BIOINFORMATICS

Vol. 24 ECCB 2008, pages i153–i159
doi:10.1093/bioinformatics/btn298

HapCUT: an efficient and accurate algorithm for the haplotype assembly problem

Vikas Bansal* and Vineet Bafna

Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA

ABSTRACT

Motivation: The goal of the haplotype assembly problem is to reconstruct the two haplotypes (chromosomes) for an individual using a mix of sequenced fragments from the two chromosomes. This problem has been shown to be computationally intractable for various optimization criteria. Polynomial time algorithms have been proposed for restricted versions of the problem. In this article, we consider the haplotype assembly problem in the most general setting, i.e. fragments of any length and with an arbitrary number of gaps.

Results: We describe a novel combinatorial approach for the haplotype assembly problem based on computing max-cuts in certain graphs derived from the sequenced fragments. Levy *et al*. have sequenced the complete genome of a human individual and used a greedy heuristic to assemble the haplotypes for this individual. We have applied our method HapCUTto infer haplotypes from this data and demonstrate that the haplotypes inferred using HapCUT are significantly more accurate (20–25% lower maximum error correction scores for all chromosomes) than the greedy heuristic and a previously published method, Fast Hare. We also describe a maximum likelihood based estimator of the absolute accuracy of the sequence-based haplotypes using population haplotypes from the International HapMap project.

Availability: A program implementing HapCUT is available on request.

Contact: vibansal@cs.ucsd.edu

1 INTRODUCTION

Humans are diploid organisms with two copies of each chromosome (except the sex chromosomes). The two chromosomes are homologous and differ at a number of sites, a large fraction of which correspond to single base pair differences commonly known as single nucleotide polymorphisms (SNPs). The genome sequence assembly for a chromosome is an arbitrary mix of the two haploid chromosomes. The two *haplotypes* (described by the combination of alleles at variant sites on a single chromosome) represent the complete information on DNA variation in an individual, and are very useful for a number of reasons: (i) different haplotypes (e.g. for genes containing multiple variants) can show different gene expression patterns and consequently varying susceptibility to disease; (ii) haplotype data from the HapMap project has allowed whole genome association studies to limit the search for disease-related genetic variants to a smaller set of 'tag' SNPs without an appreciable loss of power and (iii) knowledge of haplotype structure in humans has proved useful in understanding recombination patterns and identification of genes under positive selection (Altshuler *et al.*, 2005).

SNP chips can now interrogate upto a million SNPs in each individual, making the genotyping effort cost-effective. However, haplotype information remains difficult to obtain experimentally. Haplotypes are typically inferred from population SNP data using 'haplotype phasing' algorithms (Bafna *et al.*, 2003; Eskin *et al.*, 2003; Gusfield, 2002; Stephens *et al.*, 2001). These algorithms exploit linkage disequilibrium (LD); the correlation between alleles at neighbouring SNPs in a population to reconstruct haplotypes. The great variation in recombination rates and LD across the human genome limits the accuracy of these methods. A popular haplotype phasing method, PHASE (Stephens *et al.*, 2001), has a switch error rate of 5.4% for unrelated individuals from a European population (Marchini *et al.*, 2006); this corresponds to one switch error between the maternal and paternal chromosomes approximately every 50 kb.

An alternative approach to obtain haplotypic information is to reconstruct the two haplotypes for an individual using DNA sequence fragments. A sequence fragment that covers two or more variant sites in a genome can link or phase those variants. If a large fraction of the fragments are long enough to encompass multiple variant sites, and the shotgun sequencing has sufficient coverage to provide overlaps between fragments, the fragments can be assembled to reconstruct long haplotypes (see Fig. 1 for an illustration). The haplotype assembly problem, also known as the Single Individual Haplotyping problem, was introduced in the context of SNPs by Lancia *et al.* (2001) who described three optimization formulations for solving this problem. The problem has been shown to be computationally hard under various combinatorial objective functions (Bafna *et al.*, 2005; Cilibrasi *et al.*, 2005; Lancia *et al.*, 2001) [e.g. minimum fragment removal (MFR), minimum error correction (MEC), minimum SNP removal (MSR)]. Efficient algorithms exist for optimizing the MFR objective when all fragments are gapless (Lippert *et al.*, 2002; Rizzi *et al.*, 2002). Several heuristic methods have been proposed for handling gapped fragments (Panconesi and Sozio, 2004; Wang *et al.*, 2005). However, the lack of real sequencing data has limited the development and evaluation of computational methods for this problem.

Recently, Levy and colleagues (2007) announced the diploid genome sequence of a single human individual, referred to as *HuRef*. Interestingly, they show that a significant fraction of a population's genetic variation can be found in a single diploid genome. The coverage and quality of the HuRef sequence data also makes it amenable to 'haplotype assembly', i.e. separation of

*To whom correspondence should be addressed.

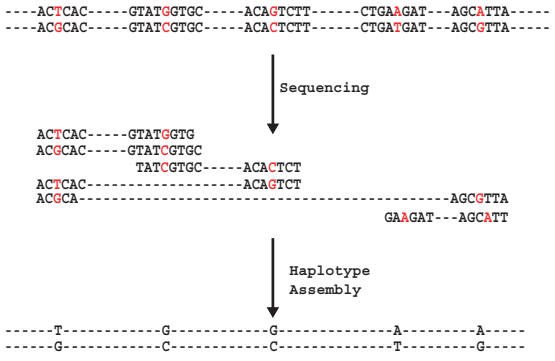


Fig. 1. A cartoon illustration of haplotype assembly. Each read originates from one of the two chromosomes. Reads that have different alleles at a common SNP can be inferred to come from different chromosomes. Reads that share an allele at a common variant site are derived from the same chromosome and can be linked together to reconstruct haplotypes. Note that only *variant* positions are relevant for haplotype assembly.

the maternal and paternal chromosomes. They have demonstrated that the quality of the data and the presence of paired-end reads allows haplotype assembly and have used a greedy heuristic for reconstructing haplotypes. The heuristic works well for the HuRef sequence data, but results indicate that it can be improved. Recently, we described HASH (Bansal *et al.*, 2008), a Markov Chain Monte Carlo (MCMC) algorithm for the haplotype assembly problem and demonstrated that the HuRef haplotypes based on HASH were much more accurate than those using the greedy heuristic. In related work, Li *et al.* (2003) and Kim *et al.* (2007) have described statistical methods (based on Gibbs sampling) for this problem, focusing on resolution in the presence of different sources of error. They apply their method to the genome of *Ciona intestinalis*.

In this article, we describe a novel combinatorial approach for the Haplotype Assembly problem based on a problem related to the MAX-CUT problem. Our algorithm HapCUTtries to minimize the MEC score of the reconstructed haplotypes by iteratively computing max-cuts in graphs derived from the sequenced fragments. Our algorithm is motivated by the HuRef sequence data and is applicable to sequenced fragments of any length with an arbitrary number of gaps. Using the HuRef sequence data, we demonstrate that our algorithm is significantly more accurate than the greedy heuristic of Levy *et al.* (2007). We also compare the performance of HapCUT with a previously proposed heuristic for this problem, namely Fast Hare (Panconesi and Sozio, 2004), and find that our algorithm consistently outperforms this heuristic. The MEC score of the haplotypes reconstructed using HapCUT is comparable to those using a MCMC algorithm (Bansal *et al.*, 2008) while being much faster to compute. While the problem of optimizing the MEC score is NP-hard even for gapless fragments (Cilibrasi *et al.*, 2005), and unlikely to admit an efficient algorithm, HapCUT represents a fast and accurate heuristic for haplotype assembly using real sequence data.

The article is organized as follows. In Section 2.1, we describe our optimization formulation, and follow it up with a solution based on computing weighted max-cuts in an associated graph (Section 2.2). In Section 3, we show the performance of our methods on a combination of simulated and real (HuRef) datasets. In Section 4, we describe a maximum likelihood (ML) estimator of the switch error rate of the HuRef haplotypes based on the CEU HapMap haplotypes and show that the HuRef haplotypes inferred using HapCUT have a low switch error rate ($1.1-1.4\%$).

2 METHODS

2.1 Preliminaries and optimization

The genome sequence assembly for a chromosome is a mix of the two haploid chromosomes. If we align all of the fragments to the assembly, certain sites (*columns* in the alignment) will show identical values (homozygous) for all fragments, while others will have different values (heterozygous) for different fragments. Note that heterozygous sites in the alignment could correspond to a single base pair (SNPs) or multiple base pairs, e.g. deletion/insertion variant. Sites that are homozygous are discarded, as they are not useful for haplotype phasing (Fig. 1). Likewise, all sites with more than two alleles are discarded, as all variant sites should be bi-allelic for a diploid genome. Arbitrarily re-labelling the variant alleles as 0 and 1, the input data can be represented as a ternary matrix X of size $m \times n$, where m is the number of fragments and n is the number of heterozygous sites. The i-th fragment (row i of X) is described by a ternary string $X_i \in \{0,1,-\}^n$, where '–' corresponds to the variant loci not covered by the fragment. The objective of haplotype assembly is to reconstruct the two haplotypes, i.e. determine the combination of alleles present on a single chromosome at the heterozygous sites. The haplotypes can be represented as an unordered pair $H=(h_1,h_2)$ of binary strings, each of length n. Since we only consider sites that are heterozygous in the individual genome, h_2 is constrained to be the bit-wise complement of h_1.

In the absence of any errors, the rows of the fragment matrix can be partitioned into two disjoint sets such that every column is homozygous in each set (Lancia *et al.*, 2001). Further, the consensus values can be used to reconstruct the two haplotypes. However, such a perfect bi-partition is not possible when there are errors in the fragment matrix. In the presence of errors, the objective of haplotype assembly is to find a bi-partition or a pair of haplotypes that minimizes some objective function. Under the MEC criterion, the objective is to change the smallest number of entries in the fragment matrix such that the resulting matrix admits a perfect bi-partition. The MEC objective is also equivalent to finding a pair of haplotypes H for which the MEC score of the fragment matrix $\mathrm{MEC}(X,H)$ is minimum.

If $d(X_i,h)$ denotes the number of mismatches between the fragment X_i and haplotype h (ignoring the '$-$' in X_i), then

$$\mathrm{MEC}(X_i,H)=\min\{d(X_i,h),d(X_i,\bar{h})\}$$

and the overall score is given by

$$\mathrm{MEC}(X,H)=\sum_i \mathrm{MEC}(X_i,H)$$

This leads to the natural parsimony-based optimization problem of computing haplotypes with minimum MEC score. For notational convenience, we will denote the error for a haplotype pair H as $\mathrm{MEC}(H)$, whenever X is implicit. The MEC optimization can also be related to a natural *likelihood*-based formulation. Assume that q is the probability that a base is miscalled. More generally, we can assume knowledge of a vector q, where $q_i[j]$ is the probability of base miscall at fragment i, position j. The likelihood of data, given the haplotype pair H is given by

$$\Pr(X\,|\,H,q)=\prod_i \Pr(X_i|H,q)=\prod_i\left(\frac{\Pr(X_i|h,q)+\Pr(X_i|\bar{h},q)}{2}\right)$$

To compute the likelihood of a fragment, define $\delta_h[i,j]=1$ if $(X_i[j]=h[j])$, and $\delta_h[i,j]=0$ otherwise. Then,

$$\Pr(X_i|h,q)=\prod_{j:X_i[j]\neq\text{`}-\text{'}}(1-q_i[j])\delta_h[i,j])+q_i[j](1-\delta_h[i,j])$$

In this article, we focus on designing an algorithm for the MEC objective function. Other objective functions for haplotype assembly have previously been proposed, such as MFR where the objective is to remove the smallest number of fragments to make the fragment matrix error free, and MSR which models false variant sites. However, the simplest (and most common) source of error is due to base miscalling and the MEC objective serves as a good model for this type of error. Moreover, MEC can also indirectly model other sources of error, e.g. a haplotype assembly with a low MEC score is also likely to be good under the MFR objective.

2.2 MEC optimization using max-cuts

Among the various objective functions for the haplotype assembly problem, MEC seems to be the most difficult to optimize. While MFR and MSR can be solved optimally for gapless fragments, finding the optimal MEC solution has been shown to be NP-hard even for gapless fragments (Cilibrasi *et al.*, 2005; Lippert *et al.*, 2002). MEC has been shown to be $O(\log n)$-approximable in the general case by Panconesi and Sauzio (2004), who also describe a heuristic Fast Hare for the problem. Here, we provide a formulation for MEC optimization based on graph cuts, which leads to a simple but effective algorithm.

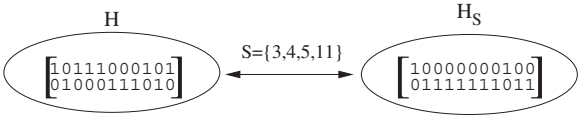
Given a fragment matrix X, and a haplotype pair H, we define the graph $G_X(H)$, with vertices defined by columns of X. We abuse notation slightly by referring to the vertex set as X. The set E_H of edges of this graph is defined by pairs of columns that are linked by some fragment. Let $X_i[j,k]$ and $H[j,k]$ represent the fragment i and haplotype pair H, respectively, when restricted to the pair of columns (j,k). The weight of the edge $(j,k) \in E_H$ connecting columns j,k is defined as

$$w_H(j,k) = \|\{i \,|\, \mathrm{MEC}(X_i[j,k], H[j,k]) = 1\}\| - \|\{i \,|\, \mathrm{MEC}(X_i[j,k], H[j,k]) = 0\}\|$$

Informally, the weight of the edge (j,k) is the number of fragments inconsistent with the current phase between the pair minus the number of fragments consistent with the phase $H[j,k]$. In other words, the weight represents the 'weakness' of the phasing between columns i and j. A *Cut* in the graph $G_X(H)$ is defined by a subset $S \subseteq X$ of vertices. The weight of a cut S in $G_X(H)$ is given by

$$w_H(S) = \sum_{j \in S, k \in X-S} w_H(j,k)$$

Given a haplotype pair H, and a cut S in $G_X(H)$, the haplotype obtained by flipping the values of the columns in S is denoted by H_S, as illustrated in the example below:



Such transformations are effective for improving the MEC score, as exemplified by the following theorem.

THEOREM 1. *Let X be a fragment matrix with each fragment of length 2. For any haplotype pair H, let $S \subseteq X$ be a positive weighted cut $w_H(S) > 0$ in the graph $G_X(H)$. Then*

$$\mathrm{MEC}(H_S) = \mathrm{MEC}(H) - w_H(S) < \mathrm{MEC}(H)$$

If S is a MAX-CUT in the graph $G_X(H)$, then H_S is an optimal MEC *solution for X.*

PROOF. Consider a cut S in the graph $G_X(H)$. Let H_S be the haplotype obtained by flipping the columns S of H. Clearly, $\mathrm{MEC}(H) - \mathrm{MEC}(H_S)$ is equal to the value of this cut $w_H(S)$. The maximum value of this difference is reached when the cut is a max-cut and therefore H_S is an optimal MEC solution.

The above theorem implies that given a current haplotype pair H, *any* positive weight ($w_H(S) > 0$) cut leads to a haplotype (H_S) with a lower MEC score. This can be repeated iteratively resulting in haplotypes with decreasing MEC scores. If we can compute the MAX-CUT in a single step, we can find the optimal MEC solution, however this is not necessary. Based on this observation, an algorithm that looks for positive cuts in $G_X(H)$ can be used to optimize the MEC score as below.

Procedure HapCUT

Initialization: Choose an initial haplotype configuration H^1 randomly.
Iteration: For $t = 1, 2, \ldots$

1. Construct the graph $G_X(H^t)$
2. Compute a cut S in $G_X(H^t)$ such that $w_H(S) \geq 0$
3. If $\mathrm{MEC}(H^t_S) \leq \mathrm{MEC}(H^t)$, $H^{t+1} = H^t_S$
4. Else $H^{t+1} = H^t$

The iterative procedure HapCUT is run until we can no longer get an improvement in the MEC score. While Theorem 1 holds only when all fragments have length 2 (and also for fragments of length 3 as we show in the next section), the algorithm HapCUT as described above works for arbitrary sized fragments. In order to ensure that HapCUT has good performance on real data, it is important to be able to compute high-scoring cuts in the graph $G_X(H)$. First, we show how the edge weights of the graph $G_X(H)$ can be weighted appropriately for long fragments.

2.3 Assigning weights to edges of $G_X(H)$

In the previous section, we described a simple formula to assign a weight to each edge of the graph $G_X(H)$. This formula gives disproportionately more weight to longer fragments, i.e. a fragment of length k contributes a total absolute weight of $\binom{k}{2}$ to the graph. The weighting scheme can be modified to ensure that Theorem 1 also holds for fragments of length 3. We simply scale the contribution of the fragment to each edge by 1/2. Now, a fragment of length 3 can have a MEC of 0 or 1. An MEC of 0 corresponds to the three vertices (columns of the fragment) being on the same side of the cut and therefore contributing 0 to the cut value. An MEC of 1 corresponds to two of the vertices being on one side and therefore the fragment contributes exactly 1 to the cut after scaling. As the scaling of $1/(k-1)$ for a fragment of length k is consistent with the weights for fragments of length 2 and 3, we adopt it for computing the edge weights of arbitrary length fragments. Results on real data indicate that this works well, even though we do not know of a scaling under which Theorem 1 holds for arbitrary length fragments.

2.4 Computing max-cuts

The problem of computing a maximum-weighted cut is known to be NP-complete (Garey and Johnson, 1979), even when all edge weights are restricted to be one. However, we only need to find a positive-weighted cut in order to improve the MEC score. Simple heuristics can find good cuts if all weights are positive. Indeed, a greedy heuristic (Sahni and Gonzalez, 1974) will give a cut which has at least 0.5 of the total weight of the edges of the graph. When the MEC score of H is poor and far away from the optimal MEC value (e.g. for a random haplotype pair), most of the edges of the graph $G_X(H)$ have positive weights and finding a positive-weighted cut is easy. However, when the MEC score of H is close to the optimum, most of the edges of the graph $G_X(H)$ have negative weights, and the greedy algorithm is not guaranteed to find a positive-weight cut. On the other hand, presence of a highly negative weight edge between two vertices s and t of the graph also implies that a positive-weight cut is unlikely to separate s and t. Therefore, for the purpose of computing a positive-weight cut, we can 'contract' the edge (s,t). We use a two-step greedy algorithm for computing a max-cut in $G_X(H)$. First, we find a cut where most of the negative weight edges do not go across the cut. In the second step, we move vertices from

one side of the cut to the other if this improves the weight of the cut. The complete greedy heuristic is described below:

Greedy-Cut($G_X(H)$)
Initialization: BestCut = $-\infty$
Iteration: Iterate $O(m \log m)$ times

1. Chose an edge (s,t) of the graph uniformly at random
2. Initialize $S_1 = \{s\}$ and $S_2 = \{t\}$
3. While $S_1 \cup S_2 \neq V$
   (a) For each vertex $v \notin S_1 \cup S_2$ compute the score $A(v) = \sum_{s_1 \in S_1} w_H(v, s_1) - \sum_{s_2 \in S_2} w_H(v, s_2)$
   (b) Let $v_{\max}$ be the vertex for which $|A(v)|$ is maximum
   (c) If $A(v_{\max}) < 0$, $S_1 = S_1 \cup v$
   (d) else if $A(v_{\max}) > 0$, $S_2 = S_2 \cup v$
   (e) else add v uniformly at random to S_1 or S_2
4. **repeat**
   (a) OldCut $= w_H(S_1)$
   (b) If $v \in S_1$ and $A(v) > 0$, move v from S_1 to S_2
   (c) If $v \in S_2$ and $A(v) < 0$, move v from S_2 to S_1

   until $w_H(S_1) \leq$ OldCut
5. If $w_H(S_1) >$ BestCut, BestCut = $w_H(S_1)$

Final: Return BestCut

The first phase of the above algorithm (Step 1–3) is designed to find a cut in which the highly negative weight edges do not cross the cut. The cut is initialized using an edge of the graph and the algorithm is repeated enough times to make sure that every edge is considered. Step 4 by itself is exactly the well-known Greedy algorithm for computing max-cuts (Sahni and Gonzalez, 1974).

3 RESULTS

We used the filtered HuRef data from Levy *et al.* (2007) to evaluate our algorithm HapCUT. The data contains a total of 1.85 million heterozygous variants for the 22 autosomes. As a typical example, chromosome 22 contained 24 968 variants encoded by 103 356 fragments. Only 53 279 of these cover more than one variant and are therefore useful for haplotype assembly. Of these fragments 18 119 correspond to mate-pairs. The chromosome is partitioned into 609 dis-connected variant 'blocks' or connected component based on the links between variants in addition to 921 isolated variants. These blocks provide large haplotypes, clearly illustrating the power of this haplotype assembly. However, as the length of our haplotypes is predetermined by the connected components, we do not discuss this further, referring the interested reader to Levy *et al.* (2007). The haplotypes for each of these blocks are assembled independently. The average number of calls for a variant is 6.7 (see Fig. 2a for distribution of the number of variant calls per fragment). A fragment spans 9.67 variants but has only 3.25 variant calls on the average (see Fig. 2b for distribution of the difference between span and length). This clearly indicates the highly 'gapped' nature of the fragment data.

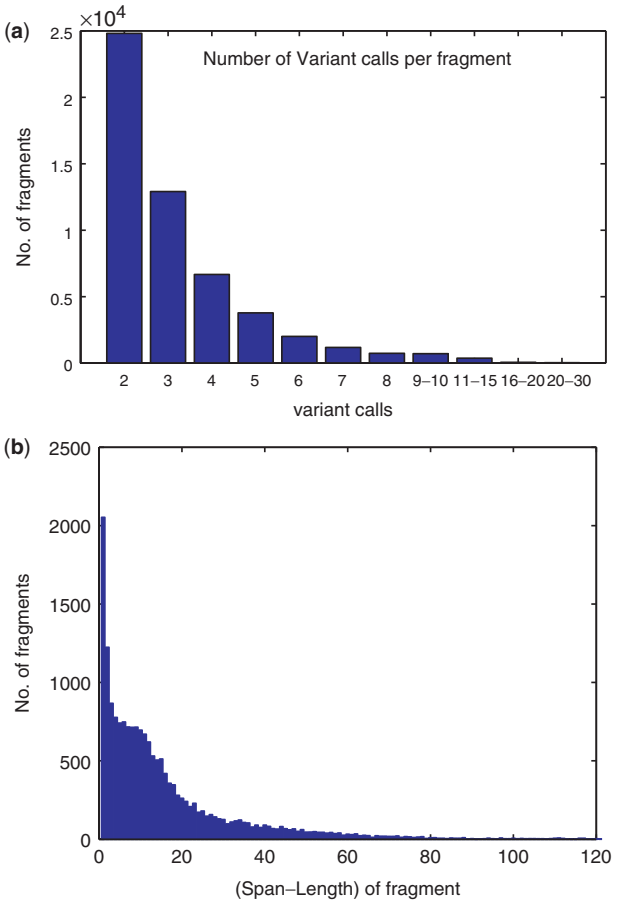


Fig. 2. (**a**) Distribution of number of variant calls per fragment. Fragments that cover only one variant site (about half of the total fragments) are not shown. (**b**) Distribution of the difference between the 'span' of a fragment (difference between the last variant call and the first variant call) and the 'length' (No. of variant calls). Fragments with span equal to length (35159/53278 fragments) are not included to improve clarity. Both plots are for chromosome 22 from HuRef genome.

3.1 MEC scores for HuRef chromosomes

We ran HapCUT for each of the HuRef chromosomes. For each chromosome, the algorithm was initialized with a randomly chosen haplotype. We found that the MEC score improves as we iteratively compute cuts and change the haplotypes. Most of the improvement in the MEC score happens in the first few iterations with no further improvement after 40–50 iterations. We compared the MEC scores for the HuRef data using four different algorithms: (i) the greedy heuristic of Levy *et al.* (2007), (ii) Fast Hare (implemented as described in Panconesi and Sozio, 2004), (iii) the MCMC algorithm HASH (Bansal *et al.*, 2008) and (iv) our algorithm HapCUT (see Fig. 3).

HapCUT performs significantly better than the greedy heuristic (the MEC scores are 20–25% lower for all chromosomes) and very similar to HASH. The performance of the heuristic Fast Hare is generally worse than that of the greedy heuristic. For a few connected components, the MEC score for HASH was slightly lower than that of HapCUT (run for 100 iterations). On all of these cases, a greedy choice of the cut did not improve the MEC score

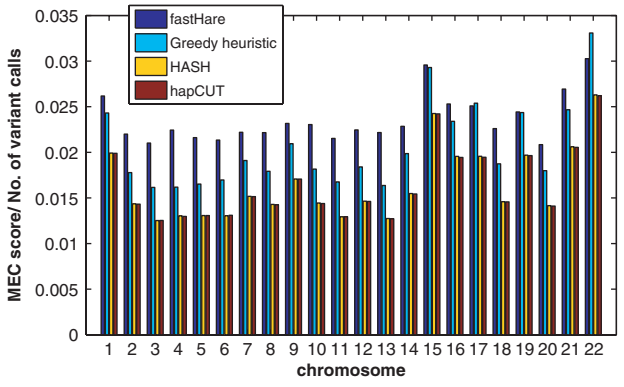


Fig. 3. MEC scores (divided by the number of variant calls) for the HuRef chromosomes for four different methods. The performance of HapCUT and HASH is comparable, and significantly better than the greedy heuristic and Fast Hare.

(data not shown). Clearly, an MCMC sampling approach has the advantage of being able to make sub-optimal choices and thereby sample a slightly better haplotype. On the other hand, for some of the chromosomes, we observed that the total MEC score of HapCUT was better than that of HASH. A possible explanation is that while the objective of HapCUT is to find the optimal MEC solution, HASH is geared towards finding the maximum likelihood solution. The two may be different when fragments have lengths greater than two. HapCUT also offers the advantage of fast computation time in comparison to MCMC sampling algorithms such as HASH. For chromosome 22, HapCUT takes less than 30 min to compute the MEC score while HASH takes more than 10 h. For all chromosomes, HapCUT was an order of magnitude faster than HASH (data not shown).

3.2 Simulations using HuRef data

We tested the performance of HapCUT on simulated data generated using the HuRef chromosomes. The fragment matrix for a chromosome was suitably modified to make it 'error free' or perfectly consistent with a particular haplotype. To generate a fragment matrix with error rate of ε ($0 \leq \varepsilon \leq 0.5$), each variant call in the fragment matrix was flipped with probability ε. We ran HapCUT on this modified fragment matrix and compared the reconstructed haplotypes with the true haplotypes. In Figure 4a, we plot the best MEC score (scaled by the number of variant calls) against the simulated error rate, i.e. the fraction of variant calls that were flipped. We observe that the MEC score is always less than the number of flipped calls and ratio of the MEC score to the number of flips decreases as the error rate increases. This is to be expected, because as the number of flipped variant calls increases, some calls might become consistent with a different (lower MEC) haplotype.

Flipped base calls could also result in errors in the reconstructed haplotypes. If the depth of coverage is low, very little can be done to recover from the error. Also, if ε is high, the optimal haplotype could well be different from the one we started with. However, we expect that at high depths of coverage and low-error rates, a correct haplotype can be recovered accurately. In Figure 4b we plot the switch error (number of switches between the original haplotype and the reconstructed haplotype) against the depth of coverage, i.e. for a particular value on the x-axis, we ignore variants with

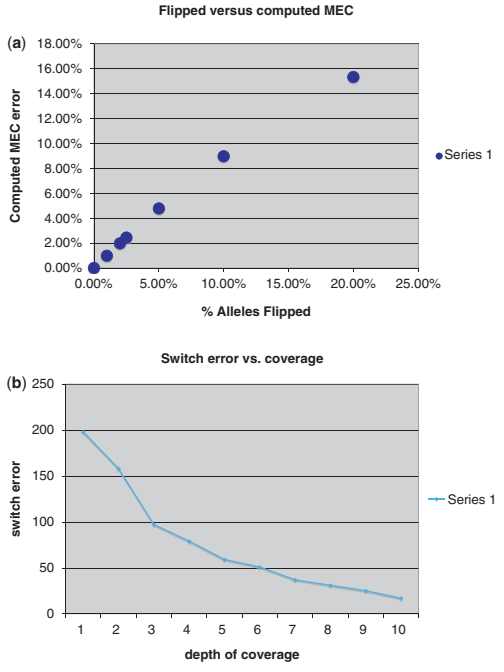


Fig. 4. (**a**): Number of simulated versus estimated errors and (**b**) haplotype switch error as a function of depth of coverage (number of calls for a variant), for $\varepsilon = 0.02$. The switch error decreases with increasing depth.

coverage below that value for computing the switch error. One can clearly see that as the depth increases, the switch error decreases from 198 (1.11% of the sites) to 17 (0.098%).

Not surprisingly, we also find that the flipped variants are largely the same as the one that mismatch the computed haplotype. Of the 3321 fragments for which some variant call was flipped, we identified 3213 that also mismatched against the computed haplotype. An additional 146 fragments that did not contain flipped alleles mismatched with the computed haplotype. Overall, these results indicate the robustness of our solution to errors in the data.

4 ML ESTIMATE OF THE ACCURACY OF HAPLOTYPE ASSEMBLY

The MEC score measures the consistency of the haplotype assembly with the fragment matrix. However, we are also interested in the absolute accuracy of the inferred haplotypes. The absolute accuracy can be measured using the switch error rate, which is the fraction of adjacent pairs of sites in the HuRef individual whose phase is incorrect. We have used the phased haplotypes from the HapMap project (Altshuler *et al.*, 2005) to obtain a ML estimate of the absolute accuracy of the HuRef haplotypes. As the HuRef individual is European in origin, we compare the HuRef haplotypes H against the set of 120 CEU HapMap haplotypes restricted to the set of SNPs heterozygous in the HuRef individual. The two HuRef haplotypes are a mosaic of the population haplotypes and a direct comparison of the full haplotypes with the HapMap haplotypes is not possible. We compute the likelihood of the haplotype H conditional on the CEU haplotypes and as a function of the switch error rate.

Consider a pair of adjacent SNPs i,j heterozygous in the HuRef individual. Let f_{00},f_{01},f_{10} and f_{11} be the frequencies of the four pairs in the HapMap CEU sample. If H_t denote the true HuRef haplotypes (unobserved), the likelihood of the phasing $H_t[i,j]$ being $(00,11)$ is given by

$$\Pr(H_t[i,j]=(00,11))=\frac{f_{00}f_{11}}{f_{00}f_{11}+f_{01}f_{10}}$$

We can similarly compute the probability $\Pr(H_t[i,j]=(01,10))$. For a *switch error rate* ε_s, the likelihood of the phasing between the pair (i,j) is given by

$$L_H(i,j)=(1-\varepsilon_s)\Pr(H_t[i,j]=H[i,j])+\varepsilon_s\Pr(H_t[i,j]\neq H[i,j])$$

The switch errors between H and H_t are a result of sequencing errors and likely to be distributed independent of LD. Therefore we can assume the switch error rate ε_s to be uniform for all pairs. We approximate the likelihood of the full haplotype H as

$$L_H=\prod_{i=1}^{n-1}L_H(i,i+1)$$

The haplotype likelihood (L_H) is a function of the switch error rate ε_s and the LD in the HapMap haplotypes. If there are very few switch errors between H and H_t, then the likelihood function L_H is expected to be maximum for values of ε_s close to 0. As the number of switch errors increases, the contribution of the second term in the likelihood $L_H(i,j)$ increases and therefore the ML estimate of ε_s should increase. We have used the HapMap haplotypes to evaluate if the ML estimator is a good estimate of the switch error rate. The haplotypes for one of the 60 CEU individuals was chosen to represent the true haplotypes H_t. We then simulated switch errors randomly with varying switch error rates ε_s (0–0.05) to generate the haplotype pair H. The likelihood function L_H was then plotted for different values of the error rate (see Fig. 5 for likelihood curves for four different values of ε_s). From the likelihood curves, we see that the ML estimate is a good estimate of the switch error rate with a tendency to slightly over-estimate the switch error rate. For a simulated switch error rate of 0.01, the likelihood was maximum for $\varepsilon_s=0.013$. Similarly for an error rate of 0.02, the ML estimate was 0.022.

We then plotted the haplotype likelihood (L_H) for the HuRef haplotypes (inferred using HapCUT) as a function of the switch error rate ε_s (see Fig. 6 for a plot for chromosome 22). The likelihood curve is flat in the region 0.013–0.015 with a maxima at 0.014. From this, we can infer that the switch error rate of the HuRef haplotypes is slightly more than 1% but not more than 1.4%. In comparison, the switch error rate of haplotypes inferred from CEU population data is 5.4% for unrelated individuals and 0.53% for parent–child trios (Marchini *et al.*, 2006).

5 DISCUSSION

With the advent of high-throughput and cost-effective sequencing technologies, personalized genomics is a distinct possibility. The availability of individual genomic sequences enables the identification of structural and copy-neutral variation, but also the separation of the maternal and paternal chromosomes.

Here, we tackle the problem of haplotype assembly, developing a novel combinatorial approach for this problem. Our approach is

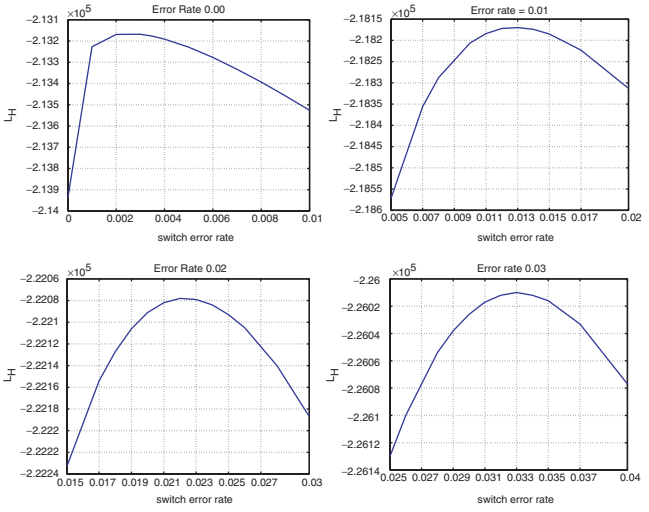


Fig. 5. Haplotype log-likelihood curves for four different values of the switch error rate.

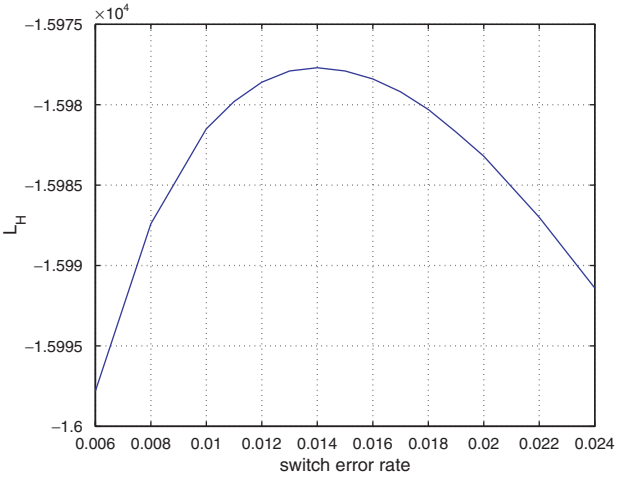


Fig. 6. The log-likelihood curve for the HuRef haplotypes for chromosome 22.

fundamentally different from previous methods for this problem [e.g. greedy heuristic (Levy *et al.*, 2007) and Fast Hare (Panconesi and Sozio, 2004)]. Previous methods attempt to reconstruct the haplotypes by clustering the fragments in a sequential fashion. On the other hand, we use the overlap structure of the fragment matrix to compute max-cuts that allow our method to find better haplotypes. Given the NP-hardness of the MEC problem, our method is also a heuristic and not guaranteed to find a provably good solution. However, results on the HuRef sequence data and comparison to an MCMC algorithm suggest that the algorithm HapCUT is able to find solutions close to the optimum MEC solution.

We have shown that HapCUT does as well as an MCMC algorithm, HASH, that we have recently developed, while being much faster. The MCMC algorithm uses graph cut computations to construct the Markov chain used for sampling the haplotype space. In comparison, HapCUT uses max-cut computations in

an associated graph to greedily move towards the optimal MEC solution. A cut corresponds to a subset of columns of the fragment matrix and choosing the right cuts is crucial for optimizing MEC and sampling the haplotype space. To develop the intuition of choosing cuts to move across solutions, we have performed a rigorous analysis of convergence on a representative family of fragment matrices (V.B. Bansal and V.B. Bafna, unpublished data). Our analysis is based on novel techniques relying upon graph conductance, and coupling arguments. As many problems in bioinformatics admit combinatorial and stochastic sampling-based solutions, such techniques could be of independent interest.

Conflict of Interest: none declared.

REFERENCES

Altshuler,D. *et al*. (2005) A haplotype map of the human genome. *Nature*, **437**, 1299–1320.

Bafna,V. *et al*. (2003) Haplotyping as perfect phylogeny: a direct approach. *J. Comput. Biol.*, **10**, 323–340.

Bafna,V. *et al*. (2005) Polynomial and APX-hard cases of individual haplotyping problems. *Theor. Comput. Sci.*, **335**, 109–125.

Bansal,V. *et al*. (2008) An MCMC algorithm for haplotype assembly for whole-genome seqeuence data. *Genome Research*.

Cilibrasi,R. *et al*. (2005) On the complexity of several haplotyping problems. In Casadio, R. and Myers, G. (eds), *WABI*, Vol. 3692 of *Lecture Notes in Computer Science*, Springer, Berlin, pp. 128–139.

Eskin,E. *et al*. (2003) Efficient reconstruction of haplotype structure via perfect phylogeny. *J. Bioinform. Computat. Biol.*, **1**, 1–20.

Garey,M.R. and Johnson,D.S. (1979) *Computers and Intractability: a Guide to the Theory of NP-completeness*. W.H. Freeman and Company, New York, NY, USA.

Gusfield, D. (2002) Haplotyping as perfect phylogeny: conceptual framework and efficient solutions (Extended abstract). In *Proceedings of the Sixth Annual International Conference on Computational Molecular Biology (RECOMB)*. ACM, New York, NY, USA, pp. 166–175.

Kim,J. *et al*. (2007) Diploid genome reconstruction of Ciona intestinalis and comparative analysis with Ciona savignyi. *Genome Res.*, **17**, 1101.

Lancia,G. *et al*. (2001) SNPs problems, complexity and algorithms. In *Proceedings of the Nineth Annual European Symposium on Algorithms (ESA)* Springer, Berlin, pp. 182–193.

Levy,S. *et al*. (2007) The diploid genome sequence of an individual human. *PLoS Bio.*, **5**, e254, doi:10.1371/journal.pbio.0050254.

Li,L. *et al*. (2003) Haplotype reconstruction from *SNP* alignment. In *RECOMB'03: Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology*. ACM Press, New York, NY, USA. pp. 207–216.

Lippert,R. *et al*. (2002) Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Brief. in Bioinform.*, **3**, 23–31.

Marchini,J. *et al*. (2006) A comparison of phasing algorithms for trios and unrelated individuals. *Am. J. Hum. Genet.*, **78**, 437–450.

Panconesi,A. and Sozio,M. (2004) Fast Hare: a fast heuristic for single individual SNP haplotype reconstruction. In Jonassen I. and Kim J. (eds), *WABI*, Vol. 3240 of *Lecture Notes in Computer Science*. Springer, Berlin, pp. 266–277.

Rizzi,R. *et al*. (2002) Practical algorithms and fixed-parameter tractability for the single individual SNP haplotyping problem. In *Proceedings of the Second International Workshop on Algorithms in Bioinformatics (WABI)*. Springer, Berlin, pp. 29–43.

Sahni,S. and Gonzalez,T. (1974) P-complete problems and approximate solutions. In *Proceedings of the 15th Annual Symposium on Switching and Automata Theory*. IEEE, pp. 28–32.

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

Wang,R.S. *et al*. (2005) Haplotype reconstruction from SNP fragments by minimum error correction. *Bioinformatics*, **21**, 2456–2462.