Title
Optimal algorithms for haplotype assembly from whole-genome sequence data.

Permalink
https://escholarship.org/uc/item/77t900hh

Journal
Bioinformatics (Oxford, England), 26(12)

ISSN
1367-4803

Authors
He, Dan
Choi, Arthur
Pipatsrisawat, Knot
et al.

Publication Date
2010-06-01

DOI
10.1093/bioinformatics/btq215

Peer reviewed
Optimal algorithms for haplotype assembly from whole-genome sequence data

Dan He*, Arthur Choi, Knot Pipatsrisawat, Adnan Darwiche, Eleazar Eskin
Department of Computer Science, University of California Los Angeles, Los Angeles, CA 90095, USA

ABSTRACT

Motivation: Haplotype inference is an important step for many types of analyses of genetic variation in the human genome. Traditional approaches for obtaining haplotypes involve collecting genotype information from a population of individuals and then applying a haplotype inference algorithm. The development of high-throughput sequencing technologies allows for an alternative strategy to obtain haplotypes by combining sequence fragments. The problem of ‘haplotype assembly’ is the problem of assembling the two haplotypes for a chromosome given the collection of such fragments, or reads, and their locations in the haplotypes, which are pre-determined by mapping the reads to a reference genome. Errors in reads significantly increase the difficulty of the problem and it has been shown that the problem is NP-hard even for reads of length 2. Existing greedy and stochastic algorithms are not guaranteed to find the optimal solutions for the haplotype assembly problem.

Results: In this article, we proposed a dynamic programming algorithm that is able to assemble the haplotypes optimally with time complexity \(O(m \times 2^k \times n)\), where \(m\) is the number of reads, \(k\) is the length of the longest read and \(n\) is the total number of SNPs in the haplotypes. We also reduce the haplotype assembly problem into the maximum satisfiability problem that can often be solved optimally even when \(k\) is large. Taking advantage of the efficiency of our algorithm, we perform simulation experiments demonstrating that the assembly of haplotypes using reads of length typical of the current sequencing technologies is not practical. However, we demonstrate that the combination of this approach and the traditional haplotype phasing approaches allow us to practically construct haplotypes containing both common and rare variants.

Contact: danhe@ics.ucla.edu

1 INTRODUCTION

Obtaining haplotypes, or the sequence of alleles on each chromosome, is an important step for many types of analyses of genetic variation in the human genome. In particular, haplotype inference is required for the application of many imputation algorithms (Marchini et al., 2007), which are now widely applied in the analysis of genome-wide association studies.

The standard approach for obtaining haplotype information involves collecting genotype data from a population of individuals. Genotype data contains information on the set of alleles at each locus, but lacks information on which chromosome a particular allele occurs on. Computational methods are then applied to these genotype data to infer the haplotypes (Browning and Browning, 2008; Halperin and Eskin, 2004; Stephens et al., 2001). These methods take advantage of the fact that alleles at neighboring loci in the genomes are correlated or are ‘in linkage disequilibrium’ (LD), as well as the fact that in any given region only a few common haplotypes account for the majority of the genetic variations in the population. Due to their reliance on LD, these methods have difficulty inferring haplotypes with rare variants and have no ability to infer haplotypes for alleles that are unique to an individual.

Recently, the development of high-throughput sequencing (HTS) technology has enabled an alternative strategy to obtain haplotypes. Since each sequence read is from a single chromosome, if a read covers two variant sites, all of the alleles present in the read must be from the same haplotype. Using this insight, it is possible to assemble the two haplotypes for a chromosome from the collection of such reads by joining reads that share alleles at common variants. The problem is referred to as ‘haplotype assembly’ (Lancia et al., 2001), which is challenging in the following two aspects:

- Reads are sampled from either of the two haplotypes and no information is given about which one they come from. The reads need to be separated for the two haplotypes in the assembly process.
- Errors in reads significantly increase the difficulty of the problem and it has been shown that the problem is NP-hard even for reads of length 2 (Cilibrasi et al., 2005; Lancia et al., 2001).

A simple greedy heuristic method (Levy et al., 2007) (which we call the Greedy algorithm), concatenates the reads with minimum conflicts and is fast but not very accurate when reads contain errors. Other stochastic algorithms, such as HASH (Bansal et al., 2008), which is a Markov chain Monte Carlo (MCMC) algorithm, and HapCut (Bansal and Bafna, 2008), which is a combinatorial approach, have been shown to be much more accurate than the Greedy algorithm on the HuRef diploid genome sequence (Levy et al., 2007).

However, both HASH and HapCut algorithms use stochastic strategies and therefore are not guaranteed to find optimal solutions for the haplotype assembly problem. In this article, we propose a dynamic programming algorithm that is able to assemble the haplotypes optimally with time complexity \(O(m \times 2^k \times n)\), where \(m\) is the number of reads, \(k\) is the length of the longest read and \(n\) is the total number of heterozygous sites in the haplotypes. Since this time complexity is exponential in \(k\), we reduce the problem to the maximum satisfiability (MaxSAT) problem for cases where \(k\) is large. MaxSAT conversion is a well-known strategy for many computational biology problem such as SNP Tagging (Choi et al., 2008). The converted MaxSAT problem can often be solved optimally in a reasonable amount of time with an MaxSAT solver. Our experiments show that the MaxSAT approach can solve 99.98% instances of the converted haplotype assembly problem optimally. We also show for the first time that the current best-known solution...
is only 1.1% from the optimal solution and our solution is the best result that has yet been achieved.

Taking advantage of the efficiency and optimality of our method, we are able to perform simulation experiments to evaluate the feasibility of assembling haplotypes using sequence reads with the length typical of the current high-throughput technologies. The current sequencing technologies are able to collect paired-end reads, where sequences of two segments are obtained separated by an approximate distance (insert length). Our experiments show that the insert length and in particular the variability in the insert length play a crucial role in our ability to assemble haplotypes. Using data from HapMap (International HapMap Consortium, 2007) we demonstrate that using current HTS technologies, the assembly of reads into haplotypes is impractical. However, we show that combining haplotype assembly from sequencing with traditional approaches for inferring haplotypes using genotypes can effectively recover haplotypes for both common and rare alleles.

2 RELATED WORK

The haplotype assembly problem was first introduced by Lancia et al. (2001). They show that the problem is computationally challenging when reads contain errors, since the reads can not be partitioned perfectly into two disjoint sets. Therefore, various combinatorial objective functions have been proposed (Lancia et al., 2001; Liptert et al., 2002) to define the best reconstruction of haplotypes such as minimum fragment removal (MFR), minimum error correction (MEC), minimum SNP removal (MSR), minimum implicit SNP removal (MISR), minimum implicit fragment removal (MIFR). Out of these objective functions, MEC, which is the number of conflicts between the sequence reads and the constructed haplotypes, is the most difficult one to optimize. The haplotype assembly problem with MEC as the objective function is NP-hard even for gapless reads of length 2, while polynomial algorithms exist for solving the problem with MFR and MSR as the objective function (Ciliberti et al., 2005; Liptert et al., 2002). Several heuristic and stochastic methods (Bansal and Bafna, 2008; Bansal et al., 2008; Levy et al., 2007; Panconesi and Sozio, 2004; Wang et al., 2005) have been proposed to optimize MEC for gapped reads. In this article we also focus on minimizing MEC. Therefore, the 'haplotype assembly problem' can be defined as following: given a set of reads that may contain errors, reconstruct the pair of haplotypes by partitioning the reads to either haplotype such that the MEC is minimized.

The Greedy heuristic algorithm (Levy et al., 2007), which concatenates the reads with minimum conflicts, is able to construct optimal haplotypes very quickly if the reads are error-free. When there are errors in the reads, the Greedy algorithm usually outputs much worse results than the optimal solution. HASH (Bansal and Bafna, 2008) and HapCut (Bansal and Bafna, 2008) algorithms are both based on the idea of building a graph from the read matrix, where each row corresponds to a read and each column corresponds to a position of the haplotype. In the graph, each column is a node and an edge between two nodes is created if there is a read spanning the corresponding two columns. The weights of the edges are determined by the number of reads that are consistent with the haplotypes minus the number of reads that are in conflict with the haplotypes in the two columns. The HASH algorithm uses graph cut computations to construct the Markov chain used for sampling the haplotype space. HapCut uses Max-Cut computations in an associated graph to greedily move towards the optimal MEC solution. Both HASH and HapCut algorithms obtain much more accurate haplotypes than the Greedy algorithm. Since convergence of Markov chain is slow, HapCut is much faster than HASH with almost the same accuracy.

Although HASH and HapCut achieve reasonably good results, they are stochastic and therefore can not guarantee optimal solutions. In this article we propose a dynamic programming algorithm that is able to find the optimal solution with time complexity $O(n \times 2^k \times n)$ for gapless reads, where $m$ is the number of reads, $k$ is the length of the longest read and $n$ is the total number of SNPs in the haplotypes.

The time complexity implies that this algorithm is most effective when $k$ is relatively small. Since only heterozygous sites in the reads are considered for assembly, and the number of heterozygous sites, or variants, is small, and these sites are often far from one another, most of the reads only cover a small number of heterozygous sites.

The dynamic programming algorithm is computationally practical for up to reads of length 15, where length is defined as the number of heterozygous sites covered by the read. We later show that in the HaRef data, >90% of the reads are of length less than 15, which indicates the dynamic programming algorithm is practical for the majority of the data. In our experiments, the run time of the dynamic programming algorithm we propose here is comparable to that of the HapCUT algorithm. Furthermore, to handle reads of length greater than 15, we propose to take advantage of recent advances in the field of logical reasoning, by modeling the haplotype problem as an MaxSAT problem. We show that modern MaxSAT solvers are powerful enough to solve such haplotype problems optimally in practice.

3 METHODS

3.1 The Haplotype assembly problem

We will follow the notation by Bansal and Bafna (2008) for the haplotype assembly problem. Given a reference genome sequence and the set of reads containing sequence from both chromosomes, after aligning all the reads to the reference genome (Li et al., 2008), the homozygous sites (columns in the alignment with identical values) are discarded since they are not informative. The heterozygous sites (columns in the alignment with different values) correspond to alleles that differ between chromosomes and they are labeled as 0 or 1 arbitrarily. A matrix $X$ of size $m \times n$ can be built from the alignment, where $m$ is the number of reads and $n$ is the number of heterozygous sites. The $i$-th read is described as a ternary string $X_i$ $\in \{0, 1, \sim\}$, where $\sim$ indicates a gap, namely that the allele is not covered by the fragment (again following the notation of Bansal and Bafna (2008) for clarity). The start position and end position of a read are the first and last positions in the corresponding row that are not $\sim$. Therefore, the $\sim$'s in the head and tail of each row will not be considered as part of the corresponding read. However, there can be $\sim$'s inside each read, which correspond to either missing data for single reads or gaps connecting a pair of single reads (called paired-end reads). Reads without $\sim$'s are called gapless reads; otherwise they are called gapped reads. Assuming a read's end position is $j$, start position is $i$, the length of the read is defined as $j-i+1$. We also assume all the reads have already been correctly aligned to the reference genome by some mapper, which may not be true since the mapper may introduce mapping errors and the reads may come from repeat-rich regions. However, the mapping process is out of the scope of this article and we thus do not evaluate the effects of mapping errors on the quality of our haplotype assembly solution.

The haplotypes can be represented as an unordered pair of binary strings $H = (h_1, h_2)$, each of length $n$. Since all the sites are heterozygous, $h_j$ is
Table 1. As we can see in this example, each read corresponds to one row where ‘-’ indicates missing information. Reads often contain errors. For example, if we only consider reads 1, 2, 3, we can partition them perfectly into two sets ({read1, read3}, {read2}) and reconstruct the haplotypes as \( H = (h_1 = [0000], h_2 = [11111]) \) by assigning reads 1 and 3 to \( h_1 \) and assigning read2 to \( h_2 \). However, reads is in conflict with this partition and there is no perfect partition for reads 1, 2, 3 and 4.

Therefore, in the presence of errors, we need to reconstruct the haplotypes such that some objective function is minimized. The objective function we use is MEC (minimum number of changes, or corrections, that need to be made in the read matrix such that the resulting matrix admits a perfect bi-partition, where each corrected read maps to either haplotype perfectly). Alternatively speaking, for any pair of complementary haplotypes, the set of reads can be partitioned into two subsets that satisfy the following property: if both subsets of the reads are mapped to one haplotype at their corresponding intervals indicated by their starting positions and lengths, where one subset is mapped to only one of the haplotypes, the number of errors or mismatches for the mapping is minimized. This minimum number of errors is the MEC score of the reads for the pair of haplotypes. In our example, if we only consider reads 1, 2, 3 and 4, we can change read4 from (101) to (000) such that now we can obtain a perfect bi-partition (\( \{\text{read1}, \text{read3}\}, \{\text{read2}\} \)) with reconstructed haplotypes \( (101) \) to \( (001) \) such that now we can obtain a perfect bi-partition (\( \{\text{read1}, \text{read4}\}, \{\text{read2}\} \)).

We take read 1–read 4 from Figure 1 as an example. We need to order them according to their start positions. (1) At position \( i \geq 0 \), we have read 1 and read 3, \( k \)-length(read 3) + 4.

Therefore, we compute \( s(i, r) \) for all length \( k = 4 \) binary strings \( r \), using only read 1 and read 3:

\[
00 \rightarrow 0000 = 0, 0000 = 0, 00000 = 0
\]

We also observe that we can split up the reads into sets where there is no read that spans any two sets. We call such a set a block. The set of reads can thus be partitioned into many blocks. Since no read spans any two blocks, which means those blocks are independent, we can reconstruct haplotypes for each block in parallel using the dynamic programming algorithm developed above and then concatenate the solutions for each block to construct the complete haplotypes.

Next, we show a simple example for the dynamic programming algorithm. We take read 1–read 4 from Figure 1 as an example. We need to order them according to their start positions.
min multiple individuals may be used to infer the most likely haplotypes for each 001) we get the optimal haplotype pair ({00001,11110}). Both the pairs
the above example, s(2, 000) = 1 and it is from s(0, 0000). By tracing back
not necessarily the case that there is only one pair of optimal haplotypes. In
information. These reads are called paired-end reads
usually span a long fragment, which can be as long as a few hundred
haplotypes is the number of heterozygous sites in each chromosome. So far, we only discussed a dynamic programming algorithm for single
3.3 MaxSAT conversion for haplotype assembly
Fig. 1. (a) The number of short reads, all reads and (b) the length of haplotypes for each chromosome. The threshold for short reads is 15. The length of haplotypes is the number of heterozygous sites in each chromosome.

Therefore, the optimal MEC is min(s(2, r)) for all length k = 3 binary strings
and the optimal MEC is 1.
Notice that when we trace back to obtain the optimal haplotypes, it is not
necessarily the case that there is only one pair of optimal haplotypes. In the
above example, s(2, 000) = 1 and it is from s(0, 0000). By tracing back
from s(2, 001) we get the optimal haplotype pair ((0000), (1111)). We also have s(2, 001) = 1 and it is from s(0, 0000). By tracing back from s(2, 001) we get the optimal haplotype pair ((0000), (1111)). Both the pairs have optimal MEC of 1. When there are multiple optimal solutions, data on
multiple individuals may be used to infer the most likely haplotypes for each
ambiguous individual.

3.3 MaxSAT conversion for haplotype assembly
So far, we only discussed a dynamic programming algorithm for single
reads. Consider reads 5, 6 and 10 in Figure 1. In each of these reads, there are two continuous strings connected by ‘-’, which indicates missing
information. These reads are called paired-end reads and are generated
by modern sequencing technologies. The problem becomes much more
complicated when paired end reads are considered since paired-end reads
usually span a long fragment, which can be as long as a few hundred
positions. Although paired-end read can be considered as a special case of a single read, the dynamic programming algorithm introduced above
becomes impractical, since we need to enumerate all positions the paired-
end read covers. As concluded above, the time complexity of the dynamic
programming algorithm is $O(n \times 2^{k_{\text{max}}})$, where $k_{\text{max}}$ is the maximum number of alleles contained among all reads. When paired-end reads are considered, $k_{\text{max}}$ could be as large as a few hundred, making the dynamic
programming approach impractical. Even single reads can be too long to
enumerate some of the positions. We set a threshold for $k_{\text{max}}$ such that
the enumeration of all length $k_{\text{max}}$ binary strings is computationally feasible.
We call the single reads and the paired-end reads of length greater than the
threshold as long reads and the other reads as short reads.

We solve the haplotype assembly problem when long reads are also
considered by conversion to MaxSAT. The MaxSAT problem is an
optimization version of the well-known Boolean satisfiability problem
(SAT) (Biere et al, 2009). Given a set of clauses (a clause is a disjunction of
Boolean literals), the MaxSAT problem asks for a complete assignment
of all variables that maximizes the number of clauses the assignment
satisfies. For example, consider the following set of four clauses:

\[(x_1 \lor \neg x_3), (x_2 \lor \neg x_2), (\neg x_2 \lor x_4), (\neg x_2 \lor x_3)\]

The assignment $x_1 = \text{false}$, $x_2 = \text{false}$, $x_3 = \text{false}$ satisfies three clauses and
is optimal. In this work, we consider a variant of MaxSAT known as partial
MaxSAT. Partial MaxSAT allows some clauses to be labeled as hard—i.e.
their satisfiability is mandatory in any solution. The objective of the problem
is to find an assignment that satisfies all hard clauses and satisfies the most
number of non-hard (i.e., soft) clauses. For more discussion on MaxSAT and
its variations, please see (Li and Manià, 2009).

In our conversion of the haplotype assembly problem to partial MaxSAT,
we define the following boolean variables:

- $h_i$, $0 \leq i < n$, represents the binary symbol at position $i$ in the haplotype
  to be constructed.
- $r_j$, $0 \leq j < m$, represents the assignment of read $j$ to a haplotype.
The value $r_j = 0$ indicates that read $j$ is assigned to the considered haplotype,
while the value $r_j = 1$ indicates that the read is assigned the complement
haplotype.
- $e_{ij}$, $0 \leq i < n, 0 \leq j < m$, represents whether a correction is needed for
  position $i$ of read $j$ with respect to the considered haplotype. The value
  $e_{ij} = 1$ indicates that a correction is needed, while the value $e_{ij} = 0$
  indicates that no correction is needed at that position.

Given these variables, we can define the set of clauses that describes the
relationship between $h_i$, $r_j$ and $e_{ij}$. These clauses essentially specify that
there is an error whenever the value at position $i$ of read $j$ does not match
with the value at position $i$ of the haplotype that the read is assigned to. Let
read[i][j] represent the value at position $i$ of read $j$ (i.e. the value of cell[i][j]
of the read matrix). We can formally define a set of clauses for each $n$ entry
in the read matrix as follows:

\[(h_i \lor \neg e_{ij}), \quad \text{if } \text{read[i][j]} = 0,\]
\[(h_i \land r_j \lor e_{ij}), \quad \text{if } \text{read[i][j]} = 1.\]

Note that $\lor$ is the logical equivalence operator and that $(x \lor y \lor z)$ is a
shorthand notation for the clauses $(x \lor y) \lor (\neg y \lor z), (\neg x \lor y \lor z), (x \lor y \lor z) , (x \lor y \lor z)$.
The global MEC score is the sum of the scores from each block. The average number of reads that span each heterozygous site is given by the number of short reads, all reads and the number of paired-end reads for each chromosome is very large. More than 90% of the reads are short reads. While previously developed haplotype assembly approaches have been applied to the HuRef data, it is not clear how applicable these approaches would be using current high-throughput genotyping technologies that have much shorter read lengths, yet are still tens of thousands of reads of length more than 15. For example, in the block shown in Figure 2, there are around 400 long reads and the maximal length of the reads is around 200. We run all three algorithms on short reads only and the results are shown in Table 2. As we can see, on average, HapCut improves the MEC score of Greedy by 30%, while our dynamic programming algorithm runs for 24 s. The run times are even closer for small size blocks and ~90% blocks are such small size blocks. Both algorithms run for more than 10 h and finish in roughly the same time on a computational cluster for all the 22 chromosomes.

4 RESULTS

4.1 HuRef experiments

We first examine the performance of the dynamic programming algorithm on the filtered HuRef data from Levy et al., 2007) overall 22 chromosomes and directly compare our method to previous approaches (Bansal and Bafna, 2008). The data consists of 32 million DNA fragments generated by Sanger sequencing and contains a total of 1.85 million heterozygous variants for the 22 chromosomes. We show the number of short reads, all reads and number of heterozygous sites for each chromosome, where the threshold for short reads is 15 as shown in Figure 1. As we can see, the number of reads for each chromosome is very large. More than 90% of the reads are short reads. Haplotypes for each chromosome are also very long, making the haplotype assembly problem computationally intensive. The average number of reads that span each heterozygous site is between six and seven.

The whole-genome sequence data consists of many disconnected blocks where no read spans the boundary of two blocks. Therefore, we can split the sequence data independently into many blocks and then solve the haplotype assembly problem for each block. The global MEC score is the sum of the scores from each block and the optimal haplotypes are the concatenation of the haplotypes from each block. We show the read matrix for the first block of chromosome 22 in Figure 2 as an example, where we have around 2300 reads spanning a block of length around 400.

4.1.1 HuRef experiments on only short reads. We first compare the results of the dynamic programming algorithm with the results of Greedy and HapCut, on short reads, namely single reads and the paired-end reads of length less than 15 only. As we showed in Figure 1, most of the reads are very short. However, there are still tens of thousands of reads of length more than 15. For example, in the block shown in Figure 2, there are around 400 long reads and the maximal length of the reads is around 200. We run all three algorithms on short reads only and the results are shown in Table 2. As we can see, on average, HapCut improves the MEC score of Greedy by 30%, while our dynamic programming algorithm shows for the first time that the solution from HapCut is only 1.1% from the optimal solution. The run time of the dynamic programming algorithm is reasonably fast and comparable to the HapCut algorithm. For example, for the first block of chromosome 22 in Figure 2, HapCut runs for 20 s while the dynamic programming algorithm runs for 24 s. The run times are even closer for small size blocks and ~90% blocks are such small size blocks. Both algorithms run for more than 10 h and finish in roughly the same time on a computational cluster for all the 22 chromosomes.

4.1.2 HuRef experiments on all reads. As mentioned in the previous section, in order to solve the haplotype assembly problem containing reads of all lengths, we convert the problem into a partial MaxSAT problem and use partial MaxSAT solvers to solve it. In this work, we consider two MaxSAT solvers: Clone (Pipatsrisawat et al., 2008) and WBO (Manquinho et al., 2009) to solve the resulting MaxSAT problems. We will report the experimental results in the next section.

4.2 Designing haplotype assembly protocols

While previously developed haplotype assembly approaches have been successfully applied to the HuRef data, it is not clear how applicable these approaches would be using current high-throughput genotyping technologies that have much shorter read lengths, yet...
higher coverage than the HuRef data. We take advantage of the efficiency of our algorithm to perform simulations in order to design sequencing protocols for current high-throughput technology in order to effectively obtain haplotypes. Unlike the reads for HuRef data, which are sequenced with the Sanger-based whole-genome shotgun sequencing and therefore are very long (each segment is thousands of base pairs long including both homozygous and heterozygous sites), here we consider the reads generated by the HTS technology (Wheeler et al., 2008). The reads generated by HTS are usually very short (each segment is around 30–100 bp including both homozygous and heterozygous sites).

The basic parameters of sequencing technology that we explore are the sequence coverage ratio (the number of times that each base pair in the sequence is covered), the insert length of the paired-end reads (the distance between the two segments of the paired-end reads), the variance of this insert length and the read length. We explore how these parameters affect the haplotype assembly. We perform our experiments over individual genotype data from HapMap (International HapMap Consortium, 2007). For a single individual, we concatenate the heterozygous SNPs to construct a true haplotype. For the individual we downloaded, there are 505,965 heterozygous SNPs. We then mimic the sequencing process by randomly generating paired-end reads with varying parameters including coverage ratio, insert length of the reads and standard deviation of the insert length. The insert length follows a Gaussian distribution with a mean of 1000. Assume the genome length is \( n \), the sequence read length is \( l \), the coverage ratio is \( c \), the number of reads to be generated then is \( n \times c \times l \). The starting positions of the reads are randomly selected within the range of the whole genome, therefore they may cover both heterozygous and homozygous SNPs. The segment length of the sequence paired-end read is 36 (including both heterozygous and homozygous sites), here we consider the reads generated by the HTS technology (Wheeler et al., 2008). The reads generated by HTS are usually very short (each segment is around 30–100 bp including both homozygous and heterozygous sites).

The basic parameters of sequencing technology that we explore are the sequence coverage ratio (the number of times that each base pair in the sequence is covered), the insert length of the paired-end reads (the distance between the two segments of the paired-end reads), the variance of this insert length and the read length. We explore how these parameters affect the haplotype assembly. We perform our experiments over individual genotype data from HapMap (International HapMap Consortium, 2007). For a single individual, we concatenate the heterozygous SNPs to construct a true haplotype. For the individual we downloaded, there are 505,965 heterozygous SNPs. We then mimic the sequencing process by randomly generating paired-end reads with varying parameters including coverage ratio, insert length of the reads and standard deviation of the insert length. The insert length follows a Gaussian distribution with a mean of 1000. Assume the genome length is \( n \), the sequence read length is \( l \), the coverage ratio is \( c \), the number of reads to be generated then is \( n \times c \times l \). The starting positions of the reads are randomly selected within the range of the whole genome, therefore they may cover both heterozygous and homozygous SNPs. The segment length of the sequence paired-end read is 36 (including both heterozygous and homozygous sites), here we consider the reads generated by the HTS technology (Wheeler et al., 2008). The reads generated by HTS are usually very short (each segment is around 30–100 bp including both homozygous and heterozygous sites).

| Chromosome | Greedy | HapCut | DP | Greedy | HapCut | MaxSAT |
|------------|--------|--------|----|--------|--------|--------|
| 1          | 21355  | 15312  | 15292 | 29518  | 19687  | 19584  |
| 2          | 16067  | 11251  | 11107 | 22706  | 14615  | 14576  |
| 3          | 11909  | 8223   | 8181  | 16696  | 10702  | 10647  |
| 4          | 12518  | 8820   | 8775  | 17509  | 10536  | 10528  |
| 5          | 11621  | 8017   | 7944  | 16432  | 10536  | 10528  |
| 6          | 10624  | 7487   | 7369  | 15295  | 9842   | 9826   |
| 7          | 11668  | 8531   | 8423  | 17188  | 11244  | 11187  |
| 8          | 10501  | 7343   | 7311  | 14535  | 9741   | 9025   |
| 9          | 10199  | 7350   | 7312  | 13512  | 9222   | 9201   |
| 10         | 10263  | 7313   | 7236  | 15172  | 9222   | 9201   |
| 11         | 8825   | 6224   | 6196  | 12667  | 8200   | 9778   |
| 12         | 8614   | 6377   | 6155  | 12433  | 8218   | 8176   |
| 13         | 6412   | 4396   | 4341  | 8848   | 5822   | 5761   |
| 14         | 6634   | 4567   | 4532  | 9070   | 5879   | 5845   |
| 15         | 9289   | 6653   | 6623  | 13291  | 9311   | 9285   |
| 16         | 8574   | 6160   | 6093  | 12365  | 8259   | 8207   |
| 17         | 7088   | 5034   | 4955  | 10195  | 6525   | 6459   |
| 18         | 4973   | 3526   | 3398  | 8324   | 4991   | 4943   |
| 19         | 5849   | 3996   | 3907  | 7939   | 5319   | 5288   |
| 20         | 4136   | 2909   | 2891  | 5563   | 3739   | 3723   |
| 21         | 3877   | 2903   | 2796  | 5607   | 3888   | 3881   |
| 22         | 4424   | 3267   | 3250  | 6685   | 4495   | 4479   |
| Sum        | 205147 | 145619 | 144087| 291474 | 191606 | 189886 |

Table 2. The MEC scores computed by Greedy, HapCut and dynamic programming (DP), MaxSAT conversion, on short reads only and on all reads, respectively, for each chromosome.
Table 3. Average number of connected components contained in each block and average size of the connected components whose size is greater than 1 for different (coverage ratio, SD) settings

| Coverage Ratio | Num | Size |
|---------------|-----|------|
| (10, 5)       | 8   | 2    |
| (10, 50)      | 7   | 2    |
| (10, 500)     | 10  | 3    |
| (20, 5)       | 8   | 2    |
| (20, 50)      | 6   | 3    |
| (20, 500)     | 6   | 2    |
| (30, 5)       | 8   | 2    |
| (30, 50)      | 6   | 3    |
| (30, 500)     | 6   | 2    |
| (40, 5)       | 8   | 5    |
| (40, 50)      | 5   | 3    |
| (40, 500)     | 3   | 1    |
| (100, 500)    | 1   | 1    |

Table 4. The probability (%) of an SNP attached to other SNPs more than once, twice, three times, four times, for different (coverage ratio, SD) settings

| Coverage Ratio | Num 1 | Num 2 | Num 3 | Num 4 |
|---------------|-------|-------|-------|-------|
| (10, 5)       | 41    | 56    | 65    | 45    |
| (10, 50)      | 35    | 38    | 39    | 44    |
| (10, 500)     | 29    | 26    | 23    | 35    |
| (20, 5)       | 66    | 62    | 64    | 32    |
| (20, 50)      | 62    | 64    | 45    | 39    |
| (20, 500)     | 44    | 44    | 41    | 23    |
| (30, 5)       | 82    | 89    | 46    | 35    |
| (30, 50)      | 75    | 54    | 43    | 35    |
| (30, 500)     | 61    | 61    | 43    | 44    |
| (40, 5)       | 89    | 47    | 73    | 34    |
| (40, 50)      | 75    | 44    | 66    | 41    |
| (40, 500)     | 43    | 60    | 43    | 54    |
| (100, 500)    | 71    | 71    | 71    | 71    |

To evaluate how well the haplotype assembly can be done w.r.t. the sequencing protocols, we next construct a graph from the read matrix. Each heterozygous SNP is a vertex in the graph and we draw an edge between two vertices if their corresponding SNPs are covered by the same read. We construct such a graph using all the generated reads and consider the connected components in this graph since we have no information on how to phase heterozygous sites in different connected components relative to each other. The number of optimal solutions will be exponential in the number of connected components. Therefore, the smaller the number of connected components is, the better we can assemble the haplotypes.

We count the average number of connected components each block contains. We also compute the average size of the connected components. We show the experimental results in Table 3. As we can see, the number of connected components in each block decreases as coverage ratio increases and as SD increases. Meanwhile, the average size of connected components also increases. However, to reduce the number of connected components each block contains to one such that we can fully reconstruct each block, we need to use a very high-coverage ratio such as 100. Thus for any reasonable coverage ratio or the standard deviation increases. With 40 times coverage and 500 bp SD, the probability of an SNP being attached to other SNPs at least once is as high as 92%, and at least twice is also high as 83%. Therefore, with even a moderate amount of coverage, most variants are covered by at least one read to another variant when the SD of the insert length is big enough. Thus, the combined strategy of using a traditional approach to infer haplotypes using the genotypes at the common variants combined with assembly of the rare variants using the sequence reads is a practical approach for inferring haplotypes.
5 DISCUSSION

In this article, we proposed a dynamic programming algorithm for the ‘haplotype assembly’ problem, which is able to assemble the haplotypes optimally with time complexity \(O(m \times 2^k \times n)\), where \(m\) is the number of reads, \(k\) is the length of the longest read and \(n\) is the total number of SNPs in the haplotypes. Our experiments show for the first time that the current best-known solutions are very close to the optimal solution.

The most difficult part of the haplotype assembly problem is to handle the long reads. Long reads can span up to a few hundred positions. To handle these cases, we convert the problem to an MaxSAT problem, which can be solved optimally by an MaxSAT solver. We show that our MaxSAT solver is able to solve 99.98% of the problem instances optimally. For the remaining 0.02%, the MaxSAT solver also reports better results than HapCut does. Therefore, the overall solution we obtained is very close to the optimal.

Although the empirical results of our methods did not show a major advance over existing methods, we believe it is technically important and also interesting to have optimal algorithms for the haplotype assembly problem.

Our analysis on individual genotype data from HapMap shows that it is impractical to fully assemble the haplotypes as the coverage ratio needed is too high. However, combined with a traditional haplotype inference approach, our algorithm is able to infer haplotypes containing both rare and common SNPs, including SNPs that are unique to individuals.

Funding: D.H. and E.E. are supported by National Science Foundation (grants 0513612, 0731455, 0729049 and 0916676); NIH (grants K25-HL080079 and U01-DA024417). University of California, Los Angeles subcontract of contract N01-ES-45530 from the National Toxicology Program and National Institute of Health Sciences to Perlegen Sciences (in part).

REFERENCES

1000 Genomes Project (2010) A deep catalog of human genetic variation. Available at http://www.1000genomes.org (last accessed date April 23, 2010); Bansal,V. and Bafna,V. (2008) HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. Bioinformatics, 24(53).

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

Baranz et al. (eds) (2009) Handbook of Satisfiability, Vol. 185 of Frontiers in Artificial Intelligence and Applications. IOS Press, Nieuwe Hemweg, Amsterdam.

Browning,B. and Browning,S. (2008) Haplotypic analysis of Wellcome Trust Case Control Consortium data. Hum. genet., 123, 273–286.

Chen,A. et al (2008) Efficient genome wide tagging by reduction to SAT. In: Proceedings of the 8th International Workshop on Algorithms in Bioinformatics, Vol. 5251 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 135–147.

Ciliakus,R. et al. (2005) On the complexity of several haplotyping problems. In: Proceedings of the 5th International Workshop on Algorithms in Bioinformatics, Vol. 3692 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 128–139.

Fitch,W. (1977) On the problem of discovering the most parsimonious tree. Am. Nat., 111, 223–257.

Gusfield,D. (2003) Haplotype inference by pure parsimony. In: Proceedings of the Combinatorial Pattern Matching Conference. Vol. 2666 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 144–155.

Halperin,E. and Eskin,E. (2004) Haplotype reconstruction from genotype data using imperfect phylogeny. Bioinformatics, 20, 1842–1849.

International HapMap Consortium (2007) A second generation human haplotype map of over 3.1 million SNPs. Nature, 449, 851–861.

Lanciucgi et al. (2001) SNPs problems, complexity, and algorithms. In: Proceedings of the 9th Annual European Symposium on Algorithms. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 182–193.

Levy,S. et al. (2007) The diploid genome sequence of an individual human. PLoS Biol., 5, e254.

Li,C. and Manly,F. (2009) MaxSAT: hard and soft constraints. Handbook of Satisfiability, 185, 613–631.

Li,H. et al. (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Res., 18, 1851.

Liptert,R. et al. (2002) Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. Brief. Bioinform., 3, 23.

Manquinho,V. et al. (2009) Algorithms for weighted Boolean optimization. In: Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 495–508.

Marchini, J. et al. (2007) A new multipoint method for genome-wide association studies by imputation of genotypes. Nat. Genet., 39, 906–913.

Pandosyan,A. and Stain,M. (2004) Fast base: a fast heuristic for single individual SNP haplotype reconstruction. Vol. 3240 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 266–277.

Papoutsou,K. et al. (2008) Solving weighted Max-SAT problems in a reduced search space: a performance analysis. Journal on Satisfiability, Boolean Modeling and Computation, 4, 191–217.

Stephen,M. et al. (2001) A new statistical method for haplotype reconstruction from population data. Am. J. Hum. Genet., 68, 978–989.

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

Wherle,D. et al. (2008) The complete genome of an individual by massively parallel DNA sequencing. Nature, 452, 872–879.