GPhase: Greedy Approach for Accurate Haplotype Inferencing

Kshitij Tayal, Naveen Sivadasan, Rajgopal Srinivasan
Life Sciences Division
TCS Innovation Labs
Hyderabad 500081, India
Email: {kshitij, naveen, raj}@atc.tcs.com

***Abstract*—We consider the computational problem of phasing an individual genotype sample given a collection of known haplotypes in the population. We give a fast and accurate algorithm GPhase for reconstructing haplotype pair consistent with input genotype. It uses the coalescent based mutation model of Stephens and Donnelly (2000). Computing optimal solution under this model is expensive and our algorithm uses a greedy approximation for fast and accurate estimation. Our algorithm is simple, efficient and has linear time and space complexity. Experiments on real datasets revealed improved gene level phasing accuracy for GPhase tool compared to other widely used tools such as SHAPEIT, Beagle, MaCH and Impute2. On simulated data, GPhase tool was able to phase samples each containing more than 1700 markers with high accuracy. GPhase can be used for gene level phasing of individual samples using publicly available haplotype datasets such as HapMap data or 1000 genome data. This finds applications in studies on recessive Mendelian disorders where parent data is lacking. GPhase is freely available for download and use from https://github.com/kshitijtayal/GPhase/.**

I. Introduction

Identification of genomic variants is known as variant calling. Since functional consequences very often depend on having two or more such variations either as part of the same or different haplotypes, it is critical to establish the relationship between SNPs as to whether they occur as part of the same haplotype or different haplotypes. Haplotype reconstruction from genotype data is known as phasing. In this work we consider the problem of phasing an individual genotype given a collection of known haplotypes.

Diploid organisms such as humans carry two homologous copies of each chromosome, one from each parent. Mutations and recombinations results in haplotype diversity in a population. In this work, we consider only biallelic genotypes with SNPs. Biallelic haplotypes can be represented as binary vectors, where a 0 and a 1 at a given SNP location (locus) indicates reference and alternative alleles respectively. Haplotypes in Fig. 1 can thus be represented by $(1, 0, 0, 0, 0, 1)$ and $(1, 1, 0, 1, 1, 0)$ and the genotype is represented by $(1/1, 0/1, 0/0, 0/1, 0/1, 0/1)$. First and third locus in this case are homozygous and rest are heterozygous. Since there are 4 heterozygous sites in this genotype, there are in total 2^4 haplotypes consistent with it. Thus, the solution space of haplotypes consistent with a given genotype has exponential dependence on the number of heterozygous sites.

The whole gamut of haplotype determination algorithms broadly falls under either haplotype assembly or haplotype inference. Haplotype assembly uses sequence reads directly to determine constituent haplotypes whereas haplotype inference use genotype data to infer haplotypes. Our work falls under the latter.

Clark's algorithm (1990) [3] was one of the first computational techniques used for haplotype inferencing. It follows parsimony approach where it attempts to restrict the number of distinct haplotypes observed in the sample. It performs reasonably well on instances with small set of markers, however its performance suffers if markers are sparsely connected. Expectation-Maximization (EM) based algorithms (1995) [6], [8], [12] assigns alleles to haplotypes with high likelihood using estimated values for population haplotype frequencies, typically assuming uniform prior for haplotypes. EM approaches are computationally expensive in case of large number of heterozygous sites.

$$--A--G--A---T--C---G-----$$
$$--C--G--A---T--C---T-----$$
$$--C--T--A---C--T---G-----$$

Fig. 1. First sequence is reference sequence. Other two sequences are two copies of a chromosome. Non SNP positions are marked by - while the rest are SNPs. There are 6 SNPs in total.

Approximate coalescent based methods, where new haplotypes are viewed as being derived from existing ones through mutation and recombination, achieved significant improvements in phasing quality. PHASE(v1.1) (2001) [18] was the first statistical method based on approximate coalescent based mutation model of Stephens and Donnelly [16] which identifies new haplotypes as derivative of old haplotypes with mutations. It uses Gibbs sampling to sample from the underlying Markov chain on the haplotype solution space. It exhibited superior accuracy than Clarks and EM but suffered from very high computational overhead. PHASE(v2.1.1) (2005) [17] used coalescent with recombination model to account for linkage disequilibrium and recombination. Similar to PHASE(v1.1) it is also computationally expensive. PHASE [18], [17] was considered gold standard for accuracy [2] but lost its importance due to slow speed. fastPHASE (2006)[15] made it possible to phase thousands of heterozygous sites at a considerable speed at the cost of reduced accuracy than its predecessors. Speed up in fastPHASE is achieved by locally clustering the haplotype using Hidden Markov model (HMM).

Beagle (2007) [1], similar to fastPHASE, use HMM to infer haplotypes. Similar to fastPHASE, it performs local clustering of haplotypes at each position, but allowing different number of clusters at each marker. Haplotype clusters form HMM states here. Impute2 (2009) [9] and MaCH (2010) [11] can operate on large data sets than what PHASE can deal with and simultaneously achieving greater accuracy than fastPHASE. They can be used for imputation of untyped variants as well. SHAPEIT (2012)[5] is an enhancement over Impute2 and MACH that scales linearly in the number of SNPs. It is lot more faster, accurate and uses less memory as compared to other two. Availability of large amounts of public haplotype data such as 1000 genome data [4] and HapMap data [7] have contributed to further improvements in phasing accuracy. We refer the reader to [2] for a review on various phasing algorithms.

A. *Our Contribution*

We consider the problem of phasing an individual genotype sample given a collection of known haplotypes in the population. The known haplotype collection could for instance be the publicly available haplotype data such as HapMap data [7] or 1000 genome data[4]. We give a fast and accurate algorithm GPhase for reconstructing a haplotype pair consistent with input genotype. Our algorithm is based on the coalescent based mutation model of Stephens and Donnelly [16] (without recombination) which is used in PHASE (v1.1) [18]. Computing optimal solution under this model is expensive and our algorithm uses greedy approximation for fast and accurate estimation. Our algorithm is simple, efficient and has linear time and space complexity. It constructs the solution incrementally while maintaining a collection of top k candidate solutions at each step and finally uses the top most solution as its solution. Experiments on real datasets revealed improved gene level phasing accuracy for GPhase tool compared to SHAPEIT, Beagle, MaCH and Impute2. On simulated data, GPhase tool was able phase samples each containing more than 1700 markers with high accuracy. Our algorithm can be viewed as an instance based learner that, for each input sample, infers most probable solution from whole training data (known haplotype collection) using the mutation model. This is in contrast to other approaches that either use simpler mutation models or build simple generalized models such as HMM from the training data for faster inferencing, which results in information/accuracy loss. GPhase tool can be used for gene level phasing of individual samples using publicly available haplotype datasets such as 1000 genome data. This has applications in studies on recessive Mendelian disorders in the absence of parent data. The tool is freely available for download and use from https://github.com/kshitijtayal/GPhase/.

II. Algorithm

Notations: We use 0 and 1 to represent two alleles of a biallelic locus. A haplotype h is specified over l loci by an l-dimensional binary vector. A genotype G characterized over l loci is also an l-dimensional vector from $\{0,1,2\}^l$, where $0,1$ and 2 are counts of the allele coded as 1. It is assumed that genotype data has no missing information. Let $h(i)$ and $G(i)$ denote the ith locus of haplotype h and genotype G respectively. Let $h(i:j)$ and $G(i:j)$ for $i \leq j$ denote the sequence of loci $i, i+1, \ldots, j$ in h and G respectively. Pair of haplotypes (h_1, h_2), called diploid is said to be consistent with genotype G if $G(i) = h_1(i) + h_2(i)$ for all $i \in \{1, \ldots, l\}$. Let $A_n = \{a_1, \ldots, a_n\}$ denote a multi-set of n haplotypes each consisting of l loci. We also refer to A_n as an $n \times l$ haplotype matrix with rows numbered $1, \ldots, n$ and columns numbered $1, \ldots, l$ and $A_n(i,j)$ denoting the jth locus of haplotype a_i. We may conveniently refer to A_n as a set or as a matrix.

A. Review of Mutation Model [16]

In this section we review relevant parts of the coalescent based model proposed by Stephens and Donnelly (2000) [16] which is used by PHASE (v1.1) [18] for haplotype inferencing. Our algorithm uses this model for inferencing. We refer the reader to [16], [18] for details. Let A_n denote a multi-set of n known haplotypes each consisting of l loci. Let $\pi(h|A_n)$ denote the conditional distribution of observing an l loci haplotype h given A_n. Stephens and Donnelly [16], proposes an approximation to $\pi(h|A_n)$ as

$$\pi(h|A_n) = \sum_{\alpha \in A_n} \sum_{m=0}^{\infty} \frac{1}{n} \left(\frac{\theta}{n+\theta}\right)^m \frac{n}{n+\theta} (P^m)_{\alpha h},$$

where P is the mutation matrix and θ is a scaled mutation rate. That is, $\pi(h|A_n)$ corresponds to the probability of choosing one haplotype say α uniformly at random from A_n and applying m mutations to α to obtain h, where m is geometrically distributed with parameter $\theta/(n+\theta)$. Since computing the above expression can be expensive, Stephens and Donnelly [16] makes further simplifying assumption that each locus mutates independently at rate $\theta/2$ and thus with a total rate of $l\theta/2$ and according to a 2×2 transition matrix (for biallelic case) P. In this simplified model, a haplotype α is chosen uniformly at random from A_n and thereafter m locations (with repetition) are chosen uniformly from α and mutated according to mutation matrix P, where m is geometrically distributed with parameter $l\theta/(n+l\theta)$. Using properties of Poisson distribution, this process can be equivalently viewed as drawing a time t from exponential distribution with rate parameter 1 and applying m_i mutations to each locus $i \in \{1, \ldots, l\}$ where m_i values are independent and are Poisson distributed with parameter $\theta t/n$. Mutations at each locus are again as per the mutation matrix P. This yields the following modified expression

$$\pi(h|A_n) = \sum_{\alpha \epsilon A_n} \frac{1}{n} \int exp(-t) F^{(\theta,t,n)}_{\alpha(1)h(1)} \cdots F^{(\theta,t,n)}_{\alpha(l)h(l)} dt$$

where

$$F^{(\theta,t,n)}_{\alpha(i)h(i)} = \sum_{m=0}^{\infty} \frac{(\theta t/n)^m}{m!} \exp\left(-\frac{\theta t}{n}\right) (P^m)_{\alpha(i)h(i)}$$

The integral in the above expression is further approximated using Gaussian quadrature to finally obtain

$$\pi(h|A_n) = \frac{1}{n} \sum_{\alpha \in A_n} f(\alpha, h) \tag{1}$$

where

$$f(\alpha, h) = \sum_{i=1}^{s} w_i F^{(\theta,t_i,n)}_{\alpha(1)h(1)} \cdots F^{(\theta,t_i,n)}_{\alpha(l)h(l)} \tag{2}$$

Here $t_1, \ldots, t_s$ are the quadrature points and $w_1, \ldots, w_s$ are the quadrature weights (marginal probabilities). Matrices $F^{(\theta,t_i,n)}_{\alpha(i)h(i)}$ can be precomputed where the infinite sum can be well approximated by a finite sum of a large number of terms.

For a haplotype pair $H = (h_1, h_2)$ we have

$$\Pr(H|A_n) \propto \pi(h_1|A_n, h_2)\pi(h_2|A_n) \tag{3}$$

In fact, for large n, finding $H = (h_1, h_2)$ that maximizes eq. (3) can be approximated by finding H that maximizes

$$\gamma(h_1, h_2) = \pi(h_1|A_n)\pi(h_2|A_n). \tag{4}$$

PHASE algorithm uses the above model to phase a collection of genotypes $G = \{G_1, \ldots, G_n\}$ [18].

Their algorithm uses Gibbs sampling where in each iteration, a random genotype G_i from G is phased assuming all remaining genotypes in G are correctly phased. If H_{-i} denote the reconstructed haplotypes for all genotypes in G except G_i, then most probable haplotype h_i consistent with G_i is inferred using equation (1) for $\pi(h|H_{-i})$ and is selected as a haplotype for G_i. The process is repeated until the underlying Markov chain on the solution space mixes.

Size of the set of haplotypes $\{h_i\}$ consistent with a genotype G_i has exponential dependency on the number of heterozygous loci in G_i. Hence, $|\{h_i\}|$ can be very large in general and this makes finding most probable h_i consistent with genotype G_i computationally expensive. This issue is partially addressed in PHASE [18] by considering only few random heterozygous loci in G_i in each iteration of Gibbs sampling and thereby restricting the size of $\{h_i\}$ for each updation [18]. However this can result in increased mixing time. Simplified alternative models were also considered in other tools such as SHAPEIT and fastPHASE for performance improvements. In the next section, we describe our algorithm that approximately solves the problem of computing a haplotype pair (h_1, h_2) consistent with genotype G that maximizes eq (4) given a collection of known haplotypes A_n. Our algorithm runs in time linear in the number of heterozygous loci in G. We consider only biallelic genotypes.

B. GPhase Algorithm

We describe GPhase algorithm to solve the following problem: given a collection A_n of known haplotypes and an unphased genotype G, reconstruct the haplotype pair $H = (h_1, h_2)$ that is consistent with G and maximizes $\gamma(h_1, h_2)$ given by equation (4). If there is a small collection of genotypes to be phased given the known haplotype collection A_n, then the algorithm can be used to phase each genotype in the collection separately. We describe a greedy algorithm to solve this problem approximately. Our experiments for gene level phasing show that the phasing quality achieved by our greedy approximation exhibit superior quality compared to other widely used phasing tools.

Let l denote the number of loci in G and in haplotypes belonging to A_n. For brevity of notation, given θ and n, let

$$\Delta^i_{ab} = F^{(\theta,t_i,n)}_{ab} \tag{5}$$

where $a, b \in \{0, 1\}$. We remark that $\Delta^i_{00}, \Delta^i_{01}, \Delta^i_{10}$ and Δ^i_{11} for $i \in \{1, \ldots, s\}$ are fixed for a given problem as θ and n are fixed. Given two haplotypes α and β, $f(\alpha, \beta)$ can be written as

$$f(\alpha, \beta) = \sum_{i=1}^{s} w_i c_i(\alpha, \beta) \tag{6}$$

where

$$c_i(\alpha, \beta) = \prod_{j=1}^{l} \Delta^i_{\alpha(j)\beta(j)} \tag{7}$$

For $i \in \{1, \ldots, s\}$, $c_i(\alpha, \beta)$ can be interpreted as a 'product similarity' between α and β where for each locus j, depending on $(\alpha(j)\beta(j)) \in \{00, 01, 10, 11\}$, a multiplicative term $\Delta^i_{\alpha(j)\beta(j)}$ is contributed to $c_i(\alpha, \beta)$. Function $f(\alpha, \beta)$ can hence be interpreted as the weighted sum of s similarity values $c_1(\alpha, \beta), \ldots, c_s(\alpha, \beta)$. It is easy to see that each locus contributes independently to similarity values $c_i(\alpha, \beta)$ and the final weighted sum $f(\alpha, \beta)$ is invariant to the relative ordering of loci. It is also easy to verify that the same holds true for $\pi(h|A_n)$ and that the optimal solution that maximizes $\gamma(h_1, h_2)$ is thus invariant any fixed permutation of columns of A_n and loci of G where the same permutation is applied to both.

For ease of exposition, before describing the GPhase algorithm, we consider a related problem: given a collection A_n of known haplotypes and an unphased genotype G, find haplotype h that is consistent with G and maximizes $\pi(h|A_n)$. We give a fast greedy algorithm GPhaseSingle (Algorithm 1) to find an approximate solution to this problem. We remark that this is a fundamental and computationally expensive sub-problem in some of the existing phasing tools. Our approach could hence be of use in other phasing methods as well. We will later extend this algorithm to solve the original problem. The algorithm considers one locus at a time and builds candidate solutions incrementally in a greedy fashion. It maintains its top k candidate solutions (for a fixed parameter k) that are updated after each

incremental extension. The order in which loci of G is considered for phasing is governed by the column ordering of matrix A_n resulting from the following *skewness permutation* of its columns.

Skewness permutation of $\mathbf{A_n, G}$: Let $A_n(j)$ be the jth column of matrix A_n. Let $\delta(j)$ denote the absolute value of difference between total number of 1s and total number of 0s in column $A_n(j)$. Skewness permutation of the columns of matrix A_n and correspondingly genotype G are done in the following manner. Let $L \subseteq \{1, \ldots, l\}$ denote the subset of heterozygous loci in G and let $l' = |L|$. Reorder columns of A_n such that its first $l - l'$ columns correspond to columns in $\{1, \ldots, l\} - L$ (corresponding to homozygous loci in G) in an arbitrary order. Remaining l' columns correspond to columns in L arranged in the decreasing order of their corresponding $\delta(j)$ values. Hence, in the permuted matrix of A_n, columns corresponding to homozygous loci in G appear first followed by columns corresponding to heterozygous loci in G in the decreasing order of their 0/1 skewness ($\delta(j)$ values).

By considering columns of the permuted matrix from left to right, loci (columns) with larger 0/1 skewness are considered earlier because phasing them can be done with lesser uncertainty. From now we assume that columns of A_n (and correspondingly loci in G) are reordered based on its skewness permutation. Recalling the notations, let $h(1 : j)$ for $j < l$ denote a partial haplotype solution for loci $1, \ldots, j$. Let $\pi(h(1 : j)|A_n)$ denote the value of $\pi(h|A_n)$ when both h and A_n are restricted to columns $1, \ldots, j$. The greedy algorithm maintains top-k partial solutions $h_1(1 : j), \ldots, h_k(1 : j)$, for a fixed parameter k, using a k element min heap data structure based on their $\pi(h_i(1 : j)|A_n)$ values. Recalling that the first $l - l'$ loci of G are all the homozygous loci of G (after permutation), the heap is initialized with a single partial solution $h(1 : l-l')$ where $h(j) = G(j)$ for $j \in \{1, \ldots, l - l'\}$, with its corresponding value $\pi(h_i(1 : l - l')|A_n)$. Each candidate top-k partial solution $h(1 : j)$ from the heap is extended by one locus $j+1$ in the following manner. Let $h(1 : j)0$ denote the sequence of length $j + 1$ obtained by appending 0 as the rightmost element to the sequence $h(1 : j)$. Consider the two possible extensions $h_0 = h(1 : j)0$ and $h_1 = h(1 : j)1$ which are $j + 1$ length haplotypes obtained by appending 0 and 1 respectively to $h(1 : j)$. Both h_0 and h_1 are considered for insertion into the top k heap of extended solutions based on their values $\pi(h_0(1 : j+1)|A_n)$ and $\pi(h_1(0 : j+1)|A_n)$. Finally, after scanning all l loci, haplotype h with maximum value in the k-heap is output as the solution. Top k partial solutions are maintained instead of only the current best in order to handle situations where final top solutions are suboptimal for the partial solutions considered during extension steps. Larger values of k would improve the quality of final solution. These steps are given below as Algorithm 1. Parameters for the algorithm are mutation rate θ, quadrature points $\{t_1, \ldots, t_s\}$, quadrature weights $\{w_1, \ldots, w_s\}$ and heap size k.

Algorithm 1. *GPhaseSingle*

Input: A_n and G on l loci.
Output: Haplotype h consistent with G.

/* Let l' be the number of heterozygous loci in G. */
Do skewness permutation of A_n, G
Initialize partial solution $h(1 : l - l')$ with $h(j) = G(j)$ for $j \in \{1, \ldots, l - l'\}$.
Insert $h(1 : l - l')$ in $\mathcal{T}$ with value $\pi(h(1 : l - l')|A_n)$.
for $j = l - l'$ to $l - 1$ **do**
  $S :=$ Elements of $\mathcal{T}$.
  Empty $\mathcal{T}$.
  for all $h(1 : j)$ in S **do**
    Consider extensions $h_0 = h(1 : j)0$ and $h_1 = h(1 : j)1$.
    Insert h_0 in $\mathcal{T}$ with value $\pi(h_0(1 : j + 1)|A_n)$.
    Insert h_1 in $\mathcal{T}$ with value $\pi(h_1(1 : j + 1)|A_n)$.
  end for
end for
return $h \in \mathcal{T}$ with maximum value.
end

Implementation and Run time: Permutation of A_n and G involves computing $\delta(j)$ values and sorting them, which can be achieved in $O(nl + l\log(l)) = O(nl)$ time. We note that since min heap $\mathcal{T}$ maintains top-k largest solutions, inserting a new element h in $\mathcal{T}$ succeeds only if either heap size is less than

k or if heap minimum is less than value of h, in which case the minimum element is replaced with h. In order to compute $\pi(h(1:j)|A_n)$ efficiently, we maintain k two dimensional arrays of the form $B[1..n][1..s]$, where s is the number of quadrature points, one such array for each of the top k partial solutions, with total $O(nk)$ space. For each top k partial solution $h(1:j)$, $B[i][r]$ entries in its corresponding B array store $c_r(a_i(1:j), h(1:j))$ values as given by eq. (7). That is, for a top k partial solution $h(1:j)$, we store its s component similarity values $c_1(), \ldots, c_s()$ between each haplotype $a_i \in A_n$ and h, both restricted to loci $1, \ldots, j$. From eq.(1) and eq.(6), it is easy to see that value of partial solution $\pi(h|A_n)$ is obtained from its corresponding B array as $(1/n)\sum_{i=1}^{n}\sum_{r=1}^{s} w_r \cdot B[i][r]$. From eq. (7), it is straightforward to verify that if a partial solution $h(1:j)$ is extended by say 0, values in its $B[i][r]$ array entries can be easily updated as either $B[i][r] \times \Delta_{00}^r$ or $B[i][r] \times \Delta_{10}^r$ depending on whether $j+1$st locus of a_i is either 0 or 1. From Algorithm 1, it follows that the total cost of one iteration of the outer loop is $O(nk)$ and the total run time is thus $O(nlk)$. Thus, for fixed k, the algorithm runs in linear time and uses linear space.

GPhase algorithm: The final algorithm is obtained by straightforward modification of the previous GPhaseSingle algorithm. Instead of maintaining top k partial solutions which are single haplotypes, we maintain k haplotype pairs $(h_1(1:j), h_2(1:j))$ consistent with $G(1:j)$ as partial solutions. That is, $G(i) = h_1(i) + h_2(i)$ for $i = 1, \ldots, j$. Value of the partial solution is given by $\gamma(h_1(1:j), h_2(1:j)) = \pi(h_1(1:j)|A_n) \cdot \pi(h_2(1:j|A_n)$ (see eq.(4)). Extending a partial solution pair $(h_1(1:j), h_2(1:j))$ give rise to two new solution pairs viz. $(h_1(1:j)0, h_2(1:j)1)$ and $(h_1(1:j)1, h_2(1:j)0)$. Finally, the solution pair with maximum value is output. The final algorithm is given in Algorithm 2. Parameters for the algorithm remain the same.

By arguments similar to that for GPhaseSingle, it follows that GPhase require total $O(nlk)$ time and uses $O(nk)$ space. Hence it runs in linear time and uses linear space for fixed k.

Algorithm 2. *GPhase*

Input: A_n and G.
Output: Haplotype pair (h_1, h_2) consistent with G.

/* Let l' be the number of heterozygous loci in G. */
Do skewness permutation of A_n, G
Initialize partial solution $(h_1(1:l-l'), h_2(1:l-l'))$ with $h_1(j) = h_2(j) = G(j)$ for $j \in \{1, \ldots, l-l'\}$.
Insert $(h_1(1:l-l'), h_2(1:l-l'))$ in $\mathcal{T}$ with value $\gamma(h_1(1:l-l'), h_2(1:l-l'))$.
for $j = l-l'$ to $l-1$ **do**
  $S :=$ Elements of $\mathcal{T}$.
  Empty $\mathcal{T}$.
  for all $(h_1(1:j), h_2(1:j))$ in S **do**
    $h_0' = h_1(1:j)0$ and $h_1' = h_2(1:j)1$
    $h_1'' = h_1(1:j)1$ and $h_0'' = h_2(1:j)0$
    Insert (h_0', h_1') in $\mathcal{T}$ with value $\gamma(h_0'(1:j+1), h_1'(1:j+1))$.
    Insert (h_0'', h_1'') in $\mathcal{T}$ with value $\gamma(h_0''(1:j+1), h_1''(1:j+1))$.
  end for
end for
return $(h_1, h_2) \in \mathcal{T}$ with maximum value.
end

III. Results

We conducted experiments where we compared GPhase phasing quality with other existing tools. We performed experiments on both real datasets and simulated datasets. In particular, we compared GPhase output with phasing output of Beagle [1], MaCH [11], Impute2 [9] and SHAPEIT [5]. For comparison, we used the standard metrics of accuracy and switch accuracy [13]. Accuracy is defined as $(l-1-ms)/(l-1)$, where l is the number of heterozygous loci in the input genotype and ms is the number of mismatching loci between inferred haplotype and true haplotype. Switch accuracy is defined as $(l-1-sw)/(l-1)$ where sw is the number of switches to recover true haplotype from inferred haplotype.

In all our experiments, model parameters used by GPhase, namely mutation rate θ, quadrature points, quadrature weights and matrix P were same as that in PHASE implementation [17]. Value θ was set to

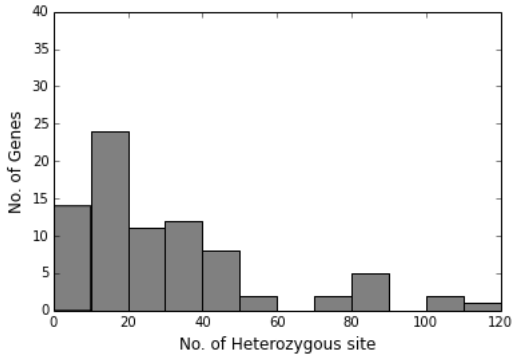

Fig. 2. Distribution of gene counts based on the average no. of heterozygous sites for 10 individuals. Heterozygous count range is split into 12 equal sized intervals and no. of genes falling in each interval is plotted separately.

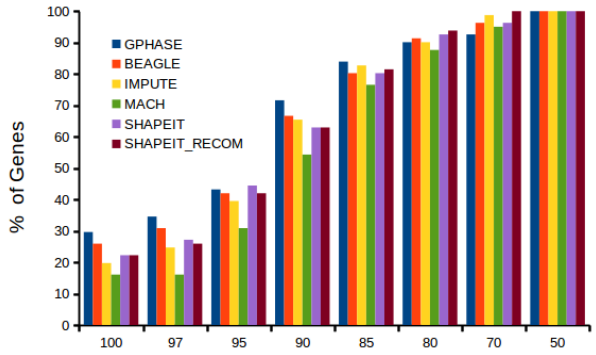

Fig. 3. Tool-wise cumulative distributions where for different accuracy values, the percentage of genes whose mean switch accuracy is at least the specified accuracy value is plotted.

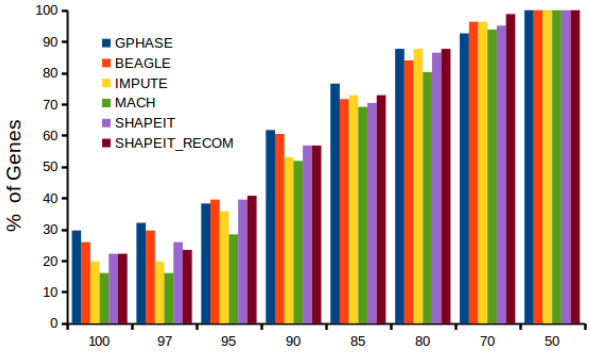

Fig. 4. Tool-wise cumulative distributions where for different accuracy values, the percentage of genes whose mean accuracy is at least the specified accuracy value is plotted.

$1/\log(2n)$ and number of quadrature points were 2. The 2×2 transition matrix P required for precomputing Δ_{ij}^r has entries $P_{00} = P_{11} = 0$ and $P_{01} = P_{10} = 1$. Value of k was set to 20.

A. *Real Dataset*

For real dataset, we used in-house data that consisted of genotype samples from 10 unrelated individuals. For each individual, there were 1529 SNP markers from across chromosomes. These markers spanned 81 genes. There were 2463 heterozygous loci in total across all individuals. Constituent haplotypes for these individual genotype samples were already known from their nuclear family data. This was used for validation. We conducted separate phasing experiment on each gene level sample for each individual using the collection of known haplotypes extracted from 1000 genome data. Fig 2 shows the distribution of genes based on the average number of heterozygous loci in their corresponding genotypes (from 10 individuals).

We used 1000 genome public dataset[4] to create the known haplotype set A_n. In our experiments, A_n consisted of 5008 individual haplotypes from 26 populations. All tools were run with default settings. We ran SHAPEIT on the inputs both under default recombination setting and under variable recombination setting. In the latter case, recombination data was provided using standard chromosome map files for human chromosomes [10]. Flag -no-mcmc was used with SHAPEIT and SHAPEIT_RECOM.

Figures 4 and 3 provide accuracy plots for phasing outputs from all the candidate tools. They provide cumulative distribution of the percentage of genes (out of 81) whose mean accuracy and mean switch accuracy (across 10 individuals) are at least w for different accuracy values w. From these plots we see that GPhase phasing quality is superior to other tools for most of the plotted accuracy values. This is true in particular for higher accuracy values.

Figures 5 and 6 give plots of mean accuracy and mean switch accuracy of GPhase separately for inputs whose heterozygous loci count falls in different ranges. As evident from these plots, GPhase performs comparably well on inputs with wide range of heterozygous loci counts.

B. *Simulated Data*

For simulated data generation, we used Cosi2 [14] as haplotype simulation tool with parameters calibrated to empirical human data. We ran

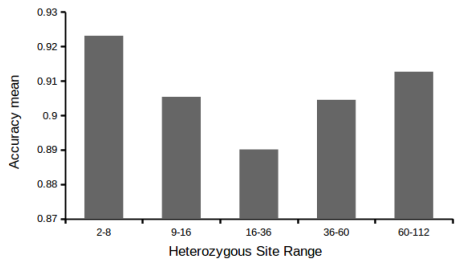

Fig. 5. Mean accuracy of GPhase for genotype inputs falling under different heterozygous loci count ranges.

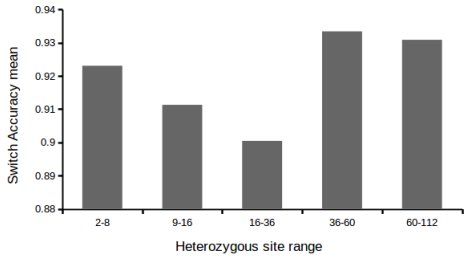

Fig. 6. Mean switch accuracy of GPhase for genotype inputs falling under different heterozygous loci count ranges.

it with default settings for European population. Three separate experiments were conducted based on the total number of simulated haplotypes generated, viz. 1000 haplotypes, 5000 haplotypes and 10,000 haplotypes. Haplotypes in the collection had $l = 1760$ markers. Each simulated data set has a recombination rate sampled from a distribution matching the decode map[10] with recombination clustered into hotspots. The simulation package can be obtained from http://www.broad.mit.edu/sfs/cosi.

Simulated data were devoid of SNP ids. Hence public reference haplotype datasets were not included in these experiments. We used leave-one-out cross-validation (LOOCV) where known set A_n is created by including all haplotypes except one pair and we infer constituent haplotypes from the combined genotype of the remaining pair. We ran these experiments with GPhase and with SHAPEIT. Default recombination settings were used for SHAPEIT. Mean switch accuracy results for GPhase and SHAPEIT for these cross validation experiments are given in table I. GPhase phasing exhibited comparable or improved quality for all three values of n.

We note that genotype samples considered in this experiment consists of about 1700 markers which is more than the typical number of markers seen at gene level. GPhase is able to obtain good phasing accuracy here as well even without explicitly modeling recombination. We believe that this is due to sufficiently large collection of known haplotypes A_n in these experiments as larger collections would contain several haplotype instances resulting from different recombination possibilities. This perhaps alleviates the need for explicitly factoring recombination into the model.

TABLE I
AVERAGE PERCENTAGE SWITCH ACCURACY FOR CROSS VALIDATION EXPERIMENTS ON SIMULATED DATASETS WITH DIFFERENT SIZES n.

	$n = 1000$	$n = 5000$	$n = 10,000$
GPhase	97.51	97.5	98.73
SHAPEIT	96.1	97.2	98.16

We also performed leave-one-out cross-validation on 1000 genome data using GPhase. The data consisted of 5008 haplotypes each with 4859 markers. As in earlier experiment, all haplotypes except one pair were provided as A_n in each cross-validation experiment. GPhase achieved mean switch accuracy of 97.82 percentage in this case.

As discussed earlier, GPhase has linear run time and uses linear space. The tool implementation was done in Python. The tool does fast phasing and all inputs were phased in a matter of seconds.

IV. CONCLUSION

We considered the problem of phasing an individual genotype sample given a collection of known haplotypes. We give a greedy approximation algorithm GPhase that uses the coalescent based mutation model of Stephens and Donnelly [16], which is also used in PHASE (v1.1) [18], for inferring a consistent haplotype pair. Though computing optimal solution under this model is computationally expensive, our algorithm computes an approximate solution efficiently in linear time and space. Experiments on real datasets revealed improved gene level phasing accuracy for GPhase tool compared to widely used tools such as SHAPEIT, Beagle, MaCH and Impute2. On simulated data, GPhase tool was able phase samples containing more than 1700 markers with high accuracy. GPhase tool can

be used for gene level phasing of individual samples using publicly available haplotype datasets such as 1000 genome data. This has applications in studies on recessive Mendelian disorders in the absence of parent data. It would be interesting to provide theoretical bounds on the quality of the solution computed by GPhase algorithm, perhaps under certain assumptions on the known haplotype collection, that explains empirical findings. It would also be worthwhile to conduct chromosome level phasing experiments using GPhase with large collections of known haplotypes to evaluate its ability to deal with recombinations.

REFERENCES

[1] S. R. Browning and B. L. Browning. Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering. *The American Journal of Human Genetics*, 81(5):1084–1097, 2007.

[2] S. R. Browning and B. L. Browning. Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, 12(10):703–714, 2011.

[3] Andrew G Clark. Inference of haplotypes from pcr-amplified samples of diploid populations. *Molecular biology and evolution*, 7(2):111–122, 1990.

[4] 1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature*, 526(7571):69–74, 2015.

[5] O. Delaneau, J. Marchini, and J. F. Zagury. A linear complexity phasing method for thousands of genomes. *Nature methods*, 9(2):179–181, 2012.

[6] L. Excoffier and M. Slatkin. *Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population*. Molecular Biology and Evolution 12(5):921927, 1995.

[7] R. A. Gibbs, J. W. Belmont, P. Hardenbol, T. D. Willis, F. Yu, H. Yang, L. Y. Ch'ang, W. Huang, B. Liu, Y. Shen, and P. K. H. Tam. The international HapMap project. *Nature*, 426(6968):789–796, 2003.

[8] M. E. Hawley and K. K. Kidd. Haplo: A program using the EM algorithm to estimate the frequencies of multi-site haplotypes. *Journal of Heredity*, 86:409, 1995.

[9] B. N. Howie, P. Donnelly, and J. Marchini. A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. *PLoS Genet*, 5(6):e1000529, 2009.

[10] A. Kong, D. F. Gudbjartsson, J. Sainz, G. M. Jonsdottir, S. A. Gudjonsson, B. Richardsson, S. Sigurdardottir, J. Barnard, B. Hallbeck, G. Masson, A. Shlien, S. T. Palsson, M. L. Frigge, T. E. Thorgeirsson, J. R. Gulcher, and K. Stefans-son. *A high-resolution recombination map of the human genome*. Nature Genetics, 31(3):241247, 2002.

[11] Y. Li, C. J. Willer, J. Ding, P. Scheet, and G. R. Abecasis. Mach: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genetic epidemiology*, 34(8):816–834, 2010.

[12] J. C. Long, R. C. Williams, and M. Urbanek. *An E-M algorithm and testing strategy for multiple-locus haplotypes*. American Journal of Human Genetics, 56(2):799810, 1995.

[13] M.E. Zwick S. Lin, D.J. Cutler and A. Chakravarti. Haplotype inference in random population samples. *The American Journal of Human Genetics*, 71(5):1129–1137, 2002.

[14] S. F. Schaffner, C. Foo, S. Gabriel, D. Reich, M. J. Daly, and D. Altshuler. Calibrating a coalescent simulation of human genome sequence variation. *Genome research*, 15(11):1576–1583, 2005.

[15] P. Scheet and M. Stephens. A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *The American Journal of Human Genetics*, 78(4):629–644, 2006.

[16] M. Stephens and P. Donnelly. Inference in molecular population genetics. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):605–635, 2000.

[17] M. Stephens and P. Scheet. Accounting for decay of linkage disequilibrium in haplotype inference and missing-data imputation. *The American Journal of Human Genetics*, 76(3):449–462, 2005.

[18] M. Stephens, N. J. Smith, and P. Donnelly. 2001. *A new statistical method for haplotype reconstruction from population data. The American Journal of Human Genetics*, 68(4):978–989, 2001.