Single Individual Haplotyping

EE381V: Genomic Signal Processing

Electrical and Computer Engineering Department
The University of Texas at Austin

24th April, 2013
Diploid genomes (humans)

Humans have 2 copies of chromosomes - inherited from parents

Genotyping provides the allelic composition at a given SNP site

Diploid genomes and genotyped positions
Diploid genome reconstruction (Phasing)

Heterozygous diploid individual

Genotype g
pairs of alleles with association of alleles to chromosomes unknown

haplotype $h=(h_1, h_2)$
possible associations of alleles to chromosome

Phasing
Why Haplotype Assembly?

- different haplotypes show different gene expression patterns, varying susceptibility to disease
- whole genome association studies (HapMap)
- understanding recombination patterns and identification of genes under positive selection
Illustration of Haplotype Assembly from Nex-Gen Sequencing Data

Diploid genome → paired end reads → reconstruction
Haplotype Assembly from Populations

Sequences from population sequencing
### Trio based haplotype assembly

#### Table: Trio based phasing (No recombination)

<table>
<thead>
<tr>
<th>Genotype</th>
<th>Parent I</th>
<th>Parent II</th>
<th>Offspring</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haplotype I</td>
<td>A T</td>
<td>T T/A</td>
<td>T/A T/A</td>
</tr>
<tr>
<td>Haplotype II</td>
<td>AT</td>
<td>TT</td>
<td>AT</td>
</tr>
<tr>
<td></td>
<td>AT</td>
<td>TA</td>
<td>TA</td>
</tr>
</tbody>
</table>

- If child is heterozygous, and a parent is homozygous, we know which allele comes from which parent.
- AT from parent I
- TA from parent II
- Child haplotypes (AT, TA)
A Combinatorial problem

0 0 0 0 1 0 0 0 0 1 0 0
1 1 1 1 0 1 1 1 1 1 0 1 1

Original haplotype

- Binary string and its complement
- The string is revealed to us only through a collection of substrings of the string, and its complement
- Given the substrings, can the string be reconstructed?
A Combinatorial problem

**Base space SNP-fragment matrix**

- Sequences mapped
- Only biallelic genotype positions retained
- Bases are converted to binary values 0 and 1 in the SNP fragment matrix
Inference in the absence of errors

Original haplotype

- The error free reconstruction is unique.
Inference in the presence of errors

The problem becomes much harder if some of the substrings have errors (do not match the consensus)
- Alphabet $\Sigma = \{0, 1, -\}$
- diploid
- gapped reads (paired end)

**Table 1.** An example of read matrix that consists of 10 reads spanning 13 positions

<table>
<thead>
<tr>
<th>Reads</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read1</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read2</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read4</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read5</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Read10</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Read - genotype position matrix / SNP matrix
Problem Statement

- given a set of fragments obtained by DNA sequencing from the 2 copies of chromosomes
- reconstruct the 2 haplotypes from SNP values observed in the fragments
- error free case trivial bipartite graph

- with errors reformulate problem
- find the smallest number of errors in the data so that there exist 2 haplotypes compatible with all the (corrected) reads
Previous Work

- Lancia 2001 Introduction MFR MSR NP Hard
- Cilibrasi 2005 MEC NP Hard gapless
- Li 2003
- FastHare Pancoseni 2004
- Wang 2005 Branch and Bound
- Zhao 2005 k Means Clustering
- Kim 2007 Gibbs Sampling
- Levy 2007 Greedy algorithm
- HASH 2008 MCMC
- SpeedHap 2007 2008
- Chen 2008 Randomized algorithms
- HapCut 2008 Graph partitioning MaxCUT
- MaxSat 2010 Dynamic programming
- HapCompass 2012 Graph based
Lancia, G. et al. (2001) SNPs problems, complexity and algorithms.

- Introduces
  - Minimum fragment removal (MFR)
  - Minimum SNP removal (MSR)
  - Longest Haplotype Reconstruction (LHR)

Cilibrasi, R. et al. (2005) On the complexity of several haplotyping problems

- Minimum error correction (MEC)
- Given an SNP matrix $M = (m_{i,j})$, correct a minimum number of elements (0 into 1 and vice versa) so that the resulting matrix is feasible
- i.e. the corrected SNP fragments can be divided into two disjoint sets of pairwise compatible fragments, with each set determining a haplotype
- Proof of NP Hardness
Switch Errors in reconstruction

- standard measures of haplotype accuracy: MEC and switch error rate
- \( SWER = \frac{2}{8} \)
Lo et al (2012) Strobe sequence design for haplotype assembly

Haplotype block length

- Paired end sequencing
- Read length
- Insert Length
- Insert length variation
- Coverage
Haplotype block length (AN50)

(a) Insert Size
Max N50
N50-simanneal (20x)

(b) Coverage

(c) Read Length

EE381V: Genomic Signal Processing Single Individual Haplotyping
Algorithms for Haplotype reconstruction
Initial Haplotype Construction

- Initial partial haplotypes constructed by repeating following sequence of steps until all rows assigned
- From remaining set of unassigned rows (initially all), choose row with fewest missing elements
- Use row to seed partial haplotype pair
- Until no more rows share non-missing information, identify row that has the strongest signal
- Assign that row to the indicated haplotype, extending the haplotypes to include any additional columns that are nonmissing for that row
- When no unassigned rows overlap the current haplotypes, consider this pair of partial haplotypes final; go back to the beginning
Iterative Haplotype Refinement.

- When all rows have been assigned to partial haplotypes, each haplotype pair and the rows it includes can be refined iteratively, repeating the following two steps until no changes result.
  - First, for each column (variant position) in the haplotypes, determine by majority rule the state assignment of each haplotype.
  - Second, for each row (read or mate pair), determine the haplotype assignment by majority rule.

- Form the adjacency matrix of the graph $G = (V, E)$
- The weight of an edge is given by $w_{i,j}$.
- MAXCUT is NP Hard.
- Assign a variable $x_i$ to each vertex of $G(w) = (V, E)$, and define this variable as follows:

$$x_i = \begin{cases} 
1 & \text{if the } i^{th} \text{ vertex is in } S \\
-1 & \text{if the } i^{th} \text{ vertex is in } \bar{S} 
\end{cases}$$

where $S$ is a subset of $V$, and $\bar{S}$ is the complement of $S$. 
We can model the max-cut problem as

\[
\begin{align*}
\text{maximize} & \quad \frac{1}{2} \sum_{i<j} w_{i,j} (1 - x_i x_j) \\
\text{subject to} & \quad x_i \in \{+1, -1\} \quad i = 1 \ldots n
\end{align*}
\]

In vector notation:

\[
\begin{align*}
\text{maximize} & \quad \frac{1}{4} x^t L x \\
\text{subject to} & \quad x_i^2 = 1 \quad i = 1 \ldots n
\end{align*}
\]

where \( L \) is the laplacian matrix of the graph (D-A).
Define a new variable $X = xx^t$, $X \in S^n_+$. Now we can write the vector form as:

\[
\begin{align*}
\text{maximize} & \quad \frac{1}{4} \text{Tr}(LX) \\
\text{subject to} & \quad \text{Diag}(X) = e \\
& \quad \text{rank}(X) = 1 \\
& \quad X \in S^n_+
\end{align*}
\]

Problem is still hard to solve because of rank constraint
Relaxing this constraint gives us the following relaxation:

\[
\begin{align*}
\text{maximize} & \quad \frac{1}{4} \text{Tr}(LX) \\
\text{subject to} & \quad \text{Diag}(X) = e \\
& \quad X \in S_+^n
\end{align*}
\]

SDP - can be efficiently solved in polynomial-time

gives an upper bound on the max-cut problem.
The Goemans-Williamson algorithm is a rounding procedure for finding an approximate solution to the max-cut problem.

Goemans-Williamson Algorithm:
- Solve model, call the optimal solution to this model $X^*$
- Compute Cholesky factorization of $X$ i.e. $X = V^t V$. Let $v_i$ represent the normalized columns of $V$
- Rounding Procedure: set $S = \{\}$
  - Uniformly generate a random vector $r$ on the unit $n$-sphere.
  - For $i = 1 \ldots n$ If $v_i^t r > 0$ assign vertex $i$ to $S$; else assign vertex $i$ to $\bar{S}$.
  - Find the weight of the obtained cut by $\frac{1}{4} x^t L x$ where $X = xx^t$
- Repeat the rounding procedure.

Guaranteed performance of 0.878
- Solution obtained by this algorithm is better than 0.878 in many cases.
MaxCut (Bansal and Bafna(2008):HapCUT an efficient and accurate algorithm for the haplotype assembly problem.)

- Given: fragment matrix $X$ a haplotype pair $H$
- Define: graph $G_X(H)$
  - Vertices: Columns of $X$
  - Edges $E_H$: Pairs of columns linked by some fragment
  - Let $X_i[j, k]$ fragment $i$, $H[j, k]$ haplotype pair $H$, when restricted to the pair of columns $(j, k)$
  - Weight of the edge $(j, k) \in E_H$ connecting columns $(j, k)$ defined as
    
    $$w_H(j, k) = \|i\|MEC(X_i[j, k], H[j, k]) = 1\| - \|i\|MEC(X_i[j, k], H[j, k]) = 0\|$$
Weight of the edge \((j, k)\) is the number of fragments inconsistent with the current phase between the pair minus the number of fragments consistent with the phase \(H[j, k]\).

Cut in the graph \(G_X(H)\) is defined by a subset \(S \subseteq X\) of vertices.

Weight of a cut \(S\) in \(G_X(H)\) is given by
\[
w_H(S) = \sum_{j \in S, k \in X - S} w_H(j, k)
\]

**Algorithm**

- **Initialization:** Choose an initial haplotype configuration \(H^1\) randomly.
- **Iteration:** For \(t = 1, 2, \ldots\)
  - Construct the graph \(G_X(H^t)\).
  - Compute a cut \(S\) in \(G_X(H^t)\) such that \(w_H(S) \geq 0\).
  - If \(MEC(H_S^t) \leq MEC(H^t)\), \(H^{t+1} = H_S^t\).
  - Else \(H^{t+1} = H_t\).
Assigning weights to edges of $G_X(H)$

- Formula gives disproportionately more weight to longer fragments, i.e. a fragment of length $k$ contributes a total absolute weight of $\binom{k}{2}$ to the graph.
- Weighting scheme modified
- Scaling of $\frac{1}{(k-1)}$ for a fragment of length $k$
Computing max-cuts

- Assumption: only need to find a positive-weighted cut in order to improve the MEC score
- Simple heuristics can find good cuts if all weights are positive.
- MEC score poor (e.g. for a random haplotype pair), most of the edges of the graph $G_X(H)$ have positive weights - finding a positive-weighted cut is easy.
- MEC score close to the optimum, most of the edges of the graph $G_X(H)$ have negative weights - greedy algorithm is not guaranteed to find a positive-weight cut.
- contract the edge $(s, t)$ and use a two-step greedy algorithm
- find a cut where most of the negative weight edges do not go across the cut
- move vertices from one side of the cut to the other if this improves the weight of the cut
HapCut : Final Algorithm

- Initialization BestCut = ∞
- Iteration: Iterate $O(m \log m)$ times
  - Chose an edge $(s, t)$ of the graph uniformly at random
  - Initialize $S_1 = s$ and $S_2 = t$
  - While $S_1 \cup S_2 \neq V$
    - For each vertex $v \not\in S_1 \cup S_2$ compute the score
      $A(v) = \sum_{s_1 \in S_1} w_H(v, s_1) \sum_{s_2 \in S_2} w_H(v, s_2)$
    - Let $v_{\text{max}}$ be the vertex for which $|A(v)|$ is maximum
    - If $A(v_{\text{max}}) < 0$, $S_1 = S_1 \cup v$
    - else if $A(v_{\text{max}}) > 0$, $S_2 = S_2 \cup v$
    - else add $v$ uniformly at random to $S_1$ or $S_2$
  - repeat
    - OldCut = $w_H(S_1)$
    - If $v \in S_1$ and $A(v) > 0$, move $v$ from $S_1$ to $S_2$
    - If $v \in S_2$ and $A(v) < 0$, move $v$ from $S_2$ to $S_1$
  - until $w_H(S_1) \leq \text{OldCut}$
  - If $w_H(S_1) > \text{BestCut}$, $\text{BestCut} = w_H(S_1)$
- Final: Return BestCut
THE END