Gene 518 (2013) 2–6

Contents lists available at SciVerse ScienceDirect

Gene

journal homepage: www.elsevier.com/locate/gene

Hap-seqX: Expedite algorithm for haplotype phasing with imputation using sequence data

Dan He [a,*], Eleazar Eskin [b]

[a] IBM T.J. Watson Research, Yorktown Heights, NY, USA
[b] Dept. of Comp. Sci., Univ. of California Los Angeles, Los Angeles, CA 90095, USA

ARTICLE INFO

Available online 23 December 2012

Keywords:
Haplotype phasing
Imputation
Dynamic programming
Hidden Markov Model

ABSTRACT

Haplotype phasing is one of the most important problems in population genetics as haplotypes can be used to estimate the relatedness of individuals and to impute genotype information which is a commonly performed analysis when searching for variants involved in disease. The problem of haplotype phasing has been well studied. Methodologies for haplotype inference from sequencing data either combine a set of reference haplotypes and collected genotypes using a Hidden Markov Model or assemble haplotypes by overlapping sequencing reads. A recent algorithm Hap-seq considers using both sequencing data and reference haplotypes and it is a hybrid of a dynamic programming algorithm and a Hidden Markov Model (HMM), which is shown to be optimal. However, the algorithm requires extremely large amount of memory which is not practical for whole genome datasets. The current algorithm requires saving intermediate results to disk and reads these results back when needed, which significantly affects the practicality of the algorithm. In this work, we proposed the expedited version of the algorithm Hap-seqX, which addressed the memory issue by using a posterior probability to select the records that should be saved in memory. We show that Hap-seqX can save all the intermediate results in memory and improves the execution time of the algorithm dramatically. Utilizing the strategy, Hap-seqX is able to predict haplotypes from whole genome sequencing data.

1. Introduction

Haplotypes are the sequences of SNPs (Single Nucleotide Polymorphisms) present on one chromosome. Since there are two chromosomes for each individual, there are two corresponding haplotypes. Haplotypes are used for many genetic analyses including imputation (Kang et al., 2010; Li et al., 2010; Marchini et al., 2007a) and association studies (Beckmann, 2001). Another important application of haplotypes is to estimate the relatedness of individuals via IBD (identity-by-descent) (Browning and Browning, 2010, 2011; Gusev et al., 2009), which quantifies the regions of SNPs inherited from the same ancestors. Many methods to reconstruct haplotypes from sequencing data have been proposed, which fall into two categories: haplotype assembly using high throughput sequencing data and haplotype imputation using reference haplotypes.

The problem of haplotype assembly is to assemble haplotypes from the genetic information, more specifically, SNPs, contained in the reads generated by high throughput sequencing technique. As both chromosomes are sequenced at the same time, the reads from both chromosomes are mixed and the sources of each read is unknown. Therefore the haplotype assembly methods (Bansal and Bafna, 2008; Bansal et al., 2008; He et al., 2010; Levy et al., 2007) first partition the reads into two separate sets of reads, one for each chromosome, then assemble each haplotype using the SNPs from the corresponding set of reads, such that the assembled haplotypes are most consistent with the set of mixed reads. An objective function MEC (Minimum Error Correction) is widely used, where for any pair of haplotypes, we can obtain the minimum number of errors when we assign the reads to either haplotype. And the goal is to assemble a pair of haplotypes that minimizes MEC. The problem is shown to be NP-hard when the reads contain sequencing errors, which is typically the case. Statistical algorithms such as MCMC (Bansal et al., 2008) and MaxCut (Bansal and Bafna, 2008), and optimal algorithms such as dynamic programming and MaxSAT (He et al., 2010) have been proposed.

Due to the limit of the sequencing techniques, such as certain regions may not be covered by the reads, fully assembling haplotypes is usually not practical. He et al. (2010) showed that even under the case when the reads are uniformly sequenced, a very high coverage ratio is needed in order to fully assemble the haplotypes.

Another type of method to reconstruct haplotypes is haplotype imputation (Marchini et al., 2007b) using reference haplotypes. HapMap (2009) and 1000 Genome Projects (2010) provide collections of very accurate haplotypes for thousands of individuals. These haplotypes are used as reference haplotypes where a given haplotype can be represented

Abbreviation: HMM, Hidden Markov Model; SNP, Single Nucleotide Polymorphism; IBD, identity-by-descent; MEC, Minimum Error Correction; MCMC, Markov chain Monte Carlo; LD, linkage disequilibrium; MIR, Most likely Imputation based on Reads.

* Corresponding author. Tel.: +1 9149452315; fax: +1 9149454217.
E-mail addresses: dhe@us.ibm.com (D. He), eeskin@cs.ucla.edu (E. Eskin).

0378-1119/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.gene.2012.11.093

as a mosaic of these reference haplotypes. Hidden Markov Models have been applied to identify the most likely mosaic representation. Therefore when missing SNPs are present in the given haplotype, we can impute them using the corresponding SNPs in the mosaic representation, based on LD between neighboring SNPs.

Haplotype imputation has very high accuracy and usually requires low coverage. However, it is only able to impute common SNPs and is not able to handle rare SNPs. A recently proposed method, Hap-seq (He et al., 2012), uses both sequencing data and reference haplotypes to phase haplotypes where both common and rare SNPs can be handled. Hap-seq utilizes an objective function, MIR (Most likely Imputation based on Reads), to maximize the joint likelihood of the reads and the reference haplotypes. MIR for a pair of haplotypes approximates the conditional probability of the pair of haplotypes given the set of reads and the set of reference haplotypes. The goal that is to find a pair of haplotypes such that the conditional probability, or MIR, is maximized.

Hap-seq uses both dynamic programming and HMM and is shown to be optimal for reads containing no more than k SNPs with complexity $O(n\times m^2\times 4^k)$. Therefore k needs to be small for Hap-seq to be computationally feasible, which is typically the case since the reads are short and contain few SNPs. The algorithm has time and space complexities both as $O(n\times m^2\times 4^k)$, where n is the length of haplotype, m is the number of the reference sequences and k is the number of maximum SNPs that a read can have, as at each position, all pairs of length k binary strings need to be enumerated and the HMM states need to be extended. The algorithm suffers from extremely large memory usage when m is around 100, which is typical for imputation. Therefore the intermediate results are written to disk and read back when needed. The heavy I/O also affects the execution time of the algorithm significantly.

In this work, we proposed an expedited dynamic programming algorithm Hap-seqX, where we aim to reduce the space complexity from $O(m^2\times 4^k)$ by recording only the binary strings and HMM states with high posterior probability. Therefore all the intermediate results can be saved into memory which not only solves the memory issue, but also improves the running time significantly. More details will be given in the next section. As shown in our experiments, Hap-seqX has high probability to obtain optimal results and even for cases it misses the optimal results, it usually still achieves better results than IMPUTE, the state-of-the-art imputation algorithm.

2. Methods

We follow the same notion of the objective function of MIR (Most likely Imputation based on Reads) as the work of He et al., 2012, which is defined as

$$MIR(r_1, r_2|R, H) = P(r_1, r_2|R) \times P(r_1|H) \times P(r_2|H)$$

where r_1, r_2 are a pair of haplotypes, $P(r_1, r_2|R)$ is the posterior probability of them given the set of reads R, $P(r_1|H)$ is the posterior probability of r_1 given the set of reference haplotypes H. Our goal is to find a pair of haplotypes r_1, r_2 such that the above objective function is maximized. Obviously a naive solution is to enumerate all pairs of r_1, r_2 with complexity $O(4^n)$ where n is the length of the haplotypes. The solution is infeasible even for very small n.

Our algorithm Hap-seqX runs a dynamic programming process, which is based on the idea of *partial haplotype*, defined as the prefix of the full length haplotypes. For example, if the full length haplotype is 001101001 (haplotypes can be represented as binary strings as the SNPs in the pair of haplotypes are either homozygous or heterozygous), strings such as 001, 0011, 001101 etc. are all partial haplotypes of this full length haplotype.

Starting from the first position of the haplotypes, all pairs of length-k binary strings are enumerated which are partial haplotypes of length k. MIR for these binary strings are computed and recorded in the dynamic programming matrix. In the next step, again all pairs of length-k binary strings are enumerated, which are the suffixes of the partial haplotypes. To compute MIR for these new binary strings using MIR from the previous step, we require the length-$(k-1)$ suffix of the previous binary strings to be identical to the length-$(k-1)$ prefix of the current binary string in order to avoid saving all combinations of (pairs of binary strings, pairs of reference sequences) at each position. An example of the dynamic programming algorithm is shown in Fig. 1.

Give the definition of MIR as $P(r_1, r_2|R, H)$, we need to consider the likelihood of reads and the likelihood of the reference haplotypes at the same time at each position. The likelihood of reads at position i for r_1, r_2, $R(i,r_1,r_2)$, is computed according to the number of mismatches between the set of reads R_i completely covered by the pair of length-k binary suffixes r_1, r_2 starting at position i and r_1, r_2. Given the sequencing error rate e, and mismatch number as E, the likelihood of reads can be computed as $e^E\times(1-e)^{K-E}$, where K is the total number of bits for the set of reads R_i.

The likelihood of imputation is usually computed by an HMM. We can see that if the transition is on the same haplotype, the transition probability is high. On the contrary, if the transition is from one haplotype to another haplotype, the transition probability is low. Both types of transition probabilities can be obtained from IMPUTE website. As we consider a pair of haplotypes, the transition will be a combination of all possible transitions for both haplotypes. Therefore, the total number of transition needs to be considered is $O(m^4)$ where m is the number of reference haplotypes. This complexity can be reduced to $O(m^2)$ by pre-computing the transition probabilities (Li et al., 2010).

As the likelihood of imputation is usually computed by HMM, we need to define conduct HMM computation at the same time when we extend the partial haplotypes: namely when we extend the partial haplotypes by one bit, we extend HMM by one state, as shown in Fig. 2. We then compute the transition $P_{transition}$ and emission $P_{emission}$ probabilities for the new state.

For emission probabilities, every SNP in the reference haplotype can emit an SNP in the target haplotype. If two SNPs are the same, we consider that there is no mutation thus the emission probability is 1. If two SNPs are different, we suspect that there is a mutation and the emission probability is the mutation rate, which again can be obtained from IMPUTE website. Since we consider a pair of haplotypes, the emission probability will be the product of the emission probability from each haplotype.

Given $P_{transition}$, $P_{emission}$ and $R(i,r_1,r_2)$, we have the following recursion for MIR at position i: $MIR(i, r_1, r_2, j_1, j_2) = \text{argmax}\ b_1, b_2, y_1, y_2(P_{transition} \times P_{emission} \times MIR(i-1, (b_1, r_1[0, k-2]), (b_2, r_2[0, k-2]), y_1, y_2) \times R(i, r_1, r_2))$ for $b_1\in\{0,1\}, b_2\in\{0,1\}, 1\le y_1, y_2\le m, 1\le j_1, j_2\le m$ which allows us to compute $MIR_{\max(i,r_1,r_2)}$ for each i for every r_1, r_2.

2.1. Hap-seqX

As we can see from the above recursion, a total of $O(m^2\times 4^k)$ records need to be saved into memory at every position, which becomes prohibitively large even for a short region of haplotypes. In order to alleviate the memory burden, Hap-seq writes intermediate results on hard disk and read them back when necessary, namely when the back-tracking is conducted.

In order to address the memory issue, we need to reduce the number of records to be saved in memory. Our intuition is that as the error rate is generally very low, there are many obviously "bad" pairs of suffixes which conflict with the reads they covered too much. Therefore we can filter out these pairs first by a pre-processing step. In order to evaluate if a pair of suffixes r_1, r_2 is good or bad, we compute the posterior probability of them given the reads as $P(i, r_1, r_2|R)$:

$$P(i, r_1, r_2|R) = e^E \times (1-e)^{K-E} \quad (1)$$

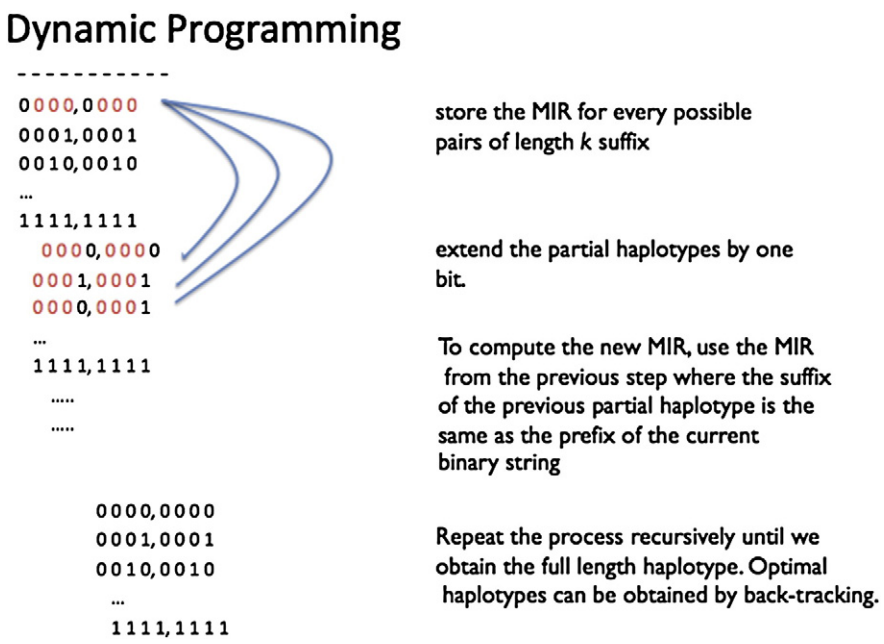

Fig. 1. Illustration of the dynamic programming process.

where e is the error rate, K is the total number of bits of the reads covered by the length k window starting from position i and E is the minimum number of errors by assigning every read to either r_1 or r_2 depending on which assignment leads to fewer errors.

Then we can sort all the 4^k pairs according to their posterior probabilities and select the top $t\%$ pairs, where t is a predefined threshold. Thus the space complexity is reduced to $O(m^2 \times 4^k \times t\%)$.

The above strategy still has two deficiencies:

- As a typical setting for the parameters is $m = 100$, $k = 3$, a reduction on 4^k would not improve the space complexity significantly.
- As 4^k is small, t needs to be big in order to guarantee a good performance. In our experiments, we observed that $t\%$ needs to be 50%–80% and thus the improvement is quite subtle.

In order to further improve the space complexity, we extend the above strategy and use the posterior probability of the (pair of length-k suffixes, pair of reference haplotypes) at each position to determine which ones should be recorded into memory. Therefore, we are trying to reduce the space complexity for the combination of the reference sequences and the suffixes instead of just for the suffixes, where we can make much more significant reduction of the space complexity.

For a pair of suffixes r_1, r_2 and a pair of reference haplotypes h_{j_1}, h_{j_2}, we need to compute their posterior probability $P(i, r_1, r_2, j_1, j_2|R, H)$ at position i. Without losing generality, assuming we assign r_1 to h_{j_1} and r_2 to h_{j_2}, is computed as

$$P(i, r_1, r_2, j_1, j_2|R, H) = P(i, r_1, r_2|R) \times P(i, r_1, j_1|H) \times P(i, r_2, j_2|H)$$

where $P(i, r_1, r_2|R)$ is the posterior probability of the pair of suffixes r_1, r_2 given the set of reads, which can be computed by Eq. (1), $P(i, r_1, j_1|H)$ is the posterior probability of the imputation for the suffix r_1 on the reference haplotype h_{j_1} given the set of reference sequences.

The computation of $P(i, r_1, j_1|H)$ involves two operations: transition and emission. Assuming the transition is from reference haplotype h_{y_1} to h_{j_1}, the transition probability $t_{(i+k-2,y_1),(i+k-1,j_1)}$ is the standard transition probability whose value depends on whether y_1 is identical to j_1. The transition probabilities when they are identical and when they are not can be both obtained from IMPUTE website and the probabilities are consistent for all i's.

The emission probability $\mu_{r_1,(i+k-1,j_1)}$ depends on the mutation rate, which can be computed as the following:

$$\mu_{r_1,(i+k-1,j_1)} = \begin{cases} 1-\mu & h_j[i+k-1] = r_1[k-1] \\ \mu & \text{otherwise} \end{cases}$$

where $h_j[i+k-1]$ is the $i+k-1$th symbol in h_j, $r_1[k-1]$ is the $k-1$th symbol in r_1. Therefore, if $h_j[i+k-1] = r_1[k-1]$, there is no mutation. Otherwise there is mutation with probability μ.

$t_{(i+k-2,y_2),(i+k-1,j_2)}$ and $\mu_{r_2,(i+k-1,j_2)}$ can be computed similarly. Thus we obtain the following formula for the posterior probability $P(i, r_1, r_2, j_1, j_2|R, H)$:

$$P(i, r_1, r_2, j_1, j_2|R, H) = \text{argmax}_{y_1,y_2}(\mu_{r_1,(i+k-1,j_1)} \times \mu_{r_2,(i+k-1,j_2)} \times t_{(i+k-2,y_1),(i+k-1,j_1)} \times t_{(i+k-2,y_2),(i+k-1,j_2)} \times R'(i, r_1, r_2))$$
$$\text{for } 1 \le y_1, y_2 \le m, 1 \le j_1, j_2 \le m$$

Therefore, at each position i, we can compute the posterior probability $P(i, r_1, r_2, j_1, j_2|R, H)$ and we can rank (r_1, r_2, j_1, j_2) by their posterior probability. Thus we can sort all these records at each position and save only the top $t\%$ records. Although optimal solution is not guaranteed any more, in our experiments, saving 1% of the records

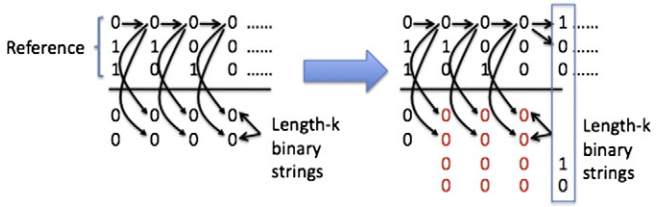

Fig. 2. Illustration of the extension of HMM at the same time as the extension of the partial haplotypes.

gives very good results. And an optimal solution for 50% of the cases is achieved.

As the number of records is $O(m^2 \times 4^k)$ at each position and we need to sort these records, the time complexity is $O(m^2 \times 4^k \times \log(m^2 \times 4^k)) = O(m^2 \times 4^k \times (2\log(m) + 2k\log(2))$. Given $m = 100$, $k = 3$, the time complexity increased by a factor of 6. However, the space complexity is reduced by a factor of 100 when $t\% = 1\%$ such that the intermediate results can now be saved completely in memory. In reality, although the time complexity increased, as we avoid the heavy IO to write the intermediate results on hard disk and read them in, the running time of Hap-seqX is indeed faster than that of Hap-seq. Therefore, Hap-seqX is a more practical algorithm.

3. Experimental results

We perform simulation experiments to compare the performance of our Hap-seqX algorithm with the standard HMM and the algorithm Hap-seq. The implementations of the methods are as follows. The standard HMM is our own implementation of the IMPUTE v1.0 model which uses the pre-defined genetic map information for the transition probability and the emission probability. The genetic map data is downloaded from the IMPUTE website. We split the reads into SNP-wise information similar to MACH (Li et al., 2010), such that IMPUTE can accept the sequence data. We apply the same genetic map data to Hap-seq and Hap-seqX. As both algorithms handle reads containing SNPs of length at most k, where k needs to be small, we set k as 3. He et al., 2012 showed that for very low coverage more than 99% of the reads contain no more than 3 SNPs. Reads containing more than 3 SNPs are split into chunks containing at most 3 SNPs, where each chunk is considered as an independent read.

We use 60 parental individuals of CEU population of HapMap Phase II data downloaded from the HapMap website. We randomly choose one target individual, generate simulated sequence reads with an error rate of 1% and coverage as 1X. The reads are of size 1000 bp in each end. The gap size is assumed to follow a normal distribution of mean 1000 bp and the standard deviation of 100 bp. We also assume the reads are uniformly generated. Our goal is to phase the target individual using the 59 individuals as the reference data set and the set of sequence data.

Our experiments focused on the whole chromosome 22, which contains 35,412 SNPs. A typical strategy to speed up the process of the imputation algorithms is to split the chromosome into small chunks and then run the algorithms on the chunks in parallel. Haplotypes reconstructed from adjacent chunks are then concatenated. In order to make the concatenation seamless, adjacent chunks are usually overlap with a buffer region. When the buffer region is big enough, the haplotypes from adjacent chunks in the overlapped buffer region should be highly consistent with each other. When they are concatenated, the best option which is able to minimize the differences between the haplotypes from adjacent chunks in the buffer region is selected. In our experiments, we split the chromosome 22 into chunks containing 1000 SNPs each and we set the buffer region length to 200 SNPs. Therefore, the first chunk covers locus [0, 1000], the second chunk covers locus [800, 2000], the third chunk covers locus [1800, 3000] and so on.

As shown in Fig. 3, as Hap-seq saves all the combinations of suffixes and reference haplotypes at each position, its performance is more accurate. Hap-seqX is generally more accurate than IMPUTE for both $t = 0.001$ and $t = 0.01$. However, when t gets small, the chances that the performance of Hap-seqX is worse than IMPUTE increase. As t increases, the performance of Hap-seqX increases steadily as we save more records in memory. For $t = 0.01$, we observed that for more than 50% chunks, Hap-seqX achieved the optimal solution, namely the switch error rate of Hap-seqX is identical to that of Hap-seq on these chunks. And Hap-seqX is almost always more accurate than IMPUTE for all chunks when $t = 0.01$. The running time of Hap-seq is 9 h and the running time of Hap-seqX is 5 h on a cluster of 300 nodes. Hap-seq requires saving intermediate results on hard drive, which generates hundreds of GB, or even TB files. Hap-seqX, on the contrary, saves everything in memory. Therefore, Hap-seqX is a more practical algorithm than Hap-seq.

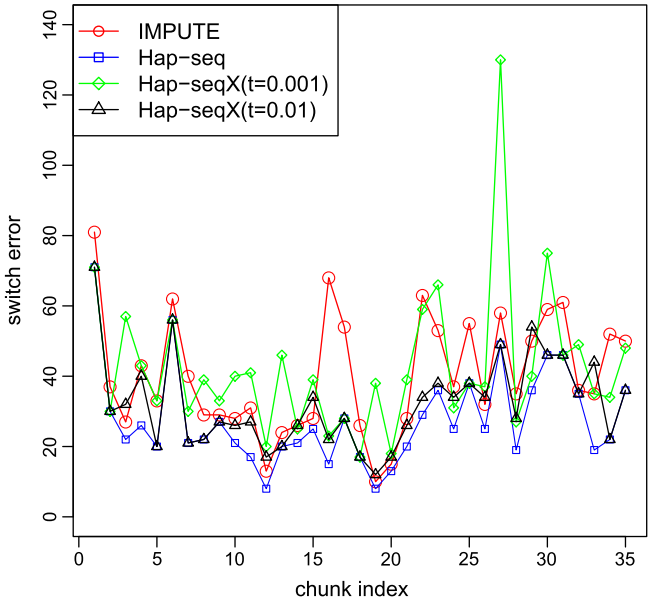

Fig. 3. The switch error rate when using IMPUTE, Hap-seq, Hap-seqX ($t = 0.001$), and Hap-seqX ($t = 0.01$) for each chunk of length 1200 SNPs for whole chromosome 22.

We also checked the performance of Hap-seqX using different amounts of memory. Hap-seqX finished in 7 h if we use 6G memory for each chunk and in 5 h if we use 10G memory for each chunk. This clearly indicates that the memory is the bottleneck of the algorithm. Indeed we suspect Hap-seqX should require much less memory. However, as our program is implemented in Java, and it is known that the automatic garbage collection of Java is not very reliable, we suspect that the memory usage should be improved a lot if we implement our algorithm in C, which will be our future work.

4. Discussion

In this paper, we presented Hap-seqX, a method for inferring haplotypes from sequence data utilizing information from both the reads and a reference dataset. Hap-seqX significantly improves current state of the art methods such as IMPUTE and unlike Hap-seq is practical to apply to whole genome datasets while obtaining very close to the optimal accuracy.

References

1000 Genomes Project, 2010. A deep catalog of human genetic variation. http://www.1000genomes.org/.
Bansal, V., Bafna, V., 2008. HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. Bioinformatics 24, i153.
Bansal, V., Halpern, A., Axelrod, N., Bafna, V., 2008. An MCMC algorithm for haplotype assembly from whole-genome sequence data. Genome Res. 18, 1336.
Beckmann, L., 2001. Haplotype sharing methods. Encyclopedia of Life Sciences (ELS). John Wiley & Sons, Ltd., Chichester.
Browning, S.R., Browning, B.L., 2010. High-resolution detection of identity by descent in unrelated individuals. Am. J. Hum. Genet. 86, 526–539.
Browning, B.L., Browning, S.R., 2011. A fast, powerful method for detecting identity by descent. Am. J. Hum. Genet. 88, 173–182.
Gusev, A., et al., 2009. Whole population, genome-wide mapping of hidden relatedness. Genome Res. 19, 318–326.
HapMap Project, 2009. http://hapmap.ncbi.nlm.nih.gov/.

He, D., Choi, A., Pipatsrisawat, K., Darwiche, A., Eskin, E., 2010. Optimal algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics 26, i183.

He, D., Han, B., Eskin, E., 2012. Hap-seq: an optimal algorithm for haplotype phasing with imputation using sequencing data. RECOMB, pp. 64–78.

Kang, H., Zaitlen, N., Eskin, E., 2010. Eminim: an adaptive and memory-efficient algorithm for genotype imputation. J. Comput. Biol. 17, 547–560.

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

Li, Y., Willer, C.J., Ding, J., Scheet, P., Abecasis, G.R., 2010. Mach: using sequence and genotype data to estimate haplotypes and unobserved genotypes. Genet. Epidemiol. 34, 816–834.

Marchini, J., Howie, B., Myers, S., McVean, G., Donnelly, P., 2007a. A new multipoint method for genome-wide association studies by imputation of genotypes. Nat. Genet. 39, 906–913.

Marchini, J., Howie, B., Myers, S., McVean, G., Donnelly, P., 2007b. A new multipoint method for genome-wide association studies by imputation of genotypes. Nat. Genet. 39, 906–913.