

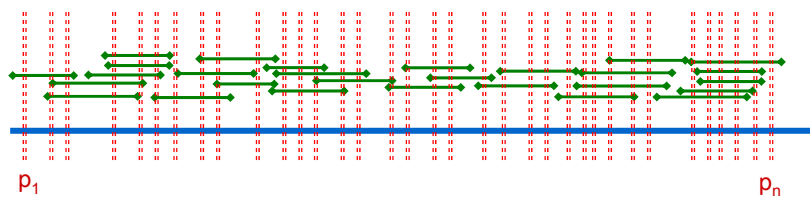
I. Retrieving DNA Sequence Information

DNA Sequencing (cont'd)

Lecture 7

Instructor: Su-In Lee

Physical mapping II – Hybridization



Short words, the *probes*, attach to complementary words

1. Construct many probes
2. Treat each BAC with all probes
3. Record which ones attach to it
4. Same words attaching to BACS X, Y \Rightarrow overlap

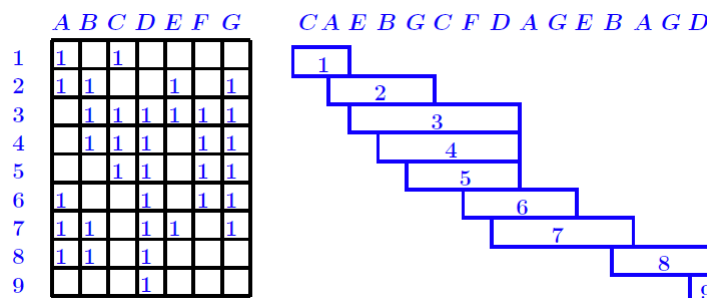
Physical mapping II – Hybridization

- Hybridization data: an $n \times m$ matrix M , where $M(i, j) = 1$ if clone C_i contains probe p_j and $M(i, j) = 0$ otherwise.
- “Fingerprint” of a clone: the set of probes hybridizing to the clone
- Two clones may overlap each other if they share part of fingerprints.

	A	B	C	D	E	F	G
1	1		1				
2	1	1			1		1
3		1	1	1	1	1	1
4		1	1	1		1	1
5			1	1		1	1
6	1			1		1	1
7	1	1		1	1		1
8	1	1		1			
9				1			

Shortest string covering problem

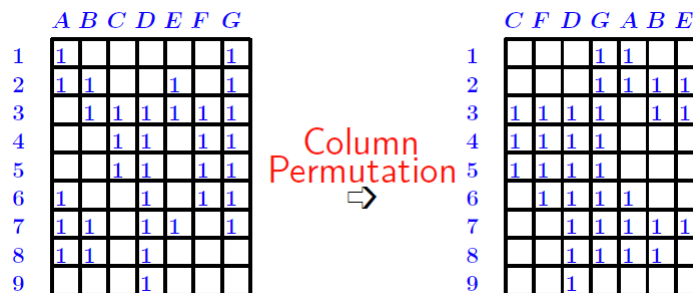
- Given hybridization data M , find a shortest string in the alphabet of probes that covers all clones.



- A string S covers a clone C if a substring of S contains exactly the same set of probes as C .
- NP-complete

Consecutive ones property

- **C1P**: the columns can be permuted in such a way that 1s in each row occur in consecutive positions
- Error-free hybridization matrix has the C1P.
- Given an matrix, determine if it has the C1P.



Algorithm for the C1P problem

- Given a binary matrix $M_{n \times m}$, determine if M has the C1P.
 - **Assume 1**: all rows are different (no two clones have the same fingerprint)
 - **Assume 2**: no row is all 0s (every clone is hybridized by at least probe)

Algorithm for the C1P problem

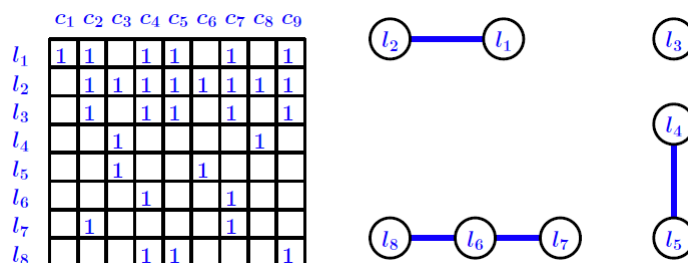
- Let $S_i = \{\text{column } k : M_{i,k} = 1\}$ for each row i in M .
- Given two rows i and j , we have (1) $S_i \cap S_j = \emptyset$, (2) $S_i \subseteq S_j$ or $S_j \subseteq S_i$, (3) $S_i \cap S_j \neq \emptyset$ and none of them is a subset of the other

- Algorithm [Fulkerson & Gross, *Pacific J. Math*, 1965]
 - Polynomial time complexity
 - 1. Separate rows into **components**
 - 2. **Permute** the columns of each component
 - 3. **Join** components together

- Other algorithms
 - Hsu, A simple test for the consecutive ones property. *ISAAC* 1992.
 - Booth & Lucker, Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms, *J. Comp & Sys Sci* 1976.

1. Separate rows into components

- Create $G_C = (V_C, E_C)$ from M such that
 - each vertex of V_C represents a row
 - $(i, j) \in E_C$ if $S_i \cap S_j \neq \emptyset$ and none of them is a subset of the other – group (3)
- **Component**: a connected component of G_C



2. Permute columns of components

- **Example:** a component with three rows as follows

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
l_1	0	1	0	0	0	0	1	1
l_2	0	1	0	0	1	0	1	0
l_3	1	0	0	1	0	0	1	1

- **Step 1:**

$$l_1 \rightarrow \begin{matrix} & \{2, 7, 8\} & \{2, 7, 8\} & \{2, 7, 8\} \\ 0 & 1 & 1 & 1 & 0 \end{matrix}$$

Permute columns of components

- **Step 2:**

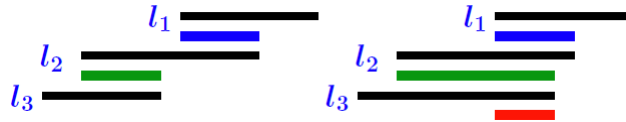
$$\begin{matrix} & \{5\} & \{2, 7\} & \{2, 7\} & \{8\} \\ l_1 \rightarrow & 0 & 0 & 1 & 1 & 1 & 0 \\ l_2 \rightarrow & 0 & 1 & 1 & 1 & 0 & 0 \end{matrix}$$

- **Step 3:**

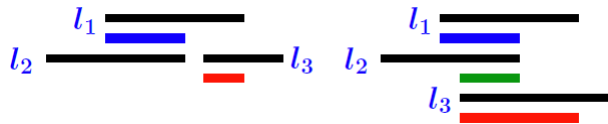
$$\begin{matrix} & \{5\} & \{2\} & \{7\} & \{8\} & \{1, 4\} & \{1, 4\} \\ l_1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ l_2 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ l_3 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{matrix}$$

Permute columns of components

- Let $x \cdot y = |\mathbf{S}_x \cap \mathbf{S}_y|$ for any two rows x and y .
- If $l_1 \cdot l_3 < \min\{l_1 \cdot l_2, l_2 \cdot l_3\}$, then l_3 goes in the same direction that l_2 was placed w.r.t. l_1 .

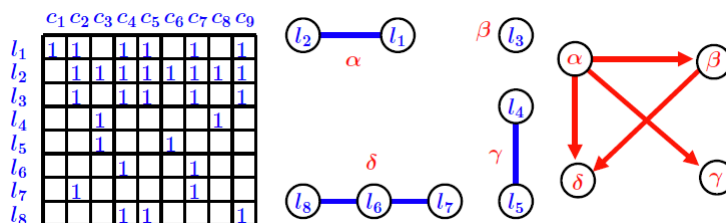


- If $l_1 \cdot l_3 > \min\{l_1 \cdot l_2, l_2 \cdot l_3\}$, then l_3 goes in the opposite direction that l_2 was placed w.r.t. l_1 .



3. Join components together

- $\alpha \rightarrow \beta$ if \mathbf{S}_i for all $i \in \beta$ are contained in at least one set \mathbf{S}_j of α .



Join components together

- Component α :

	{1}	{2, 4, 5, 7, 9}	{3, 6, 8}
l_1	1	11111	000
l_2	0	11111	111

- Component β :

	{2, 4, 5, 7, 9}
l_3	11111

- $\alpha \rightarrow \beta$:

	{1}	{2, 4, 5, 7, 9}	{3, 6, 8}
l_1	1	11111	000
l_2	0	11111	111
l_3	0	11111	000

Join components together

- Component δ :

	{9, 5}	{4}	{7}	{2}
l_6	00	1	1	0
l_7	00	0	1	1
l_8	11	1	1	0

- $\alpha \rightarrow \delta$:

	{1}	{9, 5}	{4}	{7}	{2}	{3, 6, 8}
l_1	1	11	1	1	1	000
l_2	0	11	1	1	1	111
l_3	0	11	1	1	1	000
l_6	0	00	1	1	0	000
l_7	0	00	0	1	1	000
l_8	0	11	1	0	0	000

Join components together

- Component γ : $\{6\}$ $\{3\}$ $\{8\}$

$$l_4 \quad 0 \quad 1 \quad 1$$

$$l_5 \quad 1 \quad 1 \quad 0$$

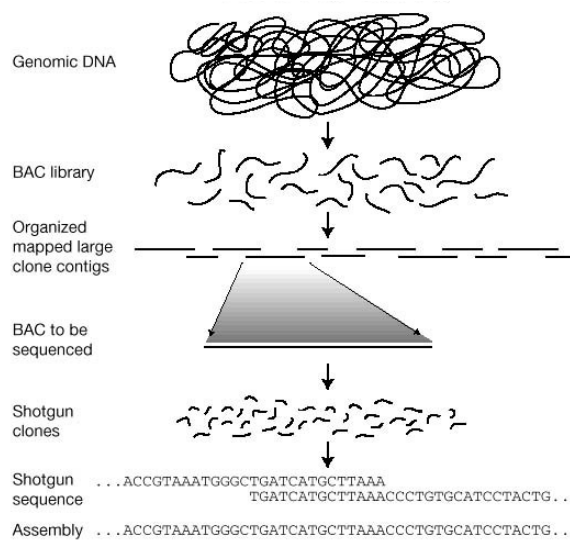
- $\alpha \rightarrow \gamma$:

	$\{1\}$	$\{9, 5\}$	$\{4\}$	$\{7\}$	$\{2\}$	$\{6\}$	$\{3\}$	$\{8\}$
l_1	1	11	1	1	1	0	0	0
l_2	0	11	1	1	1	1	1	1
l_3	0	11	1	1	1	0	0	0
l_6	0	00	1	1	0	0	0	0
l_7	0	00	0	1	1	0	0	0
l_8	0	11	1	0	0	0	0	0
l_4	0	00	0	0	0	0	1	1
l_5	0	00	0	0	0	1	1	0

Computational Molecular Biology and Genomics, Spring 2009

15

Hierarchical shotgun sequencing



Computational Molecular Biology and Genomics, Spring 2009

16

Acknowledgement

- This set of slides is based on the slides by:
 - Serafim Batzoglou (Stanford Univ)
 - Richard C.T. Lee (National Chi Nan Univ)