2008 International Conference on BioMedical Engineering and Informatics

A Heuristic Algorithm for Individual Haplotyping with Minimum Error Correction

Abdullah Al Mueen Md. Shamsuzzoha Bayzid Md. Maksudul Alam
Md. Saidur Rahman

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh.
{abdullah.al.mueen, shams.bayzid}@gmail.com, maksud@csebuet.org, saidurrahman@cse.buet.ac.bd

Abstract

Haplotype is a pattern of Single Nucleotide Polymorphisms (SNP) 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 states 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 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 a SNP matrix. Alternative gain measure is also given to reduce running time. Experimental results show that our algorithm outperforms the best known previous algorithm.

Key words: *Algorithm, Bioinformatics, DNA sequence, SNP, Haplotype, Minimum Error Correction.*

1. Introduction

A single DNA molecule is a long chain of nucleotides. There are four such nucleotides which are represented by the set of symbols {A,T,G,C}. Hence, a DNA can be thought of as a string of symbols taken from this set. Every diploid organism has a set of pairs of DNA molecules. Each pair contains a paternal copy and a maternal copy of almost identical sequence of nucleotides (considering no recombination). The copies differ at only few positions with respect to their total length. Most of the times the variations occur at single nucleotide positions (on average 1 in every 600 base pairs) that are separated by a non-empty identical subsequence. Such variation is called *Single Nucleotide Polymorphism* and abbreviated as SNP [2, 1]. The nucleotide in a SNP site is called *allele*. If a SNP site can have only two nucleotides, it is called bi-allelic. If it can have more than two alleles it is called a multi-allelic SNP. From now on, we will consider the simplest case where only bi-allelic SNPs occur in a specific pair of DNA.

Since the two copies are identical except at the SNP sites, we can describe the two copies by two shorter sequences containing information only for SNPs. These two sequences consisting of nucleotides only at SNP sites are called the *haplotypes*. Haplotyping an individual deals with determining a pair of haplotypes, one for each copy of a given chromosome. 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 to which copy of the chromosome a fragment belongs is not determined.

The problem of haplotyping has been studied extensively. The general minimum error correction(MEC) problem was proved to be NP-hard [5]. It was also proved to be NP-hard even if the SNP matrix is gapless [3]. A heuristic method based on genetic algorithm has been proposed to solve this problem in [7].

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 a SNP matrix of dimension (m, k). The algorithm is inspired from the famous Fiduccia and Mattheyses (FM) algorithm for bipartitioning a hyper graph minimizing the cut size [4].

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.

978-0-7695-3118-2/08 $25.00 © 2008 IEEE
DOI 10.1109/BMEI.2008.283

Section 4 deals with the performance comparison of our algorithm with the most recent genetic algorithm. Finally we conclude by suggesting some future directions in Section 5.

2. Preliminaries

In this section we give some definitions and preliminary ideas.

Let S be the set of k bi-allelic SNP sites over which the haplotypes will be constructed. 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 sequences of SNP sites only. Thus each fragment $f \in F$ is a string of symbols $\{0, 1, -\}$ of length k where '$-$' denotes an undetermined SNP named as $hole$. 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 of F and column j is a SNP of S. This matrix is called SNP matrix.

Table 1. A SNP Matrix.

----1101----------
-----0001110101-----
11010010011---------
---10100---010------
---------10110101011
010111---------01011

The consecutive sequence of '$-$'s that lie between two non-hole symbols is called a *gap*. A *gapless* SNP matrix is the one that has no gap in any of the fragments. In Table 1, the first, second and third rows have no gaps while each of the fourth and sixth rows has one gap.

A SNP matrix $M = < M_1, M_2, \ldots, M_m >$ can be viewed as an ordered set of m fragments where a fragment $M_i = < M_{i1}, M_{i2}, \ldots, M_{ik} >$ 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,

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

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

$$d(x, y) = \begin{cases} 1 & \text{if } x \neq - \text{ and } y \neq - \text{ and } x \neq y; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In Table 1, the distance between second and third fragment is 2, as they differ in the seventh and ninth SNP sites.

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 $C_1 \bigcap C_2 = \phi$ [7]. In Fig. 1(b), an arbitrary partition corresponding to the SNP matrix of Fig. 1(a) is shown. Then 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, i.e., $D(x, y) = 0$. Such a partition is called an error-free partition.The partition in the Fig. 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 . The definition of the haplotype $H_i, i \in \{1, 2\}$ will be given in the next section.

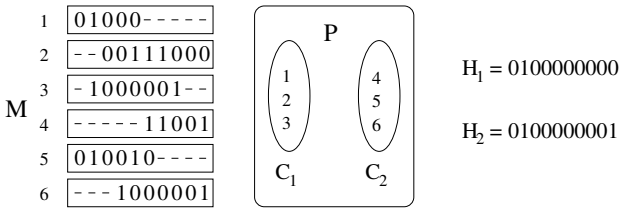

Figure 1. SNP matrix and its partition

We now focus on the general minimum error correction problem. If a matrix M is *not* error-free, there will be no error-free partition P. For such M there will be at least one conflicting pair of fragments in each of the classes of 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 must be corrected can be readily calculated by the following formula,

$$E(P) = \sum_{i=1}^{2} \sum_{f \in C_i} D(f, H_i) \tag{3}$$

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

3. A Heuristic Algorithm

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

this paper.

Construction of a haplotype from an erroneous class C requires correction of SNP values, i.e., alleles, in the fragments. Since, we want to correct minimum number of errors, we have to construct a haplotype which is minimum conflicting with the fellow fragments of its defining class. Therefore, for each SNP site, the haplotype should take the allele that is present in majority of the fragments. Let $N_j^0(C)$ be the number of fragments of a collection C that have 0 in jth SNP. Similarly, $N_j^1(C)$ defines the number of 1s. Therefore, to minimize the number of errors $E(P)$ for a specific partition P, the haplotype should be constructed according to the methodology that H_{ij} is 1 if $N_j^1(C_i) > N_j^0(C_i)$; H_{ij} is 0 if $N_j^0(C_i) \geq N_j^1(C_i)$ and $N_j^0(C_i) \neq 0$. Finally H_{ij} is a gap (-) if $N_j^1(C_i) = N_j^0(C_i) = 0$. Here $i \in \{1, 2\}$ and $j = 1, 2, \ldots, k$. In Fig. 1(c) the two haplotypes H_1 and H_2, associated with the partition P in Fig. 1(b), are constructed by this method.

To find the best partition we will use a heuristic search. 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. A fragment's transfer of collection can both 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 $Gain(f) = E(P_p) - E(P_n)$. Let $F = < f_i >$, $i = \{1, 2, \ldots, m\}$ 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 $Gain(f_i) \geq Gain(f_j)$ 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 upto nth fragment as $CGain(F, n) = \sum_{i=1}^{n} Gain(f_i)$. Here $Gain(f_i) = E(P_i) - E(P_{i+1})$. The maximum cumulative gain, $MCGain(F)$ is defined as

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

We shall illustrate these terms, later in this section, 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 the fragments from one partition to another for reducing the error function by an amount of $MCGain(F_c)$. The algorithm continues as long as $MCGain(F_c) > 0$ and stops whenever $MCGain(F_c) \leq 0$.

We now deal with data structures and complexity 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 transfers and locks that fragment to 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 is completed, the P_c becomes ready for the next iteration.

Later, 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 . 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's 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 $O(mk + k)$. And 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. Fig. 2 demonstrates an example iteration of HMEC. We consider that the current partition $P_c = P_1$ is the partition given in Fig. 1(b) for the SNP matrix M of Fig. 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

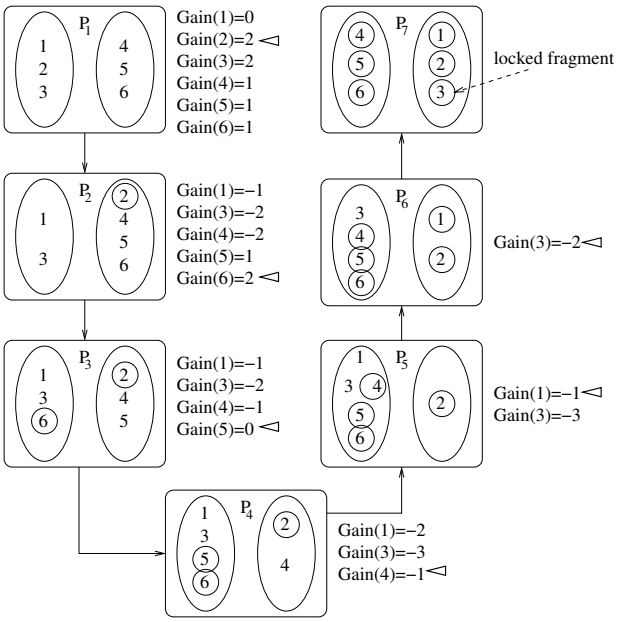

Figure 2. An example iteration of HMEC.

partition. The free fragment with maximum gain is marked in each intermediate partition. For example, the maximum gaining fragment on P_2 is fragment 6 with gain 2. After each transfer the transferred fragment is locked by a circle. Here, the ordering F_c of the fragments is $< 2, 6, 5, 4, 1, 3 >$ which is also the order of locking of the fragments. This ordering will be stored along with the $CGain$s in the log table. Table 2 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.

Table 2. An example log table.

Log Table

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

We now give an approximate gain measure to make our algorithm faster. 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,

$$ApproxGain(f) = D(H_i^p, f) - D(H_j^n, f) \quad (4)$$

where 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 reduces the run time of calculating gain to $O(k)$. The total run time of each iteration will be $O(m^2k)$.

4. Performance Comparison

We ran our program on real biological data as well as simulated data to demonstrate the performance of our program. Rieder *et al*. presented some haplotypes data and we used these as original biological data [6]. We also compared our program with the most recent genetic algorithm.

We now proceed to our testing terminologies. We first sample the original haplotype pair into many fragments with certain coverage and error. Each fragment works as distinct sample of the same specimen. Here coverage indicates how many columns of SNP matrix have been sampled out. The remaining slots are gaps. We then introduce some specific amount of error into these samples. The number of fragments, coverage and error rate are user given input for our simulated sequencing technique. The simulation was controlled in several ways. We varied the error rate while number of fragments and coverage were kept constant. Again coverage was varied while no of fragments and error rate were kept constant. Every time we compared our result with that of the Genetic algorithm.

We tested our algorithm and the Genetic algorithm thoroughly with data of various coverage and error rate. The reconstruction rate of our algorithm is very much comparable to that of the genetic algorithm and many of the times it is better. The reconstruction capability of our algorithm approaches better with the increase of coverage value. Fig. 3 illustrates the nature of two algorithm for various coverage value. The sharp slope of the corresponding graph of our algorithm is the clear testimony of superiority for our algorithm. We also simulate the algorithms for different error rate keeping the coverage value constant. Table 3 depicts that for advanced sampling technique, that is for low error rate, the performance of our algorithm is simply tremendous. For higher error rate it is also very much remarkable and comparable to that of the Genetic algorithm. Our algorithm outperforms the genetic algorithm, when time needed to reconstruct the haplotype pair is the main concern. Table 4 illustrates how fast our algorithm is compared to the genetic algorithm when programs are executed on a Pentium-III processor.

Another Salient feature of our algorithm is it's deterministic nature. In every execution, our algorithm generates same result for same data whereas Genetic algorithm, which is fully random in nature, generates different result in different execution. We have observed standard deviation up to 11 for Genetic algorithm whereas that of our algorithm is obviously 0.

5. Conclusion

In this paper we gave a heuristic algorithm based on minimum error correction which is simpler, faster and more ef-

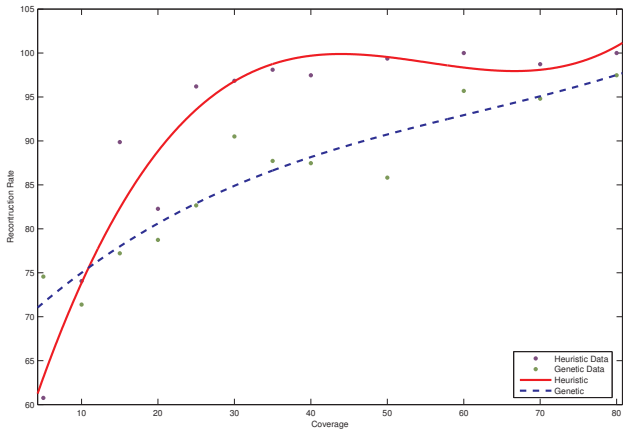

Figure 3. Reconstruction rate vs Coverage value.

Table 3. Demonstration of reconstruction rate for different error rate.

Error rate (percent)	HMEC (percent)	Genetic (percent)
2	100.00	86.453
5	99.37	91.202
7	98.73	91.264
15	84.81	86.708
20	84.81	88.164
25	93.67	82.406
30	79.75	88.355
35	92.40	81.898
40	70.89	78.482
50	85.44	78.986

Table 4. Demonstration of execution time.

Length of haplotype	HMEC (sec)	Genetic (sec)
79	0.01	9.483
158	0.01	17.806
316	0.01	35.101
632	0.03	71.142
862	0.04	100.064
964	0.04	111.119

ficient than the known algorithms for haplotyping. The accuracy of the algorithm can be improved by incorporating some prior knowledge. For example, small groups of fragments who 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.

The most important feature of this heuristic algorithm is its independency of gap. The position of holes (i.e. gaps) in the SNP matrix will not create any difference to the HMEC algorithm. For other haplotyping methods (i.e. Longest haplotype reconstruction, Minimum fragment removal), finding good heuristic algorithm would be interesting.

Acknowledgements

This research work is based on undergraduate research projects performed in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET). We thank BUET for supporting this research work.

References

[1] V. Bafna, S. Istrail, G. Lancia, and R. Rizzi. Polynomial and apx-hard cases of the individual haplotyping problem. *Theoretical Computer Science*, 335(1):109–125, 2005.

[2] P. Bonizzoni, G. D. Vedova, R. Dondi, and J. Li. The haplotyping problem: An overview of computational models and solutions. *Journal of Computer Science and Technology*, 18(6):675–688, 2003.

[3] R. Cilibrasi, L. van Iersel, S. Kelk, and J. Tromp. On the complexity of several haplotyping problems. In R. Casadio and G. Myers, editors, *WABI*, volume 3692 of *Lecture Notes in Computer Science*, pages 128–139. Springer, 2005.

[4] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *DAC '82: Proceedings of the 19th conference on Design automation*, pages 175–181, Piscataway, NJ, USA, 1982. IEEE Press.

[5] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail. Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Briefings in Bioinformatics*, 3(1):23–31, 2002.

[6] M. J. Rieder, S. L. Taylor, A. G. Clark, and D. A. Nickerson. Sequence variation in the human angiotensin converting enzyme. *Nature Genetics*, 22(1):59–62, 1999.

[7] R. S. Wang, L. Y. Wu, Z. P. Li, and X. S. Zhang. Haplotype reconstruction from snp fragments by minimum error correction. *Bioinformatics*, 21(10):2456–2462, 2005.