BMC Systems Biology

PROCEEDINGS Open Access

A fast and accurate algorithm for single individual haplotyping

Minzhu Xie[1*], Jianxin Wang[2], Tao Jiang[3]

From 23rd International Conference on Genome Informatics (GIW 2012)
Tainan, Taiwan. 12-14 December 2012

Abstract

Background: Due to the difficulty in separating two (paternal and maternal) copies of a chromosome, most published human genome sequences only provide genotype information, *i.e.*, the mixed information of the underlying two haplotypes. However, phased haplotype information is needed to completely understand complex genetic polymorphisms and to increase the power of genome-wide association studies for complex diseases. With the rapid development of DNA sequencing technologies, reconstructing a pair of haplotypes from an individual's aligned DNA fragments by computer algorithms (*i.e.*, Single Individual Haplotyping) has become a practical haplotyping approach.

Results: In the paper, we combine two measures "errors corrected" and "fragments cut" and propose a new optimization model, called Balanced Optimal Partition (BOP), for single individual haplotyping. The model generalizes two existing models, Minimum Error Correction (MEC) and Maximum Fragments Cut (MFC), and could be made either model by using some extreme parameter values. To solve the model, we design a heuristic dynamic programming algorithm H-BOP. By limiting the number of intermediate solutions at each iteration to an appropriately chosen small integer *k*, H-BOP is able to solve the model efficiently.

Conclusions: Extensive experimental results on simulated and real data show that when *k* = 8, H-BOP is generally faster and more accurate than a recent state-of-art algorithm ReFHap in haplotype reconstruction. The running time of H-BOP is linearly dependent on some of the key parameters controlling the input size and H-BOP scales well to large input data. The code of H-BOP is available to the public for free upon request to the corresponding author.

Background

Each human somatic cell contains 23 pairs of chromosomes, and there are about 0.5% differences between the DNA sequences of two copies of each chromosome [1]. The dominant DNA differences are single nucleotide polymorphisms (SNPs). Identification of the combination of alleles at the SNP loci on the same chromosome copy, *i.e.*, haplotyping, is needed to fully understand the human genetic variation patterns and enhance the power of genome-wide association studies for complex diseases [2,3]. Currently, it is expensive and labor-intensive to separate two copies of chromosomes by biological techniques [4], and most published human individuals' genomes contain only the mixed information, *i.e.*, genotype information, of the underlying two copies of chromosomes [5]. Therefore, to reduce the cost, accurate and fast computational haplotyping methods are of urgent importance.

There have been many computational haplotyping models [6-8] and they can be grouped into two main classes: haplotype inference and haplotype assembly [6]. Haplotype inference is to phase the haplotypes of individuals in a pedigree or a population from their genotypes. Computer algorithms of haplotype inference have been used in the International HapMap Project [9] and the 1000 Genomes Project [5] to identify haplotypes. Haplotype assembly is also called Single Individual Haplotyping (SIH). SIH assembles a pair of haplotypes from an individual's aligned DNA sequence fragments. With the dramatically dropped cost of human whole genome sequencing, more and more human individual's DNAs have been sequenced. With mate-pairs

* Correspondence: xieminzhu@hotmail.com
[1]College of Physics and Information Science, Hunan Normal University, Changsha 410081, P. R. China
Full list of author information is available at the end of the article

sequencing and read length improvements of the next-generation sequencing technologies, and with the development of new sequencing technologies, SIH methods have been used to build haplotype-resolved genome of human beings [10,11]. When there are enough DNA sequence fragments that cover two or more consecutive variant loci, SIH builds longer and more accurate haplotype blocks than haplotype inference does [12].

Since Lancia *et al.*, first formalized the SIH problem [13], many optimization models and algorithms have been introduced to solve the problem [7,14-25]. The main models are MEC (minimum error correction) [24], MFR (minimum fragment removal), and MSR (minimum SNP removal) [13]. Recently, Duitama *et al.*, proposed a new model MFC (maximum fragments cut) [16]. Most of the models are NP-hard and APX-hard [16,21,26], and their exact algorithms run in time exponentially dependend on at least one input parameter [15,17,19-21]. Therefore, a large number of heuristic algorithms have been designed to deal with the problem [16,22,23,25]. According to [16], one of the most accurate heuristic algorithms is HapCUT [25], while ReFHap [16] runs much faster than HapCUT without loss of accuracy. In this paper, we consider both quality measures “errors corrected” and “fragments cut”, and propose a new optimization model, called Balanced Optimal Partition (BOP), for the SIH problem. The model generalizes the most popular model MEC and the recent model MFC. In fact, it could be made either model by setting some parameters to extreme values. To solve the model, we propose a dynamic programming algorithm H-BOP. By limiting the number of intermediate solutions at each iteration to an appropriately chosen small integer k, H-BOP is able to solve the model efficiently. The time complexity of H-BOP linearly depends on some of the key parameters controlling the input size and the algorithm scales well to large input data.

Results and discussion

We use a public available Java package SIH [27] to test the performance of H-BOP. The package contains a simulated data generator and implements algorithm ReFHap [12,16]. The simulated data are generated according to five parameters: number of SNPs (haplotype length) n, number of fragments m, average fragment length l, sequencing error rate e, and gap rate g. In our experiments, since we only consider heterozygous SNPs, for each data set, a haplotype h_1 containing n SNPs is generated randomly first and then the other haplotype h_2 is obtained by flipping each allele of h_1. For each haplotype, $m/2$ fragments are randomly sampled from the haplotype and their lengths follow a normal distribution with mean l and variance 1. Finally for each fragment, every allele is flipped with probability e to introduce sequencing errors and, except at the first and last positions, every allele is deleted with probability g to introduce gaps. Given fragments generated as above, the average call coverage c is calculated by dividing the total number of alleles of the fragments by the haplotype length n. Please see [16] for more details.

Among many algorithms for the SIH problem, HapCUT and ReFHap are two of the most accurate heuristic algorithms [12,16]. Since ReFHap is much faster than HapCUT, we only compare our algorithm with ReFHap. We implemented our algorithm H-BOP in Java and embedded it in the java package SIH and tested the accuracies, phased haplotype lengths and running time of H-BOP and ReFHap on simulated data and a real data set provided by [12]. All tests are carried out on a Windows 7 64 bit PC (3GHz CPU, 4GB RAM). To measure the haplotype reconstruction accuracy of an SIH algorithm, the hamming distance between the reconstructed haplotype pair and the real haplotype pair was previously used widely in the literature [14,23]. However, a recent study [28] showed it over-penalizes simple switch errors. Therefore, as in [12,16], we use switch errors to measure the accuracy of an algorithm. A switch error is an inconsistency between the reconstructed haplotype pair and the real haplotype pair over two contiguous SNPs. There may be some SNP sites where an algorithm is unable to determine the alleles of a haplotype. The phased haplotype length is defined as the number of SNP sites where the alleles of the reconstructed haplotype pair are determined. And the number of switch errors divided by the phased haplotype length is called switch error rate.

If the allele of a fragment f at a SNP site s is known, we say f covers s. When there are no fragments covering two consecutive loci, it is not possible to determine the haplotype containing these consecutive loci for all SIH models. Therefore, for each test we divide a haplotype into blocks according to the input fragments as in [16]. A block corresponds to a connected component of a graph $G = (V, E)$ where V is the set of the SNP sites and there is a edge between two SNP sites s_1 and s_2 if and only if there is a fragment covers both s_1 and s_2. The switch errors of an algorithm are the sum of the switch errors in all blocks. In the following simulation tests, the haplotype length n = 100, the gap rate g = 0.1 and each result is the average over 100 repeated experiments if there is no explicit specification.

Parameters of the algorithm H-BOP

There are two parameters w and k in H-BOP. The parameter w is a weighting factor. H-BOP tries to seek a solution with the minimum number of errors corrected when w = 0, and a solution with a maximum cut of the weighted conflict graph corresponding to the input fragments [16] when w is set sufficiently large. k is the maximum number of intermediate solutions that we will keep at each iteration of H-BOP. When k is large enough, H-BOP in fact becomes an exact algorithm. To choose appropriate values for w and k, we test H-BOP on different combinations of w and k.

In Figure 1, the haplotype length n = 100 and the average fragment length l = 3. In Figure 1(a), the number of fragments m = 140 (c = 4.25) and k = 16. In Figures 1(b) and 1(c), m = 210 (c = 6.42) and w = 0.1. Figure 1(a) shows that when the sequencing error rate e increases from 0.01 to 0.05, the switch errors of H-BOP increases accordingly. The switch errors of H-BOP is larger when w = 0 than those when w >0 with e unchanged, which is obvious when e = 0.05. It indicates that the optimal objective minimum errors corrected leads large switch errors when sequencing error rate is high. When w = 0.01 and 0.1, the switch errors of H-BOP are smallest. Figure 1(b) shows that when k increases from 1 to 8, the switch errors decrease accordingly. When k increases from 8 to 64, there are no significant improvements in switch errors of H-BOP. When sequencing error rates are high, exact optimal solutions may incur large switch errors and Figure 1 (b) indicates that when k increases from 8 to 64, switch errors of H-BOP increases accordingly. Figure 1(c) shows that the running time of H-BOP increases linearly with k when the haplotype length n, number of fragments m and average fragment length l remain fixed. In the following tests, we set w = 0.1 and k = 8 for H-BOP.

Simulation results

We changed the sequencing error rate e, the average fragment length l and the number of fragments m to generate different fragment data sets, and compared the performance of H-BOP and ReFHap. Figure 2 shows that H-BOP and ReFHap are both accurate and there are only several switch errors in reconstructing haplotypes of 100 SNPs. The accuracies of both algorithms decrease with the increase of e and improves with the increase of m. These results are consistent with those in [16], which claim that the accuracy of an SIH algorithm increases with decreasing sequencing error rate and increasing call coverage. In a half of the total 48 cases shown in Figure 2, H-BOP produces fewer switch errors than ReFHap, especially when l = 3 and $e \geq 0.02$. In the other half, H-BOP presents a few more switch errors than ReFHap only in two cases (*i.e.*, Figure 2(a), m = 140, e = 0.005 and Figure 2(b), m = 111, e = 0.02). In the remaining 22 cases, H-BOP has the same switch errors as ReFHap.

The phased haplotype lengths of both algorithms generally increase with decreasing e and increasing m. Table 1 shows that when l = 3 and m <351, or when e = 0.05 (except when l = 10 and m >44), H-BOP is able to phase more SNPs than ReFHap. In other cases the phased haplotype lengths of H-BOP and ReFHap are equal.

We set e = 0.01 and varied the haplotype length n, the number of fragments m and the average fragment length l to compare the running time of H-BOP and ReFHap (Figure 3). When n = 100, l = 3 and m increases from 100 to 400, the running time of H-BOP increases linearly, but the running time of ReFHap increases sharply (Figure 3(a)). When m reaches 400, the running time of H-BOP is only about 4 seconds, while that of ReFHap is about 468 seconds. When n increases from 50 to 200 while m = 200 and l = 3, the average call coverage decreases and the running time of both algorithms decreases accordingly (Figure 3(b)). When m = 200, n = 100 and l increases from 3 to 9, the running time of ReFHap increases significantly while that of H-BOP increases slowly and remains less than 5 seconds (Figure 3(c)). When the number of fragments and the average fragment length increase, the average call coverage c increases accordingly. When c is large, H-BOP runs much faster than ReFHap.

Results on real data

We downloaded a real data set from the SIH website [27], which contains the aligned sorted fosmid-based NGS DNA sequence fragments and gold-standard

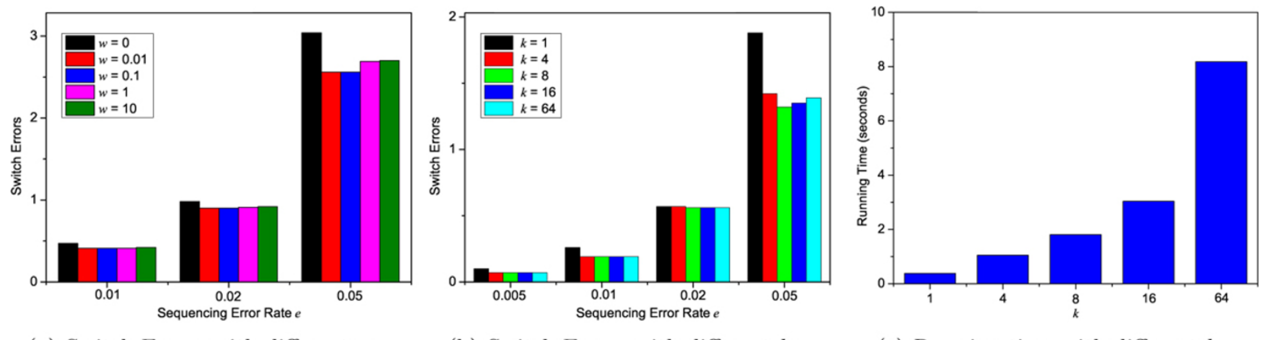


Figure 1 Performances of H-BOP with different parameter values. Here, the haplotype length n = 100, the gap rate g = 0.1 and the average fragment length l = 3. (a) The number of fragments m = 140 and k is 16. (b) w = 0.1 and m = 211. (c) w = 0.1, m = 211 and e = 0.01.

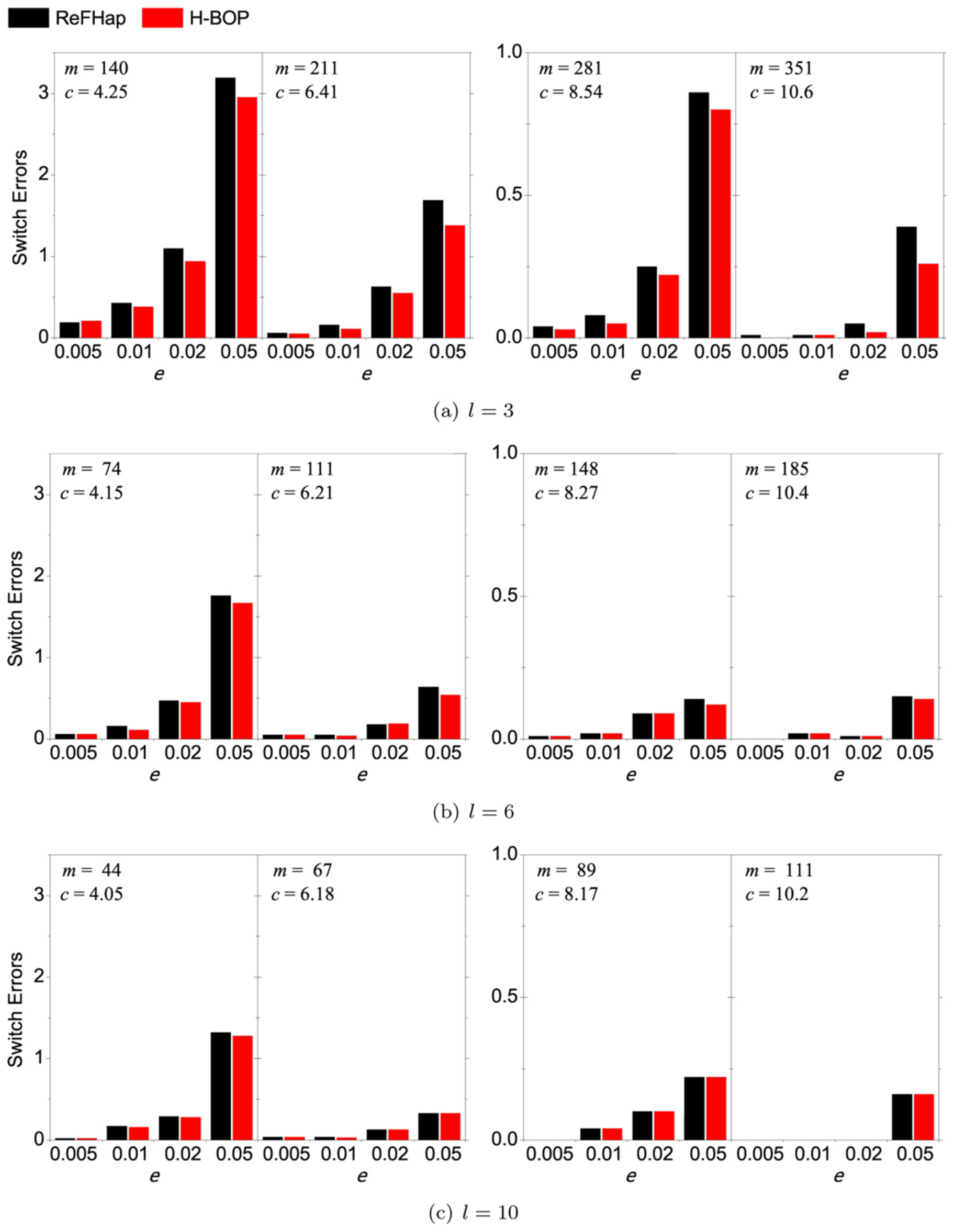


Figure 2 Switch errors of ReFHap and H-BOP with varying *e*, *m* and *l*. The parameters *k* and *w* of H-BOP are set as 8 and 0.1, respectively, and the haplotype length *n* is again set as 100.

haplotypes of a HapMap trio child, NA12878 [12]. The total heterozygous SNP sites of the data are 1,704,166, the total fragments are 285,341, the average fragment length is 18.03, and the average call coverage is 3.02. Since the coverage is low, there are many consecutive heterozygous SNP site pairs not covered by any fragments, and hence the 23 pairs of chromosomes are divided into 17,839 haplotype blocks. Due to the low

Table 1 Average phased haplotype lengths of ReFHap and H-BOP

		l = 3				*l* = 6				*l* = 10			
	m =	140	211	281	351	74	111	148	185	44	67	89	111
e = .005	ReFHap	98.16	99.59	99.79	99.97	97.2	99.04	99.39	99.66	96.24	97.7	98.51	99.06
	H-BOP	98.17	99.61	99.8	99.97	97.2	99.04	99.39	99.66	96.24	97.7	98.51	99.06
e = .01	ReFHap	97.99	99.6	99.68	99.9	97	98.87	99.32	99.59	95.68	97.81	98.73	99.13
	H-BOP	98.07	99.63	99.7	99.9	97	98.87	99.32	99.59	95.69	97.81	98.73	99.13
e = .02	ReFHap	97.42	99.39	99.71	99.86	96.94	98.59	99.48	99.63	95.31	98.06	98.24	99.22
	H-BOP	97.57	99.48	99.74	99.86	96.95	98.59	99.48	99.63	95.31	98.06	98.24	99.22
e = .05	ReFHap	96.6	98.88	99.49	99.77	95.61	98.31	99.14	99.5	94.25	97.56	98.62	98.87
	H-BOP	96.94	99.09	99.63	99.84	95.76	98.42	99.15	99.51	94.33	97.56	98.62	98.87

The numbers in the table are the average of 100 repeated tests.

coverage, both H-BOP and ReFHap ran very fast and completed the reconstruction of haplotypes for all 23 pairs of chromosomes in about half a minute. The total phased haplotype lengths of H-BOP and ReFHap are 1,563,741 and 1,562,402 respectively. Compared with the gold-standard haplotypes, the total switch errors of the haplotypes built by H-BOP and ReFHap are 21,859 and 21,835 respectively. Though the switch errors of H-BOP is larger than that of ReFHap, the switch error rates of H-BOP and ReFHap are both 0.014.

To account for both completeness and quality, Duitama *et al.* [12] proposed an alternative measure QAN50 (quality adjusted N50). QAN50 is calculated as follows [12]:

(1) Break every haplotype block into the longest possible segments containing no switch errors.

(2) Calculate span (in reference base pairs) from the first phased SNP to the last phased SNP for every segment.

(3) Adjust each span by multiplying the span with phased SNPs ratio (the number of phased SNPs divided by the number of total SNPs) inside the segment (to correct for un-phased SNPs).

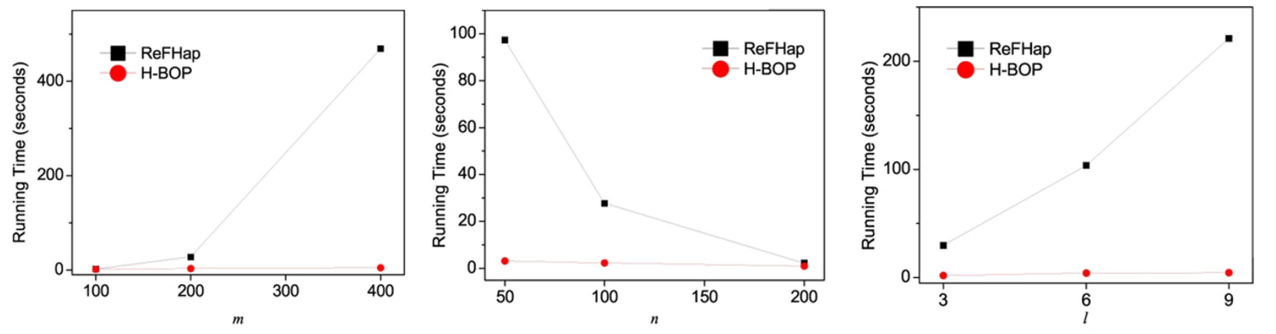
(4) Sort segments from the largest to the smallest adjusted span.

(5) Traverse the list and count the number of phased SNPs. When the count is more than a half of the total number of SNPs, the adjust span of the current segment is QAN50.

Clearly, algorithms with larger QAN50 values are more desirable. The QAN50 values of H-BOP and ReFHap on the above real data set are 114,261.09 and 113,831.67, respectively.

Conclusions

Haplotyping is regarded as one of the hardest challenges in personal genome sequencing. Though some single molecule sequencing technologies have been developed, they are still too expensive and labor-consuming. Computer algorithms are widely used to reconstruct haplotypes in personal genome sequencing. SIH uses computer algorithms to build a pair of haplotypes from an individual's aligned DNA sequence fragments. There are many different combinatorial optimization models for SIH, among which MEC is the most popular and MFC is the most recent introduction [16]. In this paper, we combine the two quality measures "errors corrected" and "fragments cut" used in MEC and MFC and introduce a new model BOP. We design a heuristic dynamic programming



(a) The number of fragments varies (b) The haplotype length varies (c) The average fragment length varies

Figure 3 Running time of ReFHap and H-BOP. (a) The number of fragments *m* varies with *n* = 100 and *l* = 3. (b) The haplotype length *n* varies with *m* = 200 and *l* = 3. (c) The average fragment length *l* varies with *m* = 200 and *n* = 100.

algorithm H-BOP to solve the model. By setting appropriate parameters of the algorithm, H-BOP is accurate and fast. Extensive simulation experiments show that H-BOP is generally faster and more accurate in assembling haplotypes than a recent state-of-art algorithm ReFHap. When the average fragment length is small and sequencing error rate is relatively high, H-BOP is significantly more accurate than ReFHap. The running time of H-BOP increases linearly as the average call coverage c increases, and H-BOP runs much faster than ReFHap when c is large. The test on a real data set also shows that H-BOP is superior to ReFHap considering both completeness and quality of the reconstructed haplotypes.

Methods

Formulation and problem

With the input of aligned DNA sequence fragments derived from a pair of chromosomes, SIH tries to reconstruct a pair of haplotypes of their underlying chromosomes. If there are no sequencing errors and for any two consecutive (but not necessarily adjacent) heterozygous SNP loci, there is at least one fragment covering both, we can easily determine the linkage relationship between two consecutive heterozygous SNPs and thus SIH is easy. However, sequencing errors are unavoidable which makes the problem complicated. In the following, we first introduce some notations and definitions similar to those in [15,16,20,23], and then propose a new optimization model.

Since we only consider the alleles at SNP loci in the SIH problem, the input aligned fragments are encoded as an $m \times n$ SNP matrix M [15,16,25], where m is the number of fragments and n the number of SNP loci. An entry at the ith row and the jth column of M is denoted as $M[i, j]$. $M[i, j]$ takes a value from $\{0, 1, -\}$, where '0' (or '1') encodes the major allele (or the minor allele, respectively) in the population and '–' represents an unknown allele. As in previous work [16,25], we assume that all SNP loci are heterozygous and every fragment covers at least two heterozygous SNP loci. If this is not the case, a simple preprocessing as in [22] and [29] can be used to remove homozygous loci and fragments covering only one heterozygous SNP locus. In the remainder of the section, the ith row of M is equivalent to the ith fragment and the j column of M is equivalent to the jth SNP locus without explicit specification for briefness.

Let $a, b \in \{0, 1, -\}$ and define

$$c(a,b) = \begin{cases} 1, \text{ if } a, b \neq - \text{and } \alpha \neq b; \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

Given an $m \times n$ SNP matrix M, let the underlying haplotype pair be $\mathcal{H} = (H_1, H_2)$. The allele at the jth SNP locus of H_1 (or H_2) is denoted as $H_1[j]$ (or $H_2[j]$, respectively). Notice that since all SNP loci of interest are heterozygous, for any $j \in \{1, ..., n\}$, $H_1[j] \neq H_2[j]$, *i.e.* when $H_1[j] = 0$, $H_2[j] = 1$ and when $H_1[j] = 1$, $H_2[j] = 0$. Let f_i denote the ith fragment (*i.e.* the ith row of M). $c(M[i_1, j], M[i_2, j]) = 1$ means that f_{i_1} and f_{i_2}*conflict* at SNP locus j (*i.e.* column j), and that if both fragments come from the same chromosome, either $M[i_1, j]$ or $M[i_2, j]$ is a sequencing error. Similarly, $c(M[i, j], H_p[j]) = 1$ ($p = 1$ or 2) means that f_i and H_p *conflict* at SNP locus j, and that if f_i comes from the chromosome with haplotype H_p, $M[i, j]$ must be a sequencing error.

Let $s_c(H_p, f_i) = \sum_{j=1,\dots,n} c(M[i, j], H_p[j])$ and $s_c(\mathcal{H}, f_i) = \min(s_c(H_1, f_i), s_c(H_2, f_i))$ and define

$$s_c(\mathcal{H}, M) = \sum_{i=1,\dots,m} (s_c(\mathcal{H}, f_i)). \tag{2}$$

If the real haplotype pair is $\mathcal{H}$, it is easy to verify that the number of sequencing errors in the input fragments is at least $s_c(\mathcal{H}, M)$, which we call the *errors corrected measure*. Based on the principle of parsimony, it is a natural optimization objective that to minimize the number of errors and hence Minimum Error Correction [20,23,24] is the most popular model for SIH.

Minimum Error Correction (MEC): Given an $m \times n$ SNP matrix M, find a haplotype pair $\mathcal{H}$ such that $s_c(\mathcal{H}, M)$ is minimized.

Based on the underlying haplotype pair $\mathcal{H} = (H_1, H_2)$, it is easy to partition the fragments into two groups G_1, G_2 according to the following rule: For each fragment f_i, if $s_c(H_1, f_i) < s_c(H_2, f_i)$, add f_i to G_1, otherwise add f_i to G_2. Let $\mathbb{P}_{\mathcal{H}}$ denote the partition (G_1, G_2) obtained by the above rule. A partition $\mathcal{P} = (G_1, G_2)$ of a fragment set R is encoded as a map such that $\mathcal{P}(f) = 0$ (or 1) if the fragment f in R belongs to G_1 (or G_2, respectively).

Conversely, given a partition $\mathcal{P} = (G_1, G_2)$, a haplotype pair $\mathcal{H} = (H_1, H_2)$ can be constructed as follows. Let $N_{g,v}[j]$ be the number of fragments of G_g whose allele at the jth locus is v for $g = 1, 2$ and $v = 0, 1$. For each SNP locus j, if $N_{1,1}[j] + N_{2,0}[j] \leq N_{1,0}[j] + N_{2,1}[j]$, $H_1[j] = 0$ and $H_2[j] = 1$; otherwise, $H_1[j] = 1$ and $H_2[j] = 0$. Let $\mathbb{H}_{\mathcal{P}}$ denote the haplotype pair obtained by the above method.

Theorem 1 *Given an SNP matrix M and a haplotype pair* $\mathcal{H}, s_c(\mathcal{H}, M) \geq s_c(\mathbb{H}_{\mathbb{P}_{\mathcal{H}}}, M)$.

There is another formulation of MEC equivalent to the above one: Given an SNP matrix M, find a partition $\mathcal{P}$ of the rows in M such that $s_c(\mathbb{H}_{\mathcal{P}}, M)$ is minimized. In the following, $s_c(\mathbb{H}_{\mathcal{P}}, M)$ is called the *errors corrected measure of* $\mathcal{P}$. While MEC aims to find a partition such that the conflict between the fragments in the same group is minimized, Maximum Fragments Cut (MFC) [16] aims to find a partition such that the conflict between the fragments in G_1 and the fragments in G_2 is maximized.

Let $a, b \in \{0, 1, -\}$ and define

$$d(a, b) = \begin{cases} -1, \text{ if } a, b \neq - \text{and } a = b; \\ 1, \quad \text{if } a, b \neq - \text{and } a \neq b; \\ 0, \quad \text{otherwise.} \end{cases} \tag{3}$$

Let f_{i_1} and f_{i_2} be two rows of M and define $d(f_{i_1}, f_{i_2}) = \sum_{j=1,\ldots,n} (d(M[i_1, j], M[i_2, j]))$. As in [16], we convert an $m \times n$ SNP matrix M into a weighted complete graph $\mathcal{G} = (V, E)$, where V is the set of rows in M and the weight of the edge between two rows f_{i_1} and f_{i_2} is $d(f_{i_1}, f_{i_2})$. Therefore, a partition $\mathcal{P}$ of the rows in M corresponds a cut of $\mathcal{G}$. Given an SNP matrix M and a partition $\mathcal{P}$, the *fragments cut measure* is defined as:

$$s_d(\mathcal{P}, M) = \sum_{\mathcal{P}(f_{i_1})=1, \mathcal{P}(f_{i_2})=2} d(f_{i_1}, f_{i_2}). \tag{4}$$

Maximum Fragments Cut (MFC) [16]: Given an SNP matrix M, find a partition $\mathcal{P}$ of the rows of M such that $s_d(\mathcal{P}, M)$ is maximized.

To take into account both the conflict between the two groups and the conflict between the fragments within the same group, we introduce a new score combining the above errors corrected measure and the fragments cut measure. Given a partition $\mathcal{P}$ of the rows of M, define the *partition score* as

$$s_p(\mathcal{P}, M) = s_c(\mathbb{H}_{\mathcal{P}}, M) - w s_d(\mathcal{P}, M), \tag{5}$$

where w is a weight factor that is used to adjust the weight of the fragments cut measure. In the following we propose a new optimization model for the SIH problem.

Balanced Optimal Partition (BOP): Given an SNP matrix M, find a partition $\mathcal{P}$ of the rows in M such that $s_p(\mathcal{P}, M)$ is minimized. A solution to *BOP* of M is denoted by *BOP(M)*, *i.e.* a partition with the minimum partition score.

Note that when $w = 0$, BOP becomes MEC which has been proved NP-hard [24] and APX-hard [26]. Therefore, BOP is NP-hard and APX-hard.

Algorithm

Given an $m \times n$ SNP matrix M, there are 2^{m-1} different partitions of m rows in M. Therefore, when m is large, it is impractical to enumerate all possible partitions and choose one with the minimum partition score. To solve the BOP model of M efficiently, we propose a dynamic programming algorithm in the subsection. We first consider the first row of M, then the first two rows and so on until we have considered all rows of M.

We need some definitions and notations. Let M [1..i, :] denote the SNP matrix consisting of only the first i rows of M. The first and last columns at which the ith row of M takes non '–' values are denoted by $l(i)$ and $r(i)$, respectively. For a column j, if $l(i) \leq j \leq r(i)$, row *i spans* column j. In the following, we assume that all the rows of M are sorted such that if $i_1 < i_2$, $l(i_1) < l(i_2)$ or $l(i_1) = l(i_2) \wedge r(i_1) \leq r(i_2)$. Let $R(i)$ denote the row set containing the rows in $M[1..i, :]$ that span column $l(i)$.

Let $\mathcal{P}$ be a partition of a row set R and $\mathcal{P}'$ a partition of a subset R' of R. If for every row $i \in R'$, $\mathcal{P}(i) = \mathcal{P}'(i)$, $\mathcal{P}'$ is called the *projection* of $\mathcal{P}$ on R' and $\mathcal{P}$ is called an *extension* of $\mathcal{P}'$ on R. Fix a row i and let $\mathcal{P}$ be a partition of $R(i)$. $\mathcal{P}'$ is an *optimal extension* of $\mathcal{P}$, if the following conditions hold: (1) $\mathcal{P}'$ is an extension of $\mathcal{P}$ on the row set $R = \{1, ..., i\}$; (2) for any possible extension $\mathcal{P}''$ of $\mathcal{P}$ on R, $s_p(\mathcal{P}', M[1..i, :]) \leq s_p(\mathcal{P}'', M[1..i, :])$.

Given a partition $\mathcal{P}$ of $R(i)$, let $\varepsilon_i(\mathcal{P})$ denote an optimal extension of $\mathcal{P}$. We call $s_p(\mathcal{E}_i(\mathcal{P}), M[1..i, :])$ the partition score of $\mathcal{P}$ and denote it as $s_p^i(\mathcal{P})$ for briefness.

Theorem 2 *For an* $m \times n$ *SNP matrix M, let* $\mathcal{P}$*be a partition of* $R(m)$. $\mathcal{E}_m(\mathcal{P})$ is a solution to BOP of M if the following condition holds: for any possible partition $\mathcal{P}'$*of* $R(i)$, $s_p^m(\mathcal{P}) \leq s_p^m(\mathcal{P}')$.

Consider the submatrix containing only the first row of M. Since $R(1)$ contains only one row, *i.e.* $R(1) = \{1\}$, there are only two possible partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ of $R(1)$ ($\mathcal{P}_1$ and $\mathcal{P}_2$ are in fact equivalent): $\mathcal{P}_1(1) = 0$ iff $\mathcal{P}_2(1) = 1$. It is easy to verify that the following equalities hold for $i = 1; 2$.

$$\mathcal{E}_1(\mathcal{P}_i) = \mathcal{P}_i; s_p^1(\mathcal{P}_i) = 0. \tag{6}$$

After $\mathcal{E}_i(\mathcal{P})$ and $s_p^i(\mathcal{P})$ have been calculated for every possible partition $\mathcal{P}$ of $R(i)$, we consider the submatrix containing the first $i + 1$ rows of M. Let $R_c(i, i + 1) = R(i) \cap R(i + 1)$, and we calculate $\mathcal{E}_{i+1}(\mathcal{P}')$ and $s_p^{i+1}(\mathcal{P}')$ for every possible partition $\mathcal{P}'$ of $R(i + 1)$ according to the following method. Let q be the number of the rows in $R(i)$ but not in $R_c(i, i + 1)$, *i.e.* $q = |R(i) - R_c(i, i + 1)|$. For a partition $\mathcal{P}''$ of $R_c(i, i + 1)$, there are 2^q distinct extensions of $\mathcal{P}''$ on $R(i)$. Suppose $\mathcal{P}_m$ is the one whose partition score is the minimum among all 2^q extensions. Then $\mathcal{E}_i(\mathcal{P}'')$ and $s_p^i(\mathcal{P}'')$ can be computed with the following equations:

$$\mathcal{E}_i(\mathcal{P}'') = \mathcal{E}_i(\mathcal{P}_m); s_p^i(\mathcal{P}'') = s_p^i(\mathcal{P}_m). \tag{7}$$

Since the rows in M are sorted, it is easy to verify that $R(i + 1) = R_c(i, i + 1) \cup \{i + 1\}$, and that the number of all possible distinct partitions of $R(i + 1)$ is two times that of $R_c(i, i + 1)$. For each partition $\mathcal{P}''$ of $R_c(i, i + 1)$, there are two distinct corresponding partitions $\mathcal{P}_1'$ and $\mathcal{P}_2'$ of $R(i + 1)$: for each $l \in R_c(i, i + 1)$, $\mathcal{P}_1'(l) = \mathcal{P}_2'(l) = \mathcal{P}''(l); \mathcal{P}_1'(i) = 0$, but $\mathcal{P}_2'(i) = 1$. Optimal extensions of $\mathcal{P}_1'$ and $\mathcal{P}_2'$ and their partition scores can be calculated with the following equations:

```
Algorithm H-BOP
input: an $m \times n$ SNP matrix $M$, a weight factor $w$, an integer $k$.
    // all columns of $M$ are heterozygous
    // all rows of $M$ span at least two columns
    // and the rows of $M$ are sorted
output: a solution to BOP of $M$.
1. initiation:
1.1. $i = 1, R(1) = \{1\}, q = 1, z = k$;
    if $2^q < k$ then $z = 2^q$;
    initiate a heap $H$ of size $z$;
    // a partition is encoded by a binary number
1.2. for $P = 0, ..., 2^q - 1$ do    // Eq.(6)
        $\mathcal{E}_1(P) = P, s_p^1(P) = 0$;
        $H$.insert($\mathcal{P}, E_1(P), s_p^1(P)$);
2. computation of $R_c(i, i+1)$:
2.1. $R_c(i, i+1) = \varnothing, d = 0$;
2.2. for each row $f \in R(i)$ do
        if $l(i+1) \leq r(f)$ then
            $R_c(i, i+1) = R_c(i, i+1) \cup \{f\}, d++$;
3. projection on $R_c(i, i+1)$:
3.1. initiate a heap $H'$ of size $z$;
3.2. for each element $L$ in $H$ do    //Eq.(7)
3.2.1. $L.P$ = the projection of $L.P$ on $R_c(i, i+1)$;
3.2.2. if $H'$ has a $L'$ such that $L'.P = L.P$ then
        {    if $L'.s > L.s$ then replace $L'$ with $L$;    }
       else $H'$.insert($L$);
4. extension on $R(i+1)$:
4.1. $R(i+1) = R_c(i, i+1) \cup \{i+1\}$; $q = | R(i+1) |$; $z = k$;
4.2. if $2^q < k$ then $z = 2^q$;
4.3. initiate a heap $H$ of size $z$;
4.4. for each element $L$ in $H'$ do //Eq.(8) – Eq.(11)
        $P'_1 = L.P, \mathcal{E}_{i+1}(P'_1) = \mathcal{E}(L.P)$;
        $s_p^{i+1}(P'_1) = s_p(P) + DeltaEC(i, P'_1)$
                              $- w \times DeltaFC(i, P'_1)$;
        $H$.insert($\mathcal{P}'_1, \mathcal{E}_{i+1}(P'_1), s_p^{i+1}(P'_1)$);
        $P'_2 = P + 2^d, \mathcal{E}_{i+1}(P'_2) = \mathcal{E}(P) + 2^i$;
        $s_p^{i+1}(P'_2) = s_p(P) + DeltaEC(i, P'_2)$
                              $- w \times DeltaFC(i, P'_2)$;
        $H$.insert($\mathcal{P}'_2, \mathcal{E}_{i+1}(P'_2), s_p^{i+1}(P'_2)$);
5. next iteration:
    if $i < m - 1$ then $i++$, go to Step 2;
6. finish: //Eq.(12)
6.1. let $L$ be the minimum element in $H$;
6.3. $BOP(M) = \mathcal{E}_m(P)$;
```

Figure 4 H-BOP Algorithm.

$$\mathcal{E}_{i+1}(\mathcal{P}_1')(l) = \begin{cases} \mathcal{E}_i(\mathcal{P}'')(l), & 1 \le l < i; \\ \mathcal{P}'_1(l), & l = i. \end{cases} \tag{8}$$

$$\mathcal{E}_{i+1}(\mathcal{P}_2')(l) = \begin{cases} \mathcal{E}_i(\mathcal{P}'')(l), & 1 \le l < i; \\ \mathcal{P}'_2(l), & l = i. \end{cases} \tag{9}$$

$$s_p^{i+1}(\mathcal{P}_1') = s_p^i(\mathcal{P}'') + \delta_c(\mathcal{P}_1') - w\delta_d(\mathcal{P}_1'); \tag{10}$$

$$s_p^{i+1}(\mathcal{P}_2') = s_p^i(\mathcal{P}'') + \delta_c(\mathcal{P}_2') - w\delta_d(\mathcal{P}_2'). \tag{11}$$

In Equation (10) (or 11), $\delta_c(\mathcal{P}_1')$ (or $\delta_c(\mathcal{P}_2')$) is the difference between the errors corrected measures of $\mathcal{E}_{i+1}(\mathcal{P}_1')$ (or $\mathcal{E}_{i+1}(\mathcal{P}_2')$) and $\mathcal{E}_i(\mathcal{P}'')$; and $\delta_d(\mathcal{P}_1')$ (or $\delta_d(\mathcal{P}_2')$) is the difference between the fragments cut measures of $\mathcal{E}_{i+1}(\mathcal{P}_1')$ (or $\mathcal{E}_{i+1}(\mathcal{P}_2')$). The values $\delta_c(\mathcal{P})$ and $\delta_d(\mathcal{P})$ are calculated by calling the following functions $DeltaEC(i, \mathcal{P})$ and $DeltaFC(i, \mathcal{P})$, respectively.

```
DeltaEC(i, $\mathcal{P}$)
{ $\delta = 0$;
  **for** $j = l(i + 1), ..., r(i + 1)$ **do**
  { **if** $M[i + 1, j]$ = = '-' **then** continue;
    $N_{1,0} = N_{2,0} = N_{1,1} = N_{2,1} = 0$;
    **for** each row $l \in R_c(i, i + 1)$ **do**
    { **if** $M[l, j]$ = = '-' **then** continue;
      $v = M[l, j]$, $g = \mathcal{P}(l)$, $N_{g,v}$ + +; }
    **if** $N_{1,1} + N_{2,0} \le N_{1,0} + N_{2,1}$ **then**
    { $\delta = \delta - (N_{1,1} + N_{2,0})$; }
    **else** { $\delta = \delta - (N_{1,0} + N_{2,1})$; }
    $v = M[i + 1, j]$, $g = \mathcal{P}(i + 1)$, $N_{g,v}$ + +;
    **if** $N_{1,1} + N_{2,0} \le N_{1,0} + N_{2,1}$ **then**
    { $\delta = \delta + (N_{1,1} + N_{2,0})$; }
    **else** { $\delta = \delta + (N_{1,0} + N_{2,1})$; }
  }
  **return** $\delta$; }
DeltaFC(i, $\mathcal{P}$)
{ $\delta = 0$, $g_0 = \mathcal{P}(i + 1)$;
  **for** $j = l(i + 1), ..., r(i + 1)$ **do**
  { **if** $M[i + 1, j]$ = = '-' **then** continue;
    **for** each row $l \in R_c(i, i + 1)$ **do**
    { **if** $M[l, j]$ = = '-' **then** continue;
      $v = M[l, j]$, $g = \mathcal{P}(l)$;
      **if** $g == g_0$ **then** continue;
      **if** $v == M[i + 1, j]$ **then** $\delta$ - -;
  **else** $\delta$ + +; }
  }
  **return** $\delta$; }
```

When $\mathcal{E}_m(\mathcal{P})$ and $s_p^m(\mathcal{P})$ are known for every possible partition $\mathcal{P}$ of $R(m)$, a solution to BOP of M is easily obtained by using the following formula:

$$BOP(M) = \mathcal{E}_m(\mathcal{P}) | s_p^m(\mathcal{P}) \text{ is minimum.} \tag{12}$$

Based on the above equations, we can construct an exact dynamic programming algorithm for BOP. However, the complexity of this exact algorithm increases exponentially with the number of rows in $R(i)$, which implies that the algorithm is impractical when the call coverage is large. Let $\mathcal{P}_i^*$ be the projection on $R(i)$ of the global optimal partition of M. If the partition score of $\mathcal{P}_i^*$ is among the k smallest ones of all possible partitions of $R(i)$, we only need to compute k partitions of $R(i)$ whose partition scores are the smallest in each iteration without losing the global optimal partition at the end. Based on the above idea, we propose a heuristic algorithm H-BOP whose pseudo-code is shown in Figure 4. In the algorithm, a partition $\mathcal{P}$ of a row set is encoded by a binary number P, and $\mathcal{P}(i)$ is represented by the ith bit of P. Therefore, the number set $\{0, ..., 2^q - 1\}$ encodes all possible partitions of a row set containing q rows (in fact, there are only 2^{q-1} different partitions, and in our implement of H-BOP, we use a binary number of $q - 1$ bits to encode a partition to save time and space). In each iteration of H-BOP, for each row i we maintain a binary max heap H to store the candidate partitions of $R(i)$ whose partition scores are among the k smallest. The heap H can store at most k elements, and each element L of H contains a partition $\mathcal{P}$, $\mathcal{E}_i(P)$ and $s_p^i(P)$, which are denoted as $L.P$, $L.\mathcal{E}$, and $L.s$ respectively. The value $s_p^i(P)$ of each element in the heap is larger than or equal to those of its two children. When the number of elements of H is smaller than its expected size (*i.e.* k), and a new element L arrives, the element is inserted into H and H is adjusted to maintain the max heap property. When the number of elements of H reaches k, L is compared with the root r of H. If $L.s < r.s$, the root is replaced by L and H is adjusted accordingly; otherwise, the new element is discarded. The above operation is denoted by H.insert(L).

The time complexity of H-BOP is $O(mkk_1k_2)$, and the space complexity is $O(mk_1 + mk)$, where $k_1 = \max_{i=1,...,m}(r(i) - l(i) + 1)$ and $k_2 = \max_{i=1,...,m}(|R(i)|)$.

Acknowledgements
This research was supported by the National Natural Science Foundation of China under Grant Nos. 61070145 and 61232001.
This article has been published as part of *BMC Systems Biology* Volume 6 Supplement 2, 2012: Proceedings of the 23rd International Conference on Genome Informatics (GIW 2012). The full contents of the supplement are available online at http://www.biomedcentral.com/bmcsystbiol/supplements/6/S2.

Author details
[1]College of Physics and Information Science, Hunan Normal University, Changsha 410081, P. R. China. [2]School of Information Science and Engineering, Central South University, Changsha 410083, P. R. China. [3]Department of Computer Science and Engineering, University of California, Riverside, CA 92521, USA.

Authors' contributions

MX designed the algorithm, performed the computational experiments, and drafted the manuscript. TJ and JW helped to draft and polish the manuscript. All authors read and approved the manuscript.

Competing interests

The authors declare that they have no competing interests.

Published: 12 December 2012

References

1. Levy S, *et al*: **The diploid genome sequence of an individual human.** *PLoS Biology* 2007, **5(10)**:e254-e254.
2. Tewhey R, Bansal V, Torkamani A, Topol EJ, Schork NJ: **The importance of phase information for human genomics.** *Nat Rev Genet* 2011, **12(3)**:215-23.
3. Casey JP, *et al*: **A novel approach of homozygous haplotype sharing identifies candidate genes in autism spectrum disorder.** *Hum Genet* 2012, **131(4)**:565-79.
4. Fan HC, Wang J, Potanina A, Quake SR: **Whole-genome molecular haplotyping of single cells.** *Nat Biotechnol* 2011, **29**:51-7.
5. The 1000 Genomes Project Consortium: **A map of human genome variation from population-scale sequencing.** *Nature* 2010, **467(7319)**:1061-73.
6. Zhang XS, Wang RS, Wu LY, Chen L: **Models and algorithms for haplotyping problem.** *Current Bioinformatics* 2006, **1**:105-114.
7. Xie M, Wang J, Chen J, Wu J, Liu X: **Computational models and algorithms for the single individual haplotyping problem.** *Current Bioinformatics* 2010, **5**:18-28.
8. Browning SR, Browning BL: **Haplotype phasing: existing methods and new developments.** *Nat Rev Genet* 2011, **12(10)**:703-14.
9. The International HapMap Consortium: **A haplotype map of the human genome.** *Nature* 2005, **437(7063)**:1299-1320.
10. Kitzman JO, *et al*: **Haplotype-resolved genome sequencing of a Gujarati Indian individual.** *Nat Biotechnol* 2011, **29**:59-63.
11. Suk EK, *et al*: **A comprehensively molecular haplotype-resolved genome of a European individual.** *Genome Res* 2011, **21(10)**:1672-85.
12. Duitama J, *et al*: **Fosmid-based whole genome haplotyping of a HapMap trio child: evaluation of Single Individual Haplotyping techniques.** *Nucleic Acids Res* 2012, **40(5)**:2041-53.
13. Lancia G, Bafna V, Istrail S, Lippert R, Schwartz R: **SNPs problems, complexity and algorithms.** In *Proc. Ann. European Symp. on Algorithms (ESA), Volume 2161 of Lecture Notes in Computer Science.* Berlin/Heidelberg: Springer;auf der Heide FM 2001:182-193.
14. Geraci F: **A comparison of several algorithms for the single individual SNP haplotyping reconstruction problem.** *Bioinformatics* 2010, **26(18)**:2217-25.
15. Xie M, Wang J, Chen J: **A model of higher accuracy for the individual haplotyping problem based on weighted SNP fragments and genotype with errors.** *Bioinformatics* 2008, **24(13)**:i105-13.
16. Duitama J, Huebsch T, McEwen G, Suk EK, Hoehe MR: **ReFHap: a reliable and fast algorithm for single individual haplotyping.** *Proceedings of the First ACM international Conference on Bioinformatics and Computational Biology* Niagara Falls, New York: ACM; 2010, 160-169.
17. He D, Choi A, Pipatsrisawat K, Darwiche A, Eskin E: **Optimal algorithms for haplotype assembly from whole-genome sequence data.** *Bioinformatics* 2010, **26(12)**:i183-90.
18. Bansal V, Bafna V: **HapCUT: an efficient and accurate algorithm for the haplotype assembly problem.** *Bioinformatics* 2008, **24(16)**:i153-9.
19. Wang J, Xie M, Chen J: **A practical exact algorithm for the individual haplotyping problem MEC/GI.** *Algorithmica* 2010, **56(3)**:283-296.
20. Xie M, Wang J, Chen J: **A practical parameterised algorithm for the individual haplotyping problem MLF.** *Mathematical Structures in Computer Science* 2010, **20(5)**:851-863.
21. Bafna V, Istrail S, Lancia G, Rizzi R: **Polynomial and APX-hard cases of the individual haplotyping problem.** *Theoretical Computer Science* 2005, **335**:109-125.
22. Panconesi A, Sozio M: **Fast hare: a fast heuristic for single individual SNP haplotype reconstruction.** In *Proc. WABI, Volume 3240 of Lecture Notes in Computer Science.* Berlin/Heidelberg: Springer;Jonassen I, Kim J 2004:266-277.
23. Wang RS, Wu LY, Li ZP, Zhang XS: **Haplotype reconstruction from SNP fragments by minimum error correction.** *Bioinformatics* 2005, **21(10)**:2456-2462.
24. Lippert R, Schwartz R, Istrail S: **Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem.** *Brief. Bioinform* 2002, **3**:1-9.
25. Genovese LM, Geraci F, Pellegrini M: **SpeedHap: an accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate and low coverage.** *IEEE/ACM Trans Comput Biol Bioinform* 2008, **5(4)**:492-502.
26. Cilibrasi R, van Iersel L, Kelk S, Tromp J: **The complexity of the single individual SNP haplotyping problem.** *Algorithmica* 2007, **49**:13-36.
27. **SIH.** [http://www.molgen.mpg.de/~genetic-variation/SIH/].
28. Lo C, Bashir A, Bansal V, Bafna V: **Strobe sequence design for haplotype assembly.** *BMC Bioinformatics* 2011, **12(Suppl 1)**:S24.
29. Xie M, Wang J: **An improved (and practical) parameterized algorithm for the individual haplotyping problem MFR with mate-pairs.** *Algorithmica* 2008, **52(2)**:250-266.

doi:10.1186/1752-0509-6-S2-S8
Cite this article as: Xie *et al.*: **A fast and accurate algorithm for single individual haplotyping.** *BMC Systems Biology* 2012 **6**(Suppl 2):S8.