Bioinformatics, 34(11), 2018, 1801–1807
doi: 10.1093/bioinformatics/bty012
Advance Access Publication Date: 12 January 2018
Original Paper

OXFORD

Genome analysis

PEATH: single-individual haplotyping by a probabilistic evolutionary algorithm with toggling

Joong Chae Na[1], Jong-Chan Lee[1], Je-Keun Rhee[2,*] and Soo-Yong Shin[3,*]

[1]Department of Computer Science and Engineering, Sejong University, Seoul 05006, Korea, [2]Cancer Research Institute, College of Medicine, The Catholic University of Korea, Seoul 06591, Korea and [3]Department of Digital Health, Samsung Advanced Institute for Health Science and Technology, Sungkyunkwan University, Seoul 06351, Korea

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on March 29, 2017; revised on December 22, 2017; editorial decision on January 8, 2018; accepted on January 10, 2018

Abstract

Motivation: Single-individual haplotyping (SIH) is critical in genomic association studies and genetic diseases analysis. However, most genomic analysis studies do not perform haplotype-phasing analysis due to its complexity. Several computational methods have been developed to solve the SIH problem, but these approaches have not generated sufficiently reliable haplotypes.

Results: Here, we propose a novel SIH algorithm, called PEATH (Probabilistic Evolutionary Algorithm with Toggling for Haplotyping), to achieve more accurate and reliable haplotyping. The proposed PEATH method was compared to the most recent algorithms in terms of the phased length, N50 length, switch error rate and minimum error correction. The PEATH algorithm consistently provides the best phase and N50 lengths, as long as possible, given datasets. In addition, verification of the simulation data demonstrated that the PEATH method outperforms other methods on high noisy data. Additionally, the experimental results of a real dataset confirmed that the PEATH method achieved comparable or better accuracy.

Availability and implementation: Source code of PEATH is available at https://github.com/jcna99/PEATH.

Contact: jkrhee@catholic.ac.kr or sooyong.shin@gmail.com

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Single-individual haplotyping (SIH) aims to reconstruct a human diploid, which is a pair of human chromosomes inherited from parents. Although each haploid from each parent is almost identical, the set of variants on each chromosome can show a few differences. These small differences play important roles in several biological processes, such as those leading to genetic diseases (Tewhey *et al.*, 2011). Therefore, haplotype-based analysis is powerful and promising for genomic association studies, where the phasing of the haplotype in the human genome has been one of the major challenges in both biological and computational aspects (Snyder *et al.*, 2015).

Haplotypes have been reconstructed by connecting overlapping heterozygous genomic sites in the sequence fragments. However, optimal accurate determination of the haplotype from the DNA sequence fragments with noise is a computationally NP-hard problem (Lancia *et al.*, 2001). However, with rapid advances in high-throughput sequencing technologies and decreased sequencing costs, many researchers have tried to practically resolve the SIH problem using real human whole-genome sequencing datasets (Rhee *et al.*, 2016). The first studies to determine the haplotype of a complete individual human genome were presented by Levy *et al.* (2007). They concatenated DNA sequence reads from shotgun sequencing by a simple greedy heuristic method, but this method was likely not

accurate. Subsequently, several more sophisticated computational approaches were proposed, including HASH, implemented by the MCMC (Markov Chain Monte Carlo) algorithm (Bansal *et al.*, 2008), and HAPCUT by solving the Max-Cut problem (Bansal and Bafna, 2008). In addition, a dynamic programming algorithm was applied (He *et al.*, 2010). These methods work for determining of the individual haplotypes, but do not show sufficiently adequate results for NGS-based long-read sequencing datasets (Duitama *et al.*, 2012). More recently, Duitama *et al.* proposed an algorithm, RefHap, to solve the SIH problem and verified its utility using fosmid-based sequencing datasets (Duitama *et al.*, 2012). Use of the new sequencing datasets, which were provided with long reads, made it more feasible to solve the SIH problem (Kitzman *et al.*, 2011; Suk *et al.*, 2011), and the state-of-the-art algorithms have primarily been validated using Dituma's fosmid-sequencing datasets. Xie *et al.* developed a heuristic dynamic programming algorithm, H-BOP (Xie *et al.*, 2012), and Matsumoto and Kiryu solved the problem by estimating the posterior distribution of the haplotypes using the variational Bayes methods in a mixture model (Matsumoto and Kiryu, 2013). ProbHap also uses a dynamic programming algorithm based on probabilistic methods and provides more accurate results for fosmid-based sequencing with a relatively low sequencing depth (Kuleshov, 2014). However, the ProbHap algorithm might not be practically applicable for sequencing datasets with a high depth of coverage. Moreover, it is not clear whether these algorithms will lead to robust results, even with high noisy data. Therefore, a new method with a reliable performance, regardless of the noise levels, is needed.

Here, we propose a novel SIH algorithm, PEATH (Probabilistic Evolutionary Algorithm with Toggling for Haplotyping), based on the estimation of distribution algorithm (EDA). EDA is an evolutionary optimization technique that offers suitable solutions in complex search spaces, including many bioinformatics problems (Armananzas *et al.*, 2008). We constructed a haplotype for a single individual using actual sequencing-based data and simulation data according to the proposed EDA-based approach. The PEATH method showed results comparable to those of state-of-the-art algorithms. Moreover, this method provided a more accurate haplotype with high noisy data. The remainder of this paper is structured as follows: the PEATH algorithm is illustrated in detail in Section 2; the experimental results via a real dataset and a simulation study are presented in Section 3; and finally, a short discussion and conclusion are provided in Section 4.

2 Materials and methods

The output of the SIH algorithm considers only one haplotype expressed as $h = [0, 1]^m$, where m is the length of the haplotype, as the other is always complementary. The overall phasing procedures of the proposed PEATH are shown in Figure 1 and Table 1.

2.1 Input data matrix

Suppose that two n x m matrices are **M** and **Q**. **M** is the input matrix; each row ($i = 1, \ldots, n$) denotes a sequence read, and each column ($j = 1, \ldots, m$) denotes a position with a heterozygous variant. $\mathbf{M}_{ij} \in$ {'0', '1', '-'} is an element for ith row and jth column of the **M** matrix. That is, the value of $\mathbf{M}_{ij}$ refers to an allele covered from ith sequence read at the given position (j). The values '0' and '1' are the two alleles at the position. For example, value '0' means that the sequence at the position is the same as the reference sequence, while '1' means the sequence heterozygous variant is

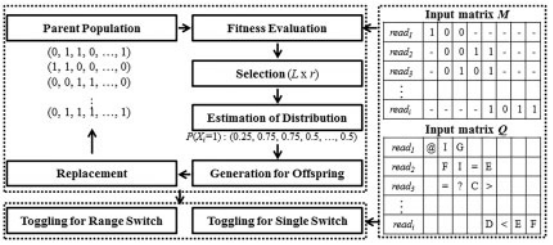

Fig. 1. Overview of the procedures for the PEATH method. The values for input matrix M indicate a position with a heterozygous variant and those for input matrix Q are shown as ASCII codes to represent the quality score for sequenced nucleotides. For the selection step, L is the population size and r is the replacement ratio

Table 1. Pseudocode for the PEATH method

```
Procedure PEATH
  input: input matrix M for sequence reads,
         input matrix Q for quality scores,
  output: haplotype with the best fitness
  for i: = 1 to phasing_iter do          //phasing_iter is a parameter
    h^i: = EDA()                         //estimation of distribution algorithm
    for k: = 1 to tog_iter do            //tog_iter is a constant parameter
      h^i: = Range_SWT (h^i)             //range switch toggling
      h^i: = Single_SWT (h^i)            //single switch toggling
    end for
  end for
  return the best haplotype among h^i's
end procedure

Procedure EDA                            //estimation of the distribution algorithm
output: haplotype with the best fitness

  parent: = Initialize the population
  while it does not meet stop condition do
    selected: = Select the best L x (1-r) individuals from parent,
      where L is the population size and r is the replacement ratio
    estimated: = Calculate the frequency of each position from selected
    offspring: = Generate L x r individuals based on estimated
    parent: = Replace the worst L x r individuals in parent by offspring
  end while
  return the best individual in parent
end procedure

Procedure Range_SWT                      //Toggling for range switch
input: haplotype h of length m
output: haplotype with the best fitness

  do
    best_h: = h
    for j: = 1 to m do
      tog_h^j: = Flip bits of best_h at positions between 1 and j
    end for
    h: = the best haplotype among tog_h^j's
  while h is better than best_h
  return best_h
end procedure

Procedure Single_SWT                     //Toggling for single switch
input: haplotype h of length m
output: haplotype with the best fitness

  do
    best_h: = h
    for j: = 1 to m do
      tog_h^j: = Flip bit of best_h at position j
    end for
    h: = the best haplotype among tog_h^j's
  while h is better than best_h
  return best_h
end procedure
```

detected at the position. The symbol '-' indicates that the sequence read does not cover the position. Within matrix **M**, the sequence reads are sorted by their start positions. **Q** is a quality score matrix with the same size as matrix **M**. The element of the **Q** matrix represents the probability of sequencing errors, expressed as a Phred Score, as shown in Figure 1.

2.2 Estimation of distribution algorithm

To determine the haplotypes from the sequencing reads with noise, an estimation of the distribution algorithm, which is a probabilistic evolutionary algorithm, was used. The evolutionary algorithm can identify good candidates for an optimization problem based on a fitness function. In the present study, the solution of the evolutionary algorithm is a haplotype. Each position of the candidate individual coincides with a position of the input matrix **M**, and each position of the candidate individual also has a value of 0 or 1.

The candidate solution was searched by using an UMDA (univariate marginal distribution algorithm), which uses the univariate marginal frequency to generate offspring (Mühlenbein, 1997). The following basic procedure was used (refer Table 1): (1) the initial populations were randomly generated with population size L; (2) fitness values were evaluated; and (3) highly ranked candidates were selected. The selected number was set to L x (1-r) (r is the ratio of replacement). (4) Probability vector $P=(p_1, p_2, \ldots, p_n)$ stores the probability for '1' at each position. (5) A set of offspring was generated according to the probability vector P and (6) the candidates with the lowest fitness values within the parents were replaced by the newly generated offspring. Procedures 2–6 were repeated until convergence.

2.3 Fitness function

The fitness of the haplotype was calculated by comparing the values at the position within h to the elements of the input matrix **M**. The fitness function is defined as:

$$\text{fitness}(h) = -\sum_i \min(D(h, \mathbf{M}_i),\ \mathrm{D}(h^*, \mathbf{M}_i)), \tag{1}$$

$$D(h, \mathbf{M}_i) = \sum_j \delta(h_j, \mathbf{M}_{ij}), \tag{2}$$

$$\delta(h_j, \mathbf{M}_{ij}) = \begin{cases} 0 & \mathbf{M}_{ij} = \text{'-'} \\ S_{ij} & \mathbf{M}_{ij} \neq \text{'-'} \ \textit{and}\ h_j = \mathbf{M}_{ij}, \\ 1 - S_{ij} & \mathbf{M}_{ij} \neq \text{'-'} \ \textit{and}\ h_j \neq \mathbf{M}_{ij} \end{cases} \tag{3}$$

$$S_{ij} = 10^{-\frac{\mathbf{Q}_{ij}}{10}}\ \mathbf{Q}_{ij} = \textit{phred quality score}. \tag{4}$$

$\mathbf{M}_i$ is the ith sequence read in the input matrix **M**. h and $h*$ are complement haplotypes. h_j is the value of the jth position of haplotype h, and $\mathbf{M}_{ij}$ is the value for the jth position of the ith sequence read in input matrix **M**. $\mathbf{Q}_{ij}$ is the phred quality score and represents the probability for a sequencing error at the jth position of the ith sequence read. That is, the algorithm attempts to identify the minimal sum of the quality-weighted errors considering the complementary characteristics of the two haplotypes, h and $h*$.

2.4 Correction of haplotype using toggling

The solution obtained by the UMDA might be not precise due to the simplicity of UMDA and still involves several switch errors, suggesting differences from the actual answer in each position. Thus, the final solution is corrected with an exhaustive toggle approach (Table 1). The switch errors were represented as range switch errors and single switch errors. For range switch errors, (1) the toggling method identifies the position where the maximum fitness value is achieved, flipping each bit from the first position to the last position. (2) Then, each value is flipped from the first position to the position obtained the maximum fitness. (3) The procedure is repeated until a higher fitness value is no longer available.

For single switch errors, the overall procedures were similar to those for the range switch errors. The only difference was to flip a single bit at a time, in contrast to a continuous flip, for steps (1) and (2) of the procedure for range switch errors.

2.5 Dataset and implementation

Performance was measured using the NA12878 sequencing dataset produced by fosmid-based technology (Duitama *et al.*, 2012). This dataset has been widely used to assess and compare SIH algorithms. The experiments were performed using a desktop computer with Ubuntu 14.04 LTS 64-bit, Intel Core i5-6600 CPU@3.30 GHz and 16 GB RAM.

For real implementation, the haplotype was determined by using a phasing blocks as a unit to solve the SIH problem. The block was defined as a set of sequence reads whose covered positions were overlapped, and phasing was performed for each block. In the present experiments, the phasing iteration number *phasing_iter* and toggling iteration number *tog_iter* were set as 50 and 10, respectively. In EDA, the population size L and ratio of the replacement r were set as 100 and 0.5, respectively.

2.6 Experiments with the simulation dataset

The simulation data were generated based on Dutima's NA12878 sequencing data as described in Section 2.5. Initially, the distribution of the quality scores ($\mathbf{Q}_{ij}$) in the original data was investigated to generate the quality value of the simulation data. The quality scores of the original data were divided into two sets, T and F, according to the concordance of $\mathbf{M}_{ij}$ with the true answer. The quality scores at the true sites of the highly concordant sequence reads (the concordance ratio >0.7) were classified as set T, and the other scores were allocated as set F. The low concordant sequence reads (concordance ratio <0.3) were assumed to be complementary sequence reads since the true answer was represented as only one haplotype among the two reads. Thus, the quality scores at the false sites of the low concordant reads were allocated to set T, while the other scores were classified as set F.

Next, the input matrix $\mathbf{M}'$ without noise was generated by correcting the noise within the original input matrix **M** based on the gold-standard information. If the concordance ratio of $\mathbf{M}_i$ was <0.5, then all of the values at $\mathbf{M}_i$ were changed to complementary values and the inaccurate values were corrected. The quality scores at the corrected positions were randomly selected from set T.

Finally, some noise was added at random positions within the matrix $\mathbf{M}'$, and the quality values at the positions with noise were randomly selected from set F. Twenty sets of simulation data using chromosome 22 were generated by a change of the random seed per specific noise ratio, and the experiments with one piece of simulation data were iteratively performed 20 times. Hereafter, the noise rate is defined as the probability that the value of each position at the input matrix is wrong.

3 Results

3.1 Experiment with simulation data

To verify the PEATH algorithm, we initially tested its performance with simulation data generated as describe in Section 2.6. We

compared the results to those obtained with state-of-the-art SIH algorithms, such as ProbHap (Kuleshov, 2014) and MixSIH (Matsumoto and Kiryu, 2013). For ProbHap, there are two executable source codes, implementation of the basic algorithm and a post-processing heuristic version, to improve the phasing accuracy. Hereafter, we distinguished these two versions as ProbHap-O (original version) and ProbHap-P (post-processing version), respectively. First, we measured the phased length and N50 length to determine the length and accuracy of the haplotype generated by the PEATH algorithm. The phased length is calculated as the total length of the generated haplotype. N50 is a measure that is mainly used for genome assembly, and its length indicates the point of half of the block distribution. Figure 2 shows the phased length according to the noise rate of the phasing matrix. The PEATH method consistently achieved the longest phased length (441 660). Moreover, the phased length obtained with PEATH was identical, regardless of the noise rate, while the phased lengths obtained with ProbHap-P and MixSIH were reduced with an increasing noise rate, 434 683–417 328 and 439 254–436 251, respectively (the phased length of MixSIH was an average value from the repeated experiments). Although ProbHap-O also provided a fixed phased length (441 520) with a different noise rate, the phased length was a slightly shorter than that obtained with the PEATH method. Similar characteristics were observed for N50 length (Table 2). The N50 length for PEATH and ProbHap-O was 420, regardless of the noise rate, whereas the N50 length for the other methods was lower and also apparently diminished at high noise rates. For the 20% noise rate, the N50 length for ProbHap-P and MixSIH was 391–404 and 405–420, respectively.

Next we measured the switch error rate (SWER) and minimum error correction (MEC) score, according to the noise rate of the phasing matrix, to evaluate the accuracy of the present approach (Fig. 3). The SWER is measured by the ratio of the number of switch errors to the length of the entire haplotype. We analyzed the SWER based on the single switch error (SSWER) and range switch error (RSWER). SSWER was calculated as the number of switch errors at a single position per megabase (sw/mb). RSWER was determined as the number of the long (two or more bases) range inversion per megabase (sw/mb). These two measures for the switch errors were the same as the short switch and long switch used in the ProbHap study (Kuleshov, 2014). Because the phased length and phased position were slightly different among the algorithms, the evaluation was performed by using identical phased positions for fair comparison. Overall, the PEATH method showed better performance for the SWERs, despite the high noise rates. For example, at a 20% noise ratio, the RSWER, SSWER and MEC of the PEATH method were significantly lower than those of ProbHap-O (P value $= 4.61 \times 10^{-56}$, 7.12×10^{-158} and 5.82×10^{-92}; t-test). Even at the ProbHap-P, the SWERs of PEATH were also significantly lower, with P values 1.16×10^{-11} and 2.13×10^{-10} for RSWER and SSWER, respectively. For MixSIH, differences were clearly observed, where the P values for RSWER, SSWER and MEC were 6.40×10^{-191}, 1.71×10^{-143} and 2.91×10^{-187}, respectively. Moreover, even at low noise rates of 5%, the two SWERs and MEC of PEATH were significantly lower than those of ProbHap-O and MixSIH (P values $< 1.00 \times 10^{-7}$), excluding SSWER of MixSIH (P value > 0.05). The results verified that the PEATH method generates relatively robust results with highly noisy datasets.

3.2 Experiment with real dataset

Next, we applied the PEATH method to two actual datasets, fosmid-based sequencing data (NA12878) (Duitama *et al.*, 2012) and HuRef data (Levy *et al.*, 2007). For the formid-based sequencing dataset, we measured the phased length and N50 length to determine the length and accuracy of the haplotype generated by the PEATH algorithm.

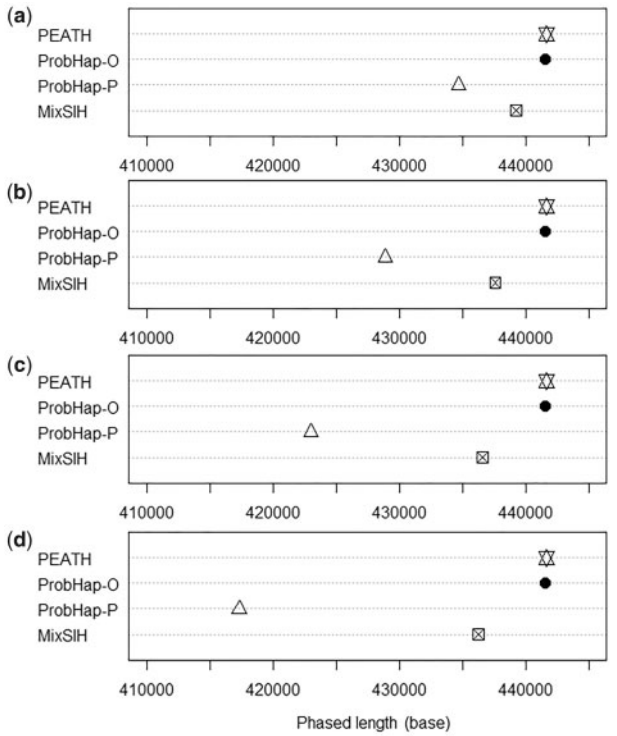

Fig. 2. Phased length according to the noise rate. Dotplot showing the phased length in each SIH algorithm with (**a**) noise 5%, (**b**) 10%, (**c**) 15%, (**d**) 20%. The *x*-axis is the phased length (bp). ProbHap-O is the ProbHap without post-processing, while ProbHap-P is the ProbHap with post-processing. For MixSIH, an average value is represented

Table 2. Average N50 length for the simulated data according to the noise rate

	5%	10%	15%	20%
PEATH	**420**	**420**	**420**	**420**
ProbHap-O	**420**	**420**	**420**	**420**
ProbHap-P	413.3	408.6	401.6	396.7
MixSIH	417.3	416.8	415.3	415.3

Note: The best result for each noise rate is indicated in bold. For ProbHap-P and MixSIH, the average value is presented.

For the phased length, the PEATH algorithm generated the longest haplotype (1 537 790) compared to that obtained with other methods because the present method thoroughly phased all positions where a sequence read exists (Table 3). For ProbHap, the phased length of the basic algorithm was 1 536 615, but after post-processing to improve the accuracy, the phased length was diminished to 1 519 565. The phased length for MixSIH was 1 532 600–1 533 773 in the repeated experiments. Table 4 represents the N50 length in each chromosome. PEATH consistently showed the best performance compared to that of other methods, although the N50 length varied according to the chromosome.

To test whether the phased haplotypes were accurately determined, we also calculated SWERs and MEC for the identically phased position. Figure 4 presents the comparison results for RSWER, SSWER and MEC using PEATH, ProbHap-O, ProbHap-P

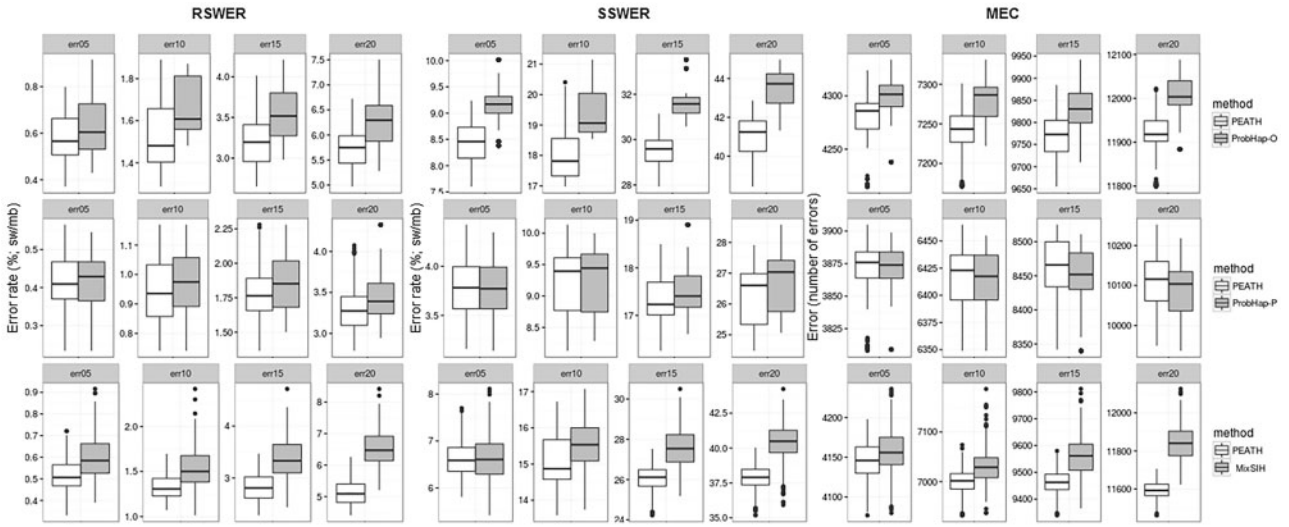

Fig. 3. Switch error rate and MEC for the simulated dataset according to the noise rate. The boxplot presents the performance of the PEATH and three other SIH algorithms (each row) using the simulated dataset. The evaluation was performed only at the identically phased positions between the two comparison methods for fairness. The noise rate is represented at the top on the boxplot as err05 (noise 5%), err10 (noise 10%), err15 (noise 15%) and err20 (noise 20%). RSWER and SSWER are represented as the error rate % (sw/mb), and MEC is shown as the number of error (base)

Table 3. Phased length for the fosmid-based sequencing data

	Phased length
PEATH	**1 537 790**
ProbHap-O	1 536 615
ProbHap-P	1 519 565
MixSIH	1 533 223

Note: The best result is indicated in bold. For MixSIH, the average value is presented.

Table 4. N50 length in each chromosome for the fosmid-based sequencing data

chr	PEATH	ProbHap-O	ProbHap-P	MixSIH
1	**231**	229	226	228.6
2	**231**	226	224	225.8
3	**208**	203	202	203.9
4	**203**	201	199	200.5
5	**227**	224	221	224.1
6	**271**	261	257	258.6
7	**232**	**232**	228	229.1
8	**258**	255	253	254.4
9	**251**	248	246	247.3
10	**243**	241	238	241.3
11	**239**	236	232	237.0
12	**242**	236	235	236.0
13	**201**	197	195	195.8
14	**267**	**267**	263	265.0
15	**241**	**241**	239	240.2
16	**365**	**365**	362	361.4
17	**263**	**263**	259	260.2
18	**223**	222	221	222.9
19	**280**	269	269	269.0
20	**279**	270	267	270.0
21	**244**	**244**	242	242.6
22	**420**	**420**	414	419.0

Note: The best result in each chromosome is marked in bold. For MixSIH, the average value is presented.

and MixSIH. The results showed that the SWERs and MEC of the PEATH method were comparable to those for the ProbHap method and better than those for MixSIH. For example, compared to ProbHap-O, the RSWER and SSWER for PEATH were 3541–3794 and 14 192–14 702, respectively, whereas the RSWER and SSWER of ProbHap-O were 3612 and 14 415, respectively, and the two results were comparable (Fig. 4a and b). Moreover, for MEC, the PEATH method showed better results than ProbHap-O. The MECs of PEATH and ProbHap-O were 121 430–121 714 and 122 001, respectively (Fig. 4c). Compared to those of ProbHap-P, the SWERs of the PEATH method were not clearly better (Fig. 4d–f). Compared to MixSIH, PEATH clearly showed better results. The average RSWERs were 3608 and 4009 for PEATH and MixSIH, respectively, and the average SSWERs were 14 088 and 15 765 for PEATH and MixSIH, respectively (Fig. 4g and h). The average MEC was 116 986 and 118 444 for PEATH and MixSIH, respectively (Fig. 4i).

We also applied the present method to the HuRef dataset. The results for PEATH are shown in Supplementary Table S1. For ProbHap and MixSIH, we could not obtain experimental results in this dataset. This result shows that our method can be used all kinds of sequencing data, in contrary to other SIH algorithms.

4 Discussion

In the present study, we proposed a new algorithm, PEATH, to solve the SIH problem. This algorithm identified reliable haplotypes (low error rates and reliably longer haplotype length) compared to the existing algorithms. Moreover, we validated that the PEATH method shows relatively better performance, even with highly noisy data, using simulation studies. In addition, the PEATH algorithm consistently represented the best phased length and N50 values, as the length of the haplotype is initialized by the number of total mutation sites and the phasing blocks are divided only in cases with no connection by the overlapped sequence reads.

The accuracy of the phased haplotypes is typically measured by SWERs and MEC. Previously, many SIH algorithms have used MEC

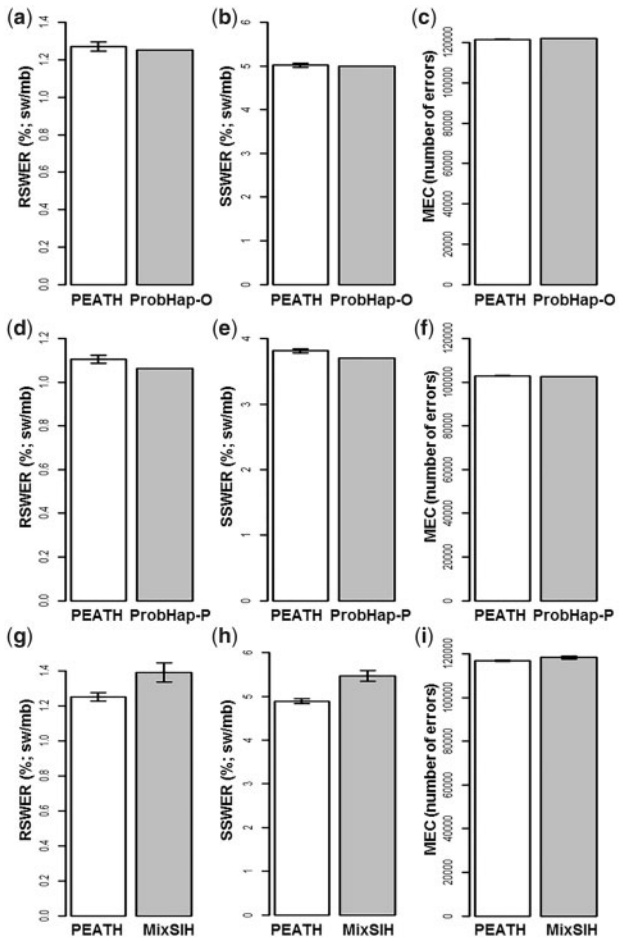

Fig. 4. Comparison of the performance. The bar graph shows the RSWER, SSWER and MEC for PEATH and three other methods. For fair comparison, these values were calculated at the identically phased positions between the two comparison methods in each barplot. (**a–c**) The results compared to ProbHap-O. (**d–f**) The results compared to ProbHap-P and (**g–i**) MixSIH. The standard deviation by the 20 repeated experiments is also represented at each bar

for the evaluation because there were no gold standard datasets. However, the evaluation score by MEC highly depends on the given input datasets and could occasionally be slightly different from the actual answer in the case of datasets with high noise. After the production of the NA12878 datasets, it was possible to verify the results using true datasets based on other measures, such as SWERs. The PEATH method showed comparable or better results to both SWERs and MEC. Moreover, as the results were validated using the simulation dataset, the PEATH method produced relatively good performances, even at high noise rates.

Some SIH algorithms have recently been developed for the examination of polyploids as well as diploids. One of the representative methods is SDhaP (Das and Vikalo, 2015). Unlike the present method, this approach does not assume that the two sequences, h and h^*, are complementary. Although a direct comparison would not be fair because of different assumptions, we compared the original results to those of SDhaP using fosmid-based sequencing datasets. The MEC result was better with SDhaP than with PEATH and other methods, including ProbHap-O, ProbHap-P and MixSIH (Supplementary Table S2, Fig. 4), although the SDhaP showed a much larger variance than the other methods. Indeed, for the HuRef dataset, the MEC result was much better for SDhaP (Supplementary Table S3). Although all of the phased positions were heterozygous at the gold-standard fosmid-based sequencing dataset, some positions (average 45 525 sites; approximately 2.96% of the phased length) were homozygous in SDhaP results, suggesting that the SDhaP results were different from the actual answers. In addition, although it is beyond the scope of the present study, we examined the potential improvement of MEC by alterations enabling h_i and h_i^* to have identical bases, and we obtained better MEC results than those obtained with SDhaP with the fosmid-based sequencing dataset (Supplementary Table S2; the modified version is represented as PEATH-homo at the Supplementary Table S2). In the future, we will examine algorithmic improvements to realize a better performance at all aspects on the evaluation measures for application on both diploid and polyploid datasets.

The main algorithm for the proposed method is based on UMDA, which determines the best solution by estimating univariate marginal probabilities based on the assumption that all input variables are independent. It is not necessary to set some factors that are related to crossover and mutation operators to appear in conventional genetic algorithms. EDA showed an outstanding performance for NP-hard problems (Larrañaga and Lozano, 2002) by combining the power of an evolutionary approach and statistical estimation. Since SIH is an NP-hard problem that is difficult for most optimization methods, EDA can be a good alternative. The drawback of EDA is that it requires a large amount of computational time since most EDA algorithms represent higher-order interactions among multiple variables based on a probabilistic tree or graph (Larrañaga, 2002). To overcome this drawback, we selected the simplest and fastest UMDA and applied a simple heuristic approach, such as toggling, to improve the performance. By combining two simple strategies, we efficiently and effectively solved the SIH problem with less computational burden. The execution time for each chromosome was 22.1–117.2 and 56.0–629.4 seconds for the fosmid-based sequencing datasets and HuRef datasets, respectively, although this time was dependent on the chromosome lengths. In addition, the PEATH method will be easily parallelized for faster execution (Lozano *et al.*, 2002).

Despite the development of massive parallel sequencing technologies, there are some limitations for determining the haplotype of a single individual directly from sequencing read data. The length of the reads from NGS was still too short to cover the neighboring heterozygous variants, and sequencing technologies producing longer reads would show relatively increased sequencing errors. However, the PEATH method showed a relatively good performance with highly noisy data, as verified through simulation data, and would be useful for the SIH problem and also for long read sequencing technologies in the future, although the sequencing error rates were still higher than those for short read sequencing data. Moreover, previous algorithms, such as ProbHap and MixSIH, did not work with short read sequencing datasets, as shown in the HuRef experiments. There have been several de novo assembly algorithms using short reads; by combining de novo assembly algorithms with the PEATH method, solving the SIH problem from short reads is possible. In addition, because the haplotype was phased based on the phasing blocks, it is feasible to connect using prior information, such as linkage disequilibrium, in the future.

Funding

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the MSIT (Ministry of Science and ICT), Republic of

Korea [grant no. NRF-2015R1C1A1A01053824], by the MSIT, Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-01629) supervised by the IITP (Institute for Information & communications Technology Promotion), by Basic Science Research Program through the NRF funded by the Ministry of Education [2017R1D1A1B03029451] and by the MSIT, Korea, under the National Program for Excellence in SW supervised by the IITP [2015-0-00938].

Conflict of Interest: none declared.

References

Armananzas,R. *et al*. (2008) A review of estimation of distribution algorithms in bioinformatics. *BioData Min*., **1**, 6.

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

Bansal,V. *et al*. (2008) An MCMC algorithm for haplotype assembly from whole-genome sequence data. *Genome Res*., **18**, 1336–1346.

Das,S. and Vikalo,H. (2015) SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics*, **16**, 260.

Duitama,J. *et al*. (2012) Fosmid-based whole genome haplotyping of a HapMap trio child: evaluation of Single Individual Haplotyping techniques. *Nucleic Acids Res*., **40**, 2041–2053.

He,D. *et al*. (2010) Optimal algorithms for haplotype assembly from whole-genome sequence data. *Bioinformatics*, **26**, i183–i190.

Kitzman,J.O. *et al*. (2011) Haplotype-resolved genome sequencing of a Gujarati Indian individual. *Nat. Biotechnol*., **29**, 59–63.

Kuleshov,V. (2014) Probabilistic single-individual haplotyping. *Bioinformatics*, **30**, i379–i385.

Lancia,G. *et al*. (2001) SNPs problems, complexity, and algorithms. In: auf der Heide, F.M. (ed.) *Algorithms—ESA 2001: 9th Annual European Symposium Århus, Denmark, August 28–31, 2001 Proceedings*. Springer, Berlin, Heidelberg. pp. 182–193.

Larrañaga,P. (2002) A Review on Estimation of Distribution Algorithms. In: Larrañaga, P. and Lozano, J.A. (eds.) *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer, Boston, MA. pp. 57–100.

Larrañaga,P. and Lozano,J.A. (2002) *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer, Boston, MA.

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

Lozano,J.A. *et al*. (2002) Parallel estimation of distribution algorithms. In: Larrañaga,P. and Lozano,J.A. (eds.) *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer, Boston, MA, pp. 129–145.

Mühlenbein,H. (1997) The Equation for Response to Selection and Its Use for Prediction. *Evol. Comput*., **5**, 303–346.

Matsumoto,H. and Kiryu,H. (2013) MixSIH: a mixture model for single individual haplotyping. *BMC Genomics*, **14**, S5.

Rhee,J.-K. *et al*. (2016) Survey of computational haplotype determination methods for single individual. *Genes Genomics*, **38**, 1–12.

Snyder,M.W. *et al*. (2015) Haplotype-resolved genome sequencing: experimental methods and applications. *Nat. Rev. Genet*., **16**, 344–358.

Suk,E.K. *et al*. (2011) A comprehensively molecular haplotype-resolved genome of a European individual. *Genome Res*., **21**, 1672–1685.

Tewhey,R. *et al*. (2011) The importance of phase information for human genomics. *Nat. Rev. Genet*., **12**, 215–223.

Xie,M. *et al*. (2012) A fast and accurate algorithm for single individual haplotyping. *BMC Syst. Biol*., **6**, S8.