WHATSHAP toy example. The small numbers next to the matrix of entries $\mathcal{F}$ denote the flipping weights.

Initialization. The first column $C(1,\cdot)$ of C is initialized to $\Delta_C(1,\cdot)$ as defined above.

Example. Assume $F(1) = \{f_0, f_1, f_2\}$ with $\mathcal{F}(0,1) = 0, \mathcal{F}(1,1) = 1$, and $\mathcal{F}(2,1) = 1$. Moreover, let $w(0,1) = 5$, $w(1,1) = 3$, and $w(2,1) = 6$. See Figure 1. Then $C(1,\cdot)$ is filled in as follows:

$$C\big(1,(\{f_0,f_1,f_2\},\emptyset)\big) = \min\{9,5\} + \min\{0,0\} = 5$$
$$C\big(1,(\{f_0,f_1\},\{f_2\})\big) = \min\{3,5\} + \min\{6,0\} = 3$$
$$C\big(1,(\{f_0,f_2\},\{f_1\})\big) = \min\{6,5\} + \min\{3,0\} = 5$$
$$C\big(1,(\{f_1,f_2\},\{f_0\})\big) = \min\{9,0\} + \min\{0,5\} = 0$$

Note that we need consider only half of the $2^{|F(1)|}$ bipartitions, because $C\big(j,(R,S)\big) = C\big(j,(S,R)\big)$ for every bipartition $B = (R,S)$ and every SNP position j.

Recurrence. We compute $C(j+1,\cdot)$ from $C(j,\cdot)$ as follows. When computing costs of bipartitions for $F(j+1)$ we need only to keep track of the effect that this has on the bipartition of $F(j)$ through their intersection, which we denote by $F^{\cap}_{j+1} = F(j) \cap F(j+1)$. For a bipartition (R,S) of $F(j+1)$ we define $R^{\cap}_{j+1} = R \cap F^{\cap}_{j+1}$ and $S^{\cap}_{j+1} = S \cap F^{\cap}_{j+1}$. The recursion then becomes:

$$C\big(j+1,(R,S)\big) = \Delta_C\big(j+1,(R,S)\big) + \min_{B \in \mathcal{B}(F(j)) \,|\, (R^{\cap}_{j+1}, S^{\cap}_{j+1}))} C(j,B) \ . \qquad (1)$$

The first term accounts for the cost of the current SNP position, while the second term accounts for costs incurred at previous SNP positions. The minimum selects the best score with respect to the first j positions over all partitions that extend (R,S).

Example (continued). We extend the example with a second SNP position. Assume $F(2) = \{f_1, f_2, f_3\}$ with $\mathcal{F}(1,2) = 0$, $\mathcal{F}(2,2) = 1$, and $\mathcal{F}(3,1) = 0$. Moreover, let $w(1,2) = 2$, $w(2,2) = 1$, and $w(3,1) = 2$. See Figure 1. Then $C(2,\cdot)$ is filled in as follows:

$$
\begin{aligned}
C\big(2,(\{f_1,f_2,f_3\},\emptyset)\big) &= \min\{4,1\}+\min\{0,0\}+ \\
&\quad \min\big\{C\big(1,(\{f_0,f_1,f_2\},\emptyset)\big), C\big(1,(\{f_1,f_2\},\{f_0\})\big)\big\} \\
&= 4+0+\min\{5,0\} = 4 \\
C\big(2,(\{f_1,f_2\},\{f_3\})\big) &= \min\{1,2\}+\min\{0,2\}+ \\
&\quad \min\big\{C\big(1,(\{f_0,f_1,f_2\},\emptyset)\big), C\big(1,(\{f_1,f_2\},\{f_0\})\big)\big\} \\
&= 1+0+\min\{5,0\} = 1 \\
C\big(2,(\{f_1,f_3\},\{f_2\})\big) &= \min\{0,4\}+\min\{1,0\}+\min\big\{C\big(1,(\{f_0,f_1\},\{f_2\})\big)\big\} \\
&= 0+0+3 = 3 \\
C\big(2,(\{f_2,f_3\},\{f_1\})\big) &= \min\{1,2\}+\min\{0,2\}+\min\big\{C\big(1,(\{f_0,f_2\},\{f_1\})\big)\big\} \\
&= 1+0+5 = 6
\end{aligned}
$$

Algorithm Engineering. To compute a column, say j, of the DP table, we have to go through all bipartitions of the active fragments $F(j) = \{f_0, \ldots, f_{|F(j)|-1}\}$ at SNP position j. Because of the observed symmetry it is sufficient to store $2^{|F(j)|-1}$ entries in column j. We order these entries by a mapping of indices $k \in \{0, \ldots, 2^{|F(j)|-1}-1\}$ to bipartitions, using a binary encoding such that each bit k_ℓ in the binary representation of k tells whether fragment f_ℓ is in the first or in the second part of the bipartition. We break the above mentioned symmetry by assigning $f_{|F(j)|-1}$ always to the first set. Formally, this results in the mapping:

$$B : k \mapsto \big(\{f_{|F(j)|-1}\} \cup \{f_\ell \,|\, k_\ell = 0\}, \{f_\ell \,|\, k_\ell = 1\} \,|\, \ell < |F(j)|-1\big)$$

for all $k \in \{0,1\}^{|F(j)|-1}$.

Example. Assume there is a SNP position j for which $F(j) = \{f_0, f_1, f_2\}$. Then $k \in \{0,1,2,3\}$ and thus $C(p,\cdot)$ has four entries each one being encoded in two bits as follows. $00 \mapsto \big(\{f_0,f_1,f_2\},\emptyset\big)$, $01 \mapsto \big(\{f_0,f_2\},\{f_1\}\big)$, $11 \mapsto \big(\{f_2\},\{f_0,f_1\}\big)$, $10 \mapsto \big(\{f_1,f_2\},\{f_0\}\big)$. Notice that $f_{|F(p)|-1} = f_2$, as a sort of *pivot*, is always in the first part of the bipartition.

For an efficient computation of $\Delta_C\big(j, B_j(k)\big)$, we enumerate all bipartitions $k \in \{0, \ldots, 2^{|F(j)|-1}-1\}$ in *Gray code* order. This ensures that at most one bit is flipped between two consecutive bipartitions. Therefore, in moving from one bipartition to the next, only one fragment swaps sides and updating the four values $W^0(j,R)$, $W^1(j,R)$, $W^0(j,S)$, and $W^1(j,S)$ can be done in constant time. As $\Delta_C\big(j,(R,S)\big)$ can be computed from these values in constant time, and moving from one Gray code to the next can be done in (amortized) constant time using the algorithm from [24], we conclude that $\Delta_C\big(j,\cdot\big)$ can be computed in $O(2^{\mathrm{cov}(j)-1})$ time, where $\mathrm{cov}(j) = |F(j)|$ denotes the physical coverage at SNP position j.

To efficiently implement the DP recursion, one can compute an *intermediate projection column* as follows. For all $B \in \mathcal{B}(F^{\cap}_{j+1})$, store

$$\overline{C}\big(j,B\big) = \min_{B' \in \mathcal{B}(F(j)\,|\,B)} C(j,B') \ .$$

Table $\overline{C}(j,\cdot)$ can be filled while computing $C(j,\cdot)$ without any additional (asymptotic) runtime expense. Using this precomputed table, Recursion (1) can be written as

$$C\big(j+1,(R,S)\big) = \Delta_C\big(j+1,(R,S)\big) + \overline{C}\big(j,(R^{\cap}_{j+1},S^{\cap}_{j+1})\big) \ .$$

The algorithm has a runtime of $O(2^{k-1}m)$, where k is the maximum value of $\mathrm{cov}(\cdot)$, and m is the number of SNP positions. Note that the runtime is independent of read length.

An optimal bipartition can be obtained by backtracking. To do this efficiently, we store tables $\overline{D}(j,\cdot)$ that store the indices of the partitions that define the minima in $\overline{C}(j,\cdot)$. Formally,

$$\overline{D}\big(j,B\big) = \operatorname*{argmin}_{B'\in\mathcal{B}(F(j)\,|\,B)} C(j,B') \ .$$

Using these auxiliary tables, the sets of fragments that are assigned to each allele can be reconstructed in $O(km)$ time. To backtrace an optimal bipartition, we need to store the rightmost DP column $C(m,\cdot)$ and the backtracking tables $\overline{D}(j,B)$ for $j\in\{1,\ldots,m-1\}$, which takes total space $O(2^{k-1}m)$. This leads to a dramatically reduced memory footprint in practice compared to storing the whole DP table C.

Backtracking gives us optimal fragment bipartitions (R^*_j,S^*_j) for each position j. It is then straightforward to derive the two haplotypes h_1 and h_2 from this as follows:

$$h_1(j) = \begin{cases} 0 & \text{if } W^0(j,R^*_j) < W^1(j,R^*_j) \\ 1 & \text{otherwise} \ , \end{cases} \quad \text{and}$$

$$h_2(j) = \begin{cases} 0 & \text{if } W^0(j,S^*_j) < W^1(j,S^*_j) \\ 1 & \text{otherwise} \ . \end{cases}$$

4 Experimental Results

The focus of the present paper is on very long reads and the promise they hold for read-based phasing. Since such data sets are not available today, we perform a simulation study. We use all variants, that is SNPs, deletions, insertions, and inversions, reported by [19] to be present in Venter's genome. These variants were introduced into the reference genome (hg18) to create a reconstructed diploid human genome with fully known variants and phasings. Using the read simulator SimSeq [10], we simulated a variety of data sets, that reflect current technology as well as possible future developments. Regarding the former, we used HiSeq and MiSeq error profiles to generate a 2x100 bp and a 2x250 bp paired-end data set, respectively. The distribution of the internal segment size (i.e., fragment size minus size of read ends) was chosen to be 100 bp and 250 bp, respectively, which reflects current library preparation protocols. Furthermore, we created an

additional MiSeq data set with 2.5 kbp internal segment size resembling mate-pair sequencing. Longer reads with 1 000 bp, 5 000 bp, 10 000 bp, and 50 000 bp were simulated with two different uniform error rates of 1 % and 5 %. All data sets were created to have 30x average coverage and were mapped to the human genome using BWA MEM [20].

To not confound results by considering positions of (possibly) wrongly called SNPs, we always used the set of true positions of heterozygous SNPs that were introduced into the genome. We extracted all reads that covered at least two such SNP positions to be used for phasing. Next, we pruned the data sets to target coverages of 5x, 10x, 15x, and 20x by removing (randomly selected) reads that violated the coverage constraints until no more such reads exist. The resulting problem instances were then solved to optimality using WhatsHap, the DP algorithm described above.

To our knowledge, no other methods exist that can solve instances of wMEC with very long reads to optimality in practice. The DP approach of He *et al.* [16] has a worst-case complexity linear in 2^r where r is the length of the longest read (in terms of the number of SNPs covered). For coverage pruned at 15x and read length 5 000, r equals 30. For read length 50 000, r reaches a value of 147, which is clearly too large to run He *et al.*'s approach. In the ILP approach of Chen *et al.* [6], the key to solving MEC to optimality is to decompose the problem into independent blocks. Such a decomposition becomes less and less possible for longer reads, thus rendering such an approach infeasible for very long reads. After submission of this article, we found the similar yet independently developed DP approach of [9] that, however, addresses only the unweighted MEC. Moreover, due to our careful algorithm engineering, we have a lower asymptotic run-time, and can practically manage coverage up to 20x rather than 12x. A detailed performance analysis against all of these tools will appear in the full version of this work.

Our approach solved any problem instance with 15x coverage or below in less than 10 minutes on a single core (of an Intel Xeon E5-2620 CPU). For coverage 20x no problem instance took longer than 2.5 hours. The accuracy performance is summarized in Figure 2. There, the percentage of chromosome 1 that could be phased (y-axis) is plotted against the percentage of errors in the predicted haplotypes (x-axis) for different read lengths and coverages. A SNP position is *unphasable* if it is not covered by any read that also covered another SNP. Furthermore, we report an unphasable position whenever one of the two haplotypes contains no read at that position. Among those positions that *are* phasable of the reported haplotypes, we compute the number of errors, which is the sum of *zygosity errors* and *switch errors*, by comparing to the true haplotypes. A zygosity error occurs when a position is reported to be homozygous when it is truly heterozygous (and vice versa). A switch operation at position t on a binary string s is defined to result in the binary string $s[1 \ldots t]\bar{s}[t+1 \ldots |s|]$, where the $\bar{\cdot}$ operation flips all bits in a binary string. The switch error is now defined as the minimum number of such operations needed to transform the predicted haplotype (after all positions with zygosity errors have been removed)

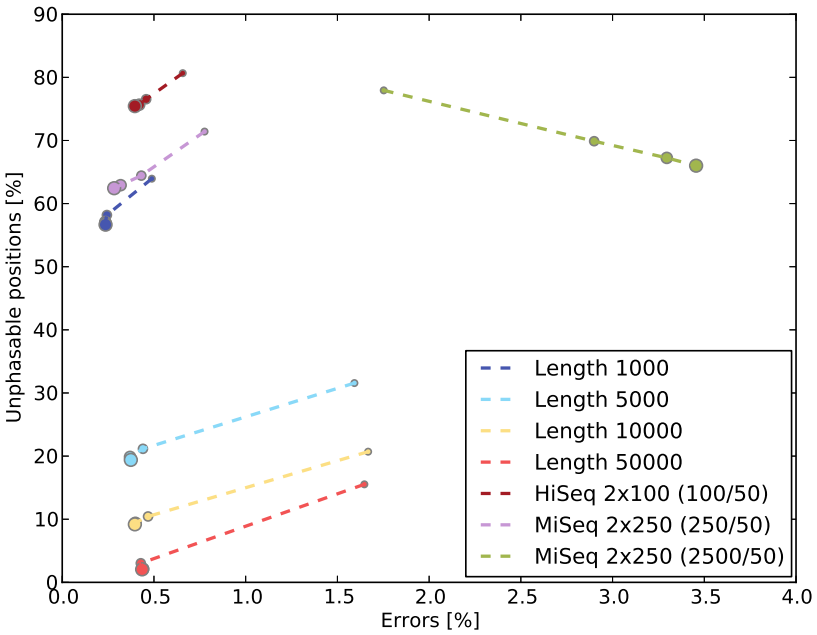

Fig. 2. Performance of phasing human chromosome 1 with 68 184 heterozygous SNPs in total using different simulated data sets and different coverages. The *unphasable positions* percentage (y-axis) gives the fraction of the SNP positions that could not be phased due to not being covered by reads that span more than one SNP position. Length 1 000, 5 000, 10 000, and 50 000 refer to reads of this length from a hypothetical sequencer with an error rate of 1%. HiSeq/MiSeq refers to using error profiles specific to these instruments during read sampling; in parentheses: mean/standard deviation of internal segment size (i.e., fragment size minus length of read ends). Data sets are pruned to four different target coverages (5x, 10x, 15x, 20x) encoded by circle diameter in the plot (larger means more coverage).

into the true haplotype. Note that the number of switch errors can increase when the number of unphasable positions goes down (see MiSeq experiments in Figure 2, for instance), as less gaps mean more contiguous fragments where such errors can be made.

Figure 2 clearly shows that long reads will indeed facilitate read-based phasing. For short reads of current HiSeq or MiSeq instruments, large portions of chromosome 1 cannot be phased. This was to be expected since short reads cannot span SNP deserts and, in general, rarely contain many SNPs. For the HiSeq data set, we found a paired-end read to cover only 2.2 SNPs on average. For the long read data sets (i.e., non-HiSeq/MiSeq) shown in Figure 2, an error rate of 1% was used. Repeating the experiments with an error rate of 5% yields nearly identical results. This exemplifies that errors are indeed corrected by solving the wMEC problem on sufficiently long reads.

Interestingly, the importance of a high coverage seems to be limited, especially for long reads. For 10x, 15x, and 20x, the corresponding circles in Figure 2 are

often close together. On the other hand, the influence of read length was much more drastic, highlighting that our approach, which is able to handle long reads but only limited coverage, is well suited for future data sets that allow for read based phasing.

5 Conclusions and Further Work

We have presented WhatsHap, a dynamic programming approach for haplotype assembly. WhatsHap is the first exact approach for the wMEC problem with runtime linear in the number of SNPs. WhatsHap is thus ready to benefit from increasing read lengths, which will boost the quality of haplotype assembly-based predictions.

While our approach handles datasets with possibly long reads, it can only deal with limited coverage. Although WhatsHap can handle coverage as large as 20x on a standard workstation, and larger coverage does not seem to significantly improve the quality of the predicted haplotypes as shown in our simulation study, a number of possible ways to cope with higher coverage are under investigation. A first possibility is a divide and conquer heuristic approach that operates on high coverage portions of the matrix by (i) (randomly/suitably) splitting the fragments into as many subsets as necessary to make each one of them a *slice* of limited coverage, (ii) solving each slice separately using the dynamic programming approach, and finally (iii) merging the resulting *super-reads* and applying iteratively the method again. Another possibility is to just properly select reads up to the manageable coverage and to discard the rest.

In the literature there are several graph representations of haplotype data (the *fragment conflict graph* defined in [18] and many of its variants), and consequently the optimization problems we have mentioned are seen there as finding the minimum number of graph editing operations that make the graph bipartite. In particular, for the conflict graph variant used in [11], the MEC problem turns out to be equivalent to finding the *Maximum Induced Bipartite Subgraph (MIBS)*. It follows that our dynamic programming approach for MEC can be generalized to a FPT approach for MIBS where the parameter is the pathwidth of the graph.

In this work we have concentrated on assembling SNP haplotypes from reads of a sequenced genome. As a next step we will integrate predictions from statistical phasers into our approach. In some sense, the *super-read* obtained from a *slice*, mentioned above, can be viewed as a reference haplotype from a *reference panel* for an existing population. Hence, reference haplotypes can be seamlessly integrated into this merging step (iii) for a hybrid approach. Hybrid methods are the future of sequencing data analysis, and the field is already moving quickly in this direction [8,15,14,27,32,33].

In addition, haplotyping mostly refers to only SNPs for historical reasons [29,30]. To fully characterize an individual genome, however, haplotyping must produce exhaustive lists of both SNPs and non-SNPs, that is, larger variants. This has become an essential ingredient of many human whole-genome projects

[5,28]. In this paper we focused on SNPs variants and we identify the integration of non-SNP-variants as a challenging future research direction.

Lastly, we used prior knowledge of the true SNP positions in the genome in our simulation study. But since our method only scales linearly in the number of SNP positions, one could conceivably use as input the raw read input, to produce a "de novo" haplotype. Since SNPs comprise roughly 5% of positions, and the runtime of our method is on the order of 10 minutes on average (for sufficient 15x coverage), such a de novo haplotype could be generated in about 3 hours. The heterozygous sites of this constructed haplotype then correspond to the SNP positions. It hence follows that this tool could be used for SNP discovery, and perhaps for larger variants as well.

Acknowledgments. Murray Patterson was funded by a Marie Curie ABCDE Fellowship of ERCIM. Leo van Iersel was funded by a Veni, and Alexander Schönhuth was funded by a Vidi grant of the Netherlands Organisation for Scientific Research (NWO).

References

1. Aguiar, D., Istrail, S.: Hapcompass: A fast cycle basis algorithm for accurate haplotype assembly of sequence data. J. of Comp. Biol. 19(6), 577–590 (2012)
2. Aguiar, D., Istrail, S.: Haplotype assembly in polyploid genomes and identical by descent shared tracts. Bioinformatics 360, i352–i360 (2013)
3. Bansal, V., Bafna, V.: HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. Bioinformatics 24(16), i153–i159 (2008)
4. Bansal, V., et al.: An MCMC algorithm for haplotype assembly from whole-genome sequence data. Genome Research 18(8), 1336–1346 (2008)
5. Boomsma, D.I., et al.: The Genome of the Netherlands: design, and project goals. European Journal of Human Genetics (2013), doi:10.1038/ejhg.2013.118
6. Chen, Z.Z., Deng, F., Wang, L.: Exact algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics 29(16), 1938–1945 (2013)
7. Cilibrasi, R., van Iersel, L., Kelk, S., Tromp, J.: On the complexity of several haplotyping problems. In: Casadio, R., Myers, G. (eds.) WABI 2005. LNCS (LNBI), vol. 3692, pp. 128–139. Springer, Heidelberg (2005)
8. Delaneau, O., Howie, B., Cox, A., Zagury, J., Marchini, J.: Haplotype estimation using sequencing reads. Am. J. of Human Genetics 93(4), 687–696 (2013)
9. Deng, F., Cui, W., Wang, L.: A highly accurate heuristic algorithm for the haplotype assembly problem. BMC Genomics 14(suppl. 2), S2 (2013)
10. Earl, D.A., et al.: Assemblathon 1: A competitive assessment of de novo short read assembly methods. Genome Research (2011), doi:10.1101/gr.126599.111
11. Fouilhoux, P., Mahjoub, A.: Solving VLSI design and DNA sequencing problems using bipartization of graphs. Comp. Optim. and Appl. 51(2), 749–781 (2012)
12. Greenberg, H., Hart, W., Lancia, G.: Opportunities for combinatorial optimization in computational biology. Informs J. on Computing 16(3), 211–231 (2004)
13. Hartl, D., Clark, A.: Principles of Population Genetics. Sinauer Associates, Inc., Sunderland (2007)
14. He, D., Eskin, E.: Hap-seqX: expedite algorithm for haplotype phasing with imputataion using sequence data. Gene. 518(1), 2–6 (2013)

15. He, D., Han, B., Eskin, E.: Hap-seq: an optimal algorithm for haplotype phasing with imputation using sequencing data. J. Comp. Biol. 20(2), 80–92 (2013)
16. He, D., et al.: Optimal algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics 26(12), i183–i190 (2010)
17. Howie, B., Donnelly, P., Marchini, J.: A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. PLoS Genetics 5(6), e1000529 (2009)
18. Lancia, G., Bafna, V., Istrail, S., Lippert, R., Schwartz, R.: SNPs problems, complexity and algorithms. In: Meyer auf der Heide, F. (ed.) ESA 2001. LNCS, vol. 2161, pp. 182–193. Springer, Heidelberg (2001)
19. Levy, S., et al.: The diploid genome sequence of an individual human. PLoS Bio. (2007), doi:10.1371/journal.pbio.0050254
20. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv (1303.3997) (2013)
21. Li, Y., et al.: MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes. Genet. Epidemiol. 34, 816–834 (2010)
22. Lippert, R., et al.: Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. Briefings in Bioinformatics 3(1), 23–31 (2002)
23. Menelaou, A., Marchini, J.: Genotype calling and phasing using next-generation sequencing reads and a haplotype scaffold. Bioinformatics 29(1), 84–91 (2013)
24. Mossige, S.: An algorithm for Gray codes. Computing 18, 89–92 (1977)
25. Panconesi, A., Sozio, M.: Fast hare: A fast heuristic for single individual SNP haplotype reconstruction. In: Jonassen, I., Kim, J. (eds.) WABI 2004. LNCS (LNBI), vol. 3240, pp. 266–277. Springer, Heidelberg (2004)
26. 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, 629–644 (2006)
27. Selvaraj, S., et al.: Whole-genome haplotype reconstruction using proximity-ligation and shotgun sequencing. Nature Biotechnology 31, 1111–1118 (2013)
28. The 1000 Genomes Project Consortium: A map of human genome variation from population-scale sequencing. Nature 467(7319), 1061–1073 (2010)
29. The International HapMap Consortium: A second generation human haplotype map of over 3.1 million SNPs. Nature 449, 851–861 (2007)
30. The International HapMap Consortium: Integrating common and rare genetic variation in diverse human populations. Nature 467, 52–58 (2010)
31. Wang, R.S., Wu, L.Y., Li, Z.P., Zhang, X.S.: Haplotype reconstruction from SNP fragments by minimum error correction. Bioinformatics 21(10), 2456–2462 (2005)
32. Yang, W.Y., Hormozdiari, F., Wang, Z., He, D., Pasaniuc, B., Eskin, E.: Leveraging reads that span multiple single nucleotide polymorphisms for haplotype inference from sequencing data. Bioinformatics 29(18), 2245–2252 (2013)
33. Zhang, Y.: A dynamic Bayesian Markov model for phasing and characterizing haplotypes in next-generation sequencing. Bioinformatics 29(7), 878–885 (2013)
34. Zhao, Y.T., Wu, L.Y., Zhang, J.H., Wang, R.S., Zhang, X.S.: Haplotype assembly from aligned weighted SNP fragments. Computational Biology and Chemistry 29, 281–287 (2005)