HapTree: A Novel Bayesian Framework for Single Individual Polyplotyping Using NGS Data

Emily Berger[1,2,3], Deniz Yorukoglu[2]*, Jian Peng[1,2], Bonnie Berger[1,2]*

1 Department of Mathematics, MIT, Cambridge, Massachusetts, United States of America, 2 Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts, United States of America, 3 Department of Mathematics, UC Berkeley, Berkeley, California, United States of America

Abstract

As the more recent next-generation sequencing (NGS) technologies provide longer read sequences, the use of sequencing datasets for complete haplotype phasing is fast becoming a reality, allowing haplotype reconstruction of a single sequenced genome. Nearly all previous haplotype reconstruction studies have focused on diploid genomes and are rarely scalable to genomes with higher ploidy. Yet computational investigations into polyploid genomes carry great importance, impacting plant, yeast and fish genomics, as well as the studies of the evolution of modern-day eukaryotes and (epi)genetic interactions between copies of genes. In this paper, we describe a novel maximum-likelihood estimation framework, HapTree, for polyploid haplotype assembly of an individual genome using NGS read datasets. We evaluate the performance of HapTree on simulated polyploid sequencing read data modeled after Illumina sequencing technologies. For triploid and higher ploidy genomes, we demonstrate that HapTree substantially improves haplotype assembly accuracy and efficiency over the state-of-the-art; moreover, HapTree is the first scalable polyplotyping method for higher ploidy. As a proof of concept, we also test our method on real sequencing data from NA12878 (1000 Genomes Project) and evaluate the quality of assembled haplotypes with respect to trio-based diplotype annotation as the ground truth. The results indicate that HapTree significantly improves the switch accuracy within phased haplotype blocks as compared to existing haplotype assembly methods, while producing comparable minimum error correction (MEC) values. A summary of this paper appears in the proceedings of the RECOMB 2014 conference, April 2–5.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: denizy@mit.edu (DY); bab@csail.mit.edu (BB)

This Methods article is associated with RECOMB 2014.

Introduction

While human and other eukaryotic genomes typically contain two copies of every chromosome, plants, yeast and fish such as salmon can have strictly more than two copies of each chromosome. By running standard genotype calling tools, it is possible to accurately identify the number of "wild type" and "mutant" alleles (A, C, G, or T) for each single-nucleotide polymorphism (SNP) site. However, in the case of two heterozygous SNP sites, genotype calling tools cannot determine whether "mutant" alleles from different SNP loci are on the same or different chromosomes (i.e. compound heterozygote). While the former would be healthy, in many cases the latter can cause loss of function; it is therefore necessary to identify the phase (*phasing*) —the copies of a chromosome on which the mutant alleles occur—in addition to the genotype (Figure 1). This necessitates efficient algorithms to obtain accurate and comprehensive phase information directly from the next-generation-sequencing read data in higher ploidy species.

Various sources of information can be utilized for the computational identification of an individual's diplotype/polyplotype: pedigree (e.g. trio-based phasing) [1–3], population

structure of variants (e.g. phasing by linkage disequilibrium) [3–6] and more recently by identity-by-descent in unrelated individuals [7,8], as well as sequencing read datasets [9–13]. Among these approaches, methods for sequence-based haplotype phasing are the only viable approach for haplotype phasing on a single individual member of a species (assuming homologous chromosomes are sequenced together), as other approaches either require family members or a population. For an individual diploid genome, the problem of reconstructing the diplotype using sequence information, the diploid phasing problem, is equivalent to the identification of the sequence of alleles on either parental haplotype. If this sequence is correctly inferred, then the other haplotype will automatically carry the corresponding opposite alleles (reference or alternative). Solving an error-free version of the diploid haplotype reconstruction problem is straightforward: the haplotype of each connected (by reads) component of heterozygous SNPs can be obtained by propagating allele information within reads. In reality, however, sequencing errors as well as false read mappings cause conflicts within sequence information, requiring a mathematical formulation of the haplotype reconstruction problem. Among various formulations suggested for this problem, the most commonly used is an NP-hard minimum error correction (MEC) definition [14,15], which aims to identify the smallest set of nucleotide changes required within

Author Summary

While human and other eukaryotic genomes typically contain two copies of every chromosome, plants, yeast and fish such as salmon can have strictly more than two copies of each chromosome. By running standard genotype calling tools, it is possible to accurately identify the number of "wild type" and "mutant" alleles (A, C, G, or T) for each single-nucleotide polymorphism (SNP) site. However, in the case of two heterozygous SNP sites, genotype calling tools cannot determine whether "mutant" alleles from different SNP loci are on the same or different chromosomes. While the former would be healthy, in many cases the latter can cause loss of function; it is therefore necessary to identify the phase—the copies of a chromosome on which the mutant alleles occur—in addition to the genotype. This necessitates efficient algorithms to obtain accurate and comprehensive phase information directly from the next-generation-sequencing read data in higher ploidy species. We introduce an efficient statistical method for this task and show that our method significantly outperforms previous ones, in both accuracy and speed, for phasing triploid and higher ploidy genomes. Our method performs well on human diploid genomes as well, as demonstrated by our improved phasing of the well known NA12878 (1000 Genomes Project).

mapped fragments that would allow a conflict-free separation of reads into two separate homologous chromosomes (or a bipartite separation of the fragment conflict graph). Some of the solutions proposed for this problem include: HapCUT[9], an algorithm for optimizing MEC score based on computing max-cuts of the fragment graph; Fast Hare [16], a heuristic that clusters reads into two sets in a greedy fashion, and HapCompass [10], a spanning tree based approach for minimizing fragment conflicts.

Unlike diploid genomes, computational identification of common chromosomal variants in polyploid genomes using sequencing data has received little attention, except in the pioneering work of Aguiar & Istrail [8]. Polyploidy studies are of importance as they allow a comprehensive investigation of variants within plant, fish, and yeast genomes and help understand mechanisms of eukaryotic evolution. However, haplotype reconstruction in polyploid genomes is fundamentally more complex, even in the error-free version of the problem (without sequencing errors or false read

mappings). Due to the newness of the NGS-based biological research in polyploid genomes, the mathematical foundations of the polyploid phasing problem have not yet been established. The solution proposed by Aguiar & Istrail for single individual polyplotyping problem is based on phasing all possible SNP loci pairs independently while further consolidating this information in a separate stage in order to infer a set of haplotypes.

Diploid phasing methods focus on a given list of heterozygous variants that are guaranteed to contain a single reference allele, as well as an alternative allele (assuming all heterozygous loci are bi-allelic). In contrast, in the polyploid phasing problem, there is no such guarantee of a single type of heterozygous SNP. Each heterozygous locus for a k-ploid chromosome can potentially contain from 1 up to $k-1$ alternative alleles within the heterozygous loci, significantly increasing the complexity of the phasing problem in comparison to the diploid case. Furthermore, in a diploid phasing setting, there are always two possible options for phasing a pair of SNP loci, regardless of what other SNPs they are phased with. These two options can be thought as parallel (alternative allele pairs and reference allele pairs are matched within themselves) or switched (each alternative allele is matched with the other reference allele). These two options are no longer relevant when the genome contains more than two copies of each chromosome, due to the fact that there are up to $k!$ options when merging a phased haplotype block with another.

In this paper, we introduce a maximum-likelihood formulation of the polyploid full haplotype reconstruction problem and present a haplotype assembly algorithm, HapTree, which concurrently performs SNP-pair phasing and full haplotype assembly based on a probabilistic framework. We observe that, on simulated polyploid data, HapTree substantially improves the phasing capabilities and performance of any existing program. Because real polyploid data is hard to come by, we also evaluate HapTree on real human diploid data and find that, when compared to the more accurate trio-based data as the ground truth [17], HapTree significantly reduces the number of switch errors, while remaining on par in terms of MEC score over existing single-individual haplotype assembly methods for diploid genomes. We also introduce a relative likelihood (RL) score definition for annotation-free evaluation of phasing quality for polyploid haplotype assembly as an alternative to MEC score. Using simulated polyploid sequencing datasets, we demonstrate that RL-score performs significantly better at capturing haplotype assembly quality than MEC-score as ploidy increases.

Figure 1. Loss of function in different polyplotypes of a sample pentaploid genome. As the loss of function is often determined by whether a healthy copy of a gene exists, knowing the genotype vector is sufficient if there is a single SNP site. In the case of two SNP sites however, the genotype vector cannot be used to unambiguously determine loss of function, and phasing is required.
doi:10.1371/journal.pcbi.1003502.g001

Method

Overview of HapTree

The HapTree pipeline is designed to perform phasing and full haplotype assembly of a single genome. The key component of HapTree is a relative likelihood function which measures the concordance between the aligned read data and a given haplotype phase under a probabilistic model that also accounts for possible sequencing errors. To identify a phasing solution of maximal likelihood, HapTree finds a collection of high-likelihood solutions for phases of the first m SNP loci and extends those to high likelihood phases of the first $m+1$ SNP loci, for each incremental m. In each step, HapTree maintains only the set of likely partial phases to be extended in next steps. Finally, a phase of maximal likelihood for all loci is obtained after the extension of the last SNP locus.

Broadly speaking, HapTree aims to discover the best, or *maximum likelihood*, haplotype based on the read data available. Theoretically, one could enumerate all possible haplotypes, compute the likelihood of each being the true haplotype (using formulas described below), and choose the most likely one; in most cases this approach is intractable as there are exponentially many possible haplotypes. HapTree therefore has a variety of ways of trimming down the solution set from all possible haplotypes to a much smaller set of more likely solutions, making the problem tractable. It does so by taking an inductive approach, generating a collection of *likely* phasing solutions for the first two SNPs in the genome, and then extending those to phasing solutions of the first three SNPs, and those to the first four SNPs, and so on. When extending any particular solution, HapTree chooses (based on computing likelihoods) how the alleles of the newly added SNP may be assigned to chromosomes; it includes only those assignments that are sufficiently likely. Additionally, if HapTree finds after extending all solutions to include the next SNP that there are too many *likely* solutions, it throws the worst (*least likely*) solutions away. Upon including all SNPs to be phased, HapTree randomly chooses a solution of maximum likelihood from amongst the solutions it has found.

Availability

An implementation of our method, HapTree, is available for download at: http://groups.csail.mit.edu/cb/haptree/

Definitions and Notation

We describe below the problem of sequence-based polyploid haplotype assembly and provide basic technical notation that will be useful for describing our method. We assume for now that each SNP locus to be phased is bi-allelic (i.e. contains only two possible alleles, one being the reference allele). We further assume that for each SNP locus s, the genotype of s is known and is defined to be the number of chromosomes carrying the alternative allele (denoted by $g(s)$). If k denotes the ploidy, $g(s)$ can range from 1 to $k-1$ for heterozygous loci s. At this point, we would like to note that these two assumptions are made for the sake of simplicity of method description and implementation, though the genotype information does tend to be available. After describing our method we also describe the changes needed to our original approach to accommodate multi-allelic and genotype-oblivious polyploid haplotype assembly. At this time our implementation accommodates the aforementioned simpler case of bi-allelic SNPs and known genotypes; it is simple to extend this implementation to the more general case, and we describe such an extension in **Discussion**.

We denote the sequence of observed nucleotides of a fragment simply as a "read" (independent from single/paired-end reads and sub-reads of a strobe read structure). The set of all reads is denoted as R. We define a read $r \in R$ as a vector with entries $r[i] \in \{0, 1, -\}$ where a 0 denotes the reference allele, a 1 the alternative allele, and a $-$ indicates one of two possibilities: First, that the read does not overlap with the corresponding SNP locus, or second, that neither the reference nor alternative allele is present and hence there must be a read error. A read $r \in R$ *contains* a SNP s if $r[s] \neq -$. A read can also be represented as a dictionary or mapping with keys the positions (from amongst the SNPs to be phased) of SNP loci it contains and values of either reference allele or alternative allele, represented by 0 and 1 respectively (e.g. $r = \{3:0, \ 4:1, \ 5:0, \ 8:1, \ 9:1\}$). As current sequencing technologies generate read data with a certain rate of sequencing errors, some of the positions within a read likely contain false nucleotide information. Among these erroneous bases, unless they are located at SNP loci and contain opposite allele information, we ignore them by representing them with $-$, and thus keep only confounding sequencing errors that can affect phased haplotype results. For each read r and for each SNP locus s, we assume an error rate of ε_r, s and a probability of opposite false allele information $r[s]$ is equal to $\varepsilon_r, s = \dfrac{\varepsilon_r, s}{1 - \frac{2}{3}\varepsilon_r, s}$. We modify this error rate by a factor of two-thirds because conditional on there being an error, we model the error as equally likely to be any of the three other alleles. Two of the three of these alleles are neither the reference nor the alternative allele and thus we know that an error has been made in this case. Therefore, two-thirds of the time the erroneous alleles produced are known as such and may be thrown out, leaving a true error only one-third of the time. We represent these error rates as matrices ε, ε. At this time our method assumes uniform error rates with respect to the SNP position; the error rate is supplied by the user and ought to depend on the read sequencing technologies used.

Upon the set of SNP loci S and read set R; we define a *Read Graph*, $G(S,R)$, such that there is a vertex for each SNP locus $s \in S$ and an edge between any two vertices s_1, s_2 if there is some read containing both s_1 and s_2; equivalently if $\exists r \in R, r[s_1] \neq - \wedge r[s_2] \neq -$. Without loss of generality, we assume that $G(S,R)$ is connected; otherwise each connected component can be processed independently.

Vector set. A *k-ploidy phase* of n SNPs with genotypes $\{g(s)\}$ is a tuple of k vectors (not necessarily distinct) $(v_1, ..., v_k) \subset \{0,1\}^n$ satisfying the genotype allele counts property, that is: $v_1[s] + v_2[s] + ... + v_k[s] = g(s)$ for all $s \in \{1, 2, ..., n\}$. We will refer to this collection as a *vector set* and we think of each vector as a row vector.

We can build a phase by selecting a permutation of the alleles present for each SNP locus s. Note that the number of distinct permutations, $C(s)$, is strictly dependent on the genotype of the SNP and in the diploid bi-allelic case is equivalent to selecting the chromosomes containing the alternative alleles, hence

$$C(s) = \binom{k}{g(s)} = \frac{k!}{g(s)!(k - g(s))!}.$$

For example, let $k = 4$, then $g(s) \in \{1, 2, 3\}$. We enumerate the possible permutations below and include an example tetraploid genome.

```
0 0 0 1    0 0 0 1 1 1    0 1 1 1
0 0 1 0    0 1 1 0 0 1    1 0 1 1
0 1 0 0    1 0 1 0 1 0    1 1 0 1
1 0 0 0    1 1 0 1 0 0    1 1 1 0
```

All possible permutations of tetraploid bi−allelic heterozygous SNPs

```
00011010
01110011
10111100
10000111
```

{(00011010),(01110011),(10111100),(10000111)}

A sample tetraploid genome and and its corresponding vector set.

The sample tetraploid genome featured above on the right has a genotype vector: [2,1,2,3,2,2,3,2]; recall this counts the number of alternative alleles present at each SNP site. For any SNP s, let P_s denote the set of distinct allele permutations at SNP locus s. Throughout we are indifferent to the order of each chromosome, with this in mind we can see that the total number of phases is bounded below by $\frac{1}{k!}\Pi_s \, C(s)$.

Likelihood of a phase. We formulate the haplotype reconstruction problem as identifying the most likely phase(s) given the read data R, all SNP loci S, as well as their genotypes, and sequencing error rates ε. We assume the sequencing errors are independent of each other, that is for all $r \in R$ and all $s \in r$, that $\{r[s]\}$ are independently correct with probabilities $(1-\varepsilon_r,s)$ and incorrect with probabilities ε_r,s. Let ε be a matrix containing all of these probabilities: $\{\varepsilon_r,s\}$. Given a vector set, V_{set}, corresponding to a phase, R, and ε; the likelihood of the phase is determined by:

$$P[V_{set}|R,\varepsilon] = \frac{P[R|V_{set},\varepsilon]P[V_{set}|\varepsilon]}{P[R|\varepsilon]}. \qquad (1)$$

As $P[R|\varepsilon]$ depends only on ε and the read set R, it is therefore the same across all vector sets. Hence, we define a *relative likelihood* measure (RL) as

$$RL[V_{set}|R,\varepsilon] = P[R|V_{set},\varepsilon]P[V_{set}|\varepsilon].$$

As for $P[V_{set}|\varepsilon]$, there are several ways this can be modeled depending on the situation. For polyploid simulated data, we can assume that $P[V_{set}|\varepsilon]$ is equal for almost all vector sets, excluding ones containing duplicate vectors. Let $M = \{m_1,m_2,...\}$ be the set of the multiplicities in V_{set}; for example, if $V_{set} = \{0001,0010,1100,0001,0010\}$ then $M = \{2,2,1\}$. The probabilities $P[V_{set}|\varepsilon]$ will differ multiplicatively by multinomial coefficients $\binom{k}{m_1,m_2,...} = \frac{k!}{m_1!m_2!...}$. Specifically:

$$P[V_{set}|\varepsilon] = \frac{\binom{k}{m_1,m_2,...}}{\Pi_s \, C(s)}.$$

For real diploid data, there will never be duplicate vectors. To model $P[V_{set}|\varepsilon]$, we might assume that since mutations tend to occur together, adjacent SNP sites are more likely to be phased in parallel (00) or (11) than switched (01) or (10). Let $V = (v,v')$ and let $P(V)$ denote the number of adjacent SNPs that are parallel in v and $S(V)$ the number of adjacent SNPs that are switched in v (we must only consider v as it determines v'). For example, if $V = ((00010111000),(11101000111))$, then $P(V) = 6$ and $S(V) = 4$. For some $p > .5$ (denoted as *parallel bias*) and $q = 1-p$, we model this vector set probability as

$$P[V_{set}|\varepsilon] = p^{P(V_{set})}q^{S(V_{set})}.$$

Finally, we consider $P[R|V_{set},\varepsilon]$. For a given $r \in R$ and $v \in V_{set}$, let $A(r,v),D(r,d)$ denote the positions of SNP loci where r and v agree and disagree respectively. For example, if $r = (-,-,1,0,1,-,-,1,0)$ and $v = (1,0,0,1,1,0,0,1,0)$, then $A(r,v) = (5,8,9)$ and $D(r,v) = (3,4)$ We may now compute the desired probability, that is:

$$P[R|V_{set},\varepsilon] = \prod_{r \in R} P[r|V_{set},\varepsilon]$$

$$P[r|V_{set},\varepsilon] = \frac{1}{k}\sum_{v \in V_{set}}\left(\prod_{s \in A(r,v)}(1-\varepsilon_r,s)\prod_{s \in D(r,v)}\varepsilon_r,s\right).$$

The goal of our haplotype reconstruction problem is to find the vector set(s) maximizing the product $P[R|V_{set},\varepsilon]P[V_{set}|\varepsilon]$, equivalently $RL[V_{set}|R,\varepsilon]$. However, as the number of possible phases is on the order k^n, checking all of these is intractable. Our solution is based on finding high likelihood phases for the first $m+1$ SNPs, conditioned on a collection of high likelihood phases for the first m SNPs.

Semi-reads and sub-reads. To properly describe our method we must first define the *semi-reads* of a SNP locus s and the *sub-reads* of a subset $S' \subset S$.

Semi-reads. To form the set of *semi-reads* of s, denoted $SR(s)$, include each read $r \in R$ that contains both s and some $s' < s$ (s' is upstream of s) and ignore all information from r on SNPs $s'' > s$ (s'' is downstream of s). Suppose the set of reads is:

{1:1, 2:1, 3:1, 4:1} {3:1, 4:1, 5:0, 6:0} {4:0, 5:1, 6:1} {4:0, 5:1, 6:1, 7:0} {5:0, 6:0, 7:1} {5:1, 6:1, 7:0}

The corresponding semi-reads for each SNP locus would be:

1 → None 2 → {1:1, 2:1} 3 → {1:1, 2:1, 3:1} 4 → {1:1, 2:1, 3:1, 4:1} {3:1, 4:1}

5 → {3:1, 4:1, 5:0} {4:0, 5:1} {4:0, 5:1}

6 → {3:1, 4:1, 5:0, 6:0} {4:0, 5:1, 6:1} {4:0, 5:1, 6:1} {5:0, 6:0} {5:1, 6:1}

7 → {4:0, 5:1, 6:1, 7:0} {5:0, 6:0, 7:1} {5:1, 6:1, 7:0}

Sub-reads. The *sub-reads* of $S' \subset S$, denoted $R(S')$, are obtained by, for each $r \in R$, removing all keys $s \in S \backslash S'$ to form r', and then adding r' to $R(S')$ if the length of r' is at least 2. Alternatively, $R(S')$ corresponds to the set of reads relevant to the problem of only phasing S'. Continuing with the example above, if

$S' = \{1,2,3,4,5\}$, then

$$R(S') = \{\{1 : 1, \quad 2 : 1, \quad 3 : 1, \quad 4 : 1\},$$
$$\{3 : 1, \quad 4 : 1, \quad 5 : 0\},$$
$$\{4 : 0, \quad 5 : 1\}, \quad \{4 : 0, \quad 5 : 1\}\}.$$

HapTree

Our main approach to solving the single individual polyploid haplotype assembly problem is by finding highly probable solutions on m SNPs and extending those to highly probable solutions on $m+1$ SNPs. Our algorithm has two fundamental parts: branching and pruning. For each connected component of the *ReadGraph*, $G(S,R)$, we inductively generate a collection of high likelihood phases on the first m SNPs. For each of these phases, we branch them to phases on $m+1$ SNPs by considering all possible orderings of alleles for position $m+1$ and including branches for those which occur with probability above a certain threshold. After doing so, we prune the tree of phases by removing all leaves that occur with probability sufficiently less than the most probable leaf. We discuss both parts in more detail below. We note that although a dynamic programming algorithm can be directly applied to infer the best solutions under HapTree's likelihood model, we instead developed HapTree, which is substantially faster than exact dynamic programming but with nearly identical empirical performance.

Extension. We first describe how to extend an existing a haplotype assembly H on $m \geq 0$ SNPs onto the $m+1^{th}$ SNP s. Recall the set of permutations of s is denoted P_s and one particular permutation as $o \in P_s$. An extension H' of H onto SNP locus s can be defined by appending some permutation $o \in P_s$ of alleles to H; $H' = H + o$. Note that it is possible for two distinct permutations to result in the same H': $H + o = H + o'$. In these cases we do not include duplicates, as they are equivalent. Observe that if H is empty, all allele permutations are the same as vector sets; we therefore include only one. For any H', we can compute the probability of it being the correct haplotype (for the first $m+1$ SNPs) conditioning on H being correct (for the first m SNPs), as well as the semi-read data $SR(s)$ and error rate ε. We express this below:

$$P[H' \mid H, SR(s), \varepsilon] = \frac{P[SR(s) \mid H', H, \varepsilon]P[H' \mid H, \varepsilon]}{P[SR(s) \mid H, \varepsilon]} \quad (2)$$

This computation is similar to those done above in equation (1). The EXTEND algorithm (Algorithm 1) is given below, which returns a list of all extensions H' of H that occur with probability above a certain threshold, ρ, given haplotype H.

Branching. Here we define *branching* a collection of haplotypes $\mathcal{H}$ with threshold ρ to SNP s: BRANCH($\mathcal{H},\rho,s$) (Algorithm 2). We assume all $H \in \mathcal{H}$ phase the first $m \geq 0$ SNPs and that SNP s is the $m+1^{th}$ SNP. The act of branching $\mathcal{H}$ returns $\mathcal{H}'$: a list of all extensions generated by EXTEND with threshold ρ for all H in $\mathcal{H}$. To initialize BRANCH we EXTEND the empty vector set to an arbitrary permutation of the alleles of the first SNP, as all permutations are equivalent as vector sets.

Pruning. For a collection of haplotypes $\mathcal{H}$ of SNPs $S' \subset S$, we can compute the relative likelihood of each haplotype conditioned on the sub-reads $R(S')$ and error rate ε; we write this as RL$[H \mid R(S'), \varepsilon]$. The same computation as performed in

Algorithm 1. EXTEND(H, ρ, s): Extending a haplotype H at SNP s to all H' that occur with probability $\geq \rho \in [0, 1)$.
Input: H, ρ, s
Output: E'
    $E = [\ ]$
    for $o \in P_s$ **do**
        $H' = H + o$
        if $H' \notin E$ **then**
            if P$[H' \mid H, SR(s), \varepsilon] > \rho$ **then**
                $E += H'$
    return E

Algorithm 2. BRANCH($\mathcal{H}$, ρ, s): Branching haplotypes $\mathcal{H}$ at SNP s with threshold $\rho \in [0, 1)$.
Input: $\mathcal{H}$, ρ, s
Output: $\mathcal{H}'$
    $\mathcal{H}' = [\ ]$
    for $H \in \mathcal{H}$ **do**
        $E = $ EXTEND(H, ρ, s)
        for $H' \in E$ **do**
            $\mathcal{H} += H'$
    return $\mathcal{H}$

equation 1 yields:

$$P[H \mid R(S'), \varepsilon] = \frac{P[R(S') \mid H, \varepsilon]P[H \mid \varepsilon]}{P[R(S') \mid \varepsilon]}.$$

Since $P[R(S')|\varepsilon]$ does not depend on H:

$$\text{RL}[H \mid R(S'), \varepsilon] = P[R(S')|H, \varepsilon]P[H|\varepsilon]. \quad (3)$$

The goal of PRUNE($\mathcal{H},\kappa,S'$) (Algorithm 3) is to return a subset $\mathcal{H}' \subset \mathcal{H}$ containing only sufficiently probable haplotypes. It does so by computing the relative likelihood of the most probable $H \in \mathcal{H}$, that is $\omega = \max_{H \in \mathcal{H}} \text{RL}[H|R(S'), \varepsilon]$, and adding $H \in \mathcal{H}$ to $\mathcal{H}'$ if RL$[H|R(S'), \varepsilon] \geq \kappa\omega$, where κ is between 0 and 1. We note that that one can compute $RL(H')$ from $RL(H)$ by only looking at the semi-reads $RS(s)$: we store the relative likelihood values for all $H \in \mathcal{H}$ and update them when branching to H'; PRUNE is therefore no more costly than BRANCH.

Main algorithm. Here we give a high-level description of our overall haplotype assembly method HapTree($R,\hat{\rho},\hat{\kappa},S$) (Algorithm 4) using the EXTEND, BRANCH, and PRUNE algorithms. We generate high likelihood phases for the first m SNPs, BRANCH those phases to include s (the $m+1^{th}$ SNP), then PRUNE the resulting phases, and repeat for $m = m+1$. We begin with an arbitrary permutation of the first SNP, since all orderings result in the same vector set. For the final step, we PRUNE with $\kappa = 1$, and therefore return only the maximally probable phases that we have found; if this set is of size greater than one, we choose a phasing from within it randomly. More generally, below we take $\hat{\rho}$ and $\hat{\kappa}$ to be vectors, as ρ and κ may depend on m, the size of $\mathcal{H}$ or other user-specified variables.

Algorithm 3. PRUNE($\mathcal{H}$, κ, S'): Pruning haplotypes $\mathcal{H}$ on S' with factor $\kappa \in [0,1]$.
  Input: $\mathcal{H}$, κ, S'
  Output: $\mathcal{H}'$
        $\mathcal{H}' = [\ ]$
        $\omega = \max\limits_{H \in \mathcal{H}} \mathrm{RL}[H | R(S'), \varepsilon]$
        for $H \in \mathcal{H}$ **do**
               if $\mathrm{RL}[H \mid R(S'), \varepsilon] \geq \kappa\omega$ **then**
                      $\mathcal{H}' += H$
        return $\mathcal{H}$

Algorithm 4. HapTree(R, $\hat{\rho}$, $\hat{\kappa}$, S): Assembling haplotype from reads R with parameters $\hat{\rho}$, $\hat{\kappa}$.
  Input: R, $\hat{\rho}$, $\hat{\kappa}$, S
  Output: $\mathcal{H}$
        $\mathcal{H} = [\ ]$
        $S' = \{\}$
        for $s \in [1, 2, \ldots, |S|]$ **do**
               $S' += s$
               $\mathcal{H} = \mathrm{BRANCH}(\mathcal{H}, \hat{\rho}(s), s)$
               $\mathcal{H} = \mathrm{PRUNE}(\mathcal{H}, \hat{\kappa}(s), S')$
        return $\mathcal{H}$

Results

Scoring and Evaluation

Determining the quality of a phasing solution depends on whether the true phase is known. When no such information is avaliable, the Minimum Error Correction (MEC) score [15] is a widely used scoring function to measure the quality of phasing solutions. The MEC score is defined as the minimum (amongst chromosomes) number of mismatches between a phase H and the read set R. A number of existing programs, including HapCut [9], find phasing solutions by optimizing the MEC score in diploid cases. For higher ploidy the MEC score can no longer be reliably used because unlike in the diploid case, the phase of any one chromosome does not determine the phases of the others. Moreover, the MEC score does not distinguish between two separate phases of a pair of SNP loci with different non-zero counts of $(0,0),(0,1),(1,0),(1,1)$ in their vector sets. Finally, unlike in the diploid case, a phase of a pair of SNP loci containing a set of parallel alleles does not prevent it from containing a set of switched alleles as well. To demonstrate these issues, consider two possible vector sets corresponding to phases of a pair of triploid SNPs both with genotype 2: $a : ((0,0),(0,0),(1,1))$ and $b : ((0,0),(0,1),(1,0))\}$. If the read data is $((0,0),(0,0),(0,0))$, it is clear from a probabilistic standpoint that phase a is a better fit, but both a and b have equal MEC scores. This effect is exaggerated as k increases.

When a true phase is available, there are a variety ways to evaluate how accurate any predicted phase is. A widely used measure in diploid phasing is switch error, which is calculated as the number of positions where the two chromosomes of a proposed phase must be switched in order to agree with the true phase. For polyploid phasing, we generalize switch error to *vector error*. In higher ploidy cases, at any SNP locus, it is possible for no chromosomes in a proposed phase to require a switch or anywhere from 2 to k chromosomes to require switches, in order for a proposed phase to agree with the true phase. We do not wish to penalize a solution where only two vectors must be switched at a given position with the same penalty to be used for a solution in which all vectors must be switched. The *vector error* of a proposed phase (with respect to the true phase) is defined by the minimum number of segments on all chromosomes for which a switch must occur; for the diploid case this score is exactly twice the switch

error. One may also think of the vector error as the minimum number of segments a proposed phase and the true phase have in common, less the ploidy. Even for triploid genomes, the vector error is more discriminative than switch error. Consider the following example in Figure 2:

In Figure 2 phase (i) is a more accurate phase than (ii), and phase (ii) more accurate than phase (iii). The segments are broken up by row and color: phase (i) having five segments, phase (ii) having six, and phase (iii) having seven. Note that there may be several ways to break a vector set into a minimal number of segments; phase (ii) is such an example. Finally, we remark that vector error can be computed in time $O(kn^2)$, where k is the ploidy and n the block size.

Results for Simulated Polyploid Data

Relative Likelihood (RL) objective function vs. MEC score for polyploid genomes. We assessed the effectiveness of our RL score by comparison to MEC score on simulated data. To do so, we simulated reads with error rate 0.02 from a pair of phased k-ploid SNP loci for different coverages ($5\times$, $10\times$, $20\times$, $100\times$) and for $k \in \{2,\ldots,10\}$. All possible phases were exhaustively enumerated, and phases of the maximal relative likelihood (RL) and phases of the minimal MEC score chosen. We computed the proportion of perfectly phased SNP pairs in both cases (perfect solution rate). Even with two SNP loci, RL significantly outperforms MEC for all $k \geq 3$ (Figure 3A). It is also worth noting that MEC (in comparison to RL) deteriorates more seriously in accuracy as ploidy (k) increases (Figure 3A). In addition, we also compared the vector error rate in both cases; for a pair of SNPs, this rate is the number of vectors from the proposed solution that cannot be matched with vectors from the true solution (Figure 3B).

The results demonstrate that the higher the ploidy, the better the relative likelihood (RL) score performs in comparison to MEC score for phasing a pair of SNPs (Figure 3). In fact, in simulations where $k \geq 8$, RL with $5\times$ the coverage already outperforms MEC with $100\times$ coverage. For the same coverage, RL always outperforms MEC for $k \geq 3$, and they are equivalent in the diploid case ($k = 2$).

Comparisons of HapTree and HapCompass. To evaluate the phasing capabilities of HapTree, we compare it with

```
1 1 1 0 0 0 1        1 1 1 0 0 0 1        1 1 1 1 1 0 1        1 1 1 0 0 0 0
1 0 1 0 0 1 0        1 0 1 1 1 0 1        1 0 1 0 0 0 1        1 0 1 1 1 0 1
0 0 0 1 1 0 1        0 0 0 0 0 1 0        0 0 0 0 0 1 0        0 0 0 0 0 1 1
   True Phase        (i): 2 Vector Errors   (ii): 3 Vector Errors   (iii): 4 Vector Errors
```

Figure 2. Examples of *Vector Error* in a sample tetraploid genome; the true phase is on the left and examples with two, three, and four vector errors are on the right.
doi:10.1371/journal.pcbi.1003502.g002
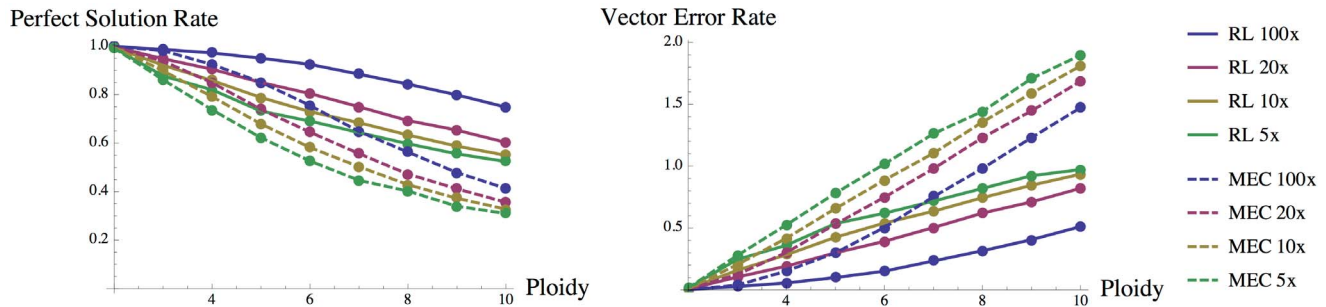
Figure 3. Proportion of perfectly phased SNP pairs and vector error rate for RL (solid line) and MEC (dashed line) optimization in 10000 trials over 5×, 10×, 20× and 100× coverage.
doi:10.1371/journal.pcbi.1003502.g003

HapCompass [8] (latest version available at: www.brown.edu/Research/Istrail_Lab/hapcompass.php), to our knowledge the only other existing program that directly addresses polyploid haplotype assembly, over multiple depth coverage values and component sizes for triploid and tetraploid simulated genomes. We simulated triploid and tetraploid genomes with different block lengths (10, 20 or 40 SNP loci), different coverages (5×, 10×, 20× and 40×), SNP positions, and SNP densities. Throughout the simulations for both the triploid and tetraploid cases, our EXTEND module is run with threshold $\rho = .01$ and PRUNE primarily with threshold $\kappa = .001$. When the current number of haplotype options generated is above 1000, we prune more aggressively with $\kappa = .01$ and when above 5000, with $\kappa = .05$. These parameters are chosen to ensure the efficiency of HapTree by only keep a tractable collection of promising solutions in each step. We also simulate a read set with uniform error rate and size dependent on coverage.

For the triploid case, we observed that HapTree finds a perfect solution at a rate independent of the number of SNPs used in the simulation; in contrast, HapCompass declines in performance the larger the block size (Figure 4). While both HapTree and HapCompass improve steadily the higher the coverage, in every case HapTree significantly outperforms HapCompass; the least significant improvement of 63% occurs in the case of 10 SNP loci and 10× coverage, whereas the most significant improvement occurs in the case of 40 SNP loci and 40× coverage. For both vector error rate and likelihood of perfect solution, we find that HapTree substantially outperforms HapCompass.

For tetraploid simulations, HapTree significantly outperforms HapCompass with block length of 10 SNP loci (Figure 5). For larger block lengths HapCompass arrives at the perfect solution at a rate of less than 1%; HapTree however does so at a rate between 40% and 70% depending on block size and coverage at least 20×.

We varied the allele error rates (.001,.02,.05, and .1) and observed decreases in accuracy that vary approximately linearly with the (uniform) allele error rates (Figure 6). The allele error rate is the likelihood of the sequencing technology to report the incorrect allele for a given position in one read. We ran 10000 trials for simulated triploid genomes of block size 10, with coverages 10×, 20×, and 40×.

For the simulations above in Figure 6, we modeled our read data on Illumina sequencing technologies; for more details, please see section **Simulated polyploid data generation** below. We also ran simulations on longer read data, modeled after 454 sequencing technologies and found almost identical results.

The primary reasons for HapTree's superior performance are, first, that HapTree's relative likelihood is more effective than HapCompass's MEC score (see **Relative likelihood vs MEC**); and second, that HapTree's inference algorithm is more accurate than the approximation algorithm used by HapCompass.

Discussion

Run-Time Evaluation

Not only does HapTree outperform HapCompass on phasing quality, it is also significantly faster, especially for longer block length. The median runtimes for block length 10 and 10× coverage were (0.00702,0.633) seconds for HapTree and HapCompass, respectively; for block length of 40 and 40× coverage, they were (0.0279,13.099) seconds, respectively.

Results on Real Diploid Data

As seen in the results of Geraci et al. [18], there is no perfect solution for diploid phasing. HapCUT is one of the methods reported that consistently performs best or close-to-best for a variety of experiments. For a proof of concept of how HapTree would perform on real data, we ran HapTree and HapCUT using
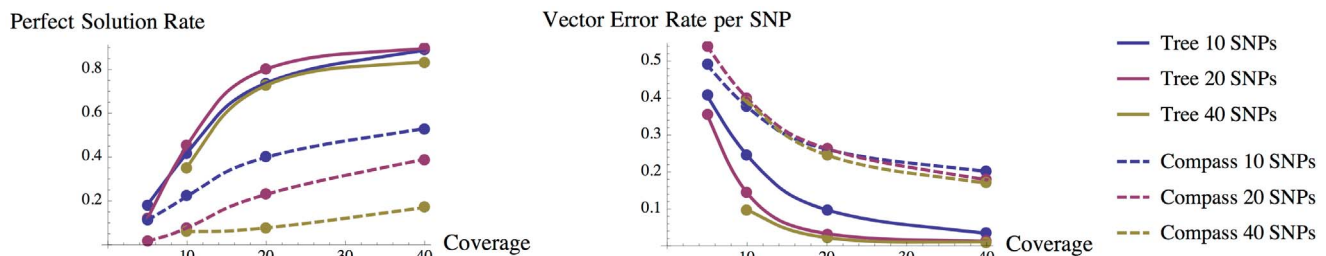
Figure 4. HapTree (solid lines) and HapCompass (dashed lines) on simulated triploid genomes: Likelihood of Perfect Solution and Vector Error Rates, 1000 Trials, Block lengths: 10, 20, and 40.
doi:10.1371/journal.pcbi.1003502.g004
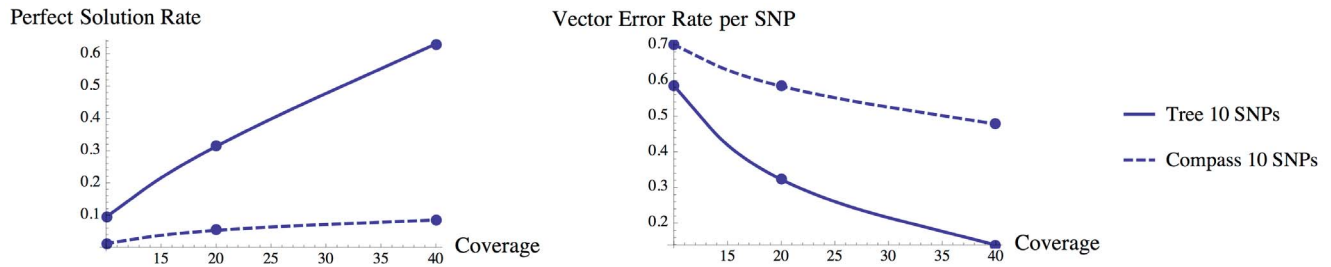
Figure 5. HapTree (solid line) and HapCompass (dashed line) on simulated tetraploid genomes: Likelihood of Perfect Solution and Vector Error Rates, 1000 Trials, Block length: 10.
doi:10.1371/journal.pcbi.1003502.g005

454 and Illumina sequencing data of the well-studied NA12878 genome (1000 Genomes Project Phase 1) [17], and compared MEC scores as well as switch errors to a trio phasing annotation accepted as ground truth; we present these results in Table 1. The trio phasing annotation represents a high quality diplotype of NA12878 for all SNP sites where either parent (NA12891 or NA12892) is homozygous [17]. Note that we computed the number of switch errors within connected SNP components only, against SNPs whose phase has been determined by the trio-based phasing; we then sum over components. In this case, HapTree was run with a uniform error rate of .02, an EXTEND threshold .001, and primarily with a PRUNE threshold of .001. We begin to prune more aggressively when we have at least 100,500, or 1000 possible haplotypes with thresholds of (.01,05,.1) respectively. For the vector set prior, from examining the read data, we ran HapTree with parallel bias $p = .8$.

We found that HapTree and HapCUT perform almost identically in MEC scores, with HapCUT having marginally smaller scores for both 454 and Illumina data sets. It is worth noting that HapCUT optimizes MEC score, and MEC score measures only the consistency between a phasing solution and read data, not with the true phase.

Notably, when comparing to the ground-truth phase as determined by trio-based phasing, we found HapTree significantly outperforms HapCut in terms of switch error rate for the phasing experiments on the NA12878 genome for 454 and Illumina datasets. Although our method is not primarily designed for phasing diploid genomes, it is still able to achieve better phasing results, when compared to the state-of-the-art diploid method. Again, the results on real-world read datasets showed the superiority of our likelihood function over MEC score for NGS-based phasing.

Simulated Polyploid Data Generation

Reads. To generate a paired-end read, we uniformly choose a starting point on the genome (we make sure the genome starts sufficiently before the first SNP and ends at the last). We fix the read-end length (read_len) to be 150. The fragment length (frag_len) is normally distributed with a mean of 550 and standard deviation of 30, but with min and max lengths of 500 and 600 respectively. The insert length (insert_len) is determined by the fragment length and read-end length, that is, insert_len = frag_len - 2read_len. Once we know the start and fragment length, we must choose from which chromosome to read; we do so uniformly from the k chromosomes. Finally, we add uniform error to the read; we choose a rate of .02, based on the reported error rate of Illumina sequencing technologies. For every SNP that the read covers, independently with probability ε we flip the allele to any other allele; two-thirds of the time when we have this error, we can see that the allele present is neither the reference nor the alternative, and therefore we delete it. Hence, conditional on seeing a SNP in a read, it is incorrect with probability $\varepsilon = \frac{\varepsilon}{1 - \frac{2}{3}\varepsilon}$ and correct with probability $1 - \varepsilon$.

Genomes. To simulate a genome, we fix a ploidy (k) and the number of SNPs (n). We determine the positions for the SNPs by randomly generating the distance between each pair of adjacent SNPs. We do so using a geometric random variable with parameter p (SNP density); this choice is equivalent to assuming that any position is a SNP independently with probability p. For phasing purposes, once one has generated the reads, the exact genomic positions are no longer relevant; they were only needed to simulate more accurate read data. We therefore refer to SNPs by their position amongst the SNPs, not their position in the genome. For each SNP, we randomly generate its haplotype, assuming for each chromosome, that the alternative and reference alleles are
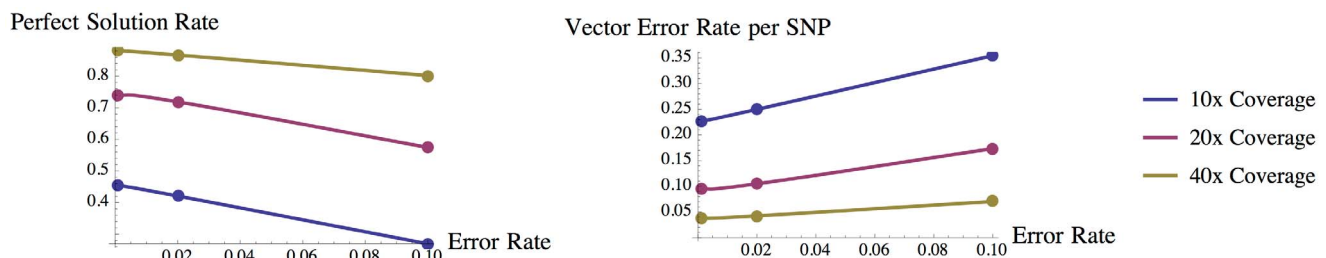
Figure 6. HapTree performance over varied error rates (.001, .02, .05, .1) and coverages (10×, 20×, 40×) on simulated triploid genomes: Likelihood of Perfect Solution and Vector Error Rates, 10000 Trials, Block length: 10.
doi:10.1371/journal.pcbi.1003502.g006

Table 1. Results of switch error (switch) and MEC score for HapTree and HapCUT of whole-genome phasing using 454 and Illumina data.

Results	454		Illumina	
Method	MEC	Switch	MEC	Switch
HapTree	32818	**2978**	20339	**1888**
HapCUT	32781	3192	20290	1933

doi:10.1371/journal.pcbi.1003502.t001

equally likely; if we generate a homozygous SNP, we try again. This procedure results in the likelihood of genotype $g(s) \in \{1, ..., k-1\}$ equal to $\binom{k}{g(s)}/(2^k - 2)$, and all orderings $o \in P_s$ being equally likely. For the simulations discussed we use this model. Note, however, that HapTree is not dependent on this model. When running HapTree on real data, different assumptions ought to be made regarding the distributions of vector sets.

Coverage. For any genome, to generate a read set with Cx coverage we need each base pair to be on average covered by C reads. To determine the number of reads to generate, we must know the length of the genome and the read length (read_len). The expected length of the genome is $\frac{n}{p}$ for SNP density p, and the read_len is 150 for each end (of which there are two); therefore we simulate $\frac{Cn}{300p}$ reads for Cx coverage. Note that many of these reads will see only zero or one SNP(s), thus for Cx coverage the number of useful reads for any SNP will be less than C.

Discussion

We have presented a scalable algorithm, HapTree, for polyplotyping using NGS sequencing data and a new metric for measuring accuracy in this context. We have described an efficient algorithm to identify phases that maximize our RL metric, a relative likelihood function which measures the quality of a given phase according to the read dataset. We have demonstrated the advantages of such a likelihood formulation over the existing MEC score in phasing both polyploid and diploid genomes. HapTree not only substantially improves the efficiency and phasing accuracy of the state-of-the-art in polyploid phasing, but also produces more accurate phased haplotype blocks for diploid genomes, as compared to HapCUT, which is designed for diploid phasing by MEC score optimization. Our results indicate that HapTree can be used in phasing individual triploid and tetraploid genomes, as well as improving phasing of real diploid genomes. HapTree also easily scales to genomes of higher ploidy.

Our algorithm can be easily extended to phase data with multi-allelic SNPs and with unknown genotype information as well. With unknown genotype information and multi-allelic SNPs, instead of $\binom{k}{g(s)}$ allele permutations, there are 4^k possibilities, since all 4 alleles (A,C,G,T) are possible for all k chromosomes. For bi-allelic SNPs with unknown genotypes, there are 2^k, as all possible reference and alliterative allele permutations are allowable. Finally, when the genotype is known but a SNP is multi-allelic, we may use multinomial coefficients to compute the number of allele permutations allowable: $\binom{k}{\#A, \#C, \#G, \#T} = \frac{k!}{\#A! \#C! \#G! \#T!}$, where $\#X$ denotes the number of alleles X according to the genotype, where $X \in \{A, C, G, T\}$. The only change to HapTree in these cases is that at each EXTEND step, we allow all allele permutation possibilities as dictated by whatever genotypic is available: we compute the probabilities for all 4^k, 2^k, or $\binom{k}{\#A, \#C, \#G, \#T}$ possibilities (depending on the situation) as opposed to $\binom{k}{g(s)}$ and EXTEND accordingly. Moreover, the type of information available does not need to be the same for all SNPs, since it only determines which allele permutations we introduce at the EXTEND step.

A future application of HapTree is genotype imputation, which can predict missing genotype from phasing results. As polyploid sequencing data becomes available, HapTree will be useful for the investigation of the role of heterozygosity in plant, fish, and other species. Moreover, accurate individual phases of diploid haplotypes can be assembled without the use of pedigree or population information.

A summary of this paper appears in the proceedings of the RECOMB 2014 conference, April 2–5 [19].

Author Contributions

Conceived and designed the experiments: EB DY BB. Performed the experiments: EB. Analyzed the data: EB DY JP BB. Wrote the paper: EB DY JP BB. Designed and guided the study: DY BB. Designed the method: EB.

References

1. Williams A, Housman D, Rinard M, Gifford D (2010) Rapid haplotype inference for nuclear families. Genome biology 11: R108.
2. Zhang K, Sun F, Zhao H (2005) Haplore: a program for haplotype reconstruction in general pedigrees without recombination. Bioinformatics 21: 90–103.
3. Browning BL, Browning SR (2009) A unified approach to genotype imputation and haplotype-phase inference for large data sets of trios and unrelated individuals. The American Journal of Human Genetics 84: 210–223.
4. Stephens M, Smith NJ, Donnelly P (2001) A new statistical method for haplotype reconstruction from population data. The American Journal of Human Genetics 68: 978–989.
5. Scheet P, Stephens M (2006) A fast and exible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. The American Journal of Human Genetics 78: 629–644.
6. Delaneau O, Coulonges C, Zagury JF (2008) Shape-it: new rapid and accurate algorithm for haplotype inference. BMC bioinformatics 9: 540.
7. Browning SR, Browning BL (2010) High-resolution detection of identity by descent in unrelated individuals. The American Journal of Human Genetics 86: 526–539.
8. Aguiar D, Istrail S (2013) Haplotype assembly in polyploid genomes and identical by descent shared tracts. Bioinformatics 29: i352–i360.

9. Bansal V, Bafna V (2008) Hapcut: an efficient and accurate algorithm for the haplotype assembly problem. Bioinformatics 24: i153–i159.

10. Aguiar D, Istrail S (2012) Hapcompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. Journal of Computational Biology 19: 577–590.

11. He D, Choi A, Pipatsrisawat K, Darwiche A, Eskin E (2010) Optimal algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics 26: i183–i190.

12. Bansal V, Halpern AL, Axelrod N, Bafna V (2008) An mcmc algorithm for haplotype assembly from whole-genome sequence data. Genome research 18: 1336–1346.

13. Efros A, Halperin E (2012) Haplotype reconstruction using perfect phylogeny and sequence data. BMC bioinformatics 13: S3.

14. Lancia G, Bafna V, Istrail S, Lippert R, Schwartz R (2001) Snps problems, complexity, and algorithms. In: AlgorithmsESA 2001, Springer. pp. 182–193.

15. Lippert R, Schwartz R, Lancia G, Istrail S (2002) Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. Briefings in bioinformatics 3: 23–31.

16. Panconesi A, Sozio M (2004) Fast hare: A fast heuristic for single individual snp haplotype reconstruction. In: Algorithms in Bioinformatics, Springer. pp. 266–277.

17. Abecasis G, Altshuler D, Auton A, Brooks L, Durbin R, et al. (2010) A map of human genome variation from population-scale sequencing. Nature 467: 1061–1073.

18. Geraci F (2010) A comparison of several algorithms for the single individual snp haplotyping reconstruction problem. Bioinformatics 26: 2217–2225.

19. Berger E, Yorukoglu D, Peng J, Berger B (2014) HapTree: A Novel Bayesian Framework for Single Individual Polyplotyping Using NGS Data. In: Research in Computational Molecular Biology, Springer. pp 18–19.