Introduction to computational and systems biology

- Lecture 5: Physical mapping

- Ion Petre
- Department of Mathematics and Statistics
- University of Turku

- Fall 2019

Turku Centre *for* Computer Science

Physical (or restriction site) mapping

- The problem: *Establish the physical location of some landmarks (e.g., the restriction sites of an enzyme) in the genome*
- Importance: in DNA sequencing, in gene finding
  - Assume a short sequence of DNA from chromosome 1 has been sequenced. Having a physical map of chromosome 1, we will find the location of the markers in the short sequence and then place the sequence on the chromosome
- Important phase in genome sequencing projects
- Short youtube video with an example (Jeri Erickson and Walt Allan at the FBR's out-reach education division, ScienceWorks for ME): https://www.youtube.com/watch?v=8FqMUF96cPE

Content

- Mapping by enzyme digestion – small molecules
  - Partial digestion (PDP)
  - Double digestion (DDP)
- Mapping by hybridization – long molecules
- Formulate the experimental setup
- Formulate the computation problem
  - Give the algorithm (if a "smart" one exists)

MAPPING BY ENZYME DIGESTION

Physical mapping: PDP or DDP

- Most physical mapping dealing with relatively short sequences (say a virus genome) correspond to the following problem:
  - if X is a set of points on a line and we know some of the differences $|x-y|$, with $x,y \in X$, deduce the position of the points in X on the line
  - Partial Digest Problem: we know all differences $|x-y|$ (experimentally difficult, its computational complexity unknown, behaves well in practice)
  - Double Digest Problem: we have two sets X,Y and we know the distance between all consecutive points in X. Same data for Y and for $X \cup Y$ – deduce the position of the points in X and Y

Difficulty of the problem

- Physical mapping is a difficult problem for a variety of reasons
  - Experimental errors
  - Lack of experimental data
  - Lack of coverage
  - It may be impossible to get from the available data one contiguous physical map
    - The pieces of map are called contigs

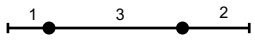
Errors in restriction site mapping

- Errors in gel electrophoresis:
  - Report the length with an error up to 5%
  - Small fragments may be ignored
  - Fragments with small difference in length may be reported having identical length
- Errors in digestion:
  - Some fragments may be completely lost
  - This leads to lack of coverage

Mapping through partial digest

Restriction site mapping by partial digest

- *Partial digest*
  - Subject the target DNA molecule to one enzyme
  - Several restriction sites of that enzyme may exist along the target molecule
  - Perform many digestion experiments with that enzyme acting on copies of the target, varying the exposure time (partial digest)
    - By allowing more or less time, the goal is to obtain at least one fragment for every pair of sites (count also the ends of the line as sites)
    - Example bellow: obtain fragments of length 1, 2, 3, 4, 5, 6
  - Based on this data, deduce the position of the restriction sites on the target DNA molecule
  - **NOTE: the problem is known in CS as the "turnpike reconstruction" problem (with exits on a highway)**



Restriction site mapping – models

- Partial digest problem (PDP):
  - X is a set of points on a line
  - The (multi)set of differences between ALL points in X is $D(X)=\{ |x_1-x_2|, x_1, x_2 \in X \}$
  - Reconstruct X based on the information given by D(X)
- (somewhat) Difficult problem
  - No polynomial algorithm is known
  - No proof that it is NP-complete
  - There are methods that prove fast in practice

A backtracking solution to PDP

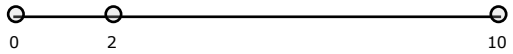
- Let N be the largest number in D(x); assume that 0 and N are in X and all other numbers in X are in between.
- Let $X_0=\{0,N\}$
- Idea: repeatedly position the longest remaining distance in D(X)
  - In each step, the longest sequence is realized from one of the outermost points of X. Thus, if m is currently the largest number in D(X), then add either m or N-m to X_0
  - We can only add m to X_0 if $|x-m| \in D(X)$, for all $x \in X_0$. In that case, we remove those differences from D(X). Similarly, we can only add N-m to X_0 if $|x-(N-m)| \in D(X)$, for all $x \in X_0$ and in that case remove those differences from D(X).
  - If neither option is possible, then we need to backtrack – remove from X_0 the last added point and continue the analysis
- Solution is found when D(X) gets empty

A backtracking solution to PDP

- The method only works if we have the full multiset of differences, i.e., we know how many fragments of a given size we have
- Computational complexity (Skiena, Smith, Lemke 1990)
  - Worst-time: $O(2^n n\log(n))$
  - Average-time: $O(n^2\log(n))$
- Good point in favor of this method: "rather small" number of possible solutions (Skiena, Smith, Lemke 1990)
  - For any D(X), let H(n) be the number of mutually homeometric (sets X with the same D(X)) sets of size n
  - $H(n)=2^k$ for some k, or else $H(n)=0$
  - $H(n)<1/2n^{1.2333}$

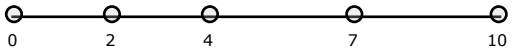
PDP – an example

- D(X)={2,2,3,3,4,5,6,7,8,10}
- Let L=D(X) and let the first point be x_1=0: X={0}
- 10∈L
  - The second point is x_2=10: X={0,10}
  - Take 10 out of L: L={2,2,3,3,4,5,6,7,8}
- 8∈L
  - Since X={0,10}, we have two possibilities: either 8∈X or 2∈X
  - The two possibilities to be considered separately – they lead to different solutions
  - In fact, they lead to "mirror/symmetric" solutions (the second one is identical to the first if we reverse the line)
  - Assume 2∈X: X={0,2,10}
  - Take 2 and 8 out of L: L={2,3,3,4,5,6,7}



PDP – an example

- X={0,2,10}, L={2,3,3,4,5,6,7}
- 7∈L
  - Two possibilities: either 7∈X or 3∈X
  - If 3∈X, then the difference 3-2=1∈D(X), a contradiction
  - Thus, 7∈X
- X={0,2,7,10}, take 7, 5, 3 out of L: L={2,3,4,6}
- 6∈L
  - Two possibilities: either 6∈X or 4∈X
  - If 6∈X: then the difference 7-6=1∈D(X), a contradiction
  - Thus, 4∈X
- X={0,2,4,7,10}, take 4, 2, 3, 6 out of L: L=∅



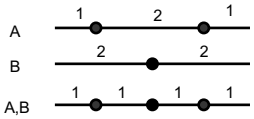
PDP – other examples

- It is crucial to have the multiset of differences rather than simply just the set of differences
- Example: assume D(X) is a given as a **set** – {2,16,18,20}
  - The backtrack method will give the answer "no solution"
  - Still, there is a solution: X={0,2,18,20}
  - This solution gives the **multiset** of differences {2,2,16,18,18,20}

Mapping through double digest

Restriction site mapping by double digest

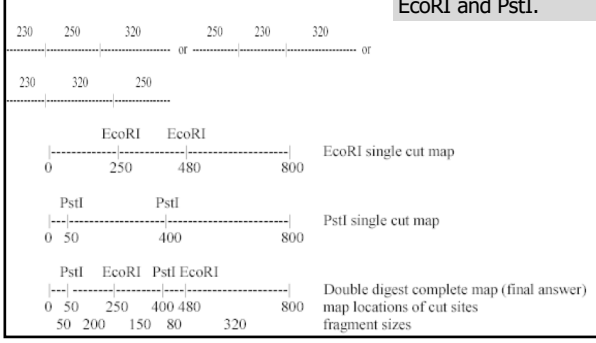
- Experimental part: subject the target DNA to complete digestion by two enzymes
  - Complete digestion by enzyme A of one copy of the target: two fragments of length 1 and one of length 2 in the example bellow
  - Complete digestion by enzyme B of another copy of the target: two fragments of length 2 in the example bellow
  - Complete digestion by enzymes A and B of another copy of the target: four fragments of length 1 in the example bellow
- Computational part: deduce the positions of the restriction sites of enzymes A and B along the target DNA molecule



DDP – **Example**

Restriction Enzyme	Fragment Lengths
EcoRI	230, 250, 320
PstI	50, 350, 400
EcoRI + PstI	50, 80, 150, 200, 320

An 800 base pair seqence of DNA was digested with EcoRI and PstI.



Restriction site mapping – models

- Double digest problem (DDP)
  - If Z is a set of n points on a line, denote by d(Z) the multiset of the n-1 distances between consecutive points of Z
  - Let A, B be two sets of points on a line and let X=A∪B
    - The outermost points of X are in both A and B; A and B have no other common points
  - Knowing d(A), d(B), and d(X), deduce the position of the points in A and B on the line

Restriction site mapping – models (2)

- Difficult problem: NP-complete
  - Easy to check a solution
  - DDP is a generalization of the set-partition problem
- The number of solutions may be *exponential*
- Difficult to cope with *coincidences*: enzyme A cuts in a point very close to enzyme B – some fragments will appear to have the same length
- Easy to get experimental data: complete digestion by two enzymes

Limitations of PDP and DDP

- They can only be applied to physical mapping of relatively small molecules, such as viral, chloroplast, or mitochondrial DNA – big difficulties for big molecules, both experimentally and computationally
  - For a large DNA molecule, biologists break into smaller pieces, map or fingerprint each fragment and then assemble the pieces together to determine the map of the entire molecule

Physical mapping of longer molecules

Physical mapping of longer molecules

- Mapping long molecules
  - Mapping starts with breaking DNA into small pieces using restriction enzymes – to study them biologists multiply them by cloning – the fragments are thus called clones
  - The clones in the clone library may overlap with each other, however the order of clones is lost
  - Process of reconstruction starts with fingerprinting the clones – describe each clone through an easily determined set of fingerprints (e.g., a set of keywords present in the clone). If two clones have substantial overlap, their fingerprints should be very similar.

Physical mapping of longer molecules

- Types of fingerprints used:
  - Restriction maps – the restriction map of a clone provides an ordered list of restriction fragments – used in 1987 for the physical map of E.coli
  - Restriction fragments sizes – this provides an unordered list of restriction fragments – used in 1986 in the yeast mapping project
  - Hybridization data – expose the clone to a number of probes (some short DNA sequences) and determine which of the probes bind to the clone – used in 1992 for the first physical map of the human genome. Consider this in details in the following

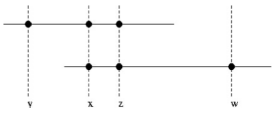
MAPPING BY HYBRIDIZATION

Mapping by hybridization

- Problem setup
  - Input: a set of DNA molecules coming from the same genome
    - Also called "probes" in this lecture
  - Output: a map giving their relative position on the genome
  - Tool to use: the hybridization of the probes on several fragments of the genome (clones)
    - "Fingerprints" of the clones

Mapping by hybridization

- The probes may be considered to be unique – each one occurs just once in the genome (such sequences exist, they are called STS – sequence tag site)
- If two clones share a large part of their fingerprints (about the same probes binding to them): they are likely to overlap in the target sequence



- Note: obtain only the *relative order* of the probes, not their exact *location*

A model for mapping by hybridization : the consecutive 1s model

- Assumptions for this model:
  - *Unique probes* – a probe can bind to a clone in at most one place (STSs)
  - *No errors* (errors considered later)
  - *All* clones x probes hybridization experiments have been done
- Data: a binary *n* x *m* matrix ($M_{ij}=1$ iff probe j hybridizes to clone i, otherwise $M_{ij}=0$)
- Problem: permute the columns (probes) such that *all 1s in each row are consecutive (the C1s property)*

A model for mapping by hybridization : the consecutive 1s model

- Intuition behind the method
  - The columns of the matrix correspond to probes
  - The rows of the matrix correspond to clones (contiguous sequences of DNA)
  - If we enumerate the probes in the order they occur on the target, then each clone has a sequence of 0 hybridization results, then a sequence of 1s, and then another sequence of 0s
  - Determining the order of the probes along the target molecule is the same as determining the permutation of columns which makes all 1s in each row to be consecutive

Example

Data offered by the lab (answers YES/NO to probe-clone hybridization, probes are marked with color dots):

Clone 1 hybridizes to the following probes: ○ ●

Clone 2 hybridizes to the following probes: ● ○

Clone 3 hybridizes to the following probes: ● ○

Lab data written as a binary matrix:

	●	○	○	●
1	1	1	0	0
2	0	1	0	1
3	1	0	1	0

Permute the columns to get the C1P:

	●	○	●	○
1	0	1	1	0
2	1	1	0	0
3	0	0	1	1

Result: the map of the molecule:

Algorithm for C1P

- Problem: Given an n x m binary matrix M, determine if M has the C1P for rows and if so, find the C1P permutation
- For each row i of M, let S_i be the set of columns k with $M_{ik}=1$ (those columns where there is 1 on row i)
- Take a look at how permuting the columns to group the 1s on one row affects other rows
- Each row can be thought of as a "witness" who gives a "statement" about which columns should be clustered together
  - "Interview" all witnesses

Algorithm for C1P

- For two rows i and j, there are three possibilities:
  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 is a subset of the other
- First case: we can clearly deal with rows i,j independently since they do not "interfere" with each other
- Second case: a row j such that $S_j \subset S_i$: we can deal with rows i, j separately since they do not "interfere" with each other
- Third case: i and j have to be treated simultaneously – they are "connected"
- Q: How to describe best these possibilities?
  - A: Graphs!
  - Build a graph corresponding to the matrix M
  - Vertices: the rows of M
  - Edge between i and j iff $S_i \cap S_j \neq \emptyset$ and none is a subset of the other (case 3 above)

Algorithm for C1P

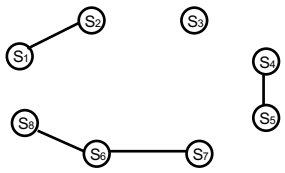
- Problem: Generate all permutations of columns that transform M into a matrix with C1s
- Basic idea:
  - Build the graph on the previous slide
  - For each connected component of the graph, describe the C1s permutations of columns for the rows in that component
  - Join the permutations for all connected component – this gives all C1s permutations for the whole matrix

Algorithm for C1P – an example

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
I_1	1	1	0	1	1	0	1	0	1
I_2	0	1	1	1	1	1	1	1	1
I_3	0	1	0	1	1	0	1	0	1
I_4	0	0	1	0	0	0	0	1	0
I_5	0	0	1	0	0	1	0	0	0
I_6	0	0	0	1	0	0	1	0	0
I_7	0	1	0	0	0	0	1	0	0
I_8	0	0	0	1	1	0	0	0	1

Solving C1P

- S_1={1,2,4,5,7,9}
- S_2={2,3,4,5,6,7,8,9}
- S_3={2,4,5,7,9}
- S_4={3,8}
- S_5={3,6}
- S_6={4,7}
- S_7={2,7}
- S_8={4,5,9}



Solving C1P: rows 6, 7, 8

- S_6={4,7}, S_7={2,7}, S_8={4,5,9}
- For row 6, any permutation that groups together columns 4 and 7 is fine (any order of 4 and 7)

			{4,7}	{4,7}		
S_6:	…	0	1	1	0	…

- For row 7, any permutation that groups together columns 2 and 7 is fine
  - Two choices to combine with row 6: either 2 goes to the right of 7 (and 4 to its left) or viceversa
  - They are symmetric – pick one

			{4}	{7}	{2}		
S_6:	…	0	1	1	0	0	…
S_7:	…	0	0	1	1	0	…

Solving C1P – rows 6, 7, 8

- S_6={4,7}, S_7={2,7}, S_8={4,5,9}

			{4}	{7}	{2}		
S_6:	…	0	1	1	0	0	…
S_7:	…	0	0	1	1	0	…

- For row 8, one needs to group together columns 4,5,9
  - Clearly, columns 5 and 9 must be placed to the left of 4 (the positions to its right are taken)
  - Columns 5 and 9 may be placed in any order

			{5,9}	{5,9}	{4}	{7}	{2}		
S_6:	…	0	0	0	1	1	0	0	…
S_7:	…	0	0	0	0	1	1	0	…
S_8:	…	0	1	1	1	0	0	0	…

  - Similarly done for the other components

Solving C1P

			{5,9}	{5,9}	{4}	{7}	{2}		
S_6:	…	0	0	0	1	1	0	0	…
S_7:	…	0	0	0	0	1	1	0	…
S_8:	…	0	1	1	1	0	0	0	…

	{1}	{2,4,5,7,9}	{2,4,5,7,9}	{2,4,5,7,9}	{2,4,5,7,9}	{2,4,5,7,9}	{3,6,8}	{3,6,8}	{3,6,8}	
S_1	0	1	1	1	1	1	1	0	0	0	0
S_2	0	0	1	1	1	1	1	1	1	1	0

		{1}	{5,9}	{5,9}	{4}	{7}	{2}	{3,6,8}	{3,6,8}	{3,6,8}
S_1:	0	1	1	1	1	1	1	0	0	0
S_2:	0	0	1	1	1	1	1	1	1	1
S_6:	0	0	0	0	1	1	0	0	0	0
S_7:	0	0	0	0	0	1	1	0	0	0
S_8:	0	0	1	1	1	0	0	0	0	0

C1P – algorithm

- **Describing all C1s permutations S for the rows in one connected component**
  - S will be a sequence of sets of integers (columns)
  - Initially S=∅
  - Traverse the component in depth-first. For each visited node i do
    - If this is the first node, then $S=S_i$
    - Let $U=S\cap S_i$, $V=S-S_i$, $W=S_i-S$
    - If the columns in U are not consecutive in S, then the component and the matrix do not have C1P, stop.
    - Same negative answer if there are integers $i,j\in V$ such that i occurs in S before the integers in U and j occurs after; stop.
    - If all integers in V occur before U in S, then S=RT, where $T=T_1T_2\ldots T_k$ consists only of integers in U and $T_1,\ldots,T_k$ are sets if integers.
      - If $T_1-U\neq\emptyset$, then replace T_1 in T with $(T_1-U)(T_1\cap U)$
      - Let S=SW
    - If all integers in V occur after U in S, then S=TR, where T=T1T2…Tk consists only of integers in U and T1,…,Tk are sets if integers.
      - If $T_k-U\neq\emptyset$, then replace Tk in T with $(T_k-U)(T_k\cap U)$
      - Let S=WS

C1P

- Complexity: O(mn)
- There is a better (but more involved) algorithm for this problem: O(m+n+r), where r is the total number of 1s in the matrix

The consecutive 1s model – discussion

- Critical (implicit) assumption so far: no errors
- Reality: there are errors of several different types

Hybridization mapping - errors

- False negatives: a probe may fail to bind where it should
- False positives: a probe may bind where it should not
  - Misreading the experimental results: *false negatives* or *false positives*
- Errors in the cloning process:
  - Two separate fragments may be joined together: chimeric clones (occurs frequently, up to 60% of the clones)
  - Deletion: an internal piece of a clone is lost – again two disjoint fragments are joined together
- Repeats in the genome: two clones may have the same fingerprint but they come from some repetitions in the genome
  - Use STSs
- Lack of data: impossible to perform all hybridization experiments

The consecutive 1s model - discussion

- In reality there are errors (in the lab one tries to minimize their number) – we need to build a model to *approximate* the solution
- Incorporate the errors into the model (i.e., find a solution which deviates as little as possible from the given data)
  - E.g., find a column permutation so that in each row there are at most k blocks of consecutive 1s (k=2,3)
  - Other idea: minimize the total number of blocks of 1 in the matrix
- The problems become *NP-complete*
- Other trouble: even if we find a C1P permutation, we cannot be sure that it is the "real" one: ideally, find all solutions

Hybridization mapping with errors

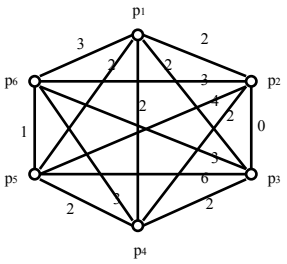
- No errors: a matrix with C1P
- A row corresponds to a chimeric clone (two fragments were joined) – two blocks of 1s separated by a number of 0 (a *gap*)
- We have a false negative on a row: we have a 0 separating two blocks of 1s – another gap
- Idea: errors – gaps
  - New problem: *Find a permutation of the columns to minimize the number of gaps*
  - Reduce the problem to another well-known optimization problem: TSP

Hybridization mapping with errors

- Problem: find a permutation of the columns to minimize the number of gaps in the rows of the matrix
- Reduce the problem to finding a Hamiltonian cycle of minimum weight
  - Add a rightmost column filled with 0s and build a graph as follows:
    - The graph is complete, a vertex for each column (probe)
    - The weight on each edge: the number of rows where the two columns differ (*the Hamming distance of the two columns*)
  - The idea of the additional column comes from two sources:
    - we will consider circular permutations of columns in such a way that if we have i blocks of 0 in a row, the "penalty" for that row will be exactly 2i (1 for entering the block of 0s, 1 for exiting it)
    - circular permutations will not penalize a row that starts with a block of 1 and ends with another – introduce an artificial all-zero last column

Example

	p_1	p_2	p_3	p_4	p_5	p_6
C_1	1	1	1	0	0	0
C_2	0	1	1	1	0	0
C_3	1	0	0	1	1	0
C_4	1	1	1	1	0	0



Hybridization mapping with errors

- Intuition: a cycle in the graph corresponds to a possible ordering of the columns
- Q: What is the relation between the gaps in rows and the weight of a cycle?
  - Gap in a row: a transition from 1 to 0 and a transition from 0 to 1 – add 2 to the weight of the cycle
  - Do not count the transitions from the last column to the first one – add the 0 column mentioned above
- Cycle weight = 2 x number of gaps + 2n
- Minimizing the number of gaps in rows is the same to minimizing the weight of the cycle:

Hybridization mapping with errors

- Result: if we want to find a column permutation giving a minimal number of gaps, find a Hamiltonian cycle with minimum weight (equivalent to TSP in the case of complete graphs)
  - Reduce the gap minimization problem to TSP, solve the corresponding TSP problem
  - TSP is NP-complete
- Use the many approximating techniques available for TSP, e.g. the Greedy approximation
- Question: *having a good approximation for TSP, do we get a good approximation for the gap minimization problem ?*
- Answer: *YES!*
  - Cycle weight = 2 x number of gaps + 2n

Summary

- Two methods to yield data for DNA mapping
  - Digestion by restriction enzymes
  - Hybridization experiments
- The computational problems
  - Measure and compare fragment lengths – PDP, DDP; DDP is NP-complete
  - C1P for hybridization matrix: no errors - polynomial algorithm, errors - NP-hard problems, approximations, reductions to other problems, prove that the reduction is good