Hindawi Publishing Corporation
ISRN Bioinformatics
Volume 2013, Article ID 291741, 10 pages
http://dx.doi.org/10.1155/2013/291741

Research Article

HMEC: A Heuristic Algorithm for Individual Haplotyping with Minimum Error Correction

Md. Shamsuzzoha Bayzid,[1] Md. Maksudul Alam,[2] Abdullah Mueen,[3] and Md. Saidur Rahman[4]

[1] *Department of Computer Science, University of Texas at Austin, Austin, TX 78712, USA*
[2] *Department of Computer Science, Virginia Tech, Blacksburg, VA 24060, USA*
[3] *Department of Computer Science and Engineering, Univerity of California, Riverside, CA 92521, USA*
[4] *Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka 1000, Bangladesh*

Correspondence should be addressed to Md. Shamsuzzoha Bayzid; shams.bayzid@gmail.com

Received 26 November 2012; Accepted 12 December 2012

Academic Editors: A. Bolshoy and A. Torkamani

Haplotype is a pattern of single nucleotide polymorphisms (SNPs) on a single chromosome. Constructing a pair of haplotypes from aligned and overlapping but intermixed and erroneous fragments of the chromosomal sequences is a nontrivial problem. Minimum error correction approach aims to minimize the number of errors to be corrected so that the pair of haplotypes can be constructed through consensus of the fragments. We give a heuristic algorithm (HMEC) that searches through alternative solutions using a gain measure and stops whenever no better solution can be achieved. Time complexity of each iteration is $O(m^3k)$ for an $m \times k$ SNP matrix where m and k are the number of fragments (number of rows) and number of SNP sites (number of columns), respectively, in an SNP matrix. Alternative gain measure is also given to reduce running time. We have compared our algorithm with other methods in terms of accuracy and running time on both simulated and real data, and our extensive experimental results indicate the superiority of our algorithm over others.

1. Introduction

A single DNA molecule is a long chain of nucleotides (base pairs). There are four such nucleotides which are represented by the set of symbols $\{A, T, G, C\}$. It is generally accepted that genomes of two humans are almost 99% identical at DNA level. However, at certain specific sites, variation is observed across the human population which is commonly known as *"single nucleotide polymorphism"* and abbreviated as "SNP" [1]. The nucleotide involved in a SNP site is called *allele*. If a SNP site can have only two nucleotides, it is called *biallelic*. If it can have more than two alleles it is called a *multiallelic* SNP [2]. From now on, we will consider the simplest case where only bi-allelic SNPs occur in a specific pair of DNA.

The single nucleotide polymorphism (SNP) is believed to be the most widespread form of genetic variation [3]. The sequence of all SNPs in a given chromosome is called haplotype. Haplotyping an individual deals with determining a pair of haplotypes, one for each copy of a given chromosome. A chromosome is a complicated structure of a DNA molecule bound by proteins. This pair of haplotypes completely define the SNP fingerprints of an individual for a specific pair of chromosomes. Given the two sequences of bases, haplotyping is straight forward and just needs to iterate through both the sequences and remove all the common alleles from them. But haplotyping becomes difficult when we want to construct haplotypes from sequencing data for higher reliability. Sequencing data for a genome does not contain the complete sequences of bases for a specific chromosome, rather it provides a set of fragments of arbitrary length for the whole genome. Therefore, the actual problem of haplotyping is to find two haplotypes from the set of overlapping fragments

of both the chromosomes, where fragments might contain errors and it is not known which copy of the chromosome a particular fragment belongs to.

The problem of haplotyping has been studied extensively. The general minimum error correction (MEC) problem was proved to be NP-hard [4]. It was also proved to be NP-hard even if the SNP matrix is gapless using a reduction from the MAX-CUT problem [1]. A method based on genetic algorithm has been proposed to solve this problem [5]. Several heuristic methods have also been proposed to find haplotypes efficiently. HapCUT [6] and ReFHap [7] are two of the most accurate algorithms in this regard.

In this paper, we give a heuristic algorithm for individual haplotyping based on minimum error correction. The complexity of each iteration is $O(m^3k)$ for an SNP matrix of dimension (m, k). The algorithm is inspired from the famous Fiduccia and Mattheyses (FM) algorithm for bipartitioning a hypergraph minimizing the cut size [8]. Extensive simulations indicate that HMEC outperforms the genetic algorithms of Wang et al. [5] in terms of both reconstruction rate and running time, and it has better (in most cases) or comparable accuracy and significantly smaller running time than that of HapCUT [6], which is the most accurate heuristic algorithm available. We also compared HMEC with some other algorithms such as SpeedHap [9], FastHare [10], MLF [11], 2 distance MEC [12], and SHR-3 [13] using the HapMap-based instance generator and comparison framework [14, 15].

The rest of the paper is organized as follows. In Section 2, we present some definitions and preliminary ideas. In Section 3, we present our algorithm for individual haplotyping. We describe the data structure and complexity of our algorithm in Section 4. We report on an extensive performance study evaluating HMEC with other available techniques in Section 5. Finally, we conclude in Section 6 by suggesting some future research directions. An earlier version of this paper was accepted for presentation at BMEI 2008 [16].

2. Preliminaries

In this section, we give some definitions and preliminary ideas.

Let S be the set of k bi-allelic SNP sites. Let F be the set of m fragments produced from two copies of the chromosome. Each fragment contains information of nonzero number of SNPs in S. Because the SNPs are bi-allelic, let the two possible alleles for each SNP site be 0 and 1, where they can be any two elements of the set $\{A, T, G, C\}$. Since all the nucleotides are the same at the sites other than SNP sites, we can remove these extraneous sites from all the fragments and consider the fragments as the sequences of the SNP sites only. Thus each fragment $f \in F$ is a string of symbols $\{0, 1, -\}$ of length k where "–" denotes an undetermined SNP which we call a *hole*. All the fragments can be arranged in an $m \times k$ matrix $M = \{M_{ij}\}, i = 1, \ldots, m, j = 1, \ldots k$, where row i is a fragment from F and column j is a SNP from S. This matrix is called the SNP matrix as follows

$$\begin{vmatrix} ----1101---------- \\ -----0001110101----- \\ 1101001001 1---------- \\ ---10100---010------ \\ ---------1011010101011 \\ 01011----------01011 \end{vmatrix} \quad (1)$$

The consecutive sequence of "–"s that lies between two nonhole symbols is called a *gap*. A *gapless* SNP matrix is the one that has no gap in any of the fragments. In (1), the first, second, and third rows have no gaps while each of the fourth and sixth rows has one gap.

A SNP matrix $M = \langle M_1, M_2, \ldots, M_m \rangle$ can be viewed as an ordered set of m fragments where a fragment $M_i = \langle M_{i1}, M_{i2}, \ldots, M_{ik} \rangle$ is an ordered set of k alleles. A fragment M_i is called to *cover* the jth SNP if $M_{ij} \in \{0, 1\}$ and called to *skip* the jth SNP if $M_{ij} = -$. Let M_s and M_t be two fragments. The distance between two fragments, $D(M_s, M_t)$, is defined as the number of SNPs that are covered by both of the fragments and have different alleles. Hence,

$$D(M_s, M_t) = \sum_{j=1}^{k} d(M_{sj}, M_{tj}), \quad (2)$$

where $d(x, y)$ is defined as

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

In (1), the distance between the second and the third fragment is two, as they differ in the seventh and ninth SNP sites (columns).

Two fragments M_s and M_t are said to be *conflicting* if $D(M_s, M_t) > 0$. Let $P(C_1, C_2)$ be a *partition* of M, where C_1 and C_2 are two sets of fragments taken from M so that $C_1 \bigcup C_2 = M$ and $C_1 \bigcap C_2 = \phi$ [5]. In Figure 1(b), an arbitrary partition corresponding to the SNP matrix of Figure 1(a) is shown. A SNP matrix M is an *error-free* matrix if and only if there exists a partition $P(C_1, C_2)$ of M such that for any two fragments $x, y \in C_i, i \in \{1, 2\}$, x and y are non-conflicting, that is, $D(x, y) = 0$. Such a partition is called an error-free partition. The partition in the Figure 1(b) is not error free since $D(M_1, M_2) > 0$ in C_1 and $D(M_5, M_6) > 0$ in C_2. For an error-free SNP matrix, a haplotype $H_i, i \in \{1, 2\}$ is constructed from its corresponding fragment class C_i using the following formula:

$$H_{ij} = \begin{cases} 1, & \text{if at least one fragment in } C_i \text{ has a 1 in } j\text{th SNP;} \\ 0, & \text{if at least one fragment in } C_i \text{ has a 0 in } j\text{th SNP;} \\ -, & \text{if all the fragments in } C_i \text{ skips } j\text{th SNP;} \end{cases} \quad (4)$$

where C_i is called the defining class of haplotype H_i, and H_{ij}, where $i \in \{1, 2\}$ and $j = 1, \ldots k$, denotes the jth element of the haplotype H_i.

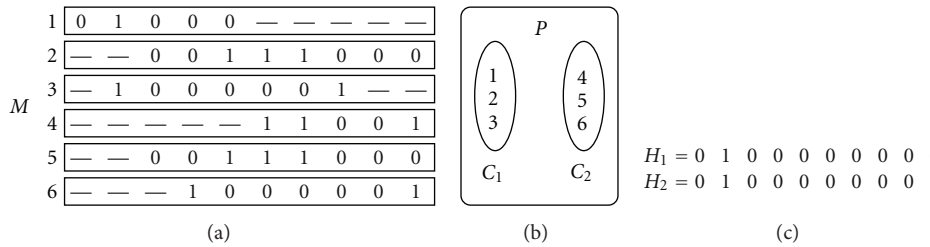

Figure 1: SNP matrix and its partition.

We now describe the general minimum error correction problem. If a matrix M is *not* error-free, there will be no error-free partition P. For such a matrix M, there will be at least one conflicting pair of fragments in each of the classes for all possible partitions. Therefore it is impossible to construct a haplotype that is non-conflicting with all the fragments in its defining class of fragments. If we are given a partition $P(C_1, C_2)$ and two haplotypes H_1 and H_2 constructed from P then the number of errors $E(P)$ that needs to be corrected can be calculated by the following formula:

$$E(P) = \sum_{i=1}^{2} \sum_{f \in C_i} D(f, H_i). \tag{5}$$

The MEC problem asks to find a partition P that minimizes the error function $E(P)$ over all such partitions of an SNP matrix M.

3. A Heuristic Algorithm

In this section, we give our heuristic algorithm based on minimum error correction which we call HMEC.

Construction of a haplotype from an erroneous class C requires correction of SNP values, that is, alleles, in the fragments. We want to minimize the number of error corrections. Therefore, for each SNP site, the haplotype should take the allele that is present in the majority of the fragments. Let $N_j^0(C)$ be the number of fragments of a collection C that have 0 in the jth SNP. Similarly, $N_j^1(C)$ defines the number of 1s [5]. Therefore, to minimize the number of errors $E(P)$ for a specific partition P, the haplotype should be constructed according to the following methodology:

$$H_{ij} = \begin{cases} 1, & \text{if } N_j^1(C_i) > N_j^0(C_i); \\ 0, & \text{if } N_j^0(C_i) \geq N_j^1(C_i) \text{ and} \\ & \quad N_j^0(C_i) \neq 0; \\ -, & \text{if } N_j^1(C_i) = N_j^0(C_i) = 0; \end{cases} \tag{6}$$

where $i \in \{1, 2\}$ and $j = 1, 2, \ldots, k$. In Figure 1(c), two haplotypes H_1 and H_2, associated with the partition P in Figure 1(b), are constructed in this way.

We will use a heuristic search to find the best partition. This algorithm starts with a current partition $P_c = P(M, \phi)$ and iteratively searches a better partition. In each iteration, the algorithm performs a sequence of transfer of fragments from their present collection to the other one so that the partition becomes less erroneous. The transfer of a fragment from one collection to the other can increase or decrease the error function $E(P)$. Let the partition before transferring a fragment f be P_p and the partition resulted is P_n. We define the gain of the transfer as $\text{Gain}(f) = E(P_p) - E(P_n)$. Figure 2 demonstrates an example calculation of the gain measure. Let $F = \langle f_i \rangle$, $i = \{1, 2, \ldots, m\}$ be an ordering of all the fragments in a partition P in such a way that fragment f_i will precede fragment f_j if all the fragments before f_i in F have already been transferred to form an intermediate partition P_i and $\text{Gain}(f_i) \geq \text{Gain}(f_j)$ over P_i. Hence, $P_1 = P_c$ at the start of each iteration. We also define the cumulative gain of a fragment ordering F up to the nth fragment as $CGain(F, n) = \sum_{i=1}^{n} \text{Gain}(f_i)$. Here $\text{Gain}(f_i) = E(P_i) - E(P_{i+1})$. The maximum cumulative gain, $MCGain(F)$ is defined as

$$MCGain(F) = \max_{1 \leq i \leq m} CGain(F, i). \tag{7}$$

In Section 4, we shall describe these terms with an example.

In each iteration, the algorithm finds the current ordering F_c of P_c and transfers only those fragments of F_c that can achieve the $MCGain(F_c)$ and the fragment that is the last to be transferred is referred as $f_{\max}$. Thus the algorithm moves from one partition to another reducing the error function by an amount of $MCGain(F_c)$. Please see Algorithm 1 for a basic description of HMEC. The algorithm continues as long as $MCGain(F_c) > 0$ and stops whenever $MCGain(F_c) \leq 0$.

4. Data Structures and Complexity

This section deals with the data structures and the complexity of our algorithm. Here we also suggest some approximation to improve the performance of our algorithm.

First, to find F_c in each iteration, the algorithm repeatedly transfers the fragment that is not transferred previously in this iteration and has maximum gain over all such fragments. To accomplish this, we use a locking mechanism. At the beginning of each iteration, all the fragments are set free. The free fragment with maximum gain is found out and tentatively transferred to the other collection. After the transfer, the fragment is locked at the new collection. This tentative transfer creates the first intermediate partition P_1. The algorithm then finds the next free fragment with maximum gain in P_1 and transfer and lock that fragment to

```
P_c = P(M, φ)
FREE_LOCKS()
CLEAR_LOG()
repeat always
   while there is an unlocked fragment in P_c do
      begin
         find a free fragment f so that Gain(f) is maximum
         transfer f to the other class
         update the haplotypes after the transfer
         LOCK(f)
         LOG_RECORD(f, Gain(f))
      end do
   FREE_LOCKS()
   check the log and find MCGain(F_c) and f_max
   if MCGain(F_c) > 0
      begin
         set new P_c by rolling back the transfers that occurred
         after the transfer of f_max
         calculate haplotypes of P_c
         CLEAR_LOG()
         continue with the the loop
   end if
   else
      begin
         terminate the algorithm and output current haplotypes
   end else
end repeat
```

Algorithm 1: The HMEC Algorithm.

create the P_2. Thus, free fragments are transferred until all the fragments are locked and the order of the transfer (F_c) is stored in the log table along with the cumulative gains (CGain). MCGain is the maximum CGain and $f_{\max}$ is the fragment corresponding to MCGain in the log table.

After finishing all such tentative transfers, P_c becomes an undefined partition. To change P_c to the desired "current" partition of the next iteration, the algorithm checks the log to find the MCGain(F_c) and $f_{\max}$, and rollback the transfer of all the fragments that were transferred after $f_{\max}$. When the rollback completes, P_c becomes ready for the next iteration.

While tentatively transferring a free fragment, the algorithm needs to find the fragment with maximum gain among the free fragments (which are not yet transferred). This requires calculating gains for each of them. To calculate the Gain(f) = $E(P_p) - E(P_n)$ for a fragment, we need to calculate two error values of two different partitions: the present intermediate partition and the next partition which will be resulted if f is transferred. Each of these error function requires calculation of two new haplotypes from their corresponding collections (see Figure 2). Although $E(P_p)$ and the haplotypes of P_p can be found from the previous transfer, calculation of $E(P_n)$ requires construction of haplotypes of P_n. Since, the difference between P_p and P_n is only one transfer, we can introduce differential calculation of haplotypes H_i^n, $i \in \{1, 2\}$ of next partition from the haplotypes of H_i^p, $i \in \{1, 2\}$ of present partition. For this purpose, the algorithm stores $N_j^1(C_i^p)$ and $N_j^0(C_i^p)$ values of the present partition. After a transfer these values will either remain same or be incremented or decremented by 1. That is why it is now possible to construct H_i^n, $i \in \{1, 2\}$ in $O(k)$ time. To compute $E(P_n)$ from the haplotypes requires $O(mk)$ time. Thus running time to compute the $E(P_n)$ as well as to compute Gain(f) is $O(mk + k)$.

For each intermediate partition $P_i, i = 1, \ldots, n$, we need to compute Gain measures for $m - i$ unlocked fragments to find the maximum one. The transfer of this fragments requires updating of $N_j^1(C_i)$ and $N_j^0(C_i)$, $i \in \{1, 2\}$ and $j = 1, 2, \ldots, k$. Hence, it also needs $O(k)$ time to run. Finally, there will be m such transfer in each iteration and maximum m rollbacks. Thus each iteration will require $O(m(m(mk + k) + k) + mk) \sim O(m^3 k)$ running time.

We now give an example illustrating a single iteration of our algorithm. Figure 3 demonstrates an example iteration of HMEC. We consider that the current partition $P_c = P_1$ is the partition given in Figure 1(b) for the SNP matrix M of Figure 1(a). All the intermediate partitions $P_i, i \in \{1, \ldots, 7\}$ are shown sequentially and the gains of each fragment over the intermediate partitions are shown on the right of each partition. The free fragment with maximum gain is marked in each intermediate partition. For example, the fragment with the maximum gain in P_2 is fragment 6 which has gain two. After each transfer, the transferred fragment is shown locked by a circle. Here, the ordering F_c of the fragments is $\langle 2, 6, 5, 1, 4, 3\rangle$ which is also the order of locking of the fragments. This order will be stored along with the CGains in

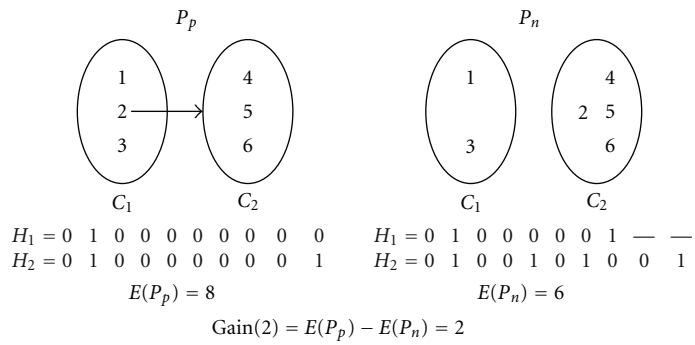

Figure 2: An example of Gain calculation.

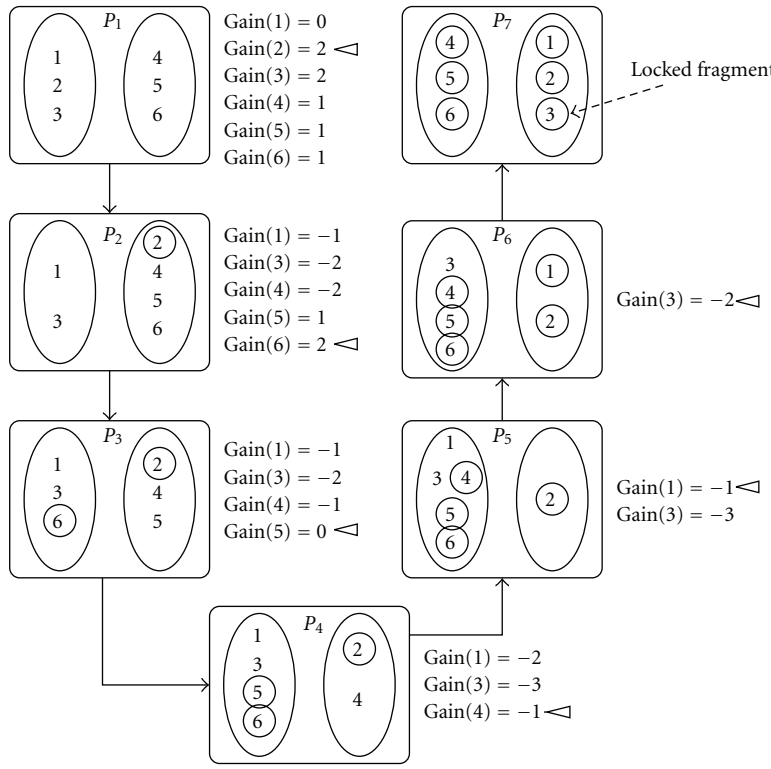

Figure 3: An example iteration of HMEC. The current partition $P_c = P_1$ is the partition given in Figure 1(b) for the SNP matrix M of Figure 1(a). Locked fragments are indicated by circles.

the log table. Figure 4 demonstrates the resulting log table of the illustrated iteration. All the tentative transfers after $f_{\max}$ have to be rolled back so that the P_3 becomes the next P_c.

We now give an approximate gain measure to make our algorithm faster. For large SNP matrix, $O(m^3k)$ running time is critical to the performance of the algorithm. We can use an approximation in the calculation of the Gain(f) by using only the fragment f and not using the $m-1$ other fragments. The approximate gain should be

$$\text{AppxGain}(f) = D\left(H_i^p, f\right) - D\left(H_j^n, f\right). \quad (8)$$

Here H_i^p is the haplotype of f's present collection C_i^p of partition P_p, and H_j^n is the haplotype of f's next collection C_j^n of partition P_n. This function ignores the effect of fragments other than f on Gain(f), but reduces the run time of gain calculation to $O(k)$. Therefore, the total run time of each iteration will be $O(m^2k)$.

5. Performance Comparison

In this section, we demonstrate the performance of our algorithm using both real biological and simulated datasets. We compared our algorithm with GMEC [5] and HapCUT [6]. We performed the simulation using the data from angiotensin-converting enzyme (ACE) [17] and public Daly set [18] to compare with GMEC [5], and used the HuRef data [19] to compare with HapCUT. We also compared HMEC

Log table

F_c	CGain
2	2
6	4
5	4
4	3
1	2
3	0

MCGain

Figure 4: The log table corresponding to the iteration illustrated in Figure 3.

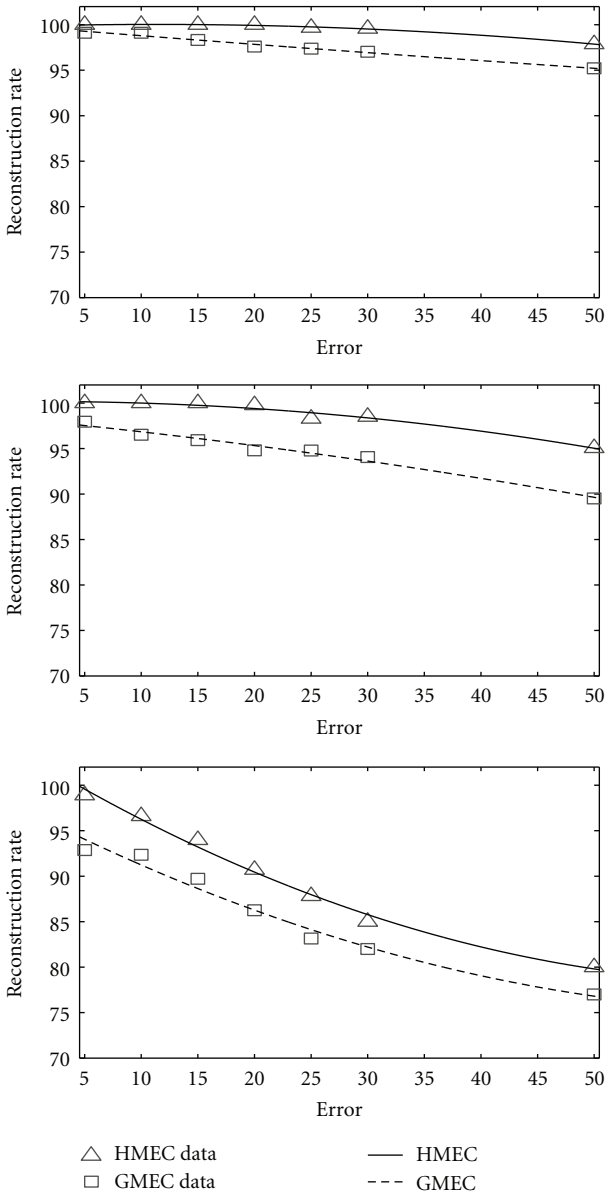

Figure 5: The comparison of HMEC and GMEC on ACE. From top to bottom, coverage = 75%, coverage = 50%, coverage = 25%.

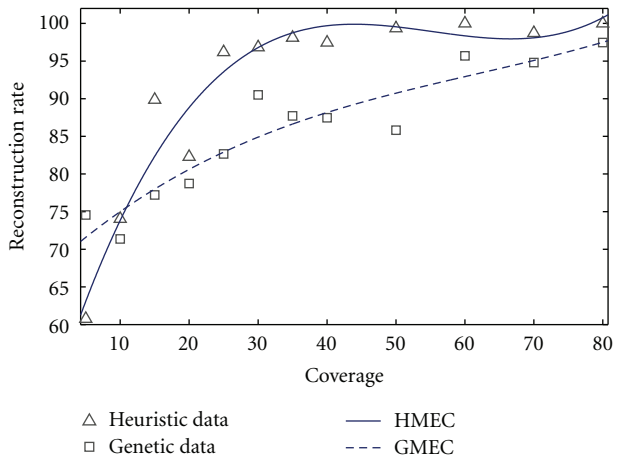

Figure 6: Reconstruction rate versus coverage value.

with some other algorithms (SpeedHap [9], FastHare [10], MLF [11], 2 distance MEC [12], SHR-3 [13]) using ReHap website interface [15], which is an HapMap-based instance generator and comparison framework [14].

5.1. Comparison with GMEC. In this section, we compare the performance of our algorithm with the genetic algorithm, which we call GMEC, described in [5].

We first sample the original haplotype pair into many fragments with different coverage and error rates. Each fragment works as a distinct sample of the same specimen. Here coverage rate indicates the percentage of the total columns of the SNP matrix that have been sampled out. The remaining slots are gaps. We then introduce some specific amount of error into these samples. The simulation was controlled in several ways. We varied the error rate while number of fragments and coverage rate were kept constant. Also, coverage was varied while number of fragments and error rate were kept constant.

Notice that the way we introduced error and controlled the coverage rate is not necessarily same as that of [5]. Therefore, the reconstruction rates of the branch and bound algorithm described in [5], which is an exact algorithm, should not be compared with those of HMEC.

5.1.1. Experiment on Angiotensin-Converting Enzyme (ACE). Angiotensin-converting enzyme catalyses the conversion of angiotensin I to the physiologically active peptide angiotensin II, which controls fluid-electrolyte balance and systematic blood pressure. Because it has a key function in the renin-angiotensin system, many association studies have been performed with DCP1 (encode angiotensin-converting anzyme) [20]. Rieder et al. completed the genomic sequencing of the DCP1 gene from 11 individuals and reported 78 SNP sites in 22 chromosomes [17].

We take six pairs of haplotypes to perform the simulation. We generate 50 fragments from each of these haplotype pairs with varying coverage and error rate. We perform the simulation for three different coverage rates (25%, 50%, and

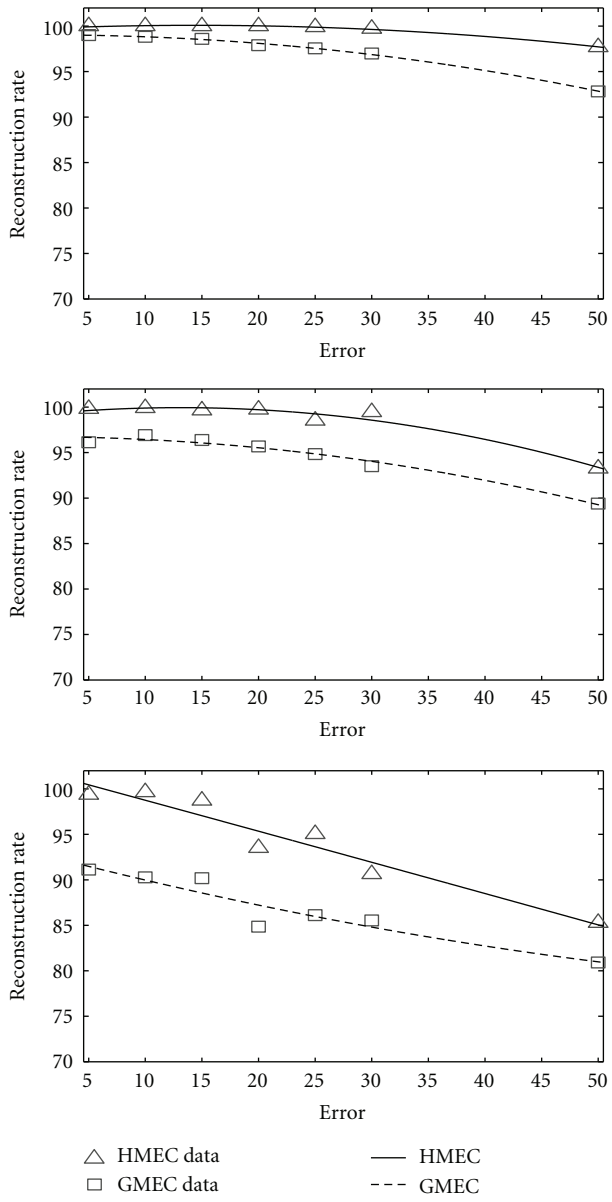

Figure 7: The comparison of HMEC and GMEC on Daly set. From top to bottom, coverage = 75%, coverage = 50%, coverage = 25%.

75%). For each of these coverage rates, we perform our simulation for different error rates such as 5%, 10%, 15%, 20%, 25%, 30%, and 50%. In every case, we compare our algorithm with GMEC. Figure 5 illustrates the comparison that bears the clear testimony to the superiority of our algorithm. For most instances, the reconstruction rate achieved by our algorithm is 100% or greater than 98%. Only for a few cases with very high error rate and low coverage value, the reconstruction rate falls below 95%. We also perform the experiment for different coverage rates while keeping the error rate constant. Figure 6 illustrates the performance of these two algorithms for various coverage values. Here also, our algorithm clearly outperforms the genetic algorithm except for very low coverage value (which is unrealistic).

Table 1: Running time of HMEC and GMEC on simulated datasets.

Length of haplotype	HMEC (sec)	GMEC (sec)
78	0.002	6.172
156	0.016	11.437
312	0.015	22.500
624	0.031	45.328
780	0.031	60.766
936	0.047	72.343

5.1.2. Simulation on Data from Chromosome 5q31. In this section, we discuss our simulation results on the data from public Daly set. Daly et al. reported a high-resolution analysis of a haplotype structure across 500 kb on chromosome 5q31 using 103 SNPs in a European derived population which consists of 129 trios [18, 20].

We performed the experiment exactly in the same way that we did for angiotensin-converting enzyme. Figure 7 demonstrates the results. Experimental results suggest that HMEC is much better than the genetic algorithm. Again, for most cases, the reconstruction rate achieved by our algorithm is 100% or greater than 98%. For every instance, our algorithm exhibits better performance than that of GMEC.

5.1.3. Experiment on Simulated Data. We used simulated data for further evaluation of HMEC. One of the very important advantages of our algorithm is that it takes very short time to reconstruct the haplotypes. Our algorithm is much faster than the GMEC. Table 1 shows the running time of HMEC and GMEC. Here, we perform the simulation by varying the length (length denotes the number of SNP sites in the haplotype pair) of the haplotypes while fixed the value of the coverage rate and the error rate at 50%. Since haplotypes with such varying lengths are not available, we rely on the simulated data. Clearly, HMEC is much faster than GMEC. For example, while HMEC can reconstruct a haplotype with 936 sites in a fraction of a second, GMEC takes 72 seconds.

5.2. Comparison with HapCUT. HapCUT [6] is one of the most accurate heuristic algorithms for individual haplotyping. HapCUT uses a random initial haplotype configuration and builds a graph. It computes max-cut on the graph to find the position to flip and iterates until no improvement in MEC score is achieved. We have performed extensive experiments to compare the performance of HMEC and HapCUT. Experimental results suggest that although HapCUT is reliable, its running time is too large to be a realistic choice for whole genome haplotyping. Our algorithm computes the haplotypes significantly faster than HapCUT without losing accuracy.

We used the filtered HuRef data from Levy et al. [19] to evaluate the performance. We generated several test data sets varying the coverage and error rate, and tested the performance of HMEC and HapCUT on these data sets. The results are shown in Tables 2, 3, 4, and 5.

Table 2: Reconstruction rate and running time of HapCUT and HMEC with 5% coverage on HuRef data.

Error rate	HapCUT		HMEC	
	Reconstruction rate	Time (sec)	Reconstruction rate	Time (sec)
10	0.986	40.582	0.859	0.119
15	0.913	38.282	0.889	0.125
20	0.875	46.324	0.811	0.129
25	0.902	47.101	0.827	0.130
30	0.840	39.196	0.802	0.125
40	0.694	44.792	0.665	0.129

Table 3: Reconstruction rate and running time of HapCUT and HMEC with 10% coverage on HuRef data.

Error rate	HapCUT		HMEC	
	Reconstruction rate	Time (sec)	Reconstruction rate	Time (sec)
10	0.988	76.081	0.966	0.141
15	0.954	90.206	0.948	0.154
20	0.953	81.169	0.957	0.144
25	0.952	74.924	0.923	0.142
30	0.951	81.828	0.893	0.177
40	0.934	69.748	0.814	0.169

Table 4: Reconstruction rate and running time of HapCUT and HMEC with 25% coverage on HuRef data.

Error rate	HapCUT		HMEC	
	Reconstruction rate	Time (sec)	Reconstruction rate	Time (sec)
10	0.988	184.642	1.000	0.156
15	0.988	171.698	1.000	0.167
20	0.988	134.406	0.998	0.163
25	0.988	133.966	0.998	0.155
30	0.987	136.572	0.996	0.172
40	0.985	171.334	0.960	0.161

Table 5: Reconstruction rate and running time of HapCUT and HMEC with 35% coverage on HuRef data.

Error rate	HapCUT		HMEC	
	Reconstruction rate	Time (sec)	Reconstruction rate	Time (sec)
10	0.988	181.025	1.000	0.158
15	0.988	132.434	1.000	0.127
20	0.988	231.989	0.999	0.105
25	0.987	210.040	0.998	0.101
30	0.988	217.292	0.996	0.102
40	0.987	215.533	0.995	0.100

Table 6: Reconstruction rate and running time of HapCUT and HMEC for ReHap samples.

Error rate	HapCUT		HMEC	
	Reconstruction rate	Time (sec)	Reconstruction rate	Time (sec)
0.2	0.870	20.343	0.985	0.035
0.3	0.865	21.109	0.970	0.030
0.4	0.840	20.106	0.945	0.035
0.5	0.790	19.549	0.865	0.050

Table 7: Comparison of different haplotyping techniques using ReHap. Min Len = 3, Max Len = 7, coverage = 8. Highest reconstruction rate for each error rate is shown in bold.

Error rate	SpeedHap	FastHare	MLF	2-Distance MEC	SHR	HMEC	HapCUT
0.2	**1**	0.91	**1**	**1**	0.65	**1**	0.9
0.3	0.935	0.7	0.945	0.61	0.685	**0.965**	0.82
0.4	0.605	0.78	0.78	0.68	0.55	0.535	**0.81**
0.5	0.535	0.48	0.485	0.505	0.526	0.525	**0.58**

Table 8: Comparison of different haplotyping techniques using ReHap. Min Len = 10, Max Len = 30, coverage = 8. Highest reconstruction rate for each error rate is shown in bold.

Error rate	SpeedHap	FastHare	MLF	2-Distance MEC	SHR	HMEC	HapCUT
0.2	0.98	0.975	**0.985**	**0.985**	0.84	**0.985**	0.87
0.3	**0.985**	**0.985**	0.97	0.97	0.74	0.97	0.865
0.4	**0.895**	0.79	0.885	0.885	0.675	0.87	0.84
0.5	0.74	0.47	0.76	0.76	0.51	0.705	**0.79**

Experimental results indicate that the reconstruction rates of both HapCUT and HMEC are reasonably good. For very low coverage, reconstruction rate of HapCUT is slightly better than HMEC. However, as the coverage rate increases, HMEC begins to outperform HapCUT. Notice that HapCUT is only better than HMEC for very low (and thus unrealistic) coverage values (5% and 10% coverage). However, for higher coverage, HMEC consistently performs better than HapCUT. Furthermore, HapCUT is significantly slower than HMEC. For an instance, with 35% coverage and 40% error rate, HapCUT takes 215.5 seconds where HMEC takes only a fraction of a second (see Table 5). Therefore, although generally HapCUT provides reliable reconstruction rate, on large dataset, it is an unrealistic choice due to its time consuming operations. On the other hand, HMEC provides the high accuracy with much less amount of time.

We also used some sample data from ReHap project [14, 15]. We created the allele matrix from the ReHap error matrix and fed the matrix to both HMEC and HapCUT algorithms. HMEC consistently produces higher reconstruction rate than HapCUT (see Table 6). Also, the running time of HMEC is clearly much better than HapCUT.

5.3. Comparison Using ReHap. We also compared HMEC with some other well-known algorithms such as SpeedHap [9], FastHare [10], MLF [11], 2 distance MEC [12], and SHR-3 [13] using HapMap-based instance generator and comparison framework [14, 15]. The results are shown in Table 7 and Table 8 that suggest that no single method clearly outperforms the others in all cases. However, reconstruction rates achieved by HMEC are the highest or very close to the highest. This bears a clear testimony to its suitability as a practical tool for individual haplotyping.

6. Conclusion

In this paper, we present a heuristic algorithm (HMEC) based on minimum error correction that computes highly accurate haplotypes significantly faster than the known algorithms for haplotyping. The algorithm is inspired from the famous Fiduccia and Mattheyses (FM) algorithm for bipartitioning a hyper graph minimizing the cut size [8]. We report on an extensive performance study evaluating our approach with other available techniques using both real and simulated datasets. Comprehensive performance study shows that our algorithm outperforms (in most cases) or matches the accuracy of other well-known methods, but runs in a fraction of the time needed for other techniques. High accuracy and very fast running time make our technique suitable for genome-wide scale data.

The accuracy of the algorithm can be improved by incorporating some prior knowledge. For example, small groups of fragments that are declared to be in the same haplotype can be identified. Probabilistic methods like expectation maximization (EM) also deserve some consideration over such optimization problems. In the near future, we intend to consider these issues to make further improvements.

Acknowledgments

The authors thank Rui-Sheng Wang (one of the authors of [5]) for providing them with the ACE and chromosome 5q31 datasets.

References

[1] R. Cilibrasi, L. V. Iersel, S. Kelk, and J. Tromp, "On the complexity of several haplotyping problems," in *Proceedings of the 5th International Workshop on Algorithms in Bioinformatics*, vol. 3692 of *Lecture Notes in Computer Science*, pp. 128–139, Springer, 2005.

[2] P. Bonizzoni, G. Della Vedova, R. Dondi, and J. Li, "The haplotyping problem: an overview of computational models and solutions," *Journal of Computer Science and Technology*, vol. 18, no. 6, pp. 675–688, 2003.

[3] A. Chakravarti, "It's raining SNPs, hallelujah?" *Nature Genetics*, vol. 19, no. 3, pp. 216–217, 1998.

[4] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail, "Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem," *Briefings in Bioinformatics*, vol. 3, no. 1, pp. 23–31, 2002.

[5] R. S. Wang, L. Y. Wu, Z. P. Li, and X. S. Zhang, "Haplotype reconstruction from SNP fragments by minimum error correction," *Bioinformatics*, vol. 21, no. 10, pp. 2456–2462, 2005.

[6] V. Bansal and V. Bafna, "HapCUT: an efficient and accurate algorithm for the haplotype assembly problem," *Bioinformatics*, vol. 24, no. 16, pp. i153–i159, 2008.

[7] J. Duitama, T. Huebsch, G. McEwen, E. K. Suk, and M. R. Hoehe, "ReFHap: a reliable and fast algorithm for single individual haplotyping," in *Proceedings of the ACM International Conference on Bioinformatics and Computational Biology (ACM-BCB '10)*, pp. 160–169, August 2010.

[8] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristics for improving network partitions," in *Proceedings of the 19th Design Automation Conference*, pp. 175–181, 1982.

[9] L. M. Genovese, F. Geraci, and M. Pellegrini, "SpeedHap: an accurate heuristic for the single individual SNP haplotyping problem with many gaps, high reading error rate, and low coverage," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 5, no. 4, pp. 492–502, 2008.

[10] A. Panconesi and M. Sozio, "Fast hare: a fast heuristic for single individual SNP haplotype reconstruction," *Workshop on Algorithms in Bioinformatics*, vol. 3240, pp. 266–277, 2004.

[11] Y. Y. Zhao, L. Y. Wu, J. H. Zhang, R. S. Wang, and X. S. Zhang, "Haplotype assembly from aligned weighted SNP fragments," *Computational Biology and Chemistry*, vol. 29, pp. 281–287, 2005.

[12] Y. Wang, E. Feng, and R. Wang, "A clustering algorithm based on two distance functions for MEC model," *Computational Biology and Chemistry*, vol. 31, no. 2, pp. 148–150, 2007.

[13] Z. Chen, B. Fu, R. Schweller, B. Yang, Z. Zhao, and B. Zhu, "Linear time probabilistic algorithms for the singular haplotype reconstruction problem from SNP fragments," *Journal of Computational Biology*, vol. 15, no. 5, pp. 535–546, 2008.

[14] F. Geraci, "A comparison of several algorithms for the single individual SNP haplotyping reconstruction problem," *Bioinformatics*, vol. 26, no. 18, Article ID btq411, pp. 2217–2225, 2010.

[15] ReHap, http://bioalgo.iit.cnr.it/rehap/.

[16] A. Al Mueen, M. S. Bayzid, M. M. Alam, and M. S. Rahman, "A heuristic algorithm for individual haplotyping with minimum error correction," in *Proceedings of the 1st International Conference on BioMedical Engineering and Informatics (BMEI '08)*, pp. 792–796, IEEE Computer Society Press, May 2008.

[17] M. J. Rieder, S. L. Taylor, A. G. Clark, and D. A. Nickrson, "Sequence variation in the human angiotensin converting enzyme," *Nature Genetics*, vol. 22, pp. 59–62, 1999.

[18] M. J. Daly, J. D. Rioux, S. F. Schaffner, T. J. Hudson, and E. S. Lander, "High-resolution haplotype structure in the human genome," *Nature Genetics*, vol. 29, no. 2, pp. 229–232, 2001.

[19] S. Levy, G. Sutton, P. C. Ng, L Feuk, and A. L. Halpern, "The deploid genome sequence of an individual human," *PLOS Biology* , vol. 5, no. 10, article e254, 2007.

[20] X. S. Zhang, R. S. Wang, L. Y. Wu, and W. Zhang, "Minimum conflict individual haplotyping from SNP fragments and related genotype," *Evolutionary Bioinformatics*, vol. 2, pp. 261–270, 2006.