PHYSICAL MAPPING OF DNA

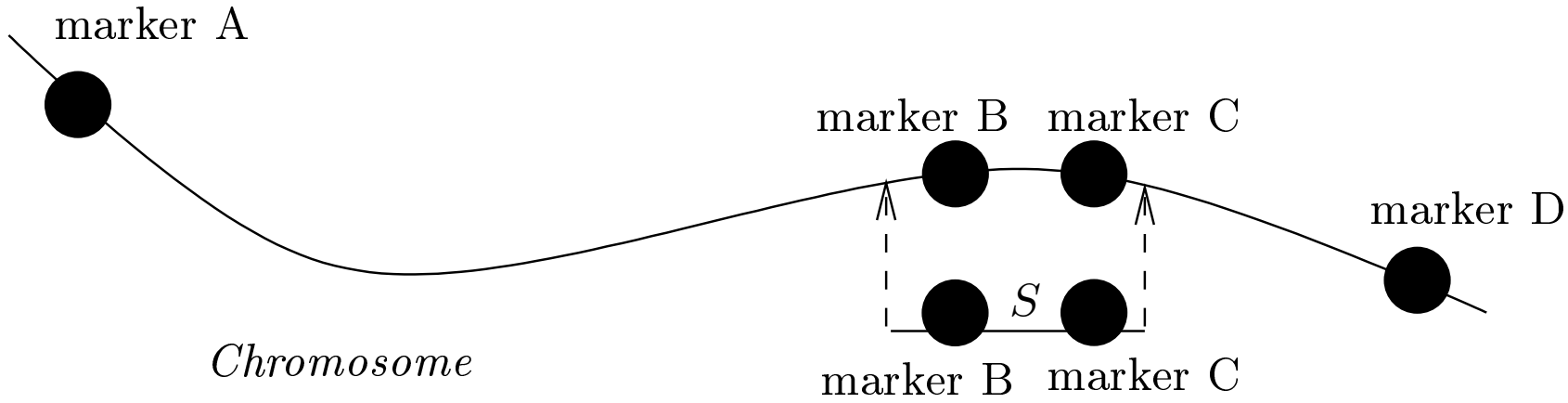
Mia Persson

Algorithms for Molecular Biology
Autumn 2004
Lund University

Biological Background

- The human chromosome is a DNA molecule with about 10^8 base pairs.
- Consider the problem of creating **physical maps** of entire chromosomes or of significants fractions of the chromosomes.
- A **physical map** tell us the location of certain **markers** along the chromosome.
- A **marker** is a precisely known small DNA sequence.

Sketch of a physical map



Use of a physical map

- **A physical map help molecular biologists to explore the genome.**
- **An example:**
  Assume that we have a completely sequenced piece S of DNA. If we know which chromosome S comes from and we have a physical map of this chromosome, then we can try to find one of the map's markers in S. If one of the map's markers is found in S, then we have located S in the chromosome.

Construction of physical maps

- Make several copies of the DNA molecule (denoted as the **target** DNA) we want to map.
- For each copy, break up this copy into several fragments (by using restriction enzymes).
- Note that in general the fragments are to long to be completely sequenced.
- Generate **fingerprints** of the fragments.
- Compare the fragments by observing overlaps between fragments and thus determine the relative order of the fragments.

Construction of physical maps

There are two commonly used approaches of getting fingerprints:

1. Restriction Site Mapping.
2. Hybridization Mapping.

Restriction Site Mapping

- A **restriction site** is a specific point in a DNA sequence.
- A **restriction map** is a map of all restriction sites in a DNA sequence.
- **Restriction enzymes** (proteins) are used in this method. A restriction enzyme cuts the DNA molecule in all places (on all restriction sites) where a certain sequence appears, thus creating a set of fragments. For example, E*co*RI is a restriction enzyme that cuts DNA wherever the sequence GAATTC is found.
- The fingerprints are the lengths of the fragments.

Double Digest Problem - Generating the Data

Assume we have two restriction enzymes, A and B, each recognizing a different sequence (restriction site).

- Apply A to one copy of the target DNA.
- Apply B to another copy of the target DNA.
- Apply both A and B to a third copy of the DNA.

High-level idea:
We want to order all the obtained fragments in such a way that the order is consistent with the experimental results and simultaneously, we locate the restriction sites (markers) in the DNA sequence.

Definition of Double Digest Problem

- **Input:**

  δA - the set of fragments lengths from the digest with first restriction enzyme A

  δB - the set of fragments lengths from the digest with second restriction enzyme B

  δX - the set of fragments lengths from the digest with both restriction enzymes A and B.

- **Output:**

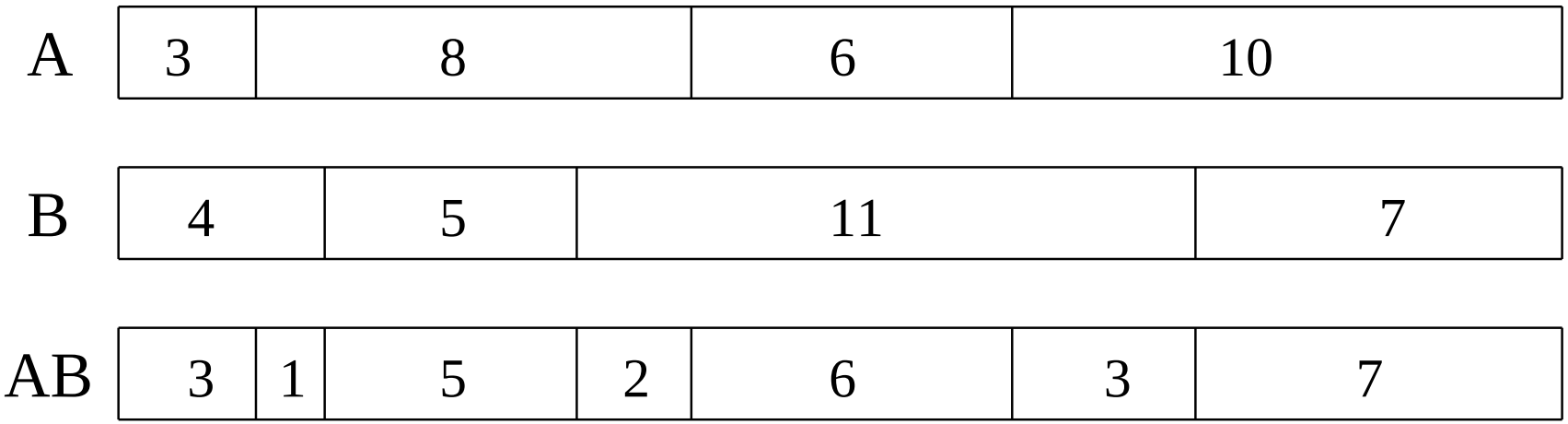
  A - location of the cuts in the restriction map for the first restriction enzyme.

  B - location of the cuts in the restriction map for the second restriction enzyme.

Double Digest Problem - An Example

Fragments of the following lengths for A: {3,6,8,10}.
Fragments of the following lengths for B: {4,5,7,11}.
Fragments of the following lengths for both A and B: {1,2,3,3,5,6,7}.

The solution:



NP-Completeness of Double Digest

1. Given the permutations of A and B, denoted by π_A and π_B. It is possible to check, in polynomial time, if the pair (π_A,π_B) is a possible solution or not.
2. Prove that the Set Partition problem can be reduced to the Double Digest problem in polynomial time.

Set Partition Problem: Given a set of integers $X = \{x_1, \ldots, x_n\}$, can the set X be partitioned into two sets X_1 and X_2 such that

$$\sum_{x_i \in X_1} x_i = \sum_{x_j \in X_2} x_j$$

i.e., the sum of the elements in X_1 equals the sum of the elements in X_2?

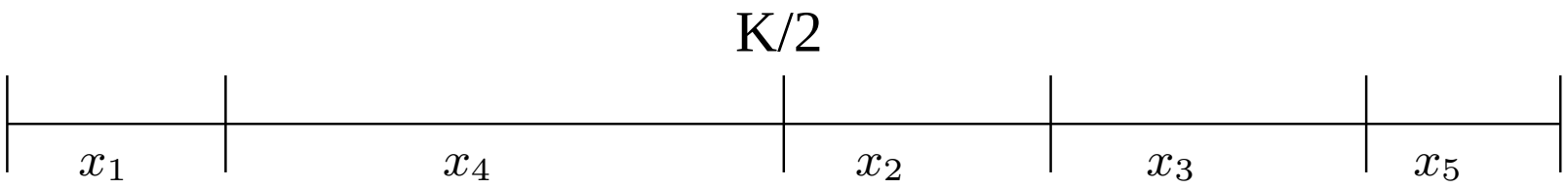
NP-Completeness of Double Digest (cont' d)

Instance of the Set Partition Problem:

$$X = \{x_1, \ldots, x_n\}$$

A solution to the following instance of Double Digest Problem would give a solution to the above instance of Set Partition Problem:
$A = X = \{x_1, \ldots, x_n\}$
$B = \{K/2, K/2\}$, where $K = \sum_{x_i \in X} x_i$
$C = A = \{x_1, \ldots, x_n\}$

NP-Completeness of Double Digest (cont' d)



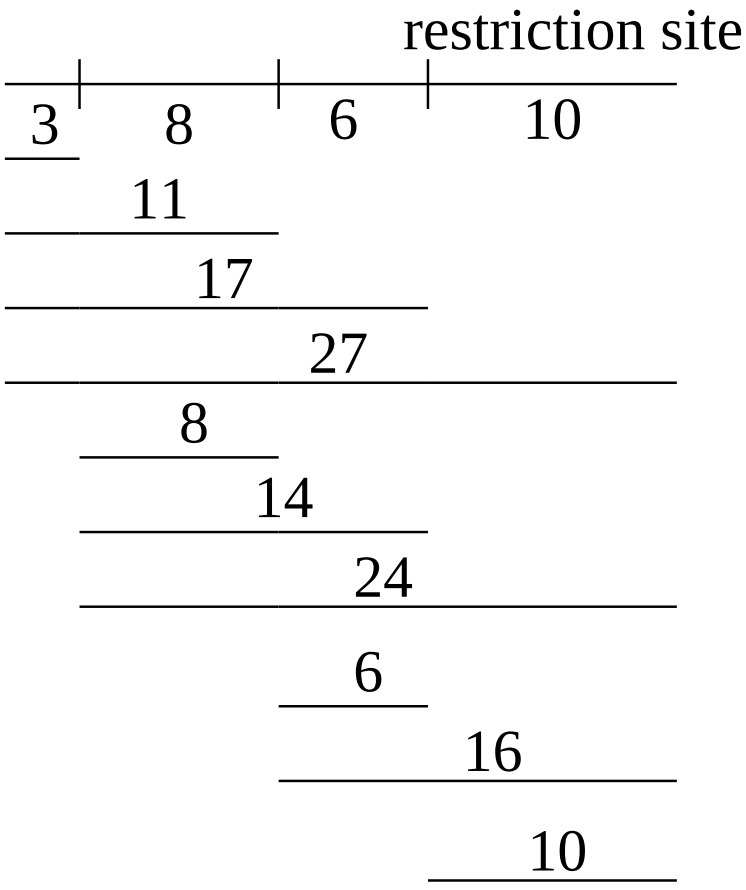
$X_1 = \{x_1, x_4\}$

$X_2 = \{x_2, x_3, x_5\}$

Partial Digest Approach

- The sample of DNA is exposed to one restriction enzyme for only a limited amount of time to prevent it from being cut at all restriction sites.
- We assume that with this method biologists can generate the set of all possible restriction fragments between every two cuts.
- We assume that multiplicity of a fragment can be detected, i.e., the number of restriction fragments of the same length can be determined.

Partial Digest Problem - Generating Data



Definition of the Partial Digest Problem

Input: A (multi)set S of lengths.

Output: A set of locations of the restriction sites (cut sites), such that the set of differences between all locations equals the set S.

More formally:

Let X be a set of points on a line.

Input: $\Delta X = \{|x_1 - x_2| : x_1, x_2 \in X\}$

Output: X

Partial Digest Problem - An Example

Example:
If $\Delta X = \{2, 5, 7, 7, 9, 9, 14, 14, 16, 23\}$, then $X = \{0, 7, 9, 14, 23\}$ is one feasible solution.

Partial Digest - Backtracking algorithm

ΔX: all pairwise distance.

1. Find L, the longest distance in ΔX.
2. $\Delta X = \Delta X$-L.
3. Find d, the longest distance in ΔX and position a cut site at point d or $L - d$.
4. Check that all the other resulting length are in ΔX.
   If they are, remove them from ΔX. If ΔX is empty we are done.
   Otherwise repeat from step 3.
   If not all distances are in ΔX backtrack.

Backtracking algorithm - An Example

$\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$

$\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8\}$

$\Delta X = \{2, 3, 3, 4, 5, 6, 7\}$

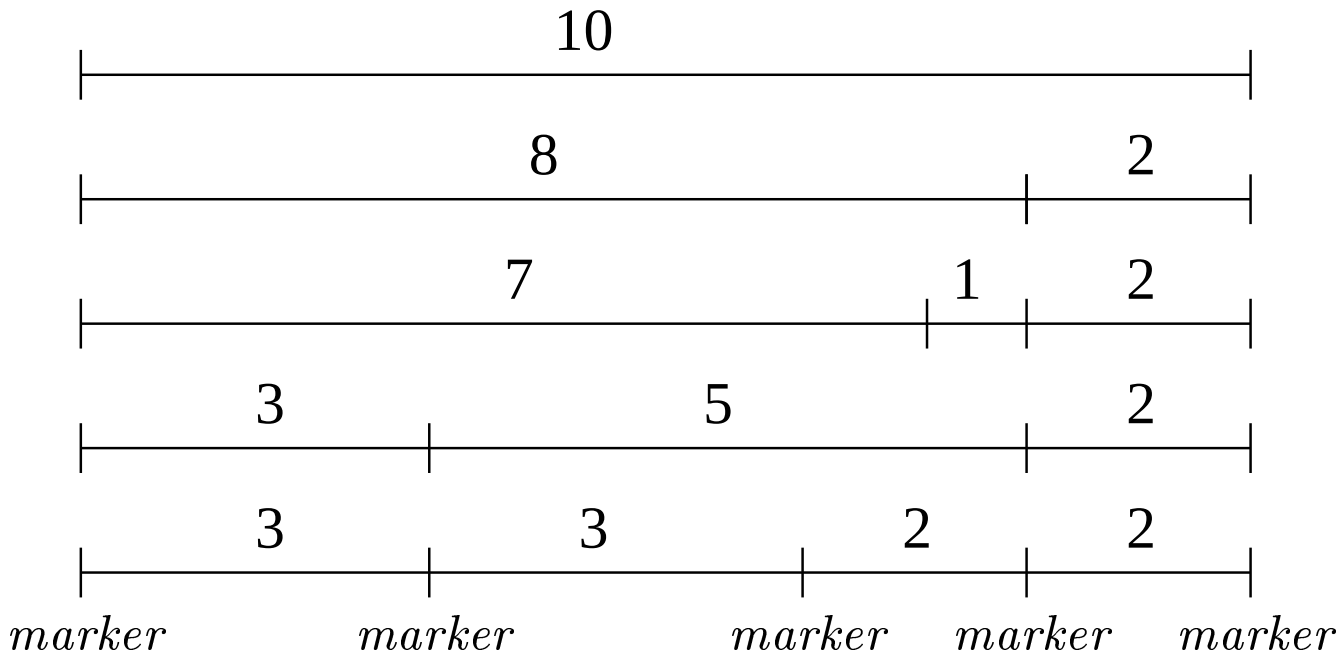
1 is not in ΔX

$\Delta X = \{2, 3, 4, 6\}$

$\Delta X = \{\}$ Done!

A solution to the PDP: $X = \{0, 2, 4, 7, 10\}$.

Backtracking algorithm - An Example



Partial Digest - Backtracking algorithm (cont' d)

Runtime:

- $O(\mathbf{n^2} \log \mathbf{n})$ expected time - fast in practice.
- Worst case exponential

But experimental PDP data is hard to obtain.

Hybridization mapping

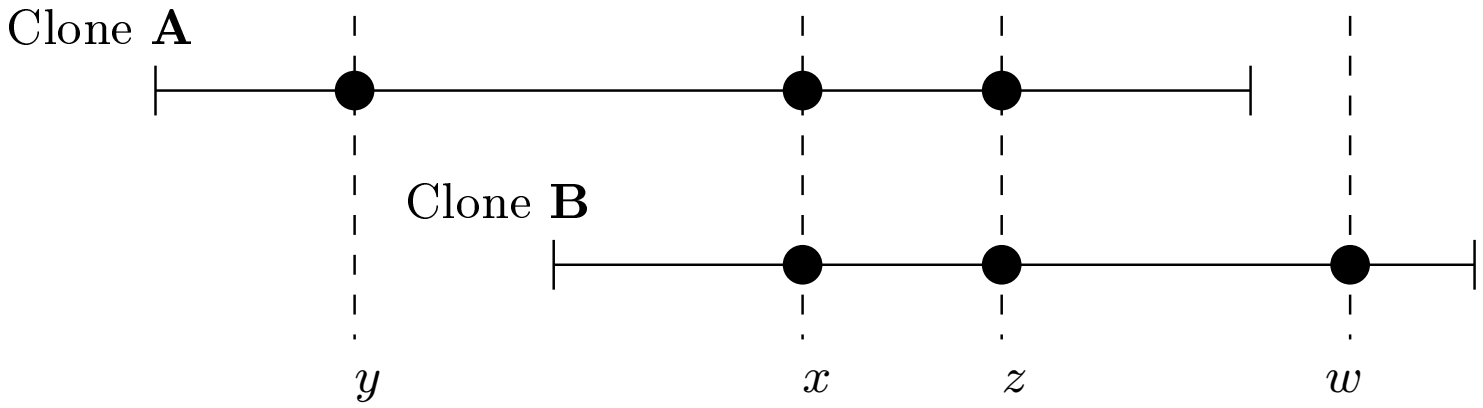
- Make several copy of the target DNA. For each of these copies, we break it up into fragments (cut at random positions in the DNA sequence). The fragments obtained are called **clones**.
- Each fragment is **cloned** (copied), and this cloning process results in a collection of many thousands of clones.

Hybridization experiment:

- For each clone we apply a set of different **probes** (short DNA sequences). If a probe occurs as a substring of the clone then the probe **hybridizes** (bind) to the clone.
- The **fingerprint** of a clone is the set of probes that hybridize to the clone.

Hybridization Mapping (cont' d)

- We assume that a probe only binds to the DNA sequence at one location.
- Two clones that bind to the same probe overlap.
- Overlaps give us the relative order of the probes along the target DNA.



Mathematical model - Interval Graph

Interval graph:

An undirected graph $G = (V, E)$ obtained from a collection C of intervals on the real line. To each interval in C there corresponds a vertex in G; we place an edge between vertices u and v if and only if their intervals have a nonempty intersection.

Hybridization mapping:

- A vertex in interval graph $H = (V, E)$ corresponds to a clone.
- There is an edge in H if clone $v_i \in V$ and clone $v_j \in V$ overlap.
- *Note:* If we had complete and correct information about clone overlapping, then H would be an interval graph.

Mathematical model - Interval Graph (cont' d)

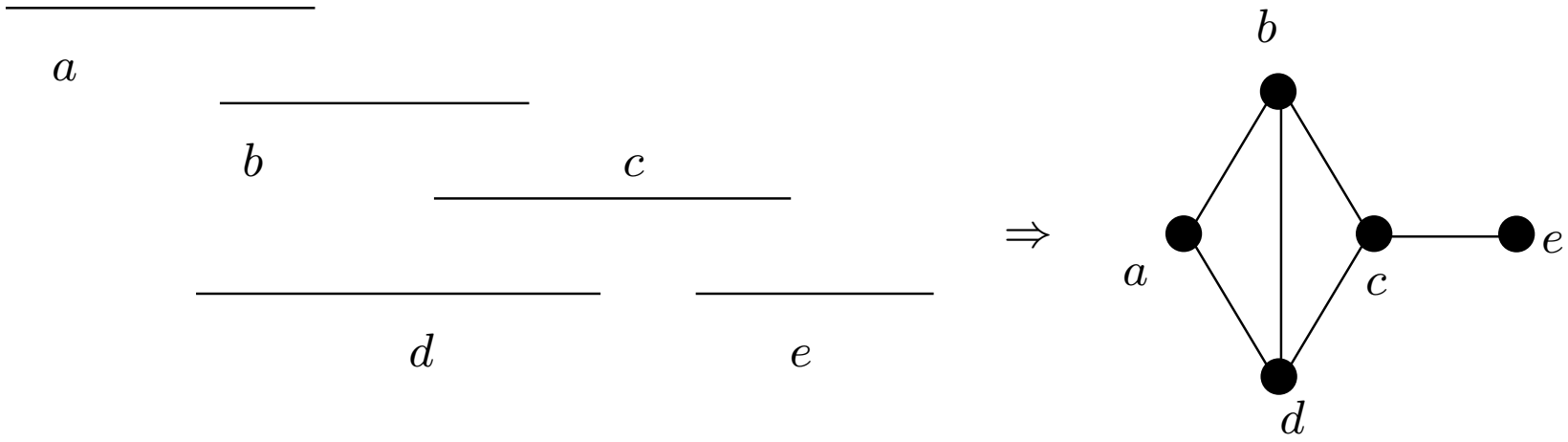
Given a graph $G_r = (V, E_r)$. If $(e_i, e_j) \in E_r$, then we know for sure that clones v_i and v_j overlap. Given a second graph $G_t = (V, E_t)$, where E_t represent known *plus* unknown overlap information (thus $E_r \subseteq E_t$). G_t is not necessarily complete (we may know for sure that certain pairs of clones do not overlap).

Problem:
Does there exist a graph $G_s = (V, E_s)$ such that $E_r \subseteq E_s \subseteq E_t$ and such that G_s is an interval graph?

Computational Complexity:

- NP-hard.

An Example



Clone overlapping *Interval graph*

Definition of CIP Problem

Given a binary matrix M, where entry M_{ij} tells whether probe j hybridize to clone i (M_{ij}=1) or not (M_{ij}=0).

Consecutive 1s Problem (C1P Problem):
Find a permutation of the columns (probes) such that all 1s in each row (clone) are consecutive.

Definition: A matrix for which there exists a permutation of the columns such that all 1s in each row are consecutive is said to have **Consecutive 1s Property (CP1)**

Physical Map Construction and the C1P problem

- A clones fingerprint is the set of probes that bind to it.
- Assume that probes only bind to DNA at one location in the target DNA.
- Assume that there are no errors.
- We have all possible data (from each clone-probe experiment).

A solution to the C1P Problem corresponds to a solution to the "physical map construction with probes as markers"-problem.

CIP Problem - An Algorithm

This algorithm finds a permutation of the rows such that each rows only has consecutive 1s.

Algorithm:

1. Divide the rows into different components by building a graph G.
2. For each component in G find permutations with C1P.
3. Join components in G to find permutation(s) for the whole matrix.

CIP Problem - An Algorithm (cont' d)

Definition:

For each row i of M, let S_i be the set of columns j where $M_{i,j} = 1$.

Step 1: Divide the rows into different components by building a graph G.

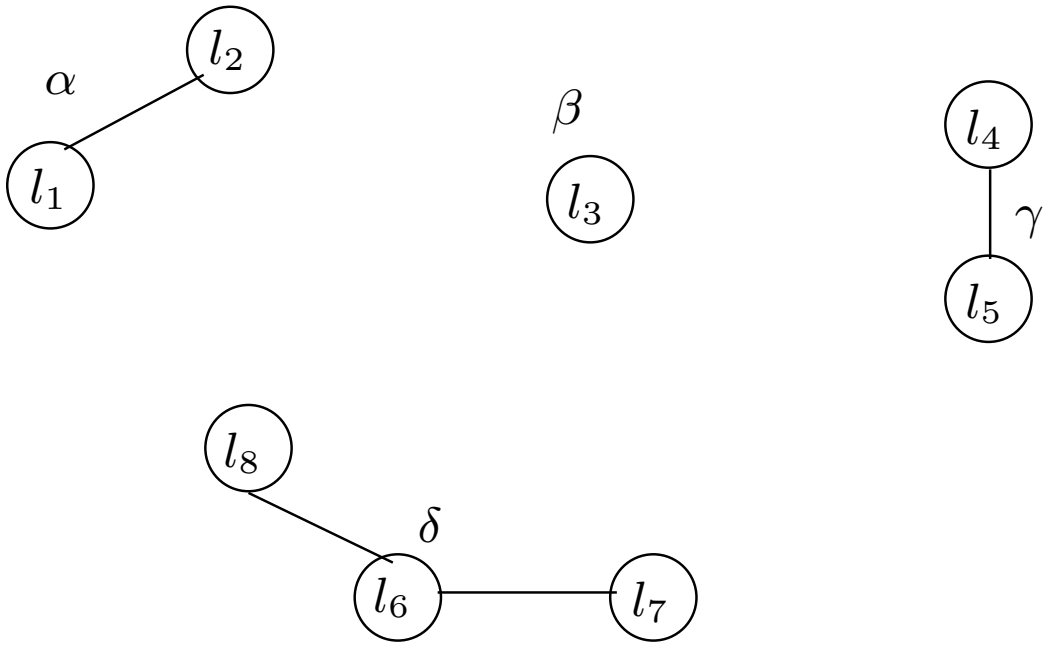
For two rows i and j we have three possible cases:

1. $S_i \cap S_j = \emptyset$.
2. $S_i \subseteq S_j$ or $S_j \subseteq S_i$.
3. $S_i \cap S_j \neq \emptyset \wedge S_i \not\subseteq S_j \wedge S_j \not\subseteq S_i$.

CIP Problem - An Algorithm (cont' d)

Build undirected graph G graph from M with vertices corresponding to rows in M. There is an edge in G if and only if case 3 holds for rows i and j.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
l_1	1	1	0	1	1	0	1	0	1
l_2	0	1	1	1	1	1	1	1	1
l_3	0	1	0	1	1	0	1	0	1
l_4	0	0	1	0	0	0	0	1	0
l_5	0	0	1	0	0	1	0	0	0
l_6	0	0	0	1	0	0	1	0	0
l_7	0	1	0	0	0	0	1	0	0
l_8	0	0	0	1	1	0	0	0	1


l_2
α
l_1
β
l_3
l_4
γ
l_5
l_8
δ
l_6
l_7

CIP Problem - An Algorithm (cont' d)

Step 2: For each component in G find permutations with C1P.

	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

		$\{2,7,8\}$			$\{2,7,8\}$			$\{2,7,8\}$				
$l_1 \rightarrow \ldots 0$		1			1			1		$0 \ldots$		
		$\{5\}$		$\{2,7\}$			$\{2,7\}$		$\{8\}$			
$l_1 \rightarrow \ldots 0$		0		1			1		1	$0 \ldots$		
$l_2 \rightarrow \ldots 0$		1		1			1		0	$0 \ldots$		
		$\{5\}$		$\{2\}$		$\{7\}$		$\{8\}$		$\{1,4\}$	$\{1,4\}$	
$l_1 \rightarrow \ldots 0$		0		1		1		1		0	0	$0 \ldots$
$l_2 \rightarrow \ldots 0$		1		1		1		0		0	0	$0 \ldots$
$l_3 \rightarrow \ldots 0$		0		0		1		1		1	1	$0 \ldots$

CIP Problem - An Algorithm (cont' d)

Step 2: For each component in G find permutations with C1P.

If $|S_1 \cap S_3| > min(|S_1 \cap S_2|, |S_2 \cap S_3|)$ then row 3 has to be placed on the opposite side of row 1 compared to row 2.

If $|S_1 \cap S_3| < min(|S_1 \cap S_2|, |S_2 \cap S_3|)$ then row 3 has to be placed on the same side of row 1 as row 2.

$S_1 = \{2, 7, 8\}$, $S_2 = \{2, 5, 7\}$, $S_3 = \{1, 4, 7, 8\}$

$S_1 \cap S_3 = \{7, 8\}$, $S_1 \cap S_2 = \{2, 7\}$, $S_2 \cap S_3 = \{7\}$

S_3 goes to the left in this example.

CIP Problem - An Algorithm (cont' d)

Step 2: For each component in G find permutations with C1P.

Add the rest of the rows in the component one by one.

Add row k by finding two rows i and j where edges (i, k) and (j, k) are in the graph and check the above condition.

Check that we still have valid permutations.

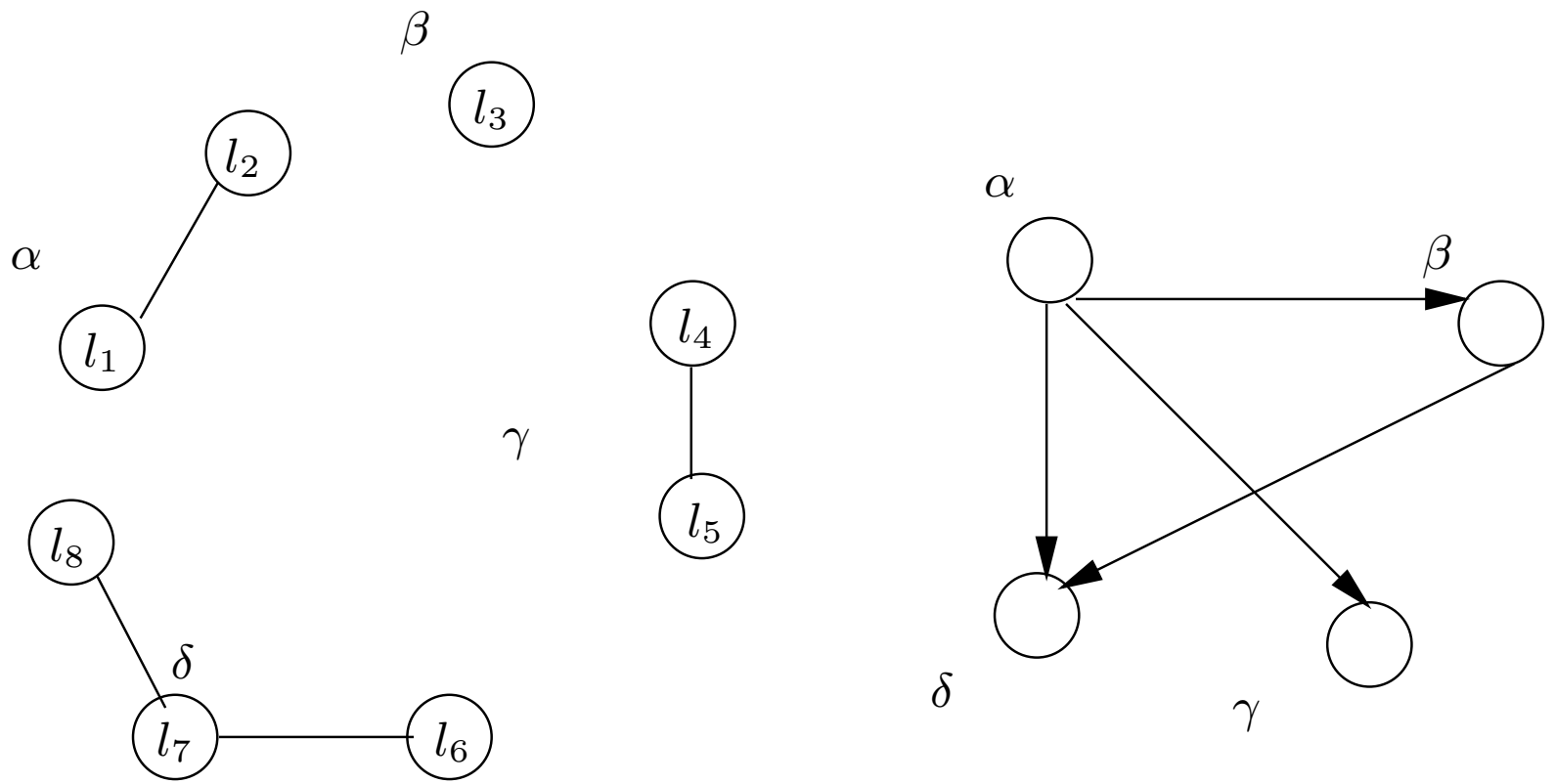
Running time:

- Time to add one row: $O(m)$
- Number of rows: $O(n)$
- Total time: $O(nm)$

CIP Problem - An Algorithm (cont' d)

Step 3: Join components in G to find permutation(s) for the whole matrix.

Build a new directed graph G_M with the components in G as vertices. There is a directed edge (α, β) between vertices α and β in G_M if the sets S_i for all $i \in \beta$ are contained in at least one set S_j of component α.


β
l_3
l_2
α
l_1
l_4
γ
l_5
l_8
δ
l_7
l_6
α
β
δ
γ

CIP Problem - An Algorithm (cont' d)

Step 3: Join components in G to find permutation(s) for the whole matrix.

Lemma: If there exist $i \in \beta$ and $j \in \alpha$ and moreover $S_i \subset S_j$ then $S_l \subset S_j$ for all $l \in \beta$.

Proof: By contradiction.

Fact: G_M is acyclic.

Process the vertices in G_M one by one in topological order. Use the topological order $\alpha, \beta, \delta, \gamma$ here.

CIP Problem - An Algorithm (cont' d)

Step 3 cont' d: Permute the columns (correctly) of all components individually. Then select all vertices in G_M without incoming edges and freeze their columns. Then, take the next vertex in topological order. Suppose we are following edge (α, β). Find a reference column in component α (this column tells us how to place the rows of β). Now, choose the row l from β that has the leftmost 1, and call the column where this 1 is c_β. We know (from Lemma) that S_l is contained in some S_i of α but not in others. Find all rows from α that contain S_l, and find the leftmost column where all such rows have 1s (and call this column c_α). This is the reference column, since we can now make c_α and c_β one and the same.

Illustrating example follow...

α:		$\{1\}$		$\{2,$	$4,$	$5,$	$7,$	$9\}$	$\{3,$	$6,$	$8\}$	
$l_1 \to \ldots$		1		1	1	1	1	1	0	0	0	$\ldots$
$l_2 \to \ldots$		0		1	1	1	1	1	1	1	1	$\ldots$
β:				$\{2,$	$4,$	$5,$	$7,$	$9\}$				
$l_3 \to$			$\ldots$	1	1	1	1	1	$\ldots$			

α, β joined:	$\{1\}$	$\{2,$	$4,$	$5,$	$7,$	$9\}$	$\{3,$	$6,$	$8\}$	
$l_1 \to \ldots$	1	1	1	1	1	1	0	0	0	$\ldots$
$l_2 \to \ldots$	0	1	1	1	1	1	1	1	1	$\ldots$
$l_3 \to \ldots$	0	1	1	1	1	1	0	0	0	$\ldots$

δ:	$\{9,$	$5\}$	$\{4\}$	$\{7\}$	$\{2\}$	
$l_6 \to \ldots$	0	0	1	1	0	$\ldots$
$l_7 \to \ldots$	0	0	0	1	1	$\ldots$
$l_8 \to \ldots$	1	1	1	0	0	$\ldots$

δ joined:	$\{1\}$	$\{9,$	$5\}$	$\{4\}$	$\{7\}$	$\{2\}$	$\{3,$	$6,$	$8\}$	
$l_1 \to \ldots$	1	1	1	1	1	1	0	0	0	$\ldots$
$l_2 \to \ldots$	0	1	1	1	1	1	1	1	1	$\ldots$
$l_3 \to \ldots$	0	1	1	1	1	1	0	0	0	$\ldots$
$l_6 \to \ldots$	0	0	0	1	1	0	0	0	0	$\ldots$
$l_7 \to \ldots$	0	0	0	0	1	1	0	0	0	$\ldots$
$l_8 \to \ldots$	0	1	1	1	0	0	0	0	0	$\ldots$

γ:	$\{6\}$	$\{3\}$	$\{8\}$	
$l_4 \to \ldots$	0	1	1	$\ldots$
$l_5 \to \ldots$	1	1	0	$\ldots$

γ joined:	$\{1\}$	$\{9,$	$5\}$	$\{4\}$	$\{7\}$	$\{2\}$	$\{6\}$	$\{3\}$	$\{8\}$
$l_1 \rightarrow$	1	1	1	1	1	1	0	0	0
$l_2 \rightarrow$	0	1	1	1	1	1	1	1	1
$l_3 \rightarrow$	0	1	1	1	1	1	0	0	0
$l_6 \rightarrow$	0	0	0	1	1	0	0	0	0
$l_7 \rightarrow$	0	0	0	0	1	1	0	0	0
$l_8 \rightarrow$	0	1	1	1	0	0	0	0	0
$l_4 \rightarrow$	0	0	0	0	0	0	0	1	1
$l_5 \rightarrow$	0	0	0	0	0	0	1	1	0

CIP Problem - An Algorithm (cont' d)

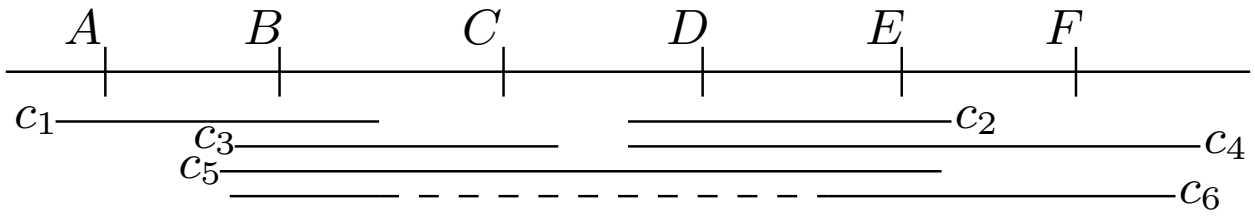
Step 3: Join components in G to find permutation(s) for the whole matrix.

Running time:

- Topological sorting: $O(n+m)$.
- Preprocessing: Store first column with 1 for each row in $O(nm)$ time.
- Joining: $O(n)$ time to find rows + $O(m)$ time to join permutations.

Total time: $O(nm)$
Best known algorithm: $O(n+m+r)$, where r is the number of 1s in the matrix.

Hybridization Mapping with Errors



	A	B	C	D	E	F	
c_1	1	1	0	0	0	0	←correct
c_2	0	0	0	1	1	0	←correct
c_3	0	1	1	0	0	0	←correct
c_4	1*	0	0	1	1	1	←false positive
c_5	0	1	0*	1	0	0	←false negative
c_6	0	1*	0	0	1*	1*	←chimeric clone

Hybridization Mapping with Errors

Three types of errors:

- False positive
- False negative
- Chimeric clone

Note: All three types give *gaps* in the (true) matrix. There may not be a solution to the C1P problem for an instance with errors.

Gap Minimization(Optimization problem): Given a binary matrix with n rows and m columns.

Problem: Find a permutation where the total number of gaps in the matrix is minimized.

Hybridization Mapping with Errors (cont' d)

Traveling Salesman Problem (TSP): Given a complete undirected weighted graph.

Problem: Find a *Hamiltonian cycle* (a cycle such that every vertex in the graph is in the cycle but each vertex appears exactly once) of minimum weight.

Fact:

TSP is NP-hard. Good approximation exist.

Hybridization Mapping with Errors (cont' d)

Reduction of Gap Minimization to TSP.

Construction:

Let M be a (clone-probe) binary matrix and $G = (V, E)$ a complete undirected graph. A column (probe) in M corresponds to a vertex in G. An edge $e \in E$ has weight equal to the *Hamming distance* between rows $u \in V$ and $v \in U$, where the Hamming distance between to binary strings is equal to the number of positions where the two strings differ, e.g., the Hamming distance between $u = 10010$ and 10100 is 2.

A minimum-weight cycle in G corresponds to a column permutation in M with minimum number of gaps.

	p_1	p_2	p_3	p_4	p_5	p_6
c_1	1	1	1	1	0	0
c_2	0	1	1	1	0	0
c_3	1	0	0	0	1	0
c_4	1	0	1	1	0	0