ReFHap: A Reliable and Fast Algorithm for Single Individual Haplotyping

Jorge Duitama *
Department of Computer Science and Engineering
University of Connecticut
371 Fairfield Rd, Unit 2155
Storrs, CT, 06269-2155, USA
jduitama@engr.uconn.edu

Thomas Huebsch
Max Planck Institute for Molecular Genetics
Ihnestr. 63-73
Berlin, Germany
huebsch@molgen.mpg.de

Gayle McEwen
Max Planck Institute for Molecular Genetics
Ihnestr. 63-73
Berlin, Germany
mcewen@molgen.mpg.de

Eun-Kyung Suk
Max Planck Institute for Molecular Genetics
Ihnestr. 63-73
Berlin, Germany
suk@molgen.mpg.de

Margret R. Hoehe
Max Planck Institute for Molecular Genetics
Ihnestr. 63-73
Berlin, Germany
hoehe@molgen.mpg.de

ABSTRACT

Full human genomic sequences have been published in the latest two years for a growing number of individuals. Most of them are a mixed consensus of the two real haplotypes because it is still very expensive to separate information coming from the two copies of a chromosome. However, latest improvements and new experimental approaches promise to solve these issues and provide enough information to reconstruct the sequences for the two copies of each chromosome through bioinformatics methods such as single individual haplotyping. Full haploid sequences provide a complete understanding of the structure of the human genome, allowing accurate predictions of translation in protein coding regions and increasing power of association studies.

In this paper we present a novel problem formulation for single individual haplotyping. We start by assigning a score to each pair of fragments based on their common allele calls and then we use these score to formulate the problem as the cut of fragments that maximize an objective function, similar to the well known max-cut problem. Our algorithm initially finds the best cut based on a heuristic algorithm for max-cut and then builds haplotypes consistent with that cut. We have compared both accuracy and running time of ReFHap with other heuristic methods on both simulated and real data and found that ReFHap performs significantly faster than previous methods without loss of accuracy.

*Also affiliated to the Max Planck Institute for Molecular Genetics. Email: duitama@molgen.mpg.de

Categories and Subject Descriptors

I.2.8 [**ARTIFICIAL INTELLIGENCE**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; J.3 [**Computer Applications**]: LIFE AND MEDICAL SCIENCES—*Biology and genetics*; I.6.5 [**SIMULATION AND MODELING**]: Model Development—*Modeling methodologies*

General Terms

Algorithms,Performance

Keywords

Haplotyping, Algorithms, Fragments, Variants, Maximum Cut, Heuristic, Efficiency

1. INTRODUCTION

DNA sequencing is at the cornerstone of current advances in genetics, enabling breakthroughs in medical and biological research [3]. The first complete human genome sequence was in fact a "representative" genome sequence based on the DNA of several individuals [10]. Advances in sequencing technology and computational methods are resulting in increasingly cost-effective, high-throughput sequencing, making the sequencing of genomes from individuals possible [3, 12, 14, 20, 24, 25]. This allows the identification of common patterns of human genetic variation between individual genomes that may affect health, disease, and individual responses to medications.

Human somatic cells are diploid, containing two sets of chromosomes, one set derived from each parent. Differences between the two copies of each chromosome are called heterozygous variants and for each variant the sequences that differ are called alleles. Most of the variation comes in form of single nucleotide variants (SNV) where the alleles are base

pairs that differ between the two copies. However, alleles can also be identified in other types of variations, for example, structural variations and insertion and deletion events (indels).

The process of grouping alleles that are present together on the same chromosome copy of an individual is called haplotyping [8, 16]; reconstruction the two sequences of each chromosome is the most advanced type of haplotyping. Besides getting the full structure of the genome, complete haplotype sequences enable improved predictions of changes in protein structure produced by mutations in coding regions and increase power for genome-wide association studies [18]. It will also allow insights into the complex interplay of alleles of genes and their regulatory elements.

At present, published sequences may be considered "mixed diploid" [9]; because they actually represent a composite of the two underlying haploid sequences. Although each study presents preliminary haplotyping results, complete construction of the "true" molecular sequences for each of the chromosome pairs remains a challenge towards full genome completion [12, 14]. This is mainly due to the fact that current sequencing technologies do not provide enough information to reliably separate alleles originating from each of the two copies of a chromosome unless parental or population information is available [1, 23, 27]. However, this situation is likely to change in the near future with improvements in second-generation sequencing methods such as longer read lengths, mate-pairs and increased throughput, and also the development of new experimental approaches. This work is part of a currently productive approach towards molecular haplotype determination which relies upon fosmid-based sequencing [5, 11]; Details about resources and approaches we have developed towards this aim are available at `http://www.molgen.mpg.de/~genetic-variation/Projects.html`.

Algorithms for haplotyping can be grouped in three categories depending on the type of source information: (i) population information, (ii) parental and individual genotype information and (iii) evidence of co-occurrence of alleles. Population information takes genotype information for a group of individuals known to come from the same population and uses an evolutionary model to phase all genotypes at the same time [7, 19, 21] whereas parental and individual genotype information enables alleles to be grouped into loci where either parent is homozygous and hence there is no doubt on deciding which allele comes from which parent [4, 13]. A combination of these approaches is used by the International HapMap Project [22] to generate haplotypes for each population. These two categories have the advantage that they require genotype information which is easy to acquire for certain loci. However, acquiring parental or population information may not feasible for all heterozygous variants of the individual of interest. In algorithms based on evidence of the allele co-occurrence, this information has so far come from reads or mate-pairs spanning at least two heterozygous variant loci but in general the evidence can come from any source. DNA sequences showing co-occurrence of two or more alleles are usually called fragments. The computational problem of haplotyping based on this kind of information is called single individual haplotyping and many formulations and algorithms have been proposed to solve it [16, 15]. Although absence of real data has been always an issue, the problem has been carefully studied from both theoretical and practical points of view and simulated data has been used to make comparisons between different approaches [1, 23, 27]. The algorithm presented in this paper is a contribution to this approach.

Previous studies on single individual haplotyping have established different problem formulations seeking for different optimization objectives. Computational properties of these formulations have been studied and it has been shown that most of them are NP-hard [16, 15]. Proposed algorithms can be divided into exact, genetic, probabilistic and heuristic. Due to the NP-hardness of the formulations, exact algorithms require an exponential dependency on at least one input parameter so they do not scale well as the size of the input gets large [23]. Genetic and probabilistic algorithms have the advantage of searching within a large set of possible haplotypes at the expense of time [2, 23]. Heuristic algorithms try to find efficiently a haplotype as close to the optimum as possible according to a specific optimization criterion. One of the most accurate of these algorithms is HapCUT [1], which starting from a random solution, builds a graph and uses max-cut to find loci that should be flipped to improve the current haplotype based on the input data. Our experiments indicate that although the algorithm is reliable, its running time is too large for whole genome haplotyping.

In this paper we present a novel problem formulation for single individual haplotyping and a heuristic algorithm called ReFHap. Like in [1] our formulation allows us to reduce the problem to max-cut but here we design a graph that enforces separation of fragments rather than variant loci. Our approach initially attempts to find the cut that groups together fragments coming from the same copy of each chromosome to subsequently build haplotypes consistent with the best cut found by our heuristic algorithm. We show through extensive simulation experiments that ReFHap represents an improvement in running time compared to previous algorithms without loss of accuracy. Simulations indicate that ReFHap is more accurate and scalable than the model of [27] and that it has comparable accuracy and higher efficiency than HapCUT [1]. Moreover, we tested both ReFHap and HapCUT with preliminary real data from fosmid-based sequencing. Results indicate that ReFHap is able to efficiently perform whole chromosome haplotyping with good accuracy.

2. METHODS

2.1 Problem Formulation

Informally, the objective of single individual haplotyping is to reconstruct the two haplotypes of an individual from a set of partial readings called fragments. Each fragment provides evidence of coocurrance of two or more alleles of different SNPs in the same haplotype. The usual strategy to predict the true haplotypes is to define a real function on the input data and an arbitrary pair of haplotypes, hoping that the real haplotypes will correspond with the result of an optimization objective on this function.

As in previous works [1, 15], we represent the input of the problem as a matrix M of size mxn where m is the number of fragments and n is the number of variant loci. Each row

of M encodes the information for one fragment as a string on the alphabet $\{0, 1, -\}$. Here, $M[i, j] \neq -$ means that fragment i calls allele $M[i, j]$ in locus j while $M[i, j] = -$ means that fragment i does not cover locus j. This problem definition inmposes a restriction of at most two alleles per locus, which is sufficient for most of the variation in many diploid organisms. There is no restriction on the type of variations considered as long as they can be mapped to a specific locus and the two alleles can be identified. Usually, the reference allele is encoded as a zero and the alternative allele as a one but this is not required by ReFHap. As in [1] ReFHap assumes that all input loci are heterozygous. If this is not the case, a preprocessing step like the one in [15] can be implemented to call genotypes and remove homozygous loci and fragments covering at most one heterozygous locus.

Given two strings f_1, f_2 of length n, we say that $f_1 = f_2$ if and only if for every $1 \leq j \leq n$, $f_1[j] = -$, $f_2[j] = -$ or $f_1[j] = f_2[j]$. In absence of errors, the problem reduces to find two haplotypes $h, \bar{h}$ such that every fragment f_i is equal to either h or $\bar{h}$ according with the definition of equality given above. This problem can be solved just by separating the fragments in two groups such that for any pair of fragments f_1, f_2 inside a group $f_1 = f_2$ and then building the haplotypes by taking the consensus allele for each locus inside each group. This is equivalent to solve max-cut on a graph $G = (V, E)$ where V is the set of fragments and $e = \langle f_1, f_2 \rangle$ if and only if $f_1 \neq f_2$. In absence of errors, G is bipartite and hence max-cut can be solved in polynomial time for G. However, the solution is not unique if the graph is not connected. The connected components of G are called in this setting haplotype blocks. The number of these blocks affects the quality of the output haplotypes because in absence of additional inputs, there is no information to decide how to connect two consecutive blocks and hence the probability of joining them consistently is $0,5$.

If fragments contain errors, G may not be bipartite anymore and conflics are created between fragments. A simple example of a conflict happens when there are two loci j_1, j_2 covered by two fragments f_1, f_2 for which $f_1[j_1] = f_2[j_1]$ and $f_1[j_2] \neq f_2[j_2]$. Clearly one of the entries must be wrong if both loci are heterozygous. Different strategies to remove conflicts lead to optimization objectives studied in previous works like finding the minimum number of fragments to remove (MFR), the minimum number of loci to remove (MSR) or the minimum number of allele calls to correct (MEC). Computational properties of these problems have been analyzed by [16, 15] and several algorithms have been proposed for MEC [1, 6, 23, 26]. If weights are available for each allele call on each freagment, the model called (WMLF) described by [27] tries to minimize the sum of weights of corrected alleles.

Our approach is to reduce the problem to max-cut as in the case without errors but adding weights to the edges of G in such a way that the cut that maximizes the sum of the weights of crossing edges resembles as accurate as possible the actual origin of each fragment. Weights are calculated based on the following scoring scheme. Given two allele calls a_1, a_2, the score $s(a_1, a_2)$ is given by:

$$s(a_1, a_2) = \begin{cases} -1 & \text{if } a_1 \neq - \wedge a_2 \neq - \wedge a_1 = a_2, \\ 1 & \text{if } a_1 \neq - \wedge a_2 \neq - \wedge a_1 \neq a_2, \\ 0 & \text{otherwise.} \end{cases}$$

Now, given two rows i_1, i_2 of M, the score $s(M, i_1, i_2)$ is just the sum of the contributions for each pair of alleles at each locus:

$$s(M, i_1, i_2) = \sum_{j=1}^{n} s(M[i_1, j], M[i_2, j])$$

Note that if two fragments do not cover any common locus then their score is zero but the opposite is not necessarily true. Given a fragments matrix M we can define a cut of fragments as a set of rows $I \subseteq \{1, \dots m\}$. Given a matrix M, the score of the cut I is given by:

$$s(M, I) = \sum_{i \in I} \sum_{k \notin I} s(M, i, k)$$

We use this scoring function to state the following problem definition:

Maximum Fragments Cut (MFC): Given a mxn matrix M of m fragments covering n loci, find a cut I such that $s(M, I)$ is maximized.

THEOREM 1. *Maximum Fragments Cut is NP-Hard.*

PROOF. Max-Cut can be reduced to MFC in the following way. Given an instance $G = (V, E)$ for max-cut, build M by creating a row for each element of V and for each edge $\langle i_1, i_2 \rangle \in E$ make a column j and set $M[i_1, j] = 0$, $M[i_2, j] = 1$ and $M[k, j] = -$ for every row l different than i_1 or i_2. The score $s(M, i_1, i_2)$ for any pair of rows is equal to 1 if and only if $\langle i_1, i_2 \rangle \in E$ otherwise, it will be zero because the two rows cover at most one common locus. Now, given a cut on G represented by a subset $V' \subseteq V$, the weight of this cut will be equal to the score $s(M, I)$ of the cut I made by selecting the rows corresponding with the vertices in V'. Hence, any algorithm that can calculate the maximum of the function $s(M, I)$ will calculate also the max-cut value for G and conversely, any algorithm that can calculate the max-cut value for G will also calculate the maximum of the function $s(M, I)$. □

2.2 Algorithm

To solve MFC we build a graph $G = (V, E, w)$, where $V = \{1, \dots, m\}$, $\langle i_1, i_2 \rangle \in E$ if and only if $s(M, i_1, i_2) \neq 0$ and for all $e = \langle i_1, i_2 \rangle \in E$, $w(e) = s(M, i_1, i_2)$. Then, we solve the weighted version of Max-Cut on G. We implemented a heuristic algorithm similar to the one used by [1] that iterates over the edges and for each one builds an initial greedy cut and then tries to improve it through local optimization. The main steps are the following:

ReFHap(M, k)

1. Build $G = (V, E, w)$ from M
2. Sort E from largest to smallest weight
3. Init I with a random subset of V
4. For each e in the first k edges of E
   (a) $I' \leftarrow$ GreedyInit(G, e)
   (b) $I' \leftarrow$ GreedyImprovement(G, I')
   (c) If $s(M, I) < s(M, I')$ then $I \leftarrow I'$

The procedure GreedyInit finds a cut I' in which e crosses from I' to $V \setminus I'$ and the procedure GreedyImprovement tries to improve I' by local optimization. The parameter k controls how many edges are considered for the initialization step and hence allows to make a compromise between accuracy and speed. Unlike the algorithm in [1] in which initial edges are chosen at random, we decided to sort the edges because edges with large weight are more likely to be part of the best cut. The maximum value that k can take is $|E|$ but we can achieve good accuracy in many cases with a small value of k. In our current implementation $k = \sqrt{|E|}$.

To show how we implemented the greedy procedures we need to expand the score function to subgraphs. Given a graph $G = (V, E, w)$ and two disjoint subsets I_1, I_2 of V we define:

$$s(G, I_1, I_2) = \sum_{i \in I_1} \sum_{k \in I_2} w(\langle i, k \rangle)$$

and for each vertex $v \in V \setminus (I_1 \cup I_2)$ we can define:

$$s(G, I_1, I_2, v) = \max(s(G, I_1 \cup \{v\}, I_2), s(G, I_1, I_2 \cup \{v\}))$$

As in [1], to avoid high negative edges crossing the cut, we build a cut from an input edge $\langle i_1, i_2 \rangle$ by initializing I_1 with i_1 and I_2 with i_2 and then adding either to I_1 or I_2 the edge that locally maximizes $s(G, I_1, I_2, v)$:

GreedyInit$(G, \langle i_1, i_2 \rangle)$

1. Init $I_1 \leftarrow \{i_1\}$ and $I_2 \leftarrow \{i_2\}$
2. While $I_1 \cup I_2 \neq V$
   (a) Find $v \in V \setminus (I_1 \cup I_2)$ maximizing $s(G, I_1, I_2, v)$
   (b) If $s(G, I_1 \cup \{v\}, I_2) > s(G, I_1, I_2 \cup \{v\})$ add v to I_1 else add v to I_2
3. return I_1

For local optimization we implemented the classical greedy algorithm of [17], which calculates for each vertex $v \in I'$ the score $s(G, I' \setminus \{v\}, V \setminus (I' \setminus \{v\}))$ and for each vertex $w \notin I'$ the score $s(G, I' \cup \{w\}, V \setminus (I' \cup \{w\}))$ and flips the vertex with maximum score if it is larger than the current score $s(G, I', V \setminus I')$. We also implemented a local optimization step flipping edges rather than vertices, which is equivalent to check the improvement after flipping every possible pair of vertices at the same time. We iterate these two optimizations until neither of them can achieve any improvement.

After finding the cut I, the algorithm uses the input matrix to find the haplotype h that minimizes the number of entries to be corrected assuming that rows in I belong to h and rows in $V \setminus I$ belong to $\bar{h}$. Since we assume that all loci are heterozygous, the output of ReFHap is just one haplotype h and $\bar{h}$ is just the haplotype obtained by flipping every allele call in h. The haplotype h can be inferred by making a single traversal of M as follows:

1. For each column j
   (a) $I_{j,0} \leftarrow \{i : (i \in I \wedge M[i, j] = 0) \vee (i \notin I \wedge M[i, j] = 1)\}$
   (b) $I_{j,1} \leftarrow \{i : (i \in I \wedge M[i, j] = 1) \vee (i \notin I \wedge M[i, j] = 0)\}$
   (c) If $|I_{j,0}| \geq |I_{j,1}|$ then $h_j \leftarrow 0$ otherwise $h_j \leftarrow 1$
2. output h

The complexity of this algorithm depends on the number of fragments m, the number of different starting edges k, the maximum number of loci covered by a single fragment and the maximum number of fragments covering a single locus, which as in [27] we call respectively k_1 and k_2. First note that the maximum degree of a vertex in G is bounded by $k_1(k_2 - 1)$. To prove this note that one fragment f calls alleles for at most k_1 loci. Each of these loci is covered by at most $k_2 - 1$ fragments different than f. Therefore, each locus contributes with at most $k_2 - 1$ edges to the vertex associated with f. In the worst case, there are no shared edges between loci and then the total number of edges is $k_1(k_2 - 1)$.

Since $k_1 k_2$ is typically much smaller than m, the total number of edges of G is $O(mk_1k_2)$. The sorting step takes then $O(mk_1k_2 \log(mk_1k_2))$ and, as shown below, it is dominated by the iterations step. For fixed G, I_1, I_2 and v, $s(G, I_1, I_2, v)$ can be calculated just by inspecting the edges of v, so, for each of the k edges considered, GreedyInit takes $O(m^2k_1k_2)$. Although the local optimization in theory could take a time equivalent to the sum of the positive edge weights which would be exponential on the input size, in practice the number of iterations is small enough to be considered a constant, so the time needed for each local optimization is $O(mk_1k_2 + mk_1^2k_2^2) = O(mk_1^2k_2^2)$. The total time is $O(k(m^2k_1k_2 + mk_1^2k_2^2))$. Comparing this result with the complexity of the algorithm designed by [27], which includes an exponential dependency on k_2, and the algorithms designed by [1] and [6], which need at least $O(mn^2)$ time, we can predict that ReFHap will perform faster especially as the number of loci covered by each fragment increases and hence less fragments are needed to achieve the same coverage. In the next section we will show simulation experiments confirming this hypothesis.

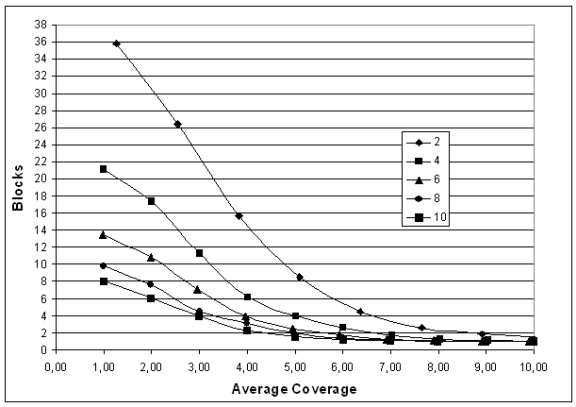

Figure 1: Number of blocks as a function of coverage for different fragment lengths.

3. RESULTS

3.1 Experimental Setup

We performed several simulation experiments to test the behavior of ReFHap under a wide range of circumstances. We generated instances varying over five different criteria: number of loci n, number of fragments f, mean fragment length l, error rate e and gap rate g. For each instance, we created a random haplotype h of size n and then we created f fragments. For each fragment we selected its length l_i drawing from a normal distribution centered at l and with standard deviation equal to 1. We then selected its starting position j as a random integer in the range from 1 to $n - l_i + 1$. The fragment f_i is then the substring of h starting at position j with length l_i. We flipped the whole fragment with probability 0.5, assuming that real fragments are equally likely to come from either of the two haplotypes. Finally, we introduce errors by flipping each allele call of f_i with probability e and we also introduced gaps by deleting each allele call of f_i, except for the first and the last position, with probability g.

We performed one experiment for each combination of selected values of the simulation parameters. For each experiment, we generated 100 random instances following the procedure described above. We implemented ReFHap in Java 1.6 and we compared it with the public available implementation of HapCUT [1] and with the implementation of the WMLF model kindly provided by the authors of [27]. Both HapCUT and WMLF were implemented in C. We ran all experiments on a RedHat Linux 64 bit server.

Before checking the performance of ReFHap we investigated how the number of haplotype blocks changes for different number of fragments and fragment lengths. As shown in section 2, the number of blocks heavily influences the quality of the haplotype no matter which model is used to solve the problem. Figure 1 shows how as the fragment length increases, less coverage is needed to connect every locus with each other. Although the simulation assumes that variants are evenly distributed in the genome (in real organisms that is not always the case) this result means that for the same coverage, few large fragments produce fewer blocks than many short fragments.

3.2 Simulation Results

For each experiment we calculated means for three measures. The first one is the Minimum Error Correction (MEC), which is the minimum number of changes within the matrix to make it consistent with the answer haplotypes. This measure divided by the total number of allele calls in the input matrix is a good estimator of the allele calling error rate. The algorithm implemented in HapCUT assumes that the true haplotype is the one that minimizes this measure. The second measure is the switch error (SE) which is calculated by traversing the resulting haplotypes from left to right and computing the number of times needed to jump from one haplotype to its complement to reconstruct the real haplotype. Assuming absence of genotyping errors, this is the true measure of quality for any solution. However, this measure can not be calculated for real instances unless a gold standard haplotype is known. For our simulations we can calculate the number of switch errors because we know the true haplotype for each instance. The third measure is the running time of the algorithm measured by running it on a single processor.

The upper panel of figure 2 shows the distribution of differences between the WMLF model and ReFHap in MEC, switch errors and running time for experiments varying coverage by increasing the number of reads and fixing the number of loci to 200, the fragment length to 6 and with no errors or gaps. ReFHap consistently produces lower MEC and switch errors. WMLF has a better runtime for small instances but that changes when coverage increases and even after $10x$, the limit on the maximum coverage for one locus of 23 is often achieved. This limit is required by WMLF because the algorithm has an exponential dependency on this parameter.

The lower panel of figure 2 shows the distribution of differences between HapCUT and ReFHap for the same criteria as above for experiments with the same number of loci and fragment length but adding an error rate of 5% and a gap rate of 10%. While the difference in switch errors between HapCUT and ReFHap is almost zero on average, ReFHap performs consistently faster than HapCUT. We performed a statistical test for each experiment to see if the differences are on average significanly different from zero and we found that this is the case in general for the time differences, in favor of ReFHap and for the MEC differences in favor of HapCUT. HapCUT provides lower MEC values because that is its optimization objective. However, switch errors are the true measure of quality, not MEC. Table 1 shows that we could not find evidence of a significant difference between HapCUT and ReFHap in switch errors for most of the experiments carried on, which means that the reliability of the two methods is similar. This table also shows that results in Figure 2 are replicated consistently by experiments increasing the mean fragments length up to 12 and the number of loci up to 1000.

We finally examined how the haplotype quality decreases as the error rate increases. Figure 3 shows that the number of switch errors increases at the same pace for both HapCUT and ReFHap as the error rate increases to an extreme value of 50%. These experiments were performed on 200 loci with 296 fragments of length 6 and a gap rate of 0.1, achieving a

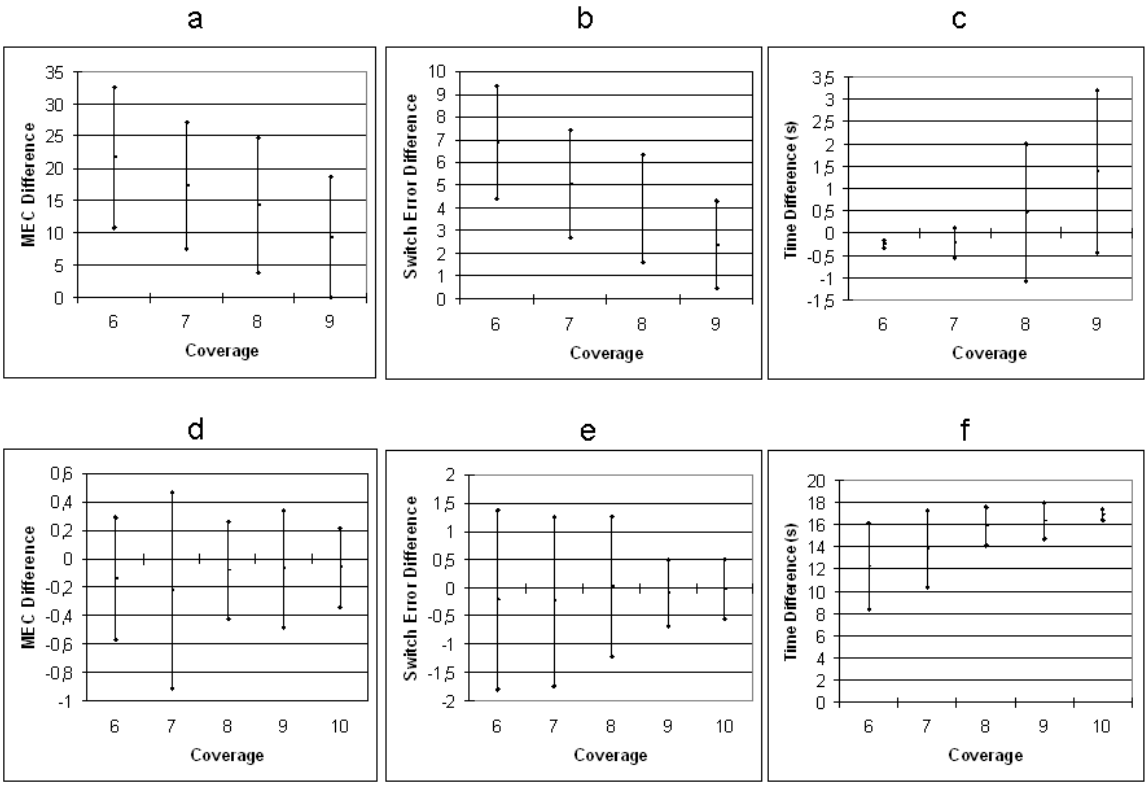

Figure 2: Distribution of differences between values reported by WMLF and HapCUT and values reported by ReFHap for experiments varying coverage by increasing the number of fragments. Markers above and below the mean correspond to the mean plus and minus one standard deviaton respectively. The upper panel shows the differences between WMLF and ReFHap in (a) MEC , (b) switch errors and (c) running time. The lower panel shows the differences between HapCUT and ReFHap in (d) MEC , (e) switch errors, and (f) running time

Table 1: Minimum Error Correction (MEC), Switch errors (SE) and running time for HapCUT and ReFHap for simulation experiments varying haplotype length (l), number of fragments (n) and mean fragment length (f). Each reported p-value is the probability that HapCUT and ReFHap report on average the same value for MEC and SE on each set of input conditions. Time p-values were also calculated but, except for the first row, they are always less than 10^{-32}

Input					HapCUT			ReFHap			p-values	
l	n	f	Avg. Blocks	Avg. Cov.	%MEC	%SE	Time	%MEC	%SE	Time	p-value MEC	p-value SE
200	200	2	26.8	2.49	3.57	12.35	0.28	3.62	12.45	0.22	$6.3 * 10^{-6}$	0.32
200	200	4	7.1	3.82	4.62	5.29	1.81	4.69	5.95	0.26	$5.9 * 10^{-11}$	$1.2 * 10^{-4}$
200	200	6	1.74	5.61	4.89	1.46	10.54	4.9	1.55	0.39	$8.8 * 10^{-4}$	0.16
200	200	8	1.06	7.41	4.94	0.75	19.06	4.94	0.74	0.58	0.02	0.4
200	200	10	1	9.19	4.91	0.34	24.55	4.91	0.29	0.68	0.16	0.12
200	200	12	1	11.0	5.0	0.21	29.27	5.0	0.2	0.73	0.5	0.29
200	222	6	1.36	6.20	4.94	1.17	12.72	4.95	1.27	0.51	$5.2 * 10^{-4}$	0.09
200	259	6	1.22	7.26	5.08	0.7	14.49	5.1	0.82	0.67	$7.2 * 10^{-4}$	0.05
200	296	6	1.06	8.29	4.97	0.43	16.74	4.97	0.42	0.91	$9.1 * 10^{-3}$	0.44
200	333	6	1.03	9.32	4.93	0.22	17.58	4.93	0.27	1.29	0.04	0.06
200	370	6	1.01	10.37	4.91	0.14	18.65	4.91	0.15	1.79	0.02	0.35
200	700	2	1.75	8.73	4.75	1.75	10.72	4.8	1.87	3.58	$1.5 * 10^{-13}$	0.09
400	700	5	1.24	8.22	4.89	0.46	79.12	4.91	0.58	4.97	$1.6 * 10^{-6}$	$1.7 * 10^{-3}$
600	700	7	1.3	7.59	4.98	0.48	300.63	4.99	0.51	4.70	$3.7 * 10^{-4}$	0.15
800	700	10	1.14	8.05	5.01	0.35	1064.62	5.01	0.35	5.47	0.05	0.48
1000	700	12	1.24	7.69	4.98	0.38	2279.25	4.98	0.45	5.89	0.16	$1.7 * 10^{-3}$
200	296	6	1.06	8.29	4.95	0.4	16.72	4.95	0.48	0.88	0.01	0.05
400	592	6	1.2	8.28	4.98	0.38	94.95	4.98	0.395	4.34	$2.9 * 10^{-3}$	0.33
600	888	6	1.28	8.3	5.02	0.4	273.87	5.02	0.46	8.05	$5.1 * 10^{-5}$	0.03
800	1185	6	1.31	8.3	4.97	0.4	595.47	4.98	0.43	13.4	$5.7 * 10^{-4}$	0.10
1000	1481	6	1.41	8.3	4.98	0.35	1019.78	4.98	0.36	20.8	$3.3 * 10^{-3}$	0.30

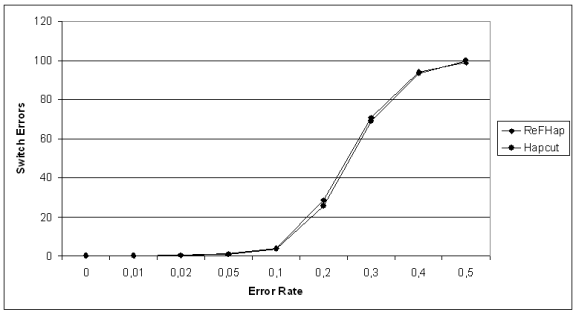

Figure 3: Switch error rate for HapCUT and ReFHap for experiments varying error rate

mean coverage of about $8x$ and a mean number of blocks of 1.06. These parameters were set up to ensure that most of the switch errors are produced by the error rate and by the behavior of the algorithms and not by the number of blocks.

3.3 Results with Real Data

We tested ReFHap and HapCUT on data resulting from the experimental fosmid based sequencing approach introduced by [5] and that we are currently developing (See `http://www.molgen.mpg.de/~genetic-variation/Projects.html`). We built a test case by sequencing and aligning fosmids generated from chromosome 22 of a caucasian individual. The input for this test case is a matrix of 32347 SNPs covered by 13905 fragments. The total number of allele calls is 178191. Hence, each SNP is covered on average 5.51 times and each fragment covers on average 12.81 SNPs. The total number of haplotype blocks is 102. Table 2 shows MEC percentage and running time values for both ReFHap and HapCUT. We included HapCUT results for its first iteration and after 50 iterations. ReFHap clearly performed faster than HapCUT by solving this test case in about one minute while even one single iteration of the heuristic implemented in HapCUT takes about one hour.

Table 2: MEC percentage and running time of ReFHap and HapCUT for a real instance with 32347 SNPs and 13905 fragments in chromosome 22

	ReFHap	HapCUT (1 It)	HapCUT (50 It)
%MEC	6.32%	6.26%	6.24%
Time	73.04 Sec	0.99 Hours	50.4 Hours

Unfortunately for this data we do not have the true haplotype so we can not calculate the exact switch error rate. However, we did several quality control steps to verify the accuracy of the assembled haplotypes. First, we estimated the switch error rate by running a simulation experiment with the same number of variants and fragments as in the real instance ($l = 32347$ and $n = 13905$), mean fragment length $f = 13$, gap probability $g = 0.1$ and error rate $e = 0.063$. Even though the total number of allele calls is on average 178004, which is less than the total for the real data, the

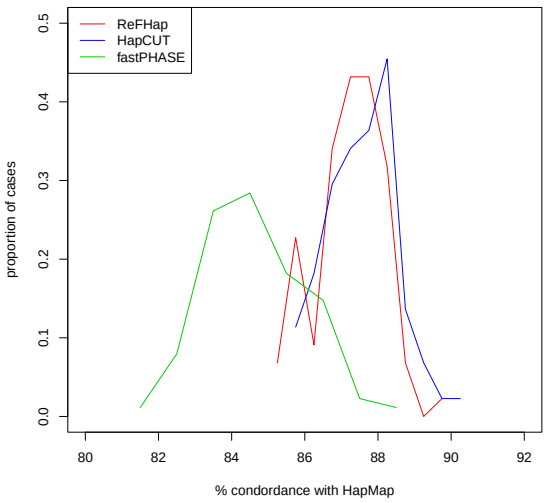

Figure 4: Distribution of percentages of concordance between assembled haplotypes for a caucasian individual and CEU HapMap haplotypes assembled from trio phasing

mean switch error rate is 1.86%.

We also had access to Affymetrix 1000k chip genotypes for one hundred individuals coming from the same population as our indvidual. We ran fastPHASE [19] on these genotypes to obtain a phasing of 3158 SNPs on chromosome 22 for the same individual. This haplotype can not be considered a gold standard but we can compare it with other haplotypes by defining a measure of concordance. Given two haplotypes, we calculate the switch error rate of the first as if the second were the gold standard and then we call percentage of concordance the result of one minus this rate. We achieved a 92.89% concordance between ReFHap and fastPHASE on 2941 SNPs shared between the two haplotypes. The percentage of concordance between HapCUT and fastPHASE was 93.30%.

Finally, we used the same measure of concordance to compare ReFHap, HapCUT and fastPHASE haplotypes with 89 haplotypes from individuals in the the CEU population of the HapMap project [22] assembled by trio phasing. We selected these haplotypes for comparison because the information provided by trios makes them more reliable and because the true haplotype of our individual should be similar to CEU haplotypes. Figure 4 shows the proportion of CEU haplotypes for different percentages of concordance, or in different words, the distribution of percentages of concordance for each assembled haplotype with the CEU haplotypes. The average percentages of concordance for HapCUT and ReFHap are 87.55% and 87.21% respectively, while the average percentage of concordance for fastPHASE is 84.67%. This comparison was done over an average of 2065 common SNPs for ReFHap and HapCUT and an average of 989 common SNPs for fastPHASE.

4. DISCUSSION

Current advances in sequencing technologies will increase the amount and types of variation discovered for individual genomes and will provide the information needed to assemble the true pair of haplotype sequences underlying each human chromosome through single individual haplotyping [11]. However, the increase in throughput, accuracy and completeness of sequencing technologies will not be reflected in improved haplotype construction if algorithms are not suitable to handle efficiently genome-wide scale data. Full haploid sequences are the ultimate goal to achieve a complete understanding of the structure of the human genome.

We have contributed to this field by introducing a novel problem formulation for single individual haplotyping and a heuristic algorithm to solve it. Our approach tries first to predict the actual separation of the input fragments into two groups, one for each chromosome copy. To this aim, we introduced a scoring scheme that allows us to build a graph of fragments and assign a weight to the relation between each pair of overlapping fragments based on their calls for common loci. After solving max-cut on this graph, we build the consensus haplotypes based on the best cut found. Since this approach resembles the well known max-cut problem, a heuristic algorithm for this problem is used as the base for our algorithm.

We compared our algorithm with HapCUT[1], which is the most accurate heuristic algorithm that we found available and with the WMLF model [27] which is the only solution including error probabilities for the input alleles. We have shown through extensive simulations that ReFHap computes haplotypes faster than these solutions without loosing accuracy. We also used experimental data to show that ReFHap scales better than other solutions for chromosome wide input, for which ReFHap finds reliable haplotypes within seconds while HapCUT takes one hour to make one iteration. We also performed comparisons with a statistical phasing approach and with high quality haplotypes from the CEU HapMap population [22].

It is difficult to establish a fair comparison between statistical phasing and phasing based on evidence of coocurance of alleles because the input information for both methods is too different to be comparable. However, we have shown that haplotypes assembled with single individual haplotyping can be more accurate than haplotypes inferred by statistical phasing if the region to assemble has enough coverage. Our simulations also help to get an idea of the coverage needed to achieve different levels of confidence.

In general, the biggest disadvantage of heuristic algorithms is that unlike exact algorithms, they do not provide the best solution for every instance. However, for this particular problem, formulations seek to optimize objective functions that are not fully correlated with the switch error rate. In that sense, even an exact algorithm cannot claim to provide the true haplotype sequences in every instance. In this scenario, an efficient heuristic algorithm with low error rates will be a better option from a practical point of view than an exact algorithm that can not ensure to have zero switch error rate.

In the near future, we intend to make further accuracy improvements by taking into account quality scores of fragment allele calls and by including other types of information like parental or population information within a single framework.

5. ACKNOWLEDGMENTS

We want to acknowledge the authors of [1] for making available an implementation of HapCUT and the authors of [27] for providing us an implementation of the WMLF model. We also want to acknowledge the entities funding this research: grant 01GS0863 from the Federal Ministry of Education and Research (BMBF) to MRH as part of the Program of Medical Genome Research (NGFN-Plus) and grants IIS-0546457 and IIS-0916948 from the United States National Science Foundation. We finally acknowledge Dr. Ralf Herwig from the bioinformatics group of the Max Planck Institute for Molecular Genetics for his support and professors Ion Mandoiu, Yufeng Wu and Sanguthevar Rajasekaran from the Computer Science Department in the University of Connecticut for providing the background needed to build the theoretical basis for this work.

6. REFERENCES

[1] V. Bansal and V. Bafna. HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics*, 24(16):i153–i159, August 2008.

[2] V. Bansal, A. L. Halpern, N. Axelrod, and V. Bafna. An MCMC algorithm for haplotype assembly from whole-genome sequence data. *Genome Research*, 18:1336–1346, August 2008.

[3] D. R. Bentley *et al.* Accurate whole human genome sequencing using reversible terminator chemistry. *Nature*, 456:53–59, October 2008.

[4] D. Brinza and A. Zelikovsky. 2SNP: scalable phasing method for trios and unrelated individuals. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 5(2):313–318, April-June 2008.

[5] C. Burgtorf, P. Kepper, M. R. Hoehe, C. Schmitt, R. Reinhardt, H. Lehrach, and S. Sauer. Clone-based systematic haplotyping (CSH): A procedure for physical haplotyping of whole genomes. *Genome Research*, 13:2717–2724, September 2003.

[6] L. M. Genovese, F. Geraci, and M. Pellegrini. SpeedHap: a fast and accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate and low coverage. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 5(4):492–502, October-December 2008.

[7] A. Gusev, I. I. Măndoiu, and B. Paşaniuc. Highly scalable genotype phasing by entropy minimization. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 5(2):252–261, April-June 2008.

[8] M. R. Hoehe. Haplotypes and the systematic analysis of genetic variation in genes and genomes. *Pharmacogenomics*, 4(5):547–570, September 2003.

[9] M. R. Hoehe, K. Köpke, B. Wendel, K. Rohde, C. Flachmeier, K. K. Kidd, W. H. Berrettini, and G. M. Church. Sequence variability and candidate gene analysis in complex disease: association of μ opioid receptor gene variation with substance dependence. *Human Molecular Genetics*, 9(19):2895–2908, September 2000.

[10] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, February 2001.

[11] J. M. Kidd, Z. Cheng, T. Graves, B. Fulton, R. K. Wilson, and E. E. Eichler. Haplotype sorting using human fosmid clone end-sequence pairs. *Genome Research*, 18:2016–2023, October 2008.

[12] S. Levy *et al.* The diploid genome sequence of an individual human. *PLoS Biology*, 5(10):e254+, September 2007.

[13] J. Marchini *et al.* A comparison of phasing algorithms for trios and unrelated individuals. *American Journal of Human Genetics*, 78(3):437–450, January 2006.

[14] K. J. McKernan *et al.* Sequence and structural variation in a human genome uncovered by short-read, massively parallel ligation sequencing using two-base encoding. *Genome Research*, 19:1527–1541, June 2009.

[15] A. Panconesi and M. Sozio. Fast Hare: a fast heuristic for single individual SNP haplotype reconstruction. *In: Jonassen, I., Kim, J. (eds.) WABI 2004. LNCS (LNBI)*, 3240:266–277, September 2004.

[16] R. Rizzi, V. Bafna, S. Istrail, and G. Lancia. Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. *In Proceedings of the Second International Workshop on Algorithms in Bioinformatics*, 2452:29–43, September 2002.

[17] S. Sahni and T. Gonzales. P-complete problems and approximate solutions. *In Proceedings of the 15th Annual Symposium on Switching and Automata Theory. IEEE*, pages 14–16, October 1974.

[18] D. J. Schaid. Evaluating associations of haplotypes with traits. *Genetic Epidemiology*, 27:348–364, November 2004.

[19] P. Scheet and M. Stephens. 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(4):629–644, February 2006.

[20] S. C. Schuster *et al.* Complete khoisan and bantu genomes from southern africa. *Nature*, 463:943–947, February 2010.

[21] M. Stephens and P. Donnelly. A comparison of bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73(5):1162–1169, October 2003.

[22] The International HapMap Consortium. A second generation human haplotype map of over 3.1 million SNPs. *Nature*, 449(18):851–861, September 2007.

[23] J. Wang, M. Xie, and J. Chen. A practical exact algorithm for the individual haplotyping problem MEC/GI. *Algorithmica*, 56(3):283–296, March 2010.

[24] J. Wang *et al.* The diploid genome sequence of an Asian individual. *Nature*, 456:60–65, October 2008.

[25] D. A. Wheeler *et al.* The complete genome of an individual by massively parallel DNA sequencing. *Nature*, 452(7189):872–876, March 2008.

[26] J. Wu, J. Wang, and J. Chen. A parthenogenetic algorithm for single individual SNP haplotyping.

Engineering Applications of Artificial Intelligence, 22(3):401–406, April 2009.

[27] M. Xie, J. Wang, and J. Chen. A model of higher accuracy for the individual haplotyping problem based on weighted SNP fragments and genotype with errors. *Bioinformatics*, 24(13):i105–i113, July 2008.