COMPARISON OF ALGORITHMS FOR THE HAPLOTYPE ASSEMBLY PROBLEM

Ömer Nebil Yaveroğlu

Computer Engineering Department, Middle East Technical University
email: nebil@ceng.metu.edu.tr

ABSTRACT

Haplotype Assembly Problem aims to construct the two haplotypes (chromosomes) given a collection of erroneous haplotype fragments. The error in the fragments is assumed to be Single Nucleotide Polymorphism (SNP). There is no limitation on the length of the fragments. Also they contain an arbitrary number of gaps. The problem of haplotype assembly problem is proven to be NP-Hard. In this paper, a comparison of three approximation algorithms on haplotype assembly problem is studied. These algorithms are HASH [1], Fast HARE [2] and HapCUT [3]. Among them HapCUT is the best algorithm by means of approximation accuracy and speed. The computational complexity of the approximate solution is calculated to be O (log n).

1. PROBLEM DEFINITION

Human are diploid organisms (2n). A human DNA sequence consists of two chromosomes which are complements of each other. Identification of the chromosome (haplotype) sequences of species is an important task in bioinformatics since it is possible to understand the genetic basis of diseases and metabolic processes with the use of this information. These sequences have an amount of genetic variability. The variability is a result of single nucleotide polymorphism (SNP's) most of the time. SNP is a single change of character in a nucleotide as a result of genetic mutation.

The current technologies for haplotype sequencing do not allow sequencing of all the nucleotides in only one experiment. SNP chips can interrogate up to a million of SNP's in an experiment. This value is not enough to identify the whole haplotype sequence of human and many other organisms. So the solution for the current technology is to partition the haplotype sequences into fragments. The fragmentation process is performed by the usage of some chemicals which breaks certain bonds in the haplotype sequence. So after the chemical separation process, a collection of haplotype fragments with no information about which strand it comes from. Also the number of fragments showing the same location and the location of the breaks are not known.

Another problem is that the reading processes of these fragments are erroneous. Some parts of the fragments cannot be read. Some read parts may be wrong. Considering all these problems all together for haplotype assemblies the aim is to find the full sequences of the two complementary haplotypes from a collection of fragments.

It is proven that the haplotype assembly problem is NP-Hard [2].Because of the NP-Hardness of the problem, it is not possible to find a deterministic or non-deterministic polynomial time solution to the problem. As a result of this, there exists a number of approximation algorithms proposed to find an approximate solution to the problem. HASH, Fast HARE and HapCUT are three well-known algorithms for the haplotype assembly problem. This paper compares these three methods by means of performance, complexity and accuracy.

2. DATASETS USED

Since human chromosome is not fully sequenced without any errors, there does not exist a ready to use dataset for applying methods. For this reason, the authors of the three algorithms have used the following methods to produce the data to work on.

Panconesi et al. [2] produces an artificial dataset using realistic assumptions. They generate a sequence of 100 characters for using as the first haplotype. Then the complement of this sequence is taken to use as the second haplotype. While taking the complement, some of the characters are not flipped with a given probability to simulate the errors in the dataset. Then the constructed samples are broken uniformly at random locations in order to generate fragments from the dataset. They claim that the only non-realistic thing about this fragment generation method is that the breaking of fragments is not performed with the same probability everywhere on a real DNA sequence. Although they generate this dataset synthetically, they claim that their method, Fast HARE, is applicable in all types of realistic data.

Bansal et al. [1,3] takes the haplotypes inferred in the HapMAP project. HapMAP haplotypes are proven to be invaluable for whole-genome association studies. On the other hand they have taken the HuRef haplotype as the reference haplotype which they aim to find. For constructing HuRef haplotype, 32 million sequenced reads have been used to generate the whole human haplotype.

3. NP-HARDNESS OF PROBLEM

There exists a short proof of NP-Hardness of Haplotype Assembly Problem in Panconesi et al.'s paper [2]. They try to solve the haplotype assembly problem by a method using minimum fragment removal. If there are no errors in reads then the matrix constructed to process the fragment dataset is bipartite. Minimum fragment removal method tries to find out a bipartite matrix by removing minimum number of rows from the fragmentation matrix. Then with the introduction of gaps, they convert minimum fragment removal to minimum element removal problem. Minimum element removal method tries to change minimum number of non-null entries in the fragmentation matrix to make this matrix bipartite.

After converting the haplotype assembly problem to minimum element removal problem, it can be proven that the haplotype assembly problem is NP-Hard. The proof NP-Completeness of minimum element removal problem can be done by reducing the minimum element removal to Hybercube Segmentation Problem. A detailed proof NP-Hardness of minimum element removal algorithm can be found in [4]. Also it can be shown that the minimum element removal algorithm is O (log n).

4. METHODS AND ALGORITHMS

The three approximation algorithms used for haplotype assembly problem has some common parts, especially for the construction of fragmentation matrix. These parts are discussed in part 2.1 and the differing parts of the algorithms are mentioned in 2.2, 2.3 and 2.4.

2.1 Common Points Of Three Methods

In all of the three methods, instead of dealing with the whole haplotype, the algorithms focus on the SNP locations. For this aim the authors extract the SNP's and form fragment matrices by using these extracted SNP's. An illustration of the construction of the haplotype assembly can be seen from Figure 1.

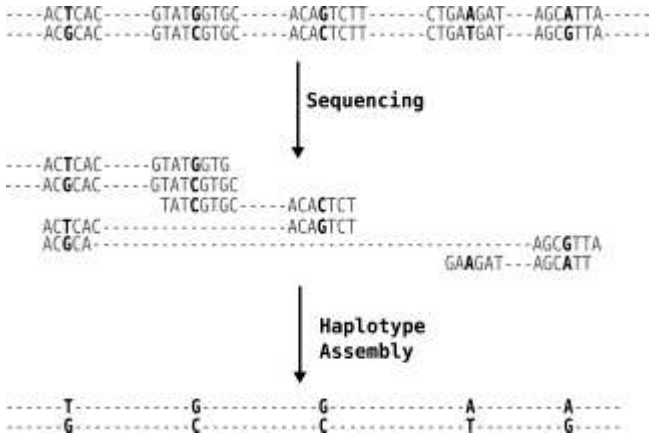

Figure 1: Selection of SNP's from haplotypes and construction of the haplotype assembly from haplotypes

Given a collection of haplotype fragments, an alignment is made according to a reference haplotype. Then the haplotype assembly is formed by taking the differing nucleotides of the alignment which correspond to SNP's.

The fragment matrix to work on is constructed by combining the SNP fragments putting the fragments to the rows of data. When this is the case, the columns of the SNP-fragment matrix correspond to SNP's. For simplicity, the characters forming the sequences are converted to 0's and 1's depending on the differing of the SNP's. Also for the non-determined segments '-' character is used to represent that the sequence information is not provided by the fragment. As a result, the fragment matrix is an m x n matrix where m is the number of fragments available and n is the number of SNP locations. Each cell in this matrix is an element of {0, 1, -}.

After the construction of this fragment matrix, all algorithms apply a different technique to get the full haplotype assembly determining the maternal and paternal chromosomes. Differing parts of these algorithms will be described in the next parts of the paper.

2.2 HASH

Since there exists an unknown number of haplotype fragments from haplotype a and b, a way to equalize them in number is required. For this aim, complements of each fragment is taken and included into the fragment set. This time there exist two copies of each fragment. To equalize their effect on Pr(X | q , H), we should divide the probability found from the constructed dataset into two.

HASH is a Markov Chain Monte Carlo (MCMC) method which takes the error probabilities into account. It iterates by performing a set of transitions of states on the SNP Fragment Matrix. A transition of fragment matrix is defined as flipping the values of a subset of SNP's (column values) forming a new fragment matrix. Optimizing the weight of the edges in a fragment matrix it is possible to find a suitable haplotype assembly that this matrix is derived.

The crucial point in MCMC is selecting the set of subsets of columns which the selection will be made for flipping. We represent this set of subsets of SNP's with Γ. A natural choice of Γ is {{1},{2}, … ,{n}} for n different SNP's. Starting with this subset and optimizing this set to get the best fitting haplotype, the haplotype assemblies can be found.

During the iteration for finding the optimum subset, a minimum cut method is used to find the optimized subset. The fragmentation matrix is converted into a graph. Each node in the graph represents a column of the fragmentation matrix, in other words a SNP. The edges shows whether some fragment that covers both columns. The weights of the edges become the number of fragments that cover both columns.

A cut is defined as a subset of nodes and edges of the original graph. Constructing a fragment graph this way and using

minimum cuts method to partition the graph into subsets, it is possible to find an optimum haplotype assembly from the given fragmentation matrix. An illustration of this method is shown in Figure 2.

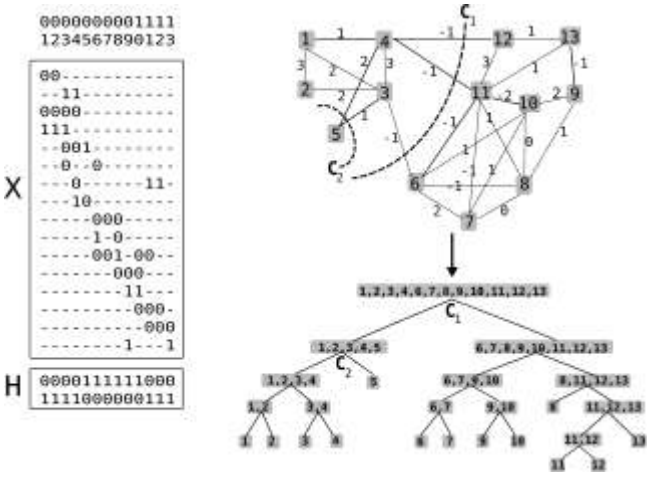

Figure 2: An illustration of the application of minimum cut method in order to find the optimum haplotype assembly.

To summarize the points I have mentioned above, it is appropriate to include a pseudocode of the algorithm used by HASH. This pseudocode can be found in Pseudocode 1.

HASH(X,q)

1. Set $\Gamma^{(0)} \leftarrow \Gamma_1$.
2. Set $H^{(0)}$ at random or otherwise.
3. For $t = 1, 2, \ldots$
   - (a) Let $H^{(t)} = \mathcal{M}(\Gamma^{(t-1)}, X, H^{(t-1)}, c)$ be the haplotype obtained after running $\mathcal{M}(\Gamma^{(t-1)})$ for $c \times n$ steps ($c \approx 1000$).
   - (b) Compute $\Gamma^{(t)}$ = WeightedGraphPartitioning(X, $H^{(t)}$).
4. Set $\Gamma \leftarrow \Gamma^{(t)}$ and discard all previous samples.
5. Run the chain $\mathcal{M}(\Gamma)$ initialized with $H^{(t)}$ for $\sim 10^6 \times n$ steps.

Pseudocode 1: HASH Algorithm

2.3 Fast HARE

Fast HARE is an algorithm developed by some researchers in the area of Computer Scientists [2]. This is the reason of the less biological depth that the paper covers. In fact the method is a heuristic to the problem of haplotype assembly problem. It takes the SNP fragment matrix defined in 2.1 and a parameter t which will be described later as the input. It outputs the haplotypes and the optimized SNP fragment matrix with a partition of the given fragments into two groups, the ones corresponding to haplotype 1 and ones that correspond to haplotype 2.

The main steps of the application of the method are as follows:

1. Eliminate the columns in which fa <= t or fb <= t.
2. Sort the rows according to the ones starting with non-null stretch.
3. Partition the rows of the matrix constructed according to which haplotype they represent.

To shortly describe these steps of the algorithms, in the first step the columns which doesn't show a SNP pattern are eliminated. fa = Na / (Na + Nb) and fb = Nb / (Na + Nb) so these values show the amount of differing in a column of haplotype fragments. Columns which are dominated by a fraction less than t are eliminated. The most different columns are included into the fragmentation matrix to be processed.

In the second step, the columns of the matrices are sorted according to the number of gap characters (-) they include at the beginning of the sequence. As a result of this sorting, the haplotype fragments representing the beginning parts of the haplotypes are taken to the front rows of the fragment matrix.

In the third step, a decision is made about which haplotype the invested fragment comes from. The first row is assigned to the first haplotype and so put to S_1 which is the set representing the haplotype fragments coming from the first haplotype. Then each row of the fragment matrix is put into either S_1 or S_2 according to the distance of the row to the elements in these sets. At the end, all of the fragments will be separated into two sets corresponding to the ones coming from haplotype 1 and haplotype 2. Reconstruction of the haplotypes is easy when these partitions are found. A basic string matching is made after flipping the values of the rows corresponding to S_2.

They have computed the complexity of this algorithm as O(nlogn +nm) where n is the number of fragments and m is the number of SNP's selected. This can be shown by recognizing that the O(nlogn) comes from second step of the algorithm and O(nm) complexity comes from the third step.

2.4 HapCUT

In fact there is nothing much to say about HapCUT after giving detailed description of HASH in the previous section. The authors of these two methods are more or less the same and the proposed method in HapCUT is just a variation of HASH.

This time instead of finding minimum cuts they have calculated the maximum cuts during the optimization of the subsets to be taken to be flipped. In other words, they perform the determination of the fragments corresponding to haplo-

type 1 and the ones corresponding to haplotype 2 by using maximum cuts instead of minimum cuts. They try to optimize the score of the new matrix. They use minimum error cuts (MEC) score as the distance metric of the graphs. All the other parts are similar when compared to HASH.

A summary pseudocode describing the algorithm can be seen in Pseudocode 2.

Procedure HapCUT

Initialization: Choose an initial haplotype configuration H^1 randomly.

Iteration: For t=1,2,...

1.Construct the graph $G_X(H^t)$
2.Compute a cut S in $G_X(H^t)$ such that $w_H(S)\geq 0$
3.If MEC($H^t{}_S$)$\leq$MEC(H^t), $H^{t+1}=H^t{}_S$
4.Else $H^{t+1}=H$

Pseudocode 2: The algorithm of HapCUT

In this paper, different scoring mechanisms can also be applied aiming to maximize their score. In fact the study can be seen as an extension of HASH which allows application of different scoring mechanism and performs faster compared to HASH.

5. COMPARISON OF THE ALGORITHMS

In the paper of Bansal et al. [3], a comparison of the three methods depending on the MEC scores is given. This comparison diagram can be seen from Figure 3.

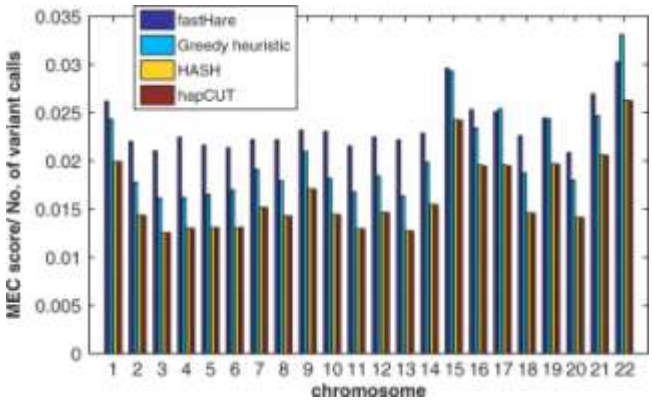

Figure 3: Comparison of the three algorithms on haplotype assembly problem according to the performance of MEC score.

As can be seen from Figure 3, HASH and HapCUT perform more or less the same on the performance criterion of MEC score by means of accuracy. These two methods outperform Fast HARE by producing less error. The MEC score of errors is found to reduce with an amount of 20 – 25%. Although this information is taken from the paper describing HapCUT [3], the amount of improvement is too high to be exaggerated.

On the other hand HapCUT seems an extension of HASH which is performed as a result of comments received for the published work. Although the performances of the methods have not increased in huge amounts, the algorithm became more open for modifications. It is possible to apply different scoring mechanisms such as minimum fragment removal (MFR), minimum error correction (MEC), minimum SNP removal. Also the results show that the HapCUT algorithm performs faster that HASH. An example comparison of the speed of algorithms is given as a computation is performed in 30 minutes by HapCUT while it takes 10 hours to complete with HASH. This shows a huge amount of reduction on the complexity of the algorithm although full comparison of complexities of two algorithms is not given in the text.

6. DISCUSSION

Within the current algorithms for the haplotype assembly problem, HapCUT seems the best performing one in between both by means of speed and accuracy. Most of the previous methods use reconstruction of the haplotype from the fragments but these three algorithms is more based on determining which haplotype produced the given fragment. Keeping this in mind, faster and easier predictions can be made using the given fragments.

Although the algorithms are mentioned to be approximation algorithms, they are erroneous. Even with these errors it is possible to produce promising results. The methods can be improved by means of applying different methods other than the used graph based methods. The three methods are more or less the same. They all try to separate the given fragments into two sets which represent the origin of the fragment. This separation information can be improved by performing cross validation. This would reduce the amount of error made and produce better results.

REFERENCES

[1] Vikas Bansal, Aaron L. Halpern and Nelson Axelrod, “An MCMC algorithm for haplotype assembly from whole-genome sequence data”, Genome Research, vol. 18, pp. 1336 - 1346, 2008.
[2] Alessandro Panconesi and Mauro Sozio, “Fast Hare: A fast heuristic for single individual SNP Haplotype Reconstruction”, WABI, pp. 266-277, 2004.
[3] Vikas Bansal and Vineet Bafna, “HAPCUT: an efficient and accurate algorithm for haplotype assembly problem”, ECCB, pp. i153-i159, 2008.
[4] G. Lancia, “Mathematical Programming Approaches for Computational Biology Problems”,In Modelli e Algorithmi per l’Ottimizzazione di Sistemi Complessi, Agnetis and Di Pillo Eds. Pitagore Editrice, 265-310.