WinHAP: An Efficient Haplotype Phasing Algorithm Based on Scalable Sliding Windows

Yun Xu^{1,2*}, Wenhua Cheng^{1,2}, Pengyu Nie^{1,2}, Fengfeng Zhou^{3*}

1 Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China, **2** Anhui Province-MOST Co-Key Laboratory of High Performance Computing and Its Application, University of Science and Technology of China, Hefei, Anhui, China, **3** Shenzhen Institutes of Advanced Technology and Key Laboratory of Health Informatics, Chinese Academy of Sciences, Shenzhen, Guangdong, China

Abstract

Haplotype phasing represents an essential step in studying the association of genomic polymorphisms with complex genetic diseases, and in determining targets for drug designing. In recent years, huge amounts of genotype data are produced from the rapidly evolving high-throughput sequencing technologies, and the data volume challenges the community with more efficient haplotype phasing algorithms, in the senses of both running time and overall accuracy. 2SNP is one of the fastest haplotype phasing algorithms with comparable low error rates with the other algorithms. The most time-consuming step of 2SNP is the construction of a maximum spanning tree (MST) among all the heterozygous SNP pairs. We simplified this step by replacing the MST with the initial haplotypes of adjacent heterozygous SNP pairs. The multi-SNP haplotypes were estimated within a sliding window along the chromosomes. The comparative studies on four different-scale genotype datasets suggest that our algorithm WinHAP outperforms 2SNP and most of the other haplotype phasing algorithms in terms of both running speeds and overall accuracies. To facilitate the WinHAP's application in more practical biological datasets, we released the software for free at: <http://staff.ustc.edu.cn/~xuyun/winhap/index.htm>.

Citation: Xu Y, Cheng W, Nie P, Zhou F (2012) WinHAP: An Efficient Haplotype Phasing Algorithm Based on Scalable Sliding Windows. PLoS ONE 7(8): e43163. doi:10.1371/journal.pone.0043163

Editor: Thomas Mailund, Aarhus University, Denmark

Received: May 17, 2012; **Accepted:** July 17, 2012; **Published:** August 14, 2012

Copyright: © 2012 Xu et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Funding: This work is supported by the National Natural Science Foundation of China (No.60970085). FZ appreciates the startup fund from the Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences (CAS). FZ is also supported in part by the National Natural Science Foundation of China (31000447) and the National Basic Research Program of China (973 Program, 2010CB732606-6). Computing resources were partly provided by the Dawning supercomputing clusters at SIAT CAS. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

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

* E-mail: ff.zhou@siat.ac.cn (FZ); xuyun@ustc.edu.cn (YX)

Introduction

Genomic variations among individuals of a population of the same species can be classified as two families based on their lengths, i.e. single nucleotide polymorphisms (SNPs) and genome structural variations (GSVs) [1,2]. Such genomic variations are supposed to be the driving force for almost all of the human phenotypes, including both non-lethal (e.g. earwax types induced by the genotypes in ABCC11 [3]) and lethal ones (e.g. chronic myelogenous leukemia induced by the Philadelphia chromosome fusion [4]). The association of each SNP with a genetic disease has been extensively studied, and promising results for Huntington's disease [5] and Marfan syndrome [6] have already been widely used for clinical diagnosis. Many genetic diseases were much more complex and were known to be associated with more than one genomic variation, e.g. tumor [7]. However, the required running time of k -SNP association with a given phenotype increases exponentially with the number k of SNPs. The number of frequent SNPs in the human population is estimated to be about 10 million, which makes it impossible to solve the general k -SNP association problem within reasonable time [8].

The diploid human genome has two copies of each chromosome. It is costly and time consuming to experimentally derive the sequence of alleles in contiguous SNP sites along each copy of the diploid chromosomes, which is called a *haplotype* [9]. So the two nucleotides/alleles for one position in the chromosome are usually

derived as an unordered pair, which is called a *genotype*. If these two alleles of a genotype are the same nucleotide, the pair is called *homozygous*, otherwise *heterozygous*. The *haplotype phasing problem* is to determine the haplotypes in a list of genomes/genotypes. The number of candidate haplotypes grows exponentially with the number of heterozygous SNPs in the population. So it's essential to design approximate algorithms for the haplotype phasing problem. The problem is usually simplified to study the phenotype association of adjacent SNPs in a continuous chromosome region [10].

The existing methods for the haplotype phasing problem can be classified into three major categories. The first category of algorithms focuses on finding the exact solution using combinatorial optimization algorithms, e.g. graph realization algorithms [11,12]. The second category applies statistical tests to estimate the haplotype frequencies [13]. And the third one proposes heuristic rules to find the suboptimal solutions instead of the optimal one [14].

A combinatorial algorithm phases the haplotypes satisfying a set of phasing principles, among which the maximum parsimony represents one of the most widely used rules. Although theoretically there may be an exponential number of candidate haplotypes for a given genotype with a given number of heterozygous SNPs, the number of possible haplotypes existing in the real population is always limited. Maximum parsimony principle has been justified by experimental results and shows that

the smallest haplotype set which can resolve all genotypes is close to the real haplotypes [12]. Gusfield proposed an Integral Linear Programming based algorithm to resolve it [12]. Wang and Xu [15] applied the branch and bound algorithm for this problem, which significantly speeds up the problem resolving. However, the basis of most haplotype phasing algorithms adopted the maximum parsimony principle, which is shown to be NP-complete [9] and APX-hard even in very restricted cases [16]. Another major principle is the perfect phylogeny tree principle [11], which claims to reconstruct the resulting haplotypes of individual genotype data from a perfect phylogeny tree. This principle assumes that any SNP mutation happened just once in the human evolutionary history. A number of approximation algorithms have been designed based on the assumption of the perfect phylogeny tree principle [11,17], but the strong assumption holds back its application in some practical cases. The combinatorial algorithm usually works fairly well on a small-scale genotype dataset. When the size of the dataset grows large, its running time or accuracy becomes unacceptable [18].

Statistical tests are also widely used to phase the genotypes. Statistical algorithms estimate the frequencies of all possible haplotypes and choose the most probable haplotype pairs as the final solution. Such algorithms test the haplotypes with the maximum likelihood using various statistical algorithms, such as expectation maximization [13] and Bayesian [19]. Other statistical algorithms employ much more complex statistical models, such as fastPhase [20] and HaploRec [21]. The problem quickly becomes intractable for the larger dataset size of dataset. In order to further reduce the computation time requirement, the partition-ligation (PL) strategy is applied in the algorithms, such as PLEM [22] and GERBIL [23]. The PL strategy focuses on the haplotypes of adjacent regional sites instead of the whole haplotypes. It works well on a larger dataset. Firstly the methods based on the PL strategy partition the candidate haplotypes into uniform blocks. However, many studies characterizing human haplotype structure have shown that the SNPs are grouped into ‘blocks’ of different size for the individuals [24,25]. Zhao YZ and Xu Y [26] presented a more accurate algorithm using a reasonable block partition and ligation strategy according to the haplotype structure. Although most statistical phasing algorithms extend the usefulness with more data processing strategies, the haplotype phasing problem still requires too much computation power, representing the major hurdles for this problem.

A large number of genotypes were generated with the large-scale sequencing technologies. In order to handle populations with more SNPs, heuristic strategies of exact algorithm are usually proposed to speed up the phasing of the genotypes. Tinini [14] designed a new heuristic algorithm (implemented as a program called CollHaps) for haplotype phasing, which based on the maximum parsimony principle and the iterative application of collapse rules. The software enables the user to process large data sets and obtain very “parsimonious” solutions in short time.

In this study, we proposed a new haplotype phasing algorithm based on scalable sliding windows and parsimony principle, which not only maintains the similar speed with the 2SNP but also has a much higher accuracy. In the *first step*, the initial haplotypes of individual genotype dataset are obtained based on simplified 2SNP method. In the *second step*, the haplotypes will be improved by the scalable sliding windows if in which a type of haplotype pair occupies the majority. The scalable sliding window is composed of consecutive SNPs which contain heterozygous SNPs, homozygous SNPs or missing SNPs. In the *final step*, the haplotypes are iteratively decreased by restricting one recombination at most in two haplotypes of each genotype based on parsimony principle.

The algorithm is implemented in a software package called WinHAP. We test WinHAP on four different-scales datasets of genotypes. The result shows that WinHAP not only runs very fast, but also has high accuracy. It outperforms most of the existing algorithms in terms of both speed and accuracy.

Methods

Datasets

Firstly, we compared the performances of WinHAP and the other Haplotype Phasing algorithms on the data of angiotensin converting enzymes (ACEs) [27]. An ACE catalyzes the conversion of angiotensin I to the physiologically active peptide angiotensin II, which controls systemic blood pressure and fluid-electrolyte balance [27]. This dataset consists of the genotypes of 11 unrelated individuals, and each genotype is 52 in length. The genotypes were resolved into 13 distinct haplotypes through experiments. This dataset is denoted as ACE.

We further tested all the algorithms with another dataset on the genetic risk factors of Crohn disease [25]. This dataset consists of 129 pedigrees (father, mother and child), each genotyped at 103 SNPs in the chromosome 5q31 region. The genotypes of another 129 children were selected as the control. The original version of the control dataset has 13,287 SNPs, including 3,873 (29%) heterozygous SNPs and 1334 (10%) missing SNPs. After pedigree resolving, the phase of 2714 heterozygous SNPs and 168 missing SNPs could be determined. These identified SNPs were used for the accuracy evaluation of various Haplotype Phasing algorithms. This dataset is denoted as 5q31.

We chose the Cystic Fibrosis Transmembrane-Conductance Regulator (CFTR) gene dataset [28] to evaluate the algorithms’ performance for the different numbers of genotypes in the same haplotype dataset. This dataset is denoted as CFTR.

We constructed a larger dataset to test the algorithms’ performance from the HapMap database [10]. We simulated genotypes based on 30 children’ data of USA Utah residents in the ENr13 region of chromosome band 4q26. A random sampling from the 30 genotypes produced 1000 genotypes with length 1393, which is denoted as HapMap.

Related Works

Brinza and Zelikovsky proposed a very fast algorithm, 2SNP, for haplotype phasing only SNP pairs [29]. 2SNP starts by calculating the certainty weight of *cis*- or *trans*-phasing on every pair of heterozygous sites i and j , where *cis*-phasing reduces the haplotype 22 to 00 and 11, and *trans*-phasing reduces 22 to 01 and 10. The formula for calculating the weight is as follows.

$$C_{ij} = \log \left(\frac{n + (F_{00}F_{11} - F_{01}F_{10})/(F_{01} + F_{10})}{n - (F_{00}F_{11} - F_{01}F_{10})/(F_{00} + F_{11})} \right) / (i - j)^2 \quad (1)$$

where n is the number of input genotypes, and F_{00} , F_{01} , F_{10} , F_{11} are the frequencies of haplotypes with the first and second binary indexes denoting alleles of the i^{th} and j^{th} SNPs. A complete graph is constructed with each node representing a heterozygous SNP and an edge weight being the certainty of *cis*- or *trans*-phasing on the two heterozygous SNPs corresponding to the two edge nodes. The maximum spanning tree can be determined and uniquely corresponds to the haplotype phasing of each genotype if they are *cis*- or *trans*-phased for all pairs of 2s. The algorithm can be finished within the time complexity $O(nm(n + m))$, where n and m are the numbers of genotypes and SNPs, respectively. It is fast enough for analyzing the high-throughput genotyping data, but its accuracy still remains to be improved.

The block structure of SNPs observed in the practical genotyping datasets is widely used to speed up the haplotype phasing algorithms and improve the phasing accuracy. The detailed patterns of SNP blocks were discovered by various linkage disequilibrium studies [30,31,32,33]. In particular, Halperin and Eskin observed that the distribution of haplotypes in a block is uneven, and four main patterns of haplotypes constitute 97% of the haplotypes in each block [33]. The uneven distribution of haplotypes in a block is illustrated in Table 1 [33]. These frequent patterns are called the *common haplotypes*. The data suggests that a genotype matching a common haplotype has a much higher probability to be in the haplotype phasing solution. So it is reasonable to prioritize the common haplotypes in the haplotype phasing process.

Most of the haplotype phasing algorithms follow the maximum parsimony principle, but its complexity of NP-completeness significantly slows down the algorithms' running. An alternative way of finding a sub-optimal solution was adopted by all these algorithms, e.g. Hapinferx [34] and parsimonious tree-grow method [35].

WinHAP Algorithm

In this section, firstly we will introduce the data structures of a haplotype and a genotype, and the haplotype phasing problem model. Then we describe in details the WinHAP algorithm, consisting of three major steps. In step 1, initial haplotypes are constructed by a simplified 2SNP-like algorithm. In step 2, the scalable sliding windows are used to correct the haplotype blocks which break apart the common haplotype pairs. In last step, the phasing solution is further refined by using a novel parsimonious strategy. We will analyze the computational complexity of the proposed algorithm at the end of this section.

The input to the haplotype phasing problem consists of n genotype vectors, each with m coordinates corresponding to m SNPs. Each SNP site is assumed to have at most two different alleles. The allele of a SNP site can be denoted by '0' or '1', where '0' represents the major allele and '1' the minor allele. A haplotype can be represented as a string of the alphabet $\{0, 1\}$. A genotype can be denoted as a string of the alphabet $\{0, 1, 2, ?\}$, where '0' and '1' represent the homozygous SNP $\{0, 0\}$ and $\{1, 1\}$, respectively, '2' is a heterozygous SNP $\{0, 1\}$, and '?' is a missing SNP. The haplotype phasing problem can be defined formally as follows.

Definition 1 (Haplotype Phasing Problem): Given a genotype $g_i = (g_i^1, \dots, g_i^m) \in \{0, 1, 2\}^m$, a pair of haplotypes $\langle h_i, \bar{h}_i \rangle$ is the solution of g_i when $h_i = (h_i^1, \dots, h_i^m)$ and $\bar{h}_i = (\bar{h}_i^1, \dots, \bar{h}_i^m)$ satisfy the

condition: for any $j \in \{1, \dots, m\}$, $h_i^j = \bar{h}_i^j = g_i^j$ if $g_i^j = 0$ or 1, and $h_i^j \neq \bar{h}_i^j$ if $g_i^j = 2$.

When the above conditions hold, we say that g_i can be phased to $\langle h_i, \bar{h}_i \rangle$, or g_i can be resolved into $\langle h_i, \bar{h}_i \rangle$, which is also denoted by $g_i = h_i \oplus \bar{h}_i$. We will describe in details the three phasing steps to resolve the phasing problem on a given input set $\{g_1, \dots, g_n\}$.

Step 1: initial phasing based on simplified 2SNP algorithm. We first simplified the 2SNP algorithm to investigate a genotype with only two heterozygous SNPs, which can be resolved into the haplotypes 00/11 or 01/10. We used *cis*-phasing and *trans*-phasing to represent a pair of heterozygous SNPs resolved into 00/11 and 01/10, respectively, as similar to the 2SNP algorithm [29].

We use the pairs of adjacent heterozygous SNPs to give the initial haplotypes, and then construct a linear tree, instead of the maximum spanning tree in the original version of 2SNP algorithm, to infer a solution to the haplotype phasing problem. For a $n \times m$ matrix of input genotypes, each row is an m -dimensional genotype g_i and its solution is haplotype h_i and $\bar{h}_i$. The k^{th} heterozygous SNP of the i^{th} genotype g_i is denoted by $g_i^{t_k}$. For each genotype, the k^{th} heterozygous SNP $g_i^{t_k}$ is paired with its next heterozygous SNP $g_i^{t_{k+1}}$. The phasing pattern (i.e. *cis*-phasing or *trans*-phasing) of each heterozygous SNP pair is determined according to the major haplotypes among the haplotypes 00/11 or 01/10. The phasing pattern of a heterozygous SNP pair is determined by the following formula (2). The first heterozygous SNP of each genotype is resolved into '0' and '1' (i.e. $h_i^{t_1} = '0'$, and $\bar{h}_i^{t_1} = '1'$). If the phasing pattern of a heterozygous SNP pair is determined, the other SNP site of this pair can be resolved uniquely. By repeating the procedures of pairing and phasing, the haplotypes of all the SNP sites can be resolved. Figure 1 shows this initial phasing process.

$$C_{ij} = \begin{cases} \frac{F_{00} \times F_{11}}{F_{01} \times F_{10}} & F_{00} \times F_{11} > 0, \text{ and } F_{01} \times F_{10} > 0 \\ \frac{(F_{00} + 1) \times (F_{11} + 1)}{(F_{01} + 1) \times (F_{10} + 1)} & \text{otherwise} \end{cases} \quad (2)$$

where F_{00} , F_{11} , F_{01} , F_{10} are the frequencies of haplotypes with the first and the second binary index, denoting alleles of the i^{th} and j^{th} SNP sites, respectively. Haplotype frequencies are determined based on all genotype frequencies except 22 and the genotypes that contains '?'.

Step 2: improvement of haplotypes using scalable sliding windows. Step 2 expands the haplotypes of adjacent SNP pairs into the results within scalable sliding windows. A *window* is a genotype segment with the same start and end SNP sites for all individuals within a population. These SNP sites could be heterozygous, homozygous or missing ones. In the process of moving window, each time the window slides to the right one

Figure 1. The initial phasing process in the first step.

doi:10.1371/journal.pone.0043163.g001

Table 1. The uneven distribution of haplotypes in a block.

Haplotype	0/1 representation	Frequency
CCGAT	00000	66
CTGAC	01001	24
ATACT	11110	10
CTGAT	01000	6
ATGAT	11000	1
ATGCC	11011	1
CCGAC	00001	1

doi:10.1371/journal.pone.0043163.t001

position from the first SNP site on the genotypes. When the window moves to some SNP site, the window starts to the right extension that the length of the window changes from l_{min} to l_{max} , where l_{min} and l_{max} are the minimum and maximum values respectively. We called these removable and scalable windows as the *scalable sliding windows*.

Four main common patterns of haplotypes can explain over 97% cases within a block of SNP sites in a practical dataset [33]. Our algorithm tries to replace the phasing with common phasing patterns, within the scalable sliding windows detected in the above procedure. A genotype segment in a window is denoted by g^w . The definition of the compatible haplotype pairs of g^w are given below.

Definition 2 (Compatible Haplotype Pair): In a window, given a genotype segment g^w and two haplotype segments h^w and $\bar{h}^w$. h^w and $\bar{h}^w$ are called the compatible haplotype pair of g^w if g^w can be denoted by $h^w \oplus \bar{h}^w$.

The scalable sliding window starts from the first column of genotype matrix, and its length is set to l_{min} . The algorithm firstly calculates the frequencies of all possible haplotypes in the genotype window. Then all compatible haplotype pairs of each genotype are detected. For some genotype, if we find that a compatible haplotype pair occupies the majority of all compatible haplotypes pairs, the genotype is denoted as a block, and the resulting haplotypes of this genotype will be revised with the common haplotype patterns. The measurement of the *majority* is determined by the following formula.

$$W_i = \frac{(F_i + 1) \times (F'_i + 1)}{\sum_{j=1}^s (F_j + 1) \times (F'_j + 1)} \quad (3)$$

where F_i and F'_i are the haplotype frequencies of the i^{th} compatible haplotype pair of g^w , and s is the number of all compatible haplotype pairs. When the maximum weight of all possible haplotype pairs is larger than threshold τ , the initial resulting haplotypes are replaced by the compatible haplotype pair with the maximum weight, otherwise keep the original haplotype pair unchanged. Each genotype segment in a window is processed in the same way. The windows with different lengths slide on the genotypes until the complete genotypes are corrected.

Let's explain the above procedures with an example haplotype block from real data, as shown in Table 1. For a genotype segment 02002, two compatible haplotype pairs (00000 and 01001, 01000 and 00001) are detected. The weight of first haplotype pair (00000 and 01001) is 99%, whereas the other is 1%. If this genotype segment is resolved into 01000 and 00001 in the first step, our algorithm will replace them with 00000 and 01001. As observed in the real haplotype block data, the number of most common haplotype (e.g. 00000, 01001, 11110 or 01000) is more than 3 times larger than the number of another uncommon haplotype (e.g. 11000, 11011 or 00001). For this reason, we set the threshold $\tau = 70\%$ by default. The compatible haplotype pair with the maximum weight is called the *common haplotype pair*. Because the length of most blocks does not exceed 10 [26], the size of scalable sliding window is set in the range from 3 to 10 (i.e. $l_{mix} = 3$, $l_{max} = 10$).

One more thing, there is a case to be considered when the distance between neighbor heterozygous SNPs is larger than l_{max} , because any window of the scalable sliding windows cannot cover this heterozygous SNP pair. For this case, we use the similarity to improve the results. A classical method for the similarity of two strings is Hamming distance. We assume these two heterozygous sites are p_s and p_e , and the genotype needing to be processed is the i^{th} genotype. The n input genotypes are resolved into $2n$ haplotypes

in previous processing, which are numbered from 1 to $2n$. We use $g_i^{p_s-p_e}$ to denote the genotype segment (a sequence of homozygous SNPs) between p_s and p_e in the i^{th} genotype, and use $h_j^{p_s-p_e}$ to denote the haplotype segment between p_s and p_e in the j^{th} haplotype. $H_{ij}^{p_s-p_e}$ represents the Hamming distance between $g_i^{p_s-p_e}$ and $h_j^{p_s-p_e}$, which is calculated by the following formulas.

$$H_{ij}^{p_s-p_e} = \sum_{k=p_s+1}^{p_e-1} \phi_{ij}(k) \quad (4)$$

where

$$\phi_{ij}(k) = \begin{cases} 1 & \text{if } g_i^k \neq h_j^k \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The weight functions of *cis*-phasing and *trans*-phasing for the heterozygous SNPs pair are as follows.

$$W_c = \sum_{h_j^{p_s}=0 \text{ and } h_j^{p_e}=0 \text{ and all } j} H_{ij}^{p_s-p_e} \times \sum_{h_j^{p_s}=1 \text{ and } h_j^{p_e}=1 \text{ and all } j} H_{ij}^{p_s-p_e} \quad (6)$$

$$W_t = \sum_{h_j^{p_s}=0 \text{ and } h_j^{p_e}=1 \text{ and all } j} H_{ij}^{p_s-p_e} \times \sum_{h_j^{p_s}=1 \text{ and } h_j^{p_e}=0 \text{ and all } j} H_{ij}^{p_s-p_e} \quad (7)$$

As the *cis*-phasing and *trans*-phasing of heterozygous SNP pair 22 are resolved into 00/11 and 01/10, respectively, the product of two sums is treated as the weight between them. Smaller Hamming distance represents greater similarity between two haplotypes. The phasing pattern (i.e. *cis*-phasing or *trans*-phasing) for this heterozygous SNP pair is updated by the one with smaller weight.

Step 3: refining haplotypes based on the maximum parsimony principle. For the genotypes containing missing data, the resulting haplotypes also have missing alleles after the first two steps. The 2SNP algorithm recovers them with the corresponding values from the haplotype with the closest Hamming distance [29]. We define a window of length 9 for each missing allele, with the missing allele in the 5^{th} position. The missing data of a haplotype in the window is resolved by the similarity weights of the all other haplotypes with this haplotype. The computation of similarity weight between the haplotypes is the same as Step 2, where the similarity weights denotes the missing alleles being '0' and '1'. Figure 2 gives an example of recovering missing data in haplotype h_1 .

The range between the dotted lines is the window to be investigated, as shown in Figure 2. Let W_0 denote the weight of missing allele being '0' in h_1 , and W_1 denote the weight of missing allele being '1'. W_0 is the sum of similarity weights of the other haplotypes where the corresponding missing position is '0' (e.g. haplotypes h_2 , h_3 and h_3), and W_1 is the sum of similarity weights of the other haplotypes where the missing position is '1' (i.e. haplotype $\bar{h}_2$). A similarity weight between h_1 and h_2 is calculated by formula (4), and added to W_0 . The missing alleles are finally determined by the allele ('0' or '1') which has the smaller weight.

After the above processing, we can further optimize the haplotype solution set based on the maximum parsimony principle, minimizing the number different haplotypes. We used

Figure 2. The recovery process of a missing allele on haplotype h_1 .

doi:10.1371/journal.pone.0043163.g002

at most one recombination in two haplotypes of each genotype to make the current haplotype set smaller. The recombination process is illustrated in Figure 3. For example, haplotypes “11010010” and “01100101” are switched at the 4th SNP site, so the resulting haplotypes are changed into “11000101” and “01110010”. The switched site must be heterozygous, and the recombination operation tries to decrease the number of different haplotypes. This investigation was iterated until the haplotype set size cannot be reduced.

Time Complexity of WinHAP

Now, let us analyze the time complexity of our algorithm. In the first step, our algorithm uses the pairs of adjacent heterozygous SNPs to obtain the initial haplotypes, which takes $O(nm)$ time since we have n genotypes each with m SNPs. In the second step, this algorithm improves the initial haplotypes using scalable sliding windows with length in the range from 3 to 10, which takes $O(n^2m)$ in time since every window should find all compatible haplotype pairs in n individuals. In the last step, our algorithm consists of two procedures. The recovery of missing data takes $O(n^2m)$ times since the process is similar to Step 2. The minimization of different haplotype number needs to investigate the recombination of each site in the worst case, which takes $O(nm \log n)$ in time since every haplotype can be queried in the ordered haplotype list. Therefore our algorithm takes $O(nm + n^2m + nm \log n) \tilde{O}(n^2m)$ in total time, and it is on the same magnitude with the 2SNP algorithm.

Measurement Criteria of Phasing Accuracy

As similar to most of the haplotype other phasing algorithms, we used the *individual error rate* (IER) [19] and the *switch error rate* (SER) [23] to evaluate the performance of phasing algorithm [36,37]. The IER is defined as the percentage of individuals whose genotypes are incorrectly resolved. Generally, the IER value of phasing algorithms decreases with the increasing number of individuals, and increases with the increasing genotype length.

Figure 3. A recombination process with the 4th SNP as the switched site.

doi:10.1371/journal.pone.0043163.g003

The SER is defined as the ratio between the numbers of switch errors and all the heterozygous loci.

We compared our algorithm with six existing programs including ISHAPE [36], fastPhase [20], GERBIL [23], BBPLEM [26], Beagle [38] and 2SNP [29]. The program PHASE was not investigated because it didn't produce the results within reasonable time. The program HaploRec [21] and CollHaps [14] was not tested because their current versions cannot handle the missing SNPs. All programs were run on the Linux operating system with a quad-core 3.1GHz CPU and 4GB memory.

Results and Discussion

Validation on Dataset ACE

We tested the performance of the algorithms on the dataset ACE, averaged over 100 independent runs. ISHAPE, GERBIL, and 2SNP were all run with the default settings. The parameter K (number of clusters) of fastPhase was set to 10 to reduce its running time. The buffer size of BBPLEM was set to 50 and the round of EM iteration was set to 20 which was the same with [26]. The parameter “*nsample*” of Beagle was set to 200 and we randomly generated the parameter “*seed*” in every independent running.

The performance of various phasing algorithms on ACE was shown in Table 2. WinHAP outperforms all the other algorithms in the error rates, with 0 in both IER and SER. Both 2SNP and GERBIL showed a slightly higher IER (0.091) and SER (0.005), which meant only one genotype was incorrectly resolved. All the other algorithms, such as ISHAPE, fastPhase, BBPLEM and Beagle, performed even worse accuracies on the ACE dataset. WinHAP also runs faster than all the other algorithms except 2SNP, with a slight increase in running time compared with 2SNP.

Validation on Dataset 5q31

Because there were missing data in 5q31, the IER and SER were computed in the case of including and not including missing data, respectively. The IER1 is defined to exclude the missing data, and IER2 is defined to include the missing data. SER1 and SER2 are defined similarly. For this large dataset with 129 genotypes and 103 SNPs, the buffer size of BBPLEM was set to 100 and the iteration round of EM was set to 20, the same as [26]. The parameter “*nsample*” of BEAGLE was set to 25 to achieve phasing in short enough time. All the other parameters were set to the default values.

Table 2. Mean IER, SER and runtime of various phasing algorithms on the ACE dataset.

Software	IER	SER	Time (s)
ISHAPE	0.092	0.011	4.237
fastPhase	0.232	0.020	3.031
GERBIL	0.091*	0.005*	1.523
BBPLEM	0.178	0.020	0.039
Beagle	0.182	0.030	0.368
2SNP	0.091*	0.005*	0.010
WinHAP	0.000	0.000	0.019*

The results corresponding to the highest performance in each column are in bold. The results corresponding to the second-best performance in each column are attached an asterisk.

doi:10.1371/journal.pone.0043163.t002

Table 3. Mean IER, SER and runtime of various phasing algorithms on the 5q31 dataset with missing data.

Software	IER1	IER2	SER1	SER2	Time (s)
ISHAPE	0.331*	0.391	0.030	0.047	489.0
fastPhase	0.337	0.385	0.026*	0.041*	74.7
GERBIL	0.380	0.434	0.030	0.045	21.3
BBPLEM	0.341	0.388*	0.030	0.043	1.1
Beagle	0.343	0.400	0.028	0.043	1.8
2SNP	0.395	0.465	0.031	0.046	0.4
WinHAP	0.326	0.388*	0.025	0.039	0.5*

The results corresponding to the highest performance in each column are in bold. The results corresponding to the second-best performance in each column are attached an asterisk.

doi:10.1371/journal.pone.0043163.t003

The seven algorithms were executed for 100 independent runs on the dataset 5q31, and the averaged performances are listed in Table 3. In most cases, WinHAP clearly outperforms the other algorithms in terms of both speed and accuracy. WinHAP achieved 0.326 in IER1, which is 1.5% lower than the second lowest value of ISHAPE. However, the running time of ISHAPE was more than 1000 times than WinHAP. As to IER2, fastPhase produced a minor improvement (0.77%) compared with WinHAP, but it runs for 149 times than WinHAP in time. As to SER1 and SER2, WinHAP achieved more than 3.8% improvement compared with the second best algorithm fastPhase, and its running time was more than 100 times less. Table 3 shows that WinHAP can produce comparable accuracy for missing data, within much shorter time.

Validation on Dataset CFTR

As similar in [14,19,39], we constructed a dataset of 57 haplotypes with no missing data from 94 experimentally identified disease haplotypes. In [19,39], the haplotypes were randomly selected to form data sets of size 28. In our experiment, we used different size (28, 30, 35, 40, 45, 50) of haplotypes pairs set to form our genotype datasets, and for each size, 100 distinct data sets were generated. The buffer size of BBPLEM was set to 100 and the iteration round of EM was set to 20. The parameter “*nsample*” of BEAGLE was set to 25. All the other parameters were set to the default values.

Table 4 gives the accuracies and running times of the seven algorithms. For the dataset CFTR, ISHAPE produced the minimum IER and SER, but it runs for very long time on these datasets. The fastPhase also outperforms WinHAP in terms of IER and SER, but its running time is similar to ISHAPE. The BBPLEM gave the excellent accuracy with much less time on this dataset. When N is 40, 45 and 50, the IER and SER of BBPLEM are second lowest. Though the algorithm WinHAP proposed in this paper had higher IER and SER for different sizes, the running speed is almost the fastest. It's highly significant that WinHAP produced an average 28.5% and 28% improvement in IER and SER for different sizes compared with 2SNP, respectively. The other algorithms, such as GERBIL and Beagle, gave the higher IER and SER compared with WinHAP, and they run for 2 times than WinHAP in time.

Validation on Dataset HapMap

For the largest dataset HapMap, the buffer size of BBPLEM was set to 100 and the iteration round of EM was set to 20. The parameter “*nsample*” of BEAGLE was set to 25. All the other parameters were set to the default values.

For the large dataset HapMap, ISHAPE, fastPhase and GERBIL failed to give a solution in 2 hours. Beagle can handle the genotype set in 55.5 seconds, but its IER and SER were the

Table 4. Mean IER, SER and runtime of various phasing algorithms on the CFTR dataset.

Software	N = 28			N = 30			N = 35		
	IER	SER	Time (s)	IER	SER	Time (s)	IER	SER	Time (s)
ISHAPE	0.221	0.044	3.90	0.214	0.045	4.14	0.191*	0.038*	4.73
fastPhase	0.252*	0.049*	3.73	0.246	0.049*	4.03	0.231	0.045	4.70
GERBIL	0.387	0.085	0.36	0.367	0.085	0.39	0.349	0.078	0.55
BBPLEM	0.261	0.054	0.02*	0.243*	0.051	0.02*	0.182	0.037	0.02*
Beagle	0.381	0.074	0.38	0.378	0.075	0.40	0.333	0.064	0.43
2SNP	0.423	0.089	0.01	0.414	0.087	0.01	0.417	0.083	0.01
WinHAP	0.316	0.065	0.01	0.312	0.066	0.01	0.301	0.061	0.01
Software	N = 40			N = 45			N = 50		
	IER	SER	Time (s)	IER	SER	Time (s)	IER	SER	Time (s)
ISHAPE	0.162	0.031	5.67	0.152	0.029	6.34	0.146	0.027	7.31
fastPhase	0.214	0.040	5.38	0.214	0.039	6.00	0.208	0.039	6.68
GERBIL	0.324	0.070	0.70	0.334	0.072	0.86	0.325	0.069	0.95
BBPLEM	0.182*	0.037*	0.02*	0.176*	0.036*	0.02*	0.178*	0.036*	0.02*
Beagle	0.304	0.055	0.05	0.276	0.050	0.48	0.272	0.048	0.50
2SNP	0.396	0.079	0.01	0.388	0.077	0.01	0.405	0.080	0.01
WinHAP	0.285	0.056	0.01	0.260	0.053	0.02*	0.275	0.056	0.02*

The results corresponding to the highest performance in each column are in bold. The results corresponding to the second-best performance in each column are attached an asterisk.

doi:10.1371/journal.pone.0043163.t004

Table 5. Mean IER, SER and runtime of various phasing algorithms on the HapMap dataset with no missing data.

Software	IER	SER	Time (s)
ISHAPE	–	–	No solution
fastPhase	–	–	No solution
GERBIL	–	–	No solution
BBPLEM	0.992*	0.027	126.1
Beagle	0.999	0.128	55.5
2SNP	0.999	0.024*	38.6
WinHAP	0.965	0.015	41.1*

The results corresponding to the highest performance in each column are in bold. The results corresponding to the second-best performance in each column are attached an asterisk.

doi:10.1371/journal.pone.0043163.t005

highest among all phasing algorithms. The algorithm BBPLEM had relatively low SER, but the running time is two times longer than Beagle. WinHAP and 2SNP outperform all the other algorithms in terms of both speed and accuracy. In particular, 37% improvement in SER was produced by WinHAP compared with 2SNP. WinHAP runs for minor longer time with significant improvement in accuracy, compared with 2SNP, as shown in Table 5. This showed that WinHAP could also deal with high-

throughput genotype data with the significantly improved accuracy, compared with the fastest algorithm 2SNP.

Conclusions

Huge amount of genotype data was being generated with the emerging large-scale sequencing technologies, at the increasing speeds. The development of a fast and more accurate haplotype phasing algorithm is necessary to meet the needs of handling massive genotype datasets. In this study, we proposed a fast haplotype phasing algorithms based on scalable sliding windows. The algorithm has better or comparable performance in both speed and accuracy in almost all cases, compared with the six other algorithms. We are working on both further improving the algorithm and applying it to study the haplotype phasing problem on our leukemia project.

Acknowledgments

We appreciate Dr. Tininini for the generous help in the performance validations, Haiyang Tu and CunLei Yan for the software website, and Jiaoyun Yang and Weihua Pan for many helpful suggestions contributing to this article.

Author Contributions

Conceived and designed the experiments: WC YX. Performed the experiments: WC. Analyzed the data: WC. Contributed reagents/materials/analysis tools: WC PN. Wrote the paper: WC YX FZ.

References

- Alkan C, Coe BP, Eichler EE (2011) Genome structural variation discovery and genotyping. *Nat Rev Genet* 12: 363–376.
- Li Y, Zheng H, Luo R, Wu H, Zhu H, et al. (2011) Structural variation in two human genomes mapped at single-nucleotide resolution by whole genome de novo assembly. *Nat Biotechnol* 29: 723–730.
- Nakano M, Miwa N, Hirano A, Yoshiura K, Niikawa N (2009) A strong association of axillary osmidrosis with the wet earwax type determined by genotyping of the ABCC11 gene. *BMC Genet* 10: 42.
- Talpaz M, Shah NP, Kantarjian H, Donato N, Nicoll J, et al. (2006) Dasatinib in imatinib-resistant Philadelphia chromosome-positive leukemias. *N Engl J Med* 354: 2531–2541.
- Walker FO (2007) Huntington's disease. *Lancet* 369: 218–228.
- Keane MG, Pyeritz RE (2008) Medical management of Marfan syndrome. *Circulation* 117: 2802–2813.
- Stadler ZK, Gallagher DJ, Thom P, Offit K (2010) Genome-wide association studies of cancer: principles and potential utility. *Oncology (Williston Park)* 24: 629–637.
- Kruglyak L, Nickerson DA (2001) Variation is the spice of life. *Nat Genet* 27: 234–236.
- Bonizzoni P, Della Vedova G, Dondi R, Li J (2003) The Haplotyping problem: An overview of computational models and solutions. *Journal of Computer Science and Technology* 18: 675–688.
- The International HapMap Consortium (2003) The International HapMap Project. *Nature* 426: 789–796.
- Gusfield D (2002) Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. *Proceedings of the sixth annual international conference on Computational biology*. Washington, DC, USA: ACM. 166–175.
- Gusfield D (2003) Haplotype inference by pure Parsimony. *Proceedings of the 14th annual conference on Combinatorial pattern matching*. Morelia, Michoacán, Mexico: Springer-Verlag. 144–155.
- Excoffier L, Slatkin M (1995) Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Mol Biol Evol* 12: 921–927.
- Tininini L, Bertolazzi P, Godi A, Lancia G (2010) CollHaps: a heuristic approach to haplotype inference by parsimony. *IEEE/ACM Trans Comput Biol Bioinform* 7: 511–523.
- Wang L, Xu Y (2003) Haplotype inference by maximum parsimony. *Bioinformatics* 19: 1773–1780.
- Lancia G, Pinotti MC, Rizzi R (2004) Haplotyping Populations by Pure Parsimony: Complexity of Exact and Approximation Algorithms. *INFORMS Journal on Computing* 16: 348–359.
- Chung RH, Gusfield D (2003) Perfect phylogeny haplotyper: haplotype inference using a tree model. *Bioinformatics* 19: 780–781.
- Jajamovich GH (2012) Maximum-Parsimony Haplotype Inference Based on Sparse Representations of Genotypes. *IEEE Transactions on Signal Processing* 60.
- Niu T, Qin ZS, Xu X, Liu JS (2002) Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *Am J Hum Genet* 70: 157–169.
- Scheet P, Stephens M (2006) A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *Am J Hum Genet* 78: 629–644.
- Eronen L, Geerts F, Toivonen H (2006) HaploRec: efficient and accurate large-scale reconstruction of haplotypes. *BMC Bioinformatics* 7: 542.
- Qin ZS, Niu T, Liu JS (2002) Partition-ligation-expectation-maximization algorithm for haplotype inference with single-nucleotide polymorphisms. *Am J Hum Genet* 71: 1242–1247.
- Kimmel G, Shamir R (2005) GERBIL: Genotype resolution and block identification using likelihood. *Proc Natl Acad Sci U S A* 102: 158–162.
- Gabriel SB, Schaffner SF, Nguyen H, Moore JM, Roy J, et al. (2002) The structure of haplotype blocks in the human genome. *Science* 296: 2225–2229.
- Daly MJ, Rioux JD, Schaffner SF, Hudson TJ, Lander ES (2001) High-resolution haplotype structure in the human genome. *Nat Genet* 29: 229–232.
- Zhao Y, Xu Y, Wang Z, Zhang H, Chen G (2008) A better block partition and ligation strategy for individual haplotyping. *Bioinformatics* 24: 2720–2725.
- Rieder MJ, Taylor SL, Clark AG, Nickerson DA (1999) Sequence variation in the human angiotensin converting enzyme. *Nat Genet* 22: 59–62.
- Kerem B, Rommens JM, Buchanan JA, Markiewicz D, Cox TK, et al. (1989) Identification of the cystic fibrosis gene: genetic analysis. *Science* 245: 1073–1080.
- Brinza D, Zelikovsky A (2006) 2SNP: scalable phasing based on 2-SNP haplotypes. *Bioinformatics* 22: 371–373.
- Patil N, Berno AJ, Hinds DA, Barrett WA, Doshi JM, et al. (2001) Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science* 294: 1719–1723.
- Goldstein DB, Weale ME (2001) Population genomics: linkage disequilibrium holds the key. *Curr Biol* 11: R576–579.
- Reich DE, Cargill M, Bolk S, Ireland J, Sabeti PC, et al. (2001) Linkage disequilibrium in the human genome. *Nature* 411: 199–204.
- Halperin E, Eskin E (2004) Haplotype reconstruction from genotype data using Imperfect Phylogeny. *Bioinformatics* 20: 1842–1849.
- Clark AG (1990) Inference of haplotypes from PCR-amplified samples of diploid populations. *Mol Biol Evol* 7: 111–122.
- Li Z, Zhou W, Zhang XS, Chen L (2005) A parsimonious tree-grow method for haplotype inference. *Bioinformatics* 21: 3475–3481.
- Delaneau O, Coulounges C, Boelle PY, Nelson G, Spadoni JL, et al. (2007) ISHAPE: new rapid and accurate software for haplotyping. *BMC Bioinformatics* 8: 205.
- Marchini J, Cutler D, Patterson N, Stephens M, Eskin E, et al. (2006) A comparison of phasing algorithms for trios and unrelated individuals. *Am J Hum Genet* 78: 437–450.

38. Browning SR, Browning BL (2007) Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering. *Am J Hum Genet* 81: 1084–1097.
39. Stephens M, Donnelly P (2003) A comparison of bayesian methods for haplotype reconstruction from population genotype data. *Am J Hum Genet* 73: 1162–1169.