HapMonster2: A Statistical Approach for Variant Calling and Haplotyping Based on Sequence Reads with Less Switching Errors

Kaname Kojima, Yosuke Kawai, Naoki Nariai, Takahiro Mimori, and Masao Nagasaki

Abstract—Haplotype phasing is essential for identifying disease-causing variants with phase-dependent interactions as well as for the coalescent-based inference of demographic history. One of approaches for estimating haplotypes is to use phase-informative reads, which span multiple heterozygous variant positions. Although the quality of estimated variants is crucial in haplotype phasing, accurate variant calling is still challenging due to errors on sequencing and read mapping. Since some of such errors can be corrected by considering haplotype phasing, simultaneous estimation of variants and haplotypes is important. We propose a statistical approach for variant calling and haplotype phasing named HapMonster2, where haplotype phasing information is used for improving the accuracy of variant calling and the improved variant calls are used for more accurate haplotype phasing. Since parameter estimation falls into bad local optima from bad initial parameters, we propose new procedures for initialization of model parameters for accurate haplotype estimation. We also devise a new algorithm for filtering out unreliable haplotypes to avoid switching errors. From the comparison with existing methods on simulation and real sequencing data, HapMonster2 is effective in both variant calling and haplotyping, while the results of HapMonster2 contain the least or second least switching errors in most of conditions.

Index Terms—next generation sequencing, variant call, haplotype phasing.

1 Introduction

Next generation sequencing (NGS) technologies enables the detection of novel rare variants in genome wide scale. For rare variant association studies, variants are often grouped in exon or gene level, and effects of multiple variants and phase-dependent interactions such as compound heterozygosity and cis-effect are considered for the analysis [2]. Thus, haplotype phasing is essential for identifying association between rare variants and disease phenotypes. In addition, haplotype phasing information is required for the coalescent-based inference of demographic history [10].

Haplotype phasing is usually estimated by two types of approaches. One approach is based on linkage disequilibrium between variant sites and estimates haplotype phasing probabilistically [3], [5], [8], [15], [19]. Although this type of approach provides accurate phasing results for common variants, it requires genotyping results for multiple samples and its accuracy for low-frequency variants or variants around recombination hot spots tends to be low. Another approach is to use phase-informative reads that span multiple heterozygous variant positions. Due to the current length of NGS reads, the range size of the estimated haplotypes is limited. However, since the length of sequence reads is growing rapidly (e.g., available read length from Illumina MiSeq has been doubled from 150 bp to 300 bp in one year), the rate of heterozygous sites that can be phased by phase-informative reads is increasing, and hence this type of approach is considered as a promising ways for phasing low-frequency variants.

Although the quality of estimated variants is essential for haplotype phasing, accurate variant calling is still challenging especially for regions with insufficient read coverage or regions containing errors on sequencing and read mapping. Since variant calling on such regions can be improved by using haplotype phasing information [4], [18], [21], considering variant calling and haplotype phasing simultaneously is important. However, SNP sites are treated independently on most of the variant callers such as Unified Genotyper in GATK and BCFtools [6], [13], [14], [16], [24], and haplotype phasing is considered separately from the variant calling. We so far proposed a new variant calling and haplotyping approach for sequencing data from individual diploid genomes named HapMonster [7], which simultaneously performs variant calling and haplotype phasing based on phase-informative reads by unifying these procedures in a statistical model. Under the model, haplotype phasing information can be used for improving the accuracy of variant calling and the improved variant calls are used for more accurate haplotype phasing as is done in LinkageMethod [18]. However, the parameter estimation tends to fall into bad local optima from bad initial parameters in the proposed model, and haplotyping results from bad local optima contain many switch errors, compared to other existing algorithms. In order to address this issue, we propose a statistical variant calling and haplotyping approach named HapMonster2 based on HapMonster. In HapMonster2, we propose a new procedure considering read assignment to haplotypes based on reliability of possible heterozygous positions. We also devise a new algorithm for filtering out

- *K. Kojima, Y. Kawai, T. Mimori, and M. Nagasaki are with Tohoku Medical Megabank Organization, Tohoku University, Sendai-shi, Miyagi, 980-8573, Japan.*
- *N. Nariai is with Institute of Genomics, University of California, La Jolla, CA, 92093-0838, USA.*
  E-mail: nagasaki@megabank.tohoku.ac.jp

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2685583, IEEE Transactions on Cloud Computing

unreliable haplotypes to avoid switching errors.

In the performance evaluation, we applied HapMonster2 and other existing methods to simulation and real sequencing datasets with various read coverages, and confirmed that HapMonster2 provides the best results in both variant calling and the number of correctly phased heterozygous positions, compared to other methods. In addition, the results of HapMonster2 contain the least or second least switching errors in most of conditions in the experiment.

2 METHODS

HapMonster2 takes mapped sequence reads to a reference genome in the SAM/BAM format as input data and estimates variants and phased haplotypes. The model of our approach is comprised of allele likelihood model and haplotype selection model. Allele likelihood model calculates the likelihood of an allele given mapped reads at a position, where errors on sequencing and read mapping are considered. Haplotype selection model represents the assignment of sequence reads to one of two haplotypes. In the following sections, we describe the details of our model and procedures for parameter estimation and inference of genotypes and haplotypes.

2.1 Modeling

2.1.1 Allele likelihood

Let R_i be the ith read in a SAM/BAM file. R_i contains its mapping quality score in Phred scale $MAPQ_i$, strings for bases aligned to position k in the reference genome r_i^k, and vectors of base quality scores in Phred scale for the aligned bases bq_i^k. Note that r_i^k is usually one nucleotide such as 'T', but it can be a string with more than one nucleotide for representing insertion, e.g., a string 'TGC' represents an insertion 'GC' right after a base 'T'. r_i^k can also be a null string to represent deletion. Based on these notations, we give allele likelihood for A at position k as:

$$\begin{aligned} P(r_i^k|A, bq_i^k) &= \sum_{b_i^k=0,1} \sum_{m_i^k=0,1} P(r_i^k|A, b_i^k)^{I(m_i^k=1)} \\ &\times P_{mis}(r_i^k)^{I(m_i^k=0)} P(m_i^k) P(b_i^k|bq_i^k), \end{aligned} \quad (1)$$

where m_i^k is a binary variable that takes one if the alignment of r_i^k is correct and zero otherwise. b_i^k is a vector of binary variables and each element indicates the correctness of each base in r_i^k, i.e., if sequencing of a base is correct, the corresponding element takes one and zero otherwise. $I(\cdot)$ is an indicator function that returns one if a condition in its argument is true, and zero otherwise. As with r_i^k, A is one nucleotide or a string with nucleotides. The term $P(r_i^k|A, b_i^k)$ in Equation (1) is the probability of read generation for the correct read alignment, and we represent the probability as:

$$P(r_i^k|A, b_i^k) = \text{Indel}(A, r_i^k) \prod_{l=1}^{\min\{|A|,|r_i^k|\}} P(r_i^k[l]|A[l], b_i^k[l]), \quad (2)$$

where $A[l]$ is the lth nucleotide of A, $r_i^k[l]$ is the lth nucleotide of r_i^k, $b_i^k[l]$ is the lth value of b_i^k, and function Indel represents base skipping errors and insertion errors. $|\cdot|$ takes a string or set as its argument and returns length for string or size for set. We model function Indel by using base skipping error rate δ and insertion error rate ι as:

$$\begin{aligned} \text{Indel}(A, r_i^k) &= \delta^{I(|A|>|r_i^k|)}(1-\delta)^{I(|A|\leq|r_i^k|)} \\ &\times \iota^{I(|A|<|r_i^k|)}(1-\iota)^{I(|A|\geq|r_i^k|)}. \end{aligned}$$

Here, we set δ and ι to 0.001. $P(r_i^k[l]|A[l], b_i^k[l])$ models base substitution error on each base and is given by:

$$P(r_i^k[l]|A[l], b_i^k[l]) = \begin{cases} 1 & r_i^k[l] = A[l] \ \& \ b_i^k[l] = 1 \\ 1/3 & r_i^k[l] \neq A[l] \ \& \ b_i^k[l] = 0 \\ 0 & \text{otherwise} \end{cases}. \quad (3)$$

$P_{mis}(r_i^k)$ represents the probability of read generation for misaligned reads. We consider that reads representing indels, i.e., read with 0 length or more than one nucleotides are generated more probably than reads with one nucleotide in the misalignment, and design $P_{mis}(r_i^k)$ as:

$$P_{mis}(r_i^k) = \begin{cases} 1/N_{mis} & |r_i^k| = 1 \\ p_{mis}/N_{mis} & \text{otherwise} \end{cases}, \quad p_{mis} \geq 1, \quad (4)$$

where N_{mis} is the normalization factor given by $\sum_{A\in\mathcal{A}_k} 1^{I(|A|=1)} p_{mis}^{I(|A|\neq 1)}$. Here, $\mathcal{A}_k$ is a set of possible alleles at position k and is given by {'A', 'T', 'G', 'C'}$\cup$ null string for deletion. In addition, if there exist reads with nucleotides more than one aligned at position k, the corresponding sequences are added $\mathcal{A}_k$. Here, we set p_{mis} to 1.0, i.e., we assume that the read is generated from possible alleles equally probably. $P(b_i^k|bq_i^k)$ is factorized as $\prod_l P(b_i^k[l]|bq_i^k[l])$, and each term is given by a binomial distribution with parameter $1 - 10^{-bq_i^k[l]/10}$:

$$\begin{aligned} P(b_i^k[l]|bq_i^k[l]) &= \left[1 - 10^{-bq_i^k[l]/10}\right]^{I(b_i^k[l]=1)} \\ &\times \left[10^{-bq_i^k[l]/10}\right]^{I(b_i^k[l]=0)}. \end{aligned} \quad (5)$$

$P(m_i^k)$ is given by a binomial distribution with parameter $p_{m_i^k}$. We also give a Beta distribution with parameter $\alpha_m(1 - 10^{-MAPQ_i/10})$ and $\alpha_m 10^{-MAPQ_i/10}$ as a prior distribution of $p_{m_i^k}$. Thus, $p_{m_i^k}$ is updated by considering both probability for alignment reliability of read r_i^k from the model and mapping quality score $MAPQ_i$. We set prior strength α_m to 10.

2.1.2 Haplotype selection

Given a genotype (A_k^1, A_k^2) at position k, haplotype selection part selects an allele, from which each read is generated, by using a binary variable h_i^k in the following manner:

$$\prod_{i\in\mathcal{I}_k} P(r_i^k|A_k^1, bq_i^k)^{I(h_i^k=1)} P(r_i^k|A_k^2, bq_i^k)^{I(h_i^k=2)} P(h_i^k), \quad (6)$$

where $\mathcal{I}_k$ is a set of indexes of reads that span position k. In position-independent variant callers, equally probable condition for haplotype selection is considered, i.e., $P(h_i^k = 1)$ and $P(h_i^k = 2)$ are 0.5. Here, instead of $P(h_i)$, we consider a conditional probability $P(h_i^k|z_k)$ given by:

$$P(h_i^k|z_k) = \begin{cases} p_{h_i} & h_i^k = 1 \ \& \ z_k = 1 \\ 1 - p_{h_i} & h_i^k = 2 \ \& \ z_k = 1 \\ 0.5 & z_k = 0 \end{cases}, \quad (7)$$

where p_{h_i} is a rate for the assignment of read R_i to a haplotype and z_k is a binary variable that determines zygosity at position k and takes value one for heterozygote and zero for homozygote. For paired-end data, p_{h_i} is shared in each read pair.

To represent zygosity with z_k, we introduce a conditional probability $P(z_k|A_k^1, A_k^2)$ that is given by:

$$P(z_k|A_k^1, A_k^2) = \begin{cases} 1.0 & \begin{array}{c} A_k^1 = A_k^2 \ \& \ z_k = 0 \\ \text{or} \\ A_k^1 \neq A_k^2 \ \& \ z_k = 1 \end{array} \\ 0 & \text{otherwise} \end{cases}. \quad (8)$$

The role of z_k is to filter out highly probably homozygous positions from data for read assignment via Equation (7), and hence only highly probably heterozygous positions are used for haplotyping.

p_{h_i} represents one of two chromosomes from which read R_i comes. Since each read comes from one of two homologous chromosomes equally probably in an ideal condition, we represent this property with the following formula:

$$P(\sum_{i \in \mathcal{I}_k} p_{h_i}) = \mathcal{N}(\sum_{i \in \mathcal{I}_k} p_{h_i}; |\mathcal{I}_k|/2, \bar{L}|\mathcal{I}_k|/4), \quad (9)$$

where $\mathcal{N}$ represents normal distribution and $\bar{L}$ is the average read length. $\sum_{i \in \mathcal{I}_k} p_{h_i}$ is considered as the number of reads from the same chromosome at position k. Since $\sum_{i \in \mathcal{I}_k} p_{h_i}$ is a continuous value, normal approximation of a binomial distribution with parameter 0.5 is employed. Although the mean and variance of the normal distribution approximating a binomial distribution with parameter 0.5 are respectively $|\mathcal{I}_k|/2$ and $|\mathcal{I}_k|/4$, we set the variance to $\bar{L}|\mathcal{I}_k|/4$ to normalize the effect with the average read length.

2.2 Parameter estimation

We use the EM algorithm for model parameters $p_{m_i^k}$ and p_{h_i}. Since the proposed model contains loop structures, it needs high time complexity to calculate the exact marginal probabilities required in E-step of the EM algorithm. Thus, we instead calculate approximated marginal probabilities with the loopy belief propagation [9], [17], [20]. In M-step, parameters $p_{m_i^k}$ and p_{h_i} are updated by using the marginal probabilities calculated in E-step.

2.3 Initialization of parameters for the assignment of reads to haplotypes

Since estimation of read assignment parameter p_{h_i} tends to fall into bad local optima from bad initial values, it is crucial to obtain better initial values for accurate haplotype phasing. We infer assignments of reads to haplotypes from possible heterozygous positions, and then use the inferred assignment to the initial values of p_{h_i}. However, due to the errors on sequencing and mapping, the contradiction on haplotyping sometimes occurs between spanning reads, and causes the difficulty in estimating haplotypes. Hence, it is important to give higher priority to more reliable pair of heterozygous positions in the inference of read assignment. We first classify pairs of heterozygous positions into classes and then define reliability score based on each class. Let (A, B) and (a, b) be heterozygous genotypes of two heterozygous positions, and denote the number of spanning reads containing allele x and y in these heterozygous positions as n_{xy}. Without loss of generality, we assume $n_{Aa} + n_{Bb} \geq n_{Ab} + n_{Ba}$ and classify pairs of heterozygous positions by Algorithm 1 given blow:

Algorithm 1: Classify a pair of heterozygous positions

1) Given n_{Aa}, n_{Bb}, n_{Ab}, and n_{Ba} for a pair of heterozygous positions, classfy the pair into six classes according to the following criteria:
   - class 1 $n_{Ab} + n_{Ba} = 0$ and $\min\{n_{Aa}, n_{Bb}\} > 0$ hold.
   - class 2 $n_{Ab} + n_{Ba} = 0$ and $\min\{n_{Aa}, n_{Bb}\} = 0$ hold.
   - class 3 $n_{Ab} + n_{Ba} \leq 0.1 \min\{n_{Aa}, n_{Bb}\}$ and $\min\{n_{Aa}, n_{Bb}\} > 0$ hold.
   - class 4 $\min\{n_{Aa}, n_{Bb}\} > n_{Ab} + n_{Ba} > 0.1 \min\{n_{Aa}, n_{Bb}\}$ holds.
   - class 5 $n_{Aa}+n_{Bb} > n_{Ab}+n_{Ba} \geq \min\{n_{Aa}, n_{Bb}\}$ holds.
   - class 6 $n_{Aa} + n_{Bb} = n_{Ab} + n_{Ba}$ holds.

Spanning reads containing A and b or B and a at two heterozygous positions cause the contradiction on haplotyping between positions. Since there exists no spanning read causing the contradiction in classes 1 and 2, heterozygous position pairs classified into these classes are considered as reliable. Class 3 is similar to class 1, but contains a small number of spanning reads causing the contradiction. Pairs in class 3 are less reliable than those in class 1, but are still considered as reliable. For class 4, 5, or 6, class with higher value is more unreliable.

We then provide reliability score to each pair of heterozygous positions according to its assigned class by Algorithm 2 given below.

Algorithm 2: Provide reliability score to a pair of heterozygous positions

1) Provide reliability score to a pair of heterozygous positions according to its assigned class:
   - class 1 $n_{Aa} + n_{Bb} + 10$
   - class 2 $2(1 - 1/(n_{Aa} + n_{Bb}))$
   - class 3 $\min\{\min\{n_{Aa}, n_{Bb}\} - 10(n_{Ab} + n_{Ba}) + 0.1, 10\}$
   - class 4 $-\min\{n_{Ab} + n_{Ba} - \min\{n_{Aa}, n_{Bb}\} + 5, 10\}$
   - class 5 $-5(1 - 1/(n_{Ab} + n_{Ba} - 0.1 \min\{n_{Aa}, n_{Bb}\} + 1.0))$
   - class 6 $-n_{Ab} - n_{Ba} - 10$

Algorithm 2 is designed to give positive values for heterozygous position pairs classified into class 1, 2, or 3, while negative values are given to unreliable heterozygous position pairs. Pairs in class 4 can take both positive and negative values. We then construct a graph comprised of heterozygous positions. If there exists at least one spanning read between heterozygous positions, an edge with the reliability score is given between the corresponding nodes in the graph. The graph is hereafter called heterozygous position graph. We trace heterozygous positions in the heterozygous position graph by selecting more reliable edges,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2685583, IEEE Transactions on Cloud Computing

and infer the assignment of reads to haplotypes when moving to the next heterozygous position. Parameters for the assignment of reads p_{h_i} are then initialized based on the inferred assignment. In the following, we explain the details of this procedure. We first estimate our model under the setting of $p_{h_i} = 0.5$, and extract positions with marginal probability of $z > 0.95$ as possible heterozygous positions. We then construct heterozygous position graph HG from the possible heterozygous positions. A queue of trees $\mathcal{QT}$ is obtained from HG by Algorithm 3 given below.

Algorithm 3: Get a queue of trees $\mathcal{QT}$ from HG

1) Let $\mathcal{V}$ be a set of nodes comprising heterozygous position graph HG.
2) Set $\mathcal{QT}$ to empty.
3) Let v be a node corresponding to the position with the smallest coordinate among nodes in $\mathcal{V}$.
4) Extract a connected graph CG including v from HG, and remove nodes comprising CG from $\mathcal{V}$.
5) Extract the maximum spanning tree T on CG based on reliability scores.
6) Put T to $\mathcal{QT}$ as the last element.
7) Repeat the above steps until $\mathcal{V}$ becomes empty.

Based on $\mathcal{QT}$, we obtain a queue of nodes $\mathcal{QN}$ which gives an ordering of heterozygous positions for tracing by Algorithm 4 given below.

Algorithm 4: Get an ordering of heterozygous positions

1) Dequeue tree T from $\mathcal{QT}$, and apply T to the following steps:
   a) Extract the maximum size subtree T_s whose nodes are connected by edges with reliability scores $> \beta$ from T.
   b) Let v be a node with the smallest coordinate among nodes comprising tree T_s.
   c) Set a set of nodes $\mathcal{U}$ to empty.
   d) Add v to $\mathcal{U}$.
   e) Let $\mathcal{W}$ be a set of nodes that comprise T and are not in $\mathcal{U}$. Find a node u that is in $\mathcal{W}$ and connected to a node in $\mathcal{U}$ by an edge with the highest reliability score.
   f) Find a node u comprising T that is not in $\mathcal{U}$ and connected to $\mathcal{U}$ by an edge with the highest reliability score.
   g) Put u to queue $\mathcal{QN}$ as the last element. Also, add u to $\mathcal{U}$.
   h) Repeat the above steps until all the node comprising T are in $\mathcal{U}$.
2) Go to step 1) if $\mathcal{QT}$ is not empty.

In our study, β in step a) is set to zero. In step a), the maximum size subtree connected with reliable edges is extracted. We consider that nodes comprising such subtree are highly reliable and give higher priority for the ordering. We trace heterozygous positions according to the ordering in $\mathcal{QN}$, and determine assignment of reads to haplotypes by Algorithm 5.

Algorithm 5: Perform initial haplotype assignment to each read

1) Dequeue node v from $\mathcal{QN}$ and apply v to the following steps:
   a) Let A and B be alleles for heterozygous position corresponding to v.
   b) Let m_{ah} be the number of reads that have allele a at position corresponding to v and are assigned to haplotype h.
   c) Let R_a be a set of reads that have allele a at position corresponding to v and are not assigned to any haplotype yet.
   d) If $m_{A1} > m_{A2}$ and $m_{B1} < m_{B2}$ hold, then assign reads in R_A to haplotype 1 and reads in R_B to haplotype 2, and go to step 2).
   e) If $m_{A1} < m_{A2}$ and $m_{B1} > m_{B2}$ hold, then assign reads in R_A to haplotype 2 and reads in R_B to haplotype 1, and go to step 2).
   f) If $m_{A1} + m_{B2} \geq m_{A2} + m_{B1}$ holds, then assign reads in R_A to haplotype 1 and reads in R_B to haplotype 2, and go to step 2).
   g) Assign reads R_A to haplotype 2 and reads in R_B to haplotype 1, and go to step 2).
2) Go to step 1) if $\mathcal{QN}$ is not empty.

p_{h_i} is set to 0.55 if the corresponding read is assigned to haplotype 1 and 0.45 if it is assigned to haplotype 2.

2.4 Variant calling and haplotype inference

For genotype inference at position k, a configuration of latent variables A_k^1 and A_k^2 that maximizes their marginal probability is searched. We use the loopy belief propagation to calculate the approximated marginal probability of A_k^1 and A_k^2 and then obtain the configuration maximizing the marginal probability. To reduce false positive variants, we calculate a lod score given by the log ratio of the marginal probability of estimated genotype and that of homozygous genotype for reference allele, and filter out the variants with lod scores less than 10.

From variant calling results, we construct a graph structure comprised of sequence termed a sequence graph. In sequence graph, all the sequence reads are represented as vertices, and if two sequence reads are spanning the same heterozygous variant position, they are connected with an edge. Each mate pair is also connected with an edge. Connected components of the read connection graph are detected with breadth-first search, and then heterozygous variants spanned by reads in the same components are considered as the same phased region. To save the memory space, we divide genome sequences into ranges with predetermined length and call variants separately on each range although overlapping is considered for neighboring ranges. The range size and overlapping region size are set to 2,500,000 bp and 1,000 bp, respectively. Since sequence graphs from neighboring ranges can be connected by merging vertices for sequence reads spanning overlapping regions in the ranges, phased regions can span several ranges.

2.5 Filtering out unreliable haplotypes

We define block as a set of positions with heterozygous genotype that are phased reliably with each other. From

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2685583, IEEE Transactions on Cloud Computing

the estimated heterozygous positions by variant calling, we extract blocks and assign integer ID called block ID to the estimated heterozygous positions. The heterozygous positions with the same block ID are in the same block. In the following, we explain more details of the algorithm for extracting blocks. After the inference of genotypes and haplotypes, we apply the inferred heterozygous positions to algorithms in Section 2.3, and obtain a queue of nodes $\mathcal{QN}$ and a queue of trees $\mathcal{QT}$. Let (A, B) and (a, b) be estimated phased genotypes for two heterozygous positions. If $(n_{Aa} + n_{Bb})/(n_{Aa} + n_{Ab} + n_{Ba} + n_{Bb}) \leq 0.5$, we consider that the estimated phase between the positions is unreliable. Based on this definition and queues $\mathcal{QN}$ and $\mathcal{QT}$, blocks are extracted by assigning block IDs to heterozygous positions Algorithm 6 given below

Algorithm 6: Assign block ID to each heterozygous position

1) Set block ID b to 0.
2) Dequeue node v from queue $\mathcal{QN}$.
3) If v has a block ID, go to step 9).
4) Set stack $\mathcal{S}$ to empty, and push v to $\mathcal{S}$.
5) If $\mathcal{S}$ is empty, go to step 8).
6) Pop a node u from $\mathcal{S}$.
7) Let T be a tree that is in $\mathcal{QT}$ and includes u. Get a queue $\mathcal{QA}$ that contains adjacent nodes of u in tree T, and apply $\mathcal{QA}$ to the following steps:
   a) If $\mathcal{QA}$ is empty, go to step 5).
   b) Dequeue node w from $\mathcal{QA}$.
   c) If w has block ID, go to step a).
   d) If the estimated phase between positions corresponding to u and w is unreliable, go to step a).
   e) Assign block ID b to w and push w to $\mathcal{S}$.
   f) Go to step a).
8) Increment block ID b.
9) If $\mathcal{QN}$ is not empty, go to step 2).

We consider that positions in the different blocks are not phased, i.e., the estimated phased between the positions is filtered out in HapMonster2.

2.6 Flow of procedures in HapMonster2

Fig. 1 illustrates the flow of procedures in HapMonster2.

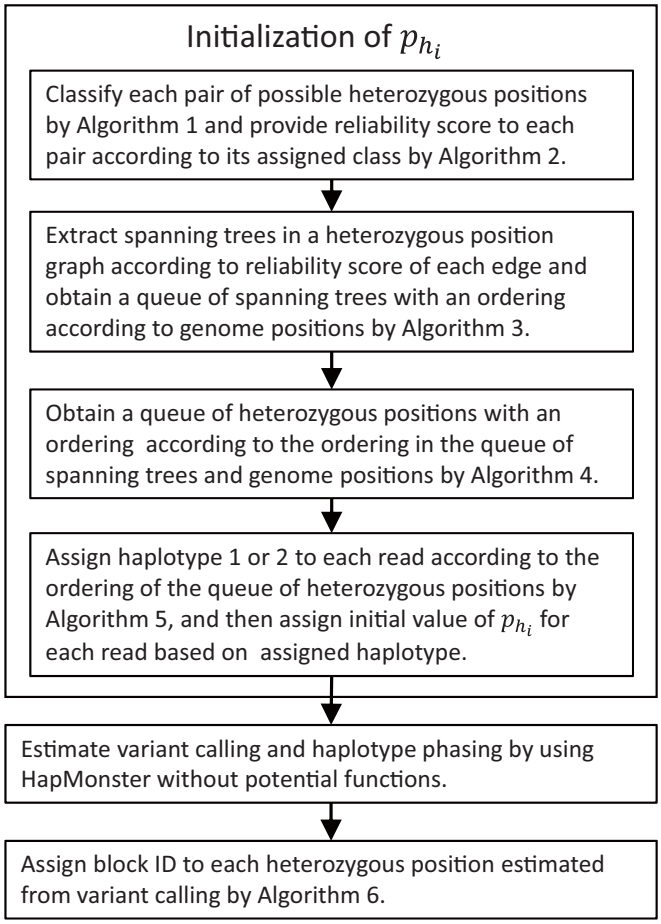

Fig. 1. Flow of procedures in HapMonster2

3 Results

3.1 Simulation analysis

We synthetically generated diploid genome sequences of chromosome 21 for a CEU individual NA12286 according to the variant calling and phasing results released in November 23, 2010 by the 1000 Genomes Project [22], [23]. The number of variants is 50,090. From the diploid genome sequences, we generated paired-end sequence reads and put 0.1% base substitution errors. Base quality scores for bases were set to Q30, which corresponds to 0.1% error. We generated three types of data sets with the following conditions:

- Read length is 100 bp, and insert size is normally distributed with mean 500 bp and standard deviation 50 bp.
- Read length is 300 bp, and insert size is normally distributed with mean 1,500 bp and standard deviation 50 bp.
- Read length is 500 bp, and insert size is normally distributed with mean 2,500 bp and standard deviation 50 bp.

A BAM file for the dataset was obtained by mapping the sequence reads to the reference genome (GRCh37) for chromosome 21 with BWA-MEM [11]. In order to evaluate the performance of HapMonster2 with various read coverages, we downsampled the BAM file to 10×, 20×, and 40× with Picard DownsampleSam (http://picard.sourceforge.net/). Downsampled BAM files were realigned with GATK Indel Realigner.

3.1.1 *Performance on variant calling*

For the comparison with existing methods, we applied HapMonster2, HapMonster, Unified Genotyper implemented in GATK, BCFtools with SAMtools mpileup [13], and Linkage Method [18] to the datasets with their default options. Like HapMonster2 and HapMonster, Linkage Method simultaneously estimates variants and haplotype phasing. Table 1 summarizes the performance of genotype concordance on

HapMonster2 and other three methods. Only if an estimated genotype and true genotype at a position are the same and not homozygous for the reference allele, the estimated genotype is counted as a true positive. In addition, only if the estimated genotype is not homozygous for the reference allele and is different from the true genotype, it is counted as a false positive. Recall and precision are respectively given by $\frac{\text{TP}}{\text{the number of trues}}$ and $\frac{\text{TP}}{\text{TP+FP}}$, where TP is the number of true positives and FP is the number of false positives. F-score is given by the harmonic mean of recall and precision as $\frac{2\times\text{recall}\times\text{precision}}{\text{recall}+\text{precision}}$, and provides the overall performance of recall and precision by capturing a trade-off between them. These are valued between zero and one, and the larger value is better. In the evaluation, only single nucleotide variants were considered.

In all the conditions, HapMonster2 gave the best results in the both recall and F-score. For precision, HapMonster was better than other methods for datasets with read coverage of 10×, and HapMonster2 was slightly worse than HapMonster. BCFtools was the best in precision for the datasets with read coverage of 20× and 40×, and HapMonster2 and HapMonster were the second best.

3.1.2 Performance on haplotype phasing

We evaluate the performance on haplotype phasing based on concordance between estimated haplotypes and haplotyping of true heterozygous variants. Let v_i and s_i be the ith true heterozygous variant and the position for v_i, respectively. We denote an estimated genotype at position s_i by g_{s_i} and also denote a genotype whose allele ordering is reversed from g_{s_i} by $\bar{g}_{s_i}$. B_i is the ID of the estimated phased region spanning position s_i, and D_i is a tertiary variable that takes 1 if $g_{s_i} = v_i$ holds, -1 if $\bar{g}_{s_i} = v_i$ holds, and 0 otherwise. Since genotyping results to be phased are different between methods, we define the following switching cost by using the above notations:

$$\begin{aligned}&\sum_{i=1}^{|\mathcal{V}|}[I(D_i=0)+I(D_i\neq 0)I(B_i\neq B_{p_i})\\ +\quad &2I(D_i\neq 0)I(D_i\neq D_{p_i})I(B_i=B_{p_i})],\end{aligned}$$

where $\mathcal{V}$ is a set of true heterozygous variants and p_i is the index of previously correctly estimated variant, e.g., if genotype of v_{i-3} is correctly estimated and those of v_{i-2} and v_{i-1} are not, p_i is $i-3$. The interpretation of the switching cost is as follows. If the estimated genotype is different from the true variant, the cost is increased by one with the first term. If the estimated genotype is correct but the estimated phased region is not spanned between positions s_i and s_{p_i}, the cost is increased by one with the second term as s_i and s_{p_i} are not phased. Finally, if the estimated haplotype is not correct between s_i and s_{p_i}, the cost is increased by two with the third term. Since a strategy for extending phased region even for uncertain case gets reward by chance for the penalty less than two in the third term, the penalty for the third term is set to two. No penalization is applied only if both the estimated genotype and estimated phase relationship between positions s_i and s_{p_i} are correct.

In addition to Linkage Method, we used Read Backed Phasing (RBP) approach implemented in GATK and HapCompass [1] as existing haplotype phasing methods based on phase-informative reads. Variant calls from Unified Genotyper were used as input variant data of RBP and HapCompass. Switching costs of HapMonster2 and other methods are summarized in Fig. 2. The haplotyping results of HapMonster2 are smaller switching costs than those of other methods for all the conditions. In the results of all the methods, switching costs are smaller for the results from the dataset of the higher read coverage and longer read length data.

TABLE 1
Comparison on genotype concordance of HapMonster2, HapMonster, Unified Genotyper (UG), BCFtools, and Linkage Method (LM) for simulation datasets with read length of 100 bp, 300 bp, and 500 bp. The best result in each condition is in bold.

Read Length	Read Coverage	Method	Recall	Precision	F-score
100bp	10×	HapMonster2	**0.9495**	0.9911	**0.9698**
		HapMonster	0.9489	**0.9912**	0.9696
		UG	0.9103	0.9911	0.9490
		BCFtools	0.9347	**0.9912**	0.9621
		LM	0.8528	0.9160	0.8833
	20×	HapMonster2	**0.9912**	0.9967	**0.9939**
		HapMonster	0.9911	0.9967	**0.9939**
		UG	0.9905	0.9963	0.9934
		BCFtools	0.9850	**0.9987**	0.9918
		LM	0.9046	0.9473	0.9254
	40×	HapMonster2	**0.9953**	0.9963	**0.9958**
		HapMonster	**0.9953**	0.9963	**0.9958**
		UG	0.9947	0.9963	0.9955
		BCFtools	0.9892	**0.9988**	0.9940
		LM	0.8729	0.9259	0.8986
300bp	10×	HapMonster2	**0.9545**	0.9923	**0.9730**
		HapMonster	0.9540	**0.9925**	0.9729
		UG	0.9141	0.9918	0.9514
		BCFtools	0.9428	0.9924	0.9669
		LM	0.8959	0.9416	0.9182
	20×	HapMonster2	**0.9926**	0.9974	**0.9950**
		HapMonster	0.9925	0.9974	0.9949
		UG	0.9916	0.9964	0.9940
		BCFtools	0.9867	**0.9991**	0.9929
		LM	0.9816	0.9906	0.9861
	40×	HapMonster2	**0.9958**	0.9971	**0.9965**
		HapMonster	**0.9958**	0.9971	**0.9965**
		UG	**0.9958**	0.9967	0.9963
		BCFtools	0.9900	**0.9993**	0.9947
		LM	0.9939	0.9960	0.9949
500bp	10×	HapMonster2	**0.9540**	0.9917	**0.9723**
		HapMonster	0.9532	**0.9918**	0.9722
		UG	0.9124	0.9908	0.9500
		BCFtools	0.9427	0.9910	0.9662
		LM	0.8975	0.9453	0.9208
	20×	HapMonster2	**0.9922**	0.9973	**0.9947**
		HapMonster	0.9921	0.9973	**0.9947**
		UG	0.9914	0.9966	0.9940
		BCFtools	0.9867	**0.9992**	0.9929
		LM	0.9786	0.9896	0.9841
	40×	HapMonster2	**0.9959**	0.9971	**0.9965**
		HapMonster	**0.9959**	0.9971	**0.9965**
		UG	0.9958	0.9967	0.9963
		BCFtools	0.9900	**0.9994**	0.9947
		LM	0.9939	0.9962	0.9951

Table 2 summarizes the number of correctly phased neighboring heterozygous position pairs (hereafter referred as correct phases) and the number of switching errors between neighboring heterozygous positions for the results of HapMonster2 and other methods. HapMonster2 provided better performance than other method in terms of the number of correct phases. The switching errors of HapMonster2 were the smallest in the datasets of 100bp and 40× and 500bp and 20×. In other datasets, its switching errors were comparable with the best results. The num-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2685583, IEEE Transactions on Cloud Computing

ber of correct phases of HapMonster was always slightly worse than that of HapMonster2. In addition, its switching errors were much more than those of HapMonster2 We also analyzed the length of correctly estimated haplotypes for each method and performance evaluations on simulation data from genome sequences with much more variants than those for human in the supplemental material.

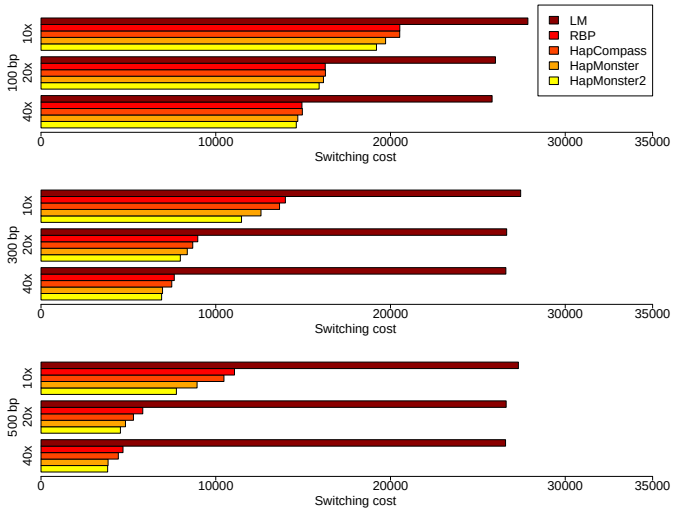

Fig. 2. Switching costs of HapMonster2, HapMonster, HapCompass, Read Backed Phasing (RBP), and Linkage Method (LM) for simulation datasets with read length of 100 bp, 300 bp, and 500 bp. The smaller switching cost is better.

3.2 Real data analysis

For real human sequencing data, we used 100 bp paired-end sequencing data of NA12878, one of samples analyzed in the 1000 Genomes Project. The data was sequenced on Illumina HiSeq 2000 with read coverage of 45× and average insert size of 300 bp. Sequence reads were mapped to the reference genome (GRCh37) with BWA [12], and the mapped dataset was downsampled to 10×, 20×, and 40× with Picard DownsampleSam. These downsampled datasets were realigned with GATK Indel Realigner, and their base quality scores were recalibrated with GATK Base Quality Score Recalibration.

3.2.1 *Performance on variant calling*

We evaluate the performance by assessing the concordance of estimated variants from these variant callers for datasets with various read coverages with SNP array genotyping results from Illumina OMNI 2.5 BeadChip. The number of SNP sites designed in the array for chr21 is 32,076, and 10,579 variants are contained in the sites. We also assessed the concordance with the variant calling result in the 1000 Genomes Project, which contain 52,454 genotyping results including 39,691 variants.

Table 3 and 4 summarize the performance of genotype concordance on HapMonster and three existing methods, Unified Genotyper, BCFtools, and Linkage Method, for NA12878 from the real datasets with read coverages of 10×, 20×, and 40×. Default options were used for these three existing methods. Accuracy is given by $\frac{TP+TN}{TP+FP+TN+FN}$, where TN is the number of true negatives and FN is the number of false negatives. HapMonster2 showed the best results in recall for all the read coverages on the agreement with SNP array genotyping results. For F-score and accuracy, HapMonster2 showed the best results for the read coverage of 10× while HapMonster is slightly better for the read coverages of 20 and 40 ×. For precision, BCFtools showed the best results for all the read coverages. On the agreement with the variant calling results from the 1000 Genomes Project, HapMonster were the best in recall, F-score, and accuracy and slightly better than HapMonster2.. In precision, HapMonster2 and HapMonster provided the best results for the datasets with 10×. For other cases, the results of BCFtools were the best, and those of HapMonster2 or HapMonster were the second best.

TABLE 2
Comparison on the number of correctly phased heterozygous position pairs (correct phases; larger value is better) and the number of switching errors (smaller value is better) in the results of HapMonster2, HapMonster, Unified Genotyper (UG), BCFtools, and Linkage Method (LM) for simulation datasets with read length of 100 bp, 300 bp, and 500 bp.

Read Length	Read Coverage	Method	No. of Correct Phases	No. of Switching errors
100bp	10×	HapMonster2	12,502	30
		HapMonster	12,271	320
		UG	11,139	2
		HapCompass	11,158	16
		LM	4,300	492
	20×	HapMonster2	15,763	10
		HapMonster	15,666	160
		UG	15,413	9
		HapCompass	15,420	26
		LM	697	1,307
	40×	HapMonster2	17,068	7
		HapMonster	17,046	71
		UG	16,747	10
		HapCompass	16,738	24
		LM	7,646	1,790
300bp	10×	HapMonster2	20,253	58
		HapMonster	19,769	689
		UG	17,698	13
		HapCompass	18,105	85
		LM	4,226	5
	20×	HapMonster2	23,708	11
		HapMonster	23,519	220
		UG	22,716	17
		HapCompass	23,084	97
		LM	5,025	3
	40×	HapMonster2	24,779	11
		HapMonster	24,761	46
		UG	24,044	0
		HapCompass	24,257	70
		LM	5,075	7
500bp	10×	HapMonster2	23,976	54
		HapMonster	23,472	734
		UG	20,601	4
		HapCompass	21,351	150
		LM	4,353	1
	20×	HapMonster2	27,127	1
		HapMonster	26,992	156
		UG	25,851	4
		HapCompass	26,524	145
		LM	5,052	1
	40×	HapMonster2	27,860	3
		HapMonster	27,855	23
		UG	26,979	1
		HapCompass	27,387	142
		LM	5,091	5

3.2.2 *Performance on haplotype phasing*

We prepared phased variant calling results as follows: i) Heterozygous sites for NA12878 were phased with SHAPEIT2 using the phased haplotypes for the Phase1

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2685583, IEEE Transactions on Cloud Computing

TABLE 3
Genotype concordance of HapMonster2, HapMonster, Unified Genotyper (UG), BCFtools, and Linkage Method (LM) with OMNI2.5 genotyping results. The best result in each condition is in bold.

Read Coverage	Method	Recall	Precision	F-score	Accuracy
10×	HapMonster2	**0.9112**	0.9891	**0.9486**	**0.9698**
	HapMonster	0.9109	0.9899	0.9487	**0.9698**
	UG	0.8854	0.9875	0.9337	0.9612
	BCFtools	0.8754	**0.9900**	0.9292	0.9583
	LM	0.7944	0.8964	0.8423	0.9285
20×	HapMonster2	**0.9618**	0.9942	0.9778	0.9864
	HapMonster	0.9617	0.9951	**0.9781**	**0.9867**
	UG	0.9579	0.9917	0.9745	0.9846
	BCFtools	0.9466	**0.9963**	0.9708	0.9818
	LM	0.8810	0.9391	0.9091	0.9560
40×	HapMonster2	**0.9663**	0.9956	0.9808	0.9880
	HapMonster	**0.9663**	0.9963	**0.9811**	**0.9882**
	UG	0.9644	0.9904	0.9772	0.9859
	BCFtools	0.9569	**0.9971**	0.9766	0.9852
	LM	0.8553	0.9199	0.8864	0.9461

TABLE 4
Genotype concordance of HapMonster2, HapMonster, Unified Genotyper (UG), BCFtools, and Linkage Method (LM) with variant calls in the 1000 Genomes Project. The best result in each condition is in bold.

Read Coverage	Method	Recall	Precision	F-score	Accuracy
10×	HapMonster2	0.9399	**0.9928**	0.9657	0.9546
	HapMonster	**0.9412**	**0.9928**	**0.9663**	**0.9554**
	UG	0.9184	0.9922	0.9539	0.9381
	BCFtools	0.8972	0.9924	0.9424	0.9222
	LM	0.8229	0.9078	0.8633	0.8658
20×	HapMonster2	0.9898	0.9977	0.9937	0.9921
	HapMonster	**0.9902**	0.9978	**0.9940**	**0.9925**
	UG	0.9885	0.9977	0.9931	0.9912
	BCFtools	0.9692	**0.9981**	0.9834	0.9767
	LM	0.8978	0.9480	0.9222	0.9225
40×	HapMonster2	0.9948	0.9981	0.9965	0.9959
	HapMonster	**0.9951**	0.9981	**0.9966**	**0.9961**
	UG	0.9948	0.9981	0.9964	0.9959
	BCFtools	0.9796	**0.9987**	0.9891	0.9846
	LM	0.8764	0.9373	0.9058	0.9062

integrated variant calls released in September, 17, 2013 as the reference panel. ii) Heterozygous sites not in the reference panel were phased by using variant calls of NA12878's parents, NA12891 and NA12892. Switching costs of HapMonster2, HapMonster, HapCompass, Read Backed Phasing (RBP), and Linkage Method (LM) are summarized in Fig. 3. Variant calls from Unified Genotyper were used for input variant data of HapCompass and RBP. For all the dataset, switching costs of HapMonster2 were smaller than those of other methods. HapMonster2 showed the best performance for the read coverage of 10× while HapMonster was slightly better than HapMonster2 for the higher read coverages. In the results of all the methods, switching costs were smaller for the results from the dataset of the higher read coverage. The number of corrected phases and the number of switching errors are summarized in Table 5. In the number of corrected phases, HapMonster was the best and HapMonster2 was the second best in all conditions although for the results of HapMonster2 contained much less switching errors than those of HapMonster. Results of HapMonster2 contained least switching errors for the datasets of read coverages of 20× and 40×. For the dataset of 10×, the number of switching errors for HapMonster2 was slightly more than that of LinkageMethod although the number of corrected phases for LinkageMethod was much less than that of HapMonster2. We also analyzed the length of correctly estimated haplotypes for each method in the supplemental material.

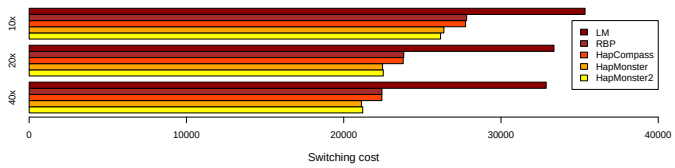

Fig. 3. Switching costs of HapMonster, HapMonster2, HapCompass, Read Backed Phasing (RBP), and Linkage Method (LM) for real sequencing datasets. The smaller switching cost is better.

TABLE 5
Comparison on the number of correct phases (larger value is better) and the number of switching errors (smaller value is better) in the results from HapMonster2, HapMonster, Unified Genotyper (UG), HapCompass, and Linkage Method (LM) for real datasets.

Read Coverage	Method	No. of Correct Phases	No. of Switching errors
10×	HapMonster2	13,090	249
	HapMonster	13,532	905
	UG	11,551	370
	HapCompass	11,725	475
	LM	3,898	245
20×	HapMonster2	16,743	261
	HapMonster	17,393	866
	UG	15,701	523
	HapCompass	15,875	658
	LM	6,283	655
40×	HapMonster2	18,022	234
	HapMonster	18,770	900
	UG	17,159	593
	HapCompass	17,291	718
	LM	7,378	1,254

3.3 Required computational resource

The computational time and memory usage of HapMonster2, HapMonster, Unified Genotyper (UG), BCFtools, Read Backed Phasing (RBP), HapCompass, and Linkage Method (LM) for the simulation datasets with read length of 500 bp and real datasets are summarized in Tables 6 and 7. HapMonster2 and HapMonster are implemented in Java. All the computation was performed on Intel Xeon CPU E5-2670 processors with a single thread. The computational time of HapMonster2 and HapMonster is similar to that of BCFtools, and slightly more than that of Unified Genotyper. Although Read Backed Phasing and HapCompass are very fast on the simulation datasets, they are slower than HapMonster2 and HapMonster on the real datasets. HapMonster2 requires slightly more memory space than other methods other than Linkage Method. However, the required memory space is less than 3GB, and hence it can work currently available laptop PCs.

4 Conclusions

HapMonster2 is a statistically unified model for variant calling and haplotyping by using phase-informative reads.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2685583, IEEE Transactions on Cloud Computing

TABLE 6
Running time and memory usage for HapMonster2, HapMonster, Unified Genotyper (UG), BCFtools, Read Backed Phasing (RBP), HapCompass, and Linkage Method (LM) for simulation datasets with read length of 500 bp.

Read Coverage	Method	Running Time	Memory Usage
10×	HapMonster2	4.3 [min]	2.3 [GB]
	HapMonster	3.1 [min]	2.3 [GB]
	UG	5.0 [min]	0.9 [GB]
	BCFtools	5.9 [min]	0.1 [GB]
	RBP	1.0 [min]	1.0 [GB]
	HapCompass	0.3 [min]	1.7 [GB]
	LM	56.8 [min]	5.6 [GB]
20×	HapMonster2	5.7 [min]	2.7 [GB]
	HapMonster	4.8 [min]	2.8 [GB]
	UG	6.8 [min]	0.9 [GB]
	BCFtools	10.8 [min]	0.1 [GB]
	RBP	2.3 [min]	1.0 [GB]
	HapCompass	0.6 [min]	1.8 [GB]
	LM	129.7 [min]	9.6 [GB]
40×	HapMonster2	14.4 [min]	3.5 [GB]
	HapMonster	13.2 [min]	4.5 [GB]
	UG	10.0 [min]	0.9 [GB]
	BCFtools	20.5 [min]	0.1 [GB]
	RBP	3.9 [min]	1.0 [GB]
	HapCompass	1.0 [min]	2.0 [GB]
	LM	396.6 [min]	27.9 [GB]

TABLE 7
Running time and memory usage for HapMonster2, HapMonster, Unified Genotyper (UG), BCFtools, Read Backed Phasing (RBP), HapCompass, and Linkage Method (LM) for real datasets.

Read Coverage	Method	Running Time	Memory Usage
10×	HapMonster2	8.9 [min]	1.8 [GB]
	HapMonster	8.4 [min]	2.1 [GB]
	UG	5.9 [min]	0.9 [GB]
	BCFtools	7.4 [min]	0.1 [GB]
	RBP	40.7 [min]	1.0 [GB]
	HapCompass	281.3 [min]	1.8 [GB]
	LM	108.3 [min]	6.2 [GB]
20×	HapMonster2	12.3 [min]	2.7 [GB]
	HapMonster	13.2 [min]	4.0 [GB]
	UG	8.1 [min]	0.9 [GB]
	BCFtools	14.4 [min]	0.1 [GB]
	RBP	72.4 [min]	1.0 [GB]
	HapCompass	419.5 [min]	2.6 [GB]
	LM	154.9 [min]	20.8 [GB]
40×	HapMonster2	20.2 [min]	2.2 [GB]
	HapMonster	26.0 [min]	5.8 [GB]
	UG	12.8 [min]	0.9 [GB]
	BCFtools	30.4 [min]	0.1 [GB]
	RBP	130.4 [min]	1.0 [GB]
	HapCompass	529.1 [min]	3.1 [GB]
	LM	633.6 [min]	85.5 [GB]

By considering variant calling and haplotyping simultaneously, phased information improved the accuracy of variant calling, and the accurately estimated variants support the reliable haplotyping synergistically. In addition, an initialization procedure of parameter for read assignment to haplotype and a filtering procedure unreliable estimated haplotyping were introduce for less switching errors in haplotyping results.

By using the simulation sequencing datasets with various read length and real human NGS datasets, we confirmed the effectiveness of our method from the comparison with Unified Genotyper, BCFtools, and Linkage Method for variant calling and with Linkage Method, Read Backed Phasing, and HapCompass for haplotype phasing. We also showed that HapMonster2 was more effective for datasets with longer reads, especially in haplotype phasing. HapMonster2 showed the best results in switching cost, while its results contain much less switching errors than those of HapMonster.

Although HapMonster2 considers only phase-informative reads for haplotype phasing, use of linkage disequilibrium is a promising way for the further improvement on accurate haplotype phasing. We would like to consider such an extension in future work.

Acknowledgments

This work was supported (in part) by MEXT Tohoku Medical Megabank Project. We thank Yukuto Sato and Yumi Yamaguchi-Kabata for their support to our study.

References

[1] Aguiar, D. and Istrail, S.: Haplotype assembly in polyploid genomes and identical by descent shared tracts. Bioinformatics, 29(13), i352–i360 (2013)
[2] Bansal, V., Libiger, O., Torkamani, A., Schork, N.J.: Statistical analysis strategies for association studies involving rare variants. Nature Reviews Genetics, 11, 773–785 (2010)
[3] Browning, R. and Browning, B.L.: Rapid and accurate haplotype phasing and missing data inference for whole genome association studies using localized haplotype clustering. Ametican Journal of Human Genetics, 81, 1084–1097 (2007)
[4] Browning, B.L. and Yu, Z.: Simultaneous genotype calling and haplotype phasing improves genotype accuracy and reduces false-positive associations for genome-wide association studies. Ametican Journal of Human Genetics, 85(6), 847–861 (2009)
[5] Delaneau, O., Marchini, J., Zagury, J.F.: A linear complexity phasing method for thousands of genomes. Nature Methods, 9(2), 179–181 (2011)
[6] DePristo, M.A. *et al.*: A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nature Genetics, 43, 491–498 (2011)
[7] Kojima, K., Nariai, N, Mimori, T., Yamaguchi-Kabata, Y., Sato, Y, Kawai, Y., Nagasaki, M.: HapMonster: a statistically unified approach for variant calling and haplotyping based on phase-informative reads. Lecture Notes in Computer Science, 8542, 107–118 (2014)
[8] Howie, B.N., Donnelly, P., Marchini, J.: A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. PLoS Genetics, 5(6) (2009)
[9] Jordan, M.I.: Graphical models, *Statistical Science*, **19**(1), 140–155 (2004).
[10] Kuhner, M.K.: Coalescent genealogy samplers: windows into population history, Trends in Ecology and Evolution, 24(2), 86–93 (2009)
[11] Li, H.: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997 (2013)
[12] Li, H., Durbin, R.: Fast and accurate short-read alignment with Burrows-Wheeler Transform. Bioinformatics, 25(14), 1754–1760 (2009)
[13] Li, H., Ruan, J., Durbin, R.: Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Research, 18(11), 1851–1858 (2008)
[14] Li, R., Li, Y., Fang, X., Yang, H., Wang, J., Kristiansen, K., Wang, J.: SNP detection for massively parallel whole-genome resequencing. Genome Research, 19(6), 1124–1132, (2009)
[15] Li, Y., Willer, C.J., Ding, J., Scheet, P., Abecasis, G.R.: MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes. Genetic Epidemiology, 34(8), 816–834, (2010)
[16] McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M.J., DePristo, M.A.: The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. Genome Research, 20(9), 1297–1303, (2010)
[17] Murphy, K.P., Weiss, Y., and Jordan, M.I.: Loopy belief propagation for approximate inference: an empirical study. Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, 467–475, 1999.

[18] Sasaki, E., Sugino, R.P., Innan, H.: The linkage method: a novel approach for SNP detection and haplotype reconstruction from a single diploid individual using next generation sequence data. Molecular Biology and Evolution, (9), 2187–2196 (2013)
[19] Scheet, P., Stephens, M.: A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. American Journal of Human Genetics, 78(4), 629–644, (2006)
[20] Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory, 51(7), 2282–2312 (2005)
[21] Yu, Z., Garner, C., Ziogas, A., Anton-Culver, H., and Schaid, D.J.: Genotype determination for polymorphisms in linkage disequilibrium. BMC Bioinformatics, 10:63 (2009)
[22] 1000 Genomes Project Consortium, Abecasis, G.R., Altshuler, D., Auton, A., Brooks, L.D., Durbin, R.M., Gibbs, R.A., Hurles, M.E., McVean, G.A.: A map of human genome variation from population-scale sequencing. Nature, 467(7319), 1061–1073 (2010)
[23] 1000 Genomes Project Consortium, Abecasis, G.R., Auton, A., Brooks, L.D., DePristo, M.A., Durbin, R.M., Handsaker, R.E., Kang, H.M., Marth, G.T., McVean, G.A.: An integrated map of genetic variation from 1,092 human genomes. Nature, 491(7422), 56–65 (2012)
[24] You, N., Murillo, G., Su, X., Zeng, X., Xu, J., Ning, K., Zhang, S., Zhu, J., Cui, X.: SNP calling using genotype model selection on high-throughput sequencing data. Bioinformatics, 28(5), 643–650 (2012)

Kaname Kojima received the bachelor's degree from Tokyo Institute of Technology in 2005, the master's degree from the University of Tokyo in 2008, and the doctor's degree from the University of Tokyo in 2011, respectively. He is currently a senior assistant professor at Tohoku Medical Megabank Organization, Tohoku University. His research interests include next generation sequencing analysis and machine learning methods.

Yosuke Kawai received the bachelor's degree from Tokyo University of Science in 2001, the master's degree from Tokyo University of Science in 2003, and the doctor's degree from Tokyo University of Science in 2006, respectively. He is currently a senior assistant professor at Tohoku Medical Megabank Organization, Tohoku University. His current research focuses on developing a method to infer demography from population genomics data.

Naoki Nariai received the bachelor's degree in information science and the master's degree in computer science from the University of Tokyo in 2003 and 2005, respectively, and the PhD degree in bioinformatics from Boston University in 2010. He is currently a project scientist at Institute for Genomic Medicine, University of California, San Diego. His areas of interest include detection of genomic structural variation, gene expression analysis, and machine learning.

Takahiro Mimori received the bachelor's degree and the master's degree in physics from Kyoto University in 2006 and 2008, respectively. He is currently a research associate at Tohoku Medical Megabank Organization, Tohoku University. His areas of interest include next generation sequencing analysis and machine learning.

Masao Nagasaki received the PhD degree (Science) in information science from the University of Tokyo in 2004. Since 2013, he is the leader of the group of in silico analysis in Tohoku Medical Megabank Organization (ToMMo) in Tohoku University. The current main mission is to create the thousands Japanese reference panel from the cohort project in ToMMo. In Mar/2014, he designed and installed a supercomputer environment around 18,000 cores (=401TF) and 12 Petabyte high performance storage in ToMMo. On this infrastructure, his group is daily processing and analyzing whole-genome sequenced data and omics data from next generation sequencers. He was also contributed to the ICGC liver project in RIKEN and NCC.