Lectures 15, 16, 17, and 18: Microarrays

Steven Skiena

Department of Computer Science
State University of New York
Stony Brook, NY 11794–4400

http://www.cs.sunysb.edu/∼skiena

Faster, Better, Cheaper

Biology used to be a hypothesis driven science...
But one of the major themes driving contemporary genomics research is the idea of doing experiments to capture raw data on “complete” state of a particular system.
Obtaining such massive amounts of data requires increases in automation and parallelism over traditional experiments.
Laboratory techniques exist to enable biologists to measure the *expression* of a single gene under certain conditions.
Microarray technology enables one to do such experiments on a vastly larger scale.

Random Access and Arrays

Suppose you want to do millions of different chemical reactions simultaneously.

You can’t do them in single test tube, because how do you know where to look for the results of any given reaction?

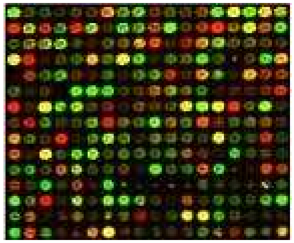
To exploit parallelism, you need to be able to associate each intended reaction with a location or address.

The traditional solution is to do experiments in 96-well or 384-well *plates*, where each chamber is kept separate from each other.

Coordinating the results from many plates involves careful labeling, e.g. with barcodes.

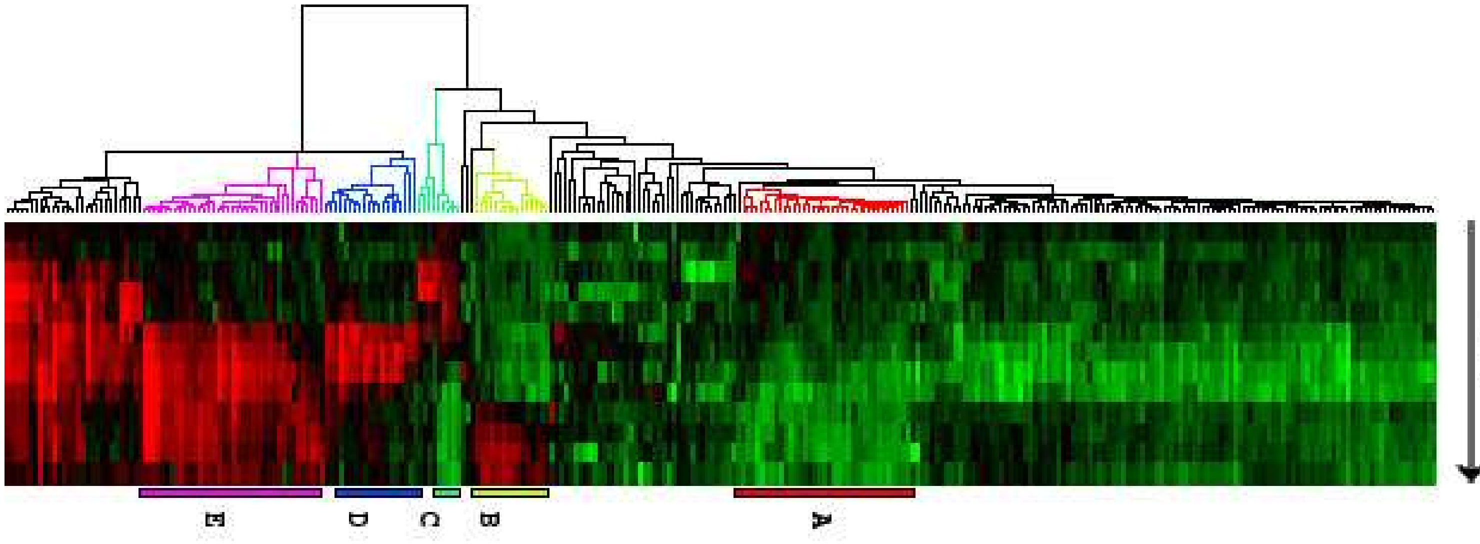
Certain technologies have been developed where different compounds are anchored to tiny *beads*, so reacting beads can be eye-balled, isolated, and identified.

But the best solution is to attach distinct compounds to different regions of a solid substrate so you know *where* they are.



What Are They Good For?

Identification of genes involved in the cell division cycle for yeast:



Sequencing variants of a *known* genome, for detecting single nucleotide polymorphisms (SNPs) or identifying a specific strain of virus (e.g. the Affymetrix HIV-1 array).

Measuring differential expression of all genes in tumor and normal cells, to determine which genes may cause/cure cancer, or identify which treatment a specific tumor should respond best to.

Measuring differential expression of all genes in different tissue types, to determine what makes one cell type different than another.

Measuring copy number variants from chromosomal anomalies or cancer.

Obtaining individual’s genotype / SNP data, e.g. 23andMe.

Microarray Technology

Single stranded DNA/RNA molecules are anchored by one end to the plate/substrate. These molecules will seek to hybridize with complementary strands floating in solution.
The target molecules are fluorescently labeled, so that the spots on the *chip/array* where hybridization occurs can be identified.
The strength of the detected signal somewhat reflects the amount of stuff which binds to it, and thus the amount of the target in solution. Such *quantitative* expression data is not very reliable, however.

Relative Target Analysis

More accurate data comes from comparing the *relative* amounts of expressed DNA/RNA in two related samples, each of which is colored with a different fluorophore. The ratio of green/red tells us about the relative expression in both samples.

Sources of Error

There are several factors which lead to errors in microarray hybridization data, beyond obvious manufacturing defects and experimental errors.

The strength of the bond formed between two single stranded DNA/RNA molecules is a function of (1) the length of the bonded molecules, (2) the base composition of the molecules, since A/T and C/G bond with different energies, (3) the number and location of base mismatches, since end mismatches cause less trouble.

Cross hybridization is a source of many false positive errors, where a closely related DNA sequence binds at the probe in the absence of the desired target.

Heat breaks these bonds, so the *stringency* of hybridization can be effected by changing the temperature and other conditions.

Self hybridization occurs when probe molecules fold and hybridize with themselves, thus rendering them less effective at hybridizing with the target. This occurs particularly in *self-palindromic* probes.

Spotted Microarrays

There are two distinct but important types of DNA/RNA array technology.

Spotted microarrays, pioneered by Pat Brown at Stanford, lay down rows of tiny drops from racks of previously prepared DNA/RNA samples.

Extensive automation makes it possible to build small glass slides containing tens of thousands of different probe spots.

Any given DNA/RNA probe can be hundreds of bases long, and in principle made from any DNA/RNA sample.

Thus reasonable spotted arrays might contain all genes in a given organism, or all EST fragments from a library.

Affymetrix Arrays

Affymetrix arrays are *synthesized* by using the same light mask technology that silicon chips are manufactured.

They exploit *photo-sensitive* reactions to (1) remove a blocking group at the end of a molecule, and (2) to extend the molecule with a given blocked base.

Thus *any* subset of DNA k-mers can be built up using at most $4k$ reactions.

Affymetrix arrays are typically limited to oligos of $k \approx 25$, but can contain hundreds of thousands of probes.

It is expensive to fabricate the masks for a new array design, so they win only if you can sell many copies.

Other Array Technologies

Agilent Technologies manufactures array makers using ink-jet printer technology.

These mimic the Affymetrix synthesis approach of growing another base at the end of a molecule by (1) deprotecting the end, and (2) attaching a new base with a protected end to avoid duplication.

These are exciting because it becomes practical to synthesize only one instance of a given array design.

Other companies use similar technologies on addressable *beads* to get around patent issues concerning arrays of oligonucleotides – or locations in friendly countries (e.g. Iceland).

Image Processing Issues

Each array image contains so much information it must be read by a computer, not a person.

The first step of image processing is *gridding*, identifying the mapping between the image and the locations of the spots on the array.

The careful mechanical construction of these arrays makes this a tractable problem.

Determining the hybridization level of a spot is not just a function of intensity, but a statistical analysis of *control probes* and *redundant* probes for the specific target.

But your data analysis has only begun once you have the hybridization level for each spot. . .

Linkage Analysis and Microarrays

Many genetic diseases (e.g. Huntington's Disease, Tay-Sachs) are caused by inheriting defective versions (allele) of genes from one or both parents, who in turn inherited them from their parents…

Trying to figure out which genes actually cause the disease is complicated by several factors, including (1) the phenotype may not show up unless you get some combination of bad alleles, (2) about 1/4 of your genes come from each grandparent, so there are many possible candidates and (3) how do you tell which alleles you got from whom?

Recombination and Linkage Analysis

Through recombination, each chromosome you inherit from Mom is a random mixture of the corresponding chromosomes of her parents.

In this mixture, alternations between parents occurs relatively rarely, so if you inherit given gene from grandma, likely the neighboring gene also came from grandma.

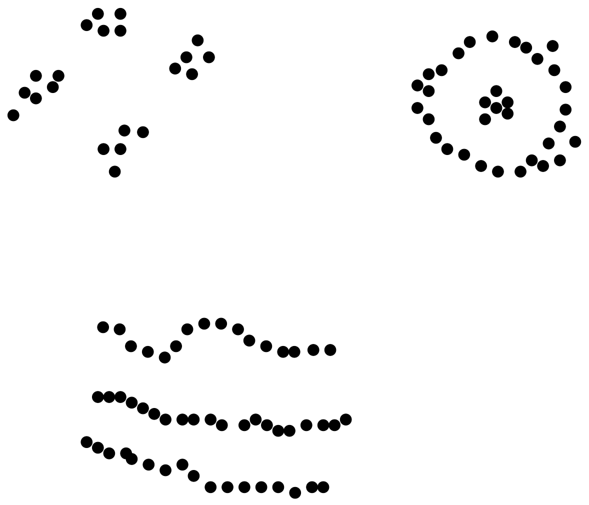
Linkage analysis mapped diseases to approximate locations on chromosomes bases on (1) pedigree analysis (i.e. family trees annotated with the presence or absence of disease), and (2) experimental data describing which flavor of genetic marker you have at each of a few hundred positions.

This analysis is done through very computationally-intensive linkage analysis programs.

Specially-designed linkage analysis microarrays with hundreds of thousands of markers will enable researchers to gather very accurate measurements of exactly where the crossovers were and what you inherited from each parent.

Data Clustering

Clustering is an inherently ill-defined problem since the correct clusters depend upon context and are in the eye of the beholder.



How many clusters are there?

Biological applications of clustering include grouping sequences/ESTs by similarity, genes by similar expression patterns, and samples by tissue type/disease.

Biological clustering is often associated with *dendograms* or phylogenic trees, although there is no reason there *need* be a tree associated with clusters.

Distance Measures

Certain mathematical properties are expected of any distance measure, or *metric*:

1. $d(x, y) \geq 0$ for all x, y.
2. $d(x, y) = 0$ iff $x = y$.
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, y) \leq d(x, z) + d(z, y)$ for all x, y, and z. (triangle inequality)

Euclidean distance $d(x, y) = \sqrt{\Sigma_{i=1}^{d} |x_i - y_i|^2}$ is probably the most commonly used metric. Note that it weights all features/dimensions “equally”.

Variations on Euclidean distance include L_k norms for $k \neq 2$:

$$d(x,y) = (\sum_{i=1}^{d} |x_i - y_i|^k)^{(1/k)}$$

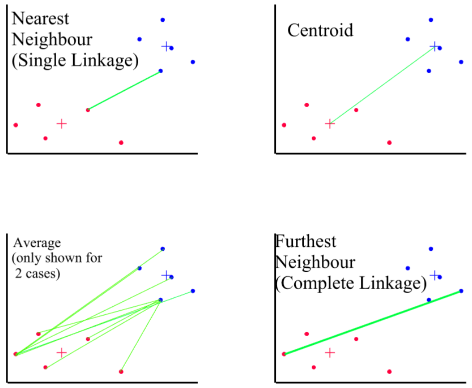
We get the "Manhattan" distance metric for $k = 1$ and the "maximum" component for $k = \infty$.

The *correlation coefficient* of sequences X and Y, yield a number from -1 to 1 but is *not* a metric.

The loss of the triangle inequality in non-metrics can cause strange clustering behavior, particularly in single linkage methods.

Distance Measures Between Clusters

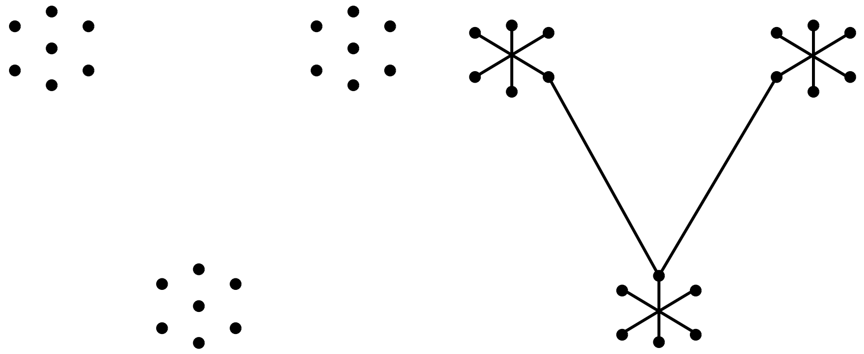
There are several ways to measure the distance between two clusters:



The choice of measure implies a tradeoff between computational efficiency and robustness.

Agglomerative Clustering

Such methods merge smaller clusters into bigger clusters. The simplest merging criteria is to join the two *nearest* clusters.
Minimum spanning tree methods define *single-link* clustering.



But does nearest mean the closest pair of examples? The closest mean/median examples?

In *complete link* clustering, we merge the pair which minimizes the maximum distance between elements. This is more expensive ($O(n^3)$ vs. $O(n^2)$), but presumably more robust.

k-Means Clustering

k-*Means clustering* is an iterative clustering technique where you specify the number of clusters k in advance.

Pick k points, and assign each example to the closest of the k points.

Pick the ‘average’ or ‘center’ point from each cluster, and repeat until sufficiently stable.

Note that such center points are not well defined in clustering non-numerical attributes, such as color, profession, favorite type of music, etc.

How Many Clusters?

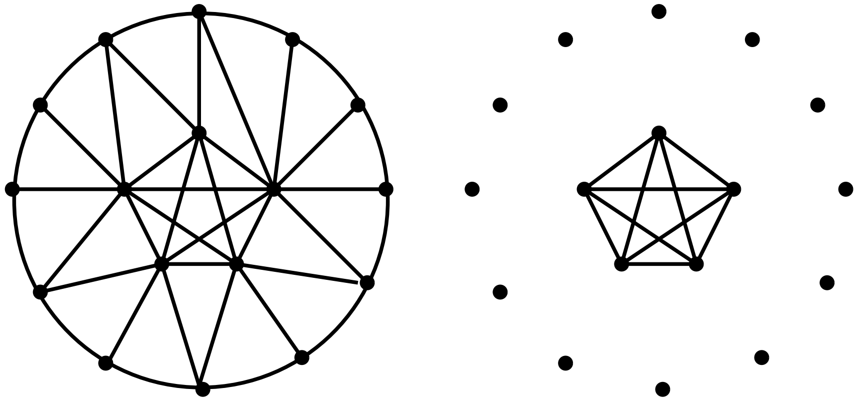
The question of how many clusters you have (or expect) is inherently fuzzy in most applications. However, the “right” number of clusters, adding another cluster should not dramatically reduce distance from the centers.

Such techniques represent a *partitioning*-based strategy, which is dual to the notion of agglomerative clustering.

Graph-Theoretic Clustering Methods

An alternate approach to clustering constructs an underlying *similarity* graph, where elements i and j are connected by an edge iff i and j are similar enough that they should/can be in the same cluster.

If the similarity measure is totally correct and consistent, the graph will consist of disjoint cliques, one per cluster.



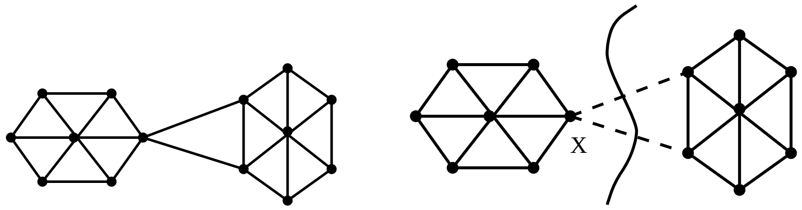
Overcoming spurious similarities reduces to finding maximum-sized cliques, an NP-complete problem.

Overcoming occasional missing edges means we really seek to find sets of vertices inducing dense graphs.

Repeatedly deleting low-degree vertices gives an efficient algorithm for finding an induced subgraph with minimum degree k, if one exists.

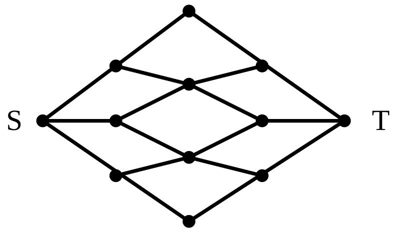
Cut-Based Algorithms

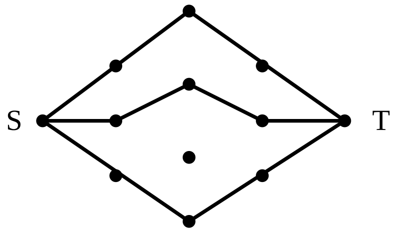
A real cluster in such a graph should be easy to disconnect by deleting a small number of edges. The *minimum edge cut* problem asks for the smallest number of edges whose deletion will disconnect the graph.



The *max flow / min cut theorem* states that the maximum possible flow between vertices i and j in a weighted graph G is the same as the minimum total edge weight needed to separate i from j.

Thus network flow can be used to find the minimum cut efficiently.



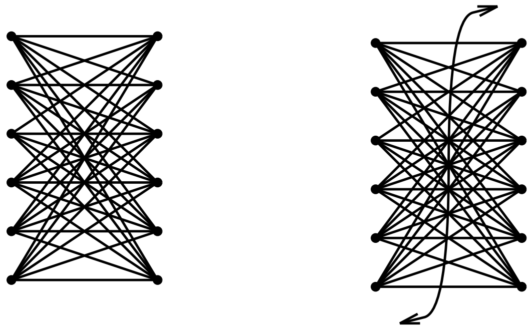


However the globally minimum cut is likely to just slice off a single vertex as opposed to an entire cluster.

Graph Partitioning Approaches

Graph partitioning problems which seek to ensure large components on either side of the cut tend to be NP-complete. By *complementing* graph G, we get a graph G' with an edge (i, j) in G' iff the edge did not exist in G.

By finding the *maximum cut* in G' and recurring on each side, we can have a top-down (i.e. divisive) clustering algorithm.



Finding the maximum cut is NP-complete, but a simple algorithm gives a cut at least half as big as the optimal cut:

For each vertex, flip a coin to decide which side of the cut (left or right) it goes. This means that edge (i, j) has a 50% chance of being cut (yes if coins i and j both came up either heads or tails).

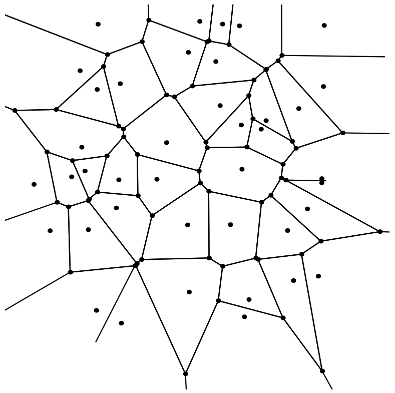
Local improvement heuristics can be used to refine this meaningless cut.

Nearest Neighbor Clustering

Assign each point to its nearest neighbor who has been clustered if the distance is sufficiently small.

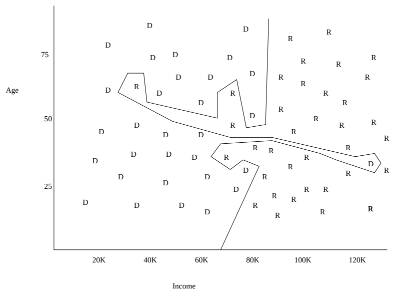
Repeat until there are sufficient number of clusters.

Nearest neighbor *classifiers* assign an unknown sample the classification associated with the closest point.



Conceptually, all classifiers carve up space into labeled regions. *Voronoi diagrams* partition space into cells defining the boundaries of the nearest neighbor regions.

Nearest neighbor classifiers are simple and reasonable, but not robust. A natural extension assigns the majority classification of the k closest points.



An important task in building classifiers is avoiding *over-fitting*, training your classifier to replicate your data too faithfully.

Other classifier techniques include *decision trees* and *support-vector machines*.

Decisions with Clustering

How many clusters *should* there be? (hint: eyeball it or look for large distances)
How do we visualize large, multidimensional datasets? (hint: read Tufte's books)
How do we handle multidimensional datasets? (hint: dimension reduction techniques such as principle component analysis and singular-value decomposition (SVD))
How do we handle noise? (hint: aim for robustness by avoiding single-linkage methods and proper distance criteria)

Which clustering algorithm should we use? (hint: this probably doesn’t make as much difference as you think).

Which distance measure should we use? (hint: this is your most important decision)

Microarray Software

Affymetrics supplies image analysis and data analysis software for customers of its microarrays.

Several third-party bioinformatics companies (e.g. Scanalytics, Silicon Genetics, Compugen Lion Bioscience) supply such software applicable for spotted microarray systems.

Data mining software uses clustering and other statistical methods to identify interesting features in microarray and heterogeneous data sets.

Websites such as Netaffx (www.affymetrix.com) provide links between various public databases, enabling you to click on genes that look interesting in your microarray data and find out what is known about them.

Cluster/TreeView

Developed by Michael Eisen, this is the original microarray clustering software.

Provides a computational and graphical environment for analyzing data from DNA micorarray experiments

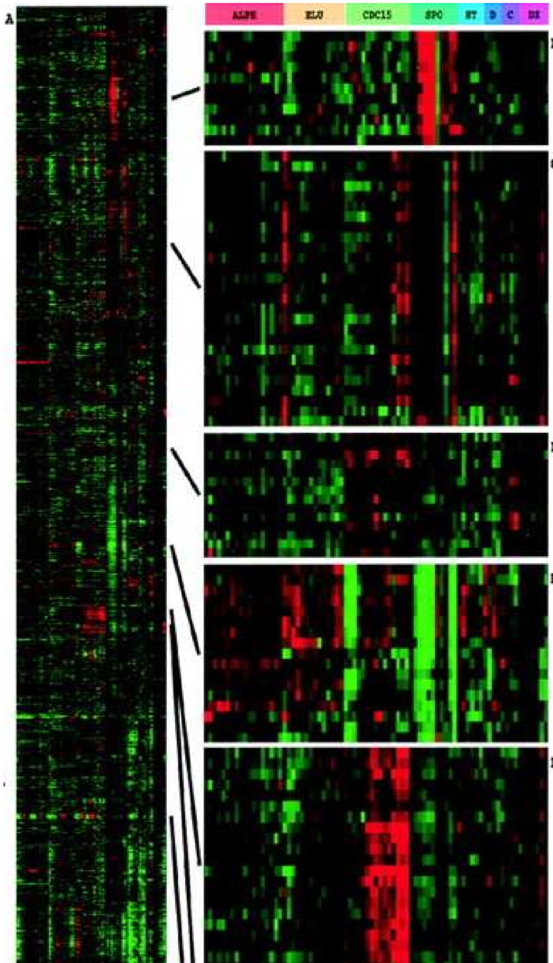
Cluster organizes and analyzes the data in a number of ways, clustering either genes or experiments by similarity.

TreeView allows interactive graphical analysis of the results from Cluster.

Cluster uses the correlation coefficient (-1 to 1) of *logarithmically normalized* primary data for each gene. Since this primary data is a ratio of red/green, this is zero for equal expression, positive for up-regulated expression, and negative for down-regulated expression.

$$corr(X,Y) = \frac{\Sigma_i(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\Sigma_i(x_i - \bar{X})^2}\sqrt{\Sigma_i(y_i - \bar{Y})^2}}$$

The denominators are the standard deviations of the two sequences, to normalize them. Positive correlation is achieved when both measurements on the same size of their respective means.



B	STU2	CYTOSKELETON	SPINDLE POLE BODY COMPONENT
	DIN7	DNA REPAIR	EXONUCLEASE; ALSO RECOMBINATION
	BNR1	CYTOSKELETON	ACTIN FILAMENT ORGANIZATION
	SPC42	CYTOSKELETON	SPINDLE POLE BODY COMPONENT
	CNM67	CYTOSKELETON	SPINDLE POLE BODY COMPONENT
	CLB4	CELL CYCLE	G2/M CYCLIN
	CDC10	CYTOKINESIS	GTP BINDING PROTEIN
	CDC3	CYTOKINESIS	SEPTIN
	CLB3	CELL CYCLE	G2/M CYCLIN
	APC4	CELL CYCLE	ANAPHASE-PROMOTING COMPLEX SUBUNIT
	CDC16	CELL CYCLE	ANAPHASE-PROMOTING COMPLEX SUBUNIT
C	RPN11	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	UFD1	PROTEIN DEGRADATION	UBIQUITIN FUSION DEGRADATION
	RPN9	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	RPT1	PROTEIN DEGRADATION	26S PROTEASOME SUBUNIT
	RPT6	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	PRE4	PROTEIN DEGRADATION	PROTEASOME SUBUNIT, B TYPE
	RPN6	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	RPT4	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	RPN7	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	RPN3	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	PUP2	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT ALPHA5
	SCL1	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT YC7ALPHA/Y8
	PRE5	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT ALPHA6
	PRE9	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT Y13 ALPHA3
	PRE1	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT C11 BETA4
	PRE2	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT BETA5
	PRE3	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT BETA1
	PRE10	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT C1 ALPHA7
	PUP1	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT BETA2
	PRE6	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT ALPHA4
	PRE7	PROTEIN DEGRADATION	20S PROTEASOME SUBUNIT
	RPN10	PROTEIN DEGRADATION	26S PROTEASOME SUBUNIT
	RPT3	PROTEIN DEGRADATION	26S PROTEASOME SUBUNIT
	RPT5	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	RPN12	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
	RPN5	PROTEIN DEGRADATION	26S PROTEASOME SUBUNIT
	RPN8	PROTEIN DEGRADATION	26S PROTEASOME REGULATORY SUBUNIT
D	POP6	TRNA PROCESSING	RNASE P AND RNASE MRP SUBUNIT
	CAF16	TRANSPORT	ATP-BINDING CASSETTE ABC FAMILY
	MSL1	MRNA SPLICING	UNKNOWN
	SMD3	MRNA SPLICING	CORE SNRNP PROTEIN
	TAF40	TRANSCRIPTION	TFIID 40 KD SUBUNIT
	PET309	RNA PROCESSING	COX1 MRNA STABILITY
	PZF1	TRANSCRIPTION	TFIIIA
	YSH1	MRNA 3 -END PROCESSING	CLEAVAGE/POLYADENYLATION FACTOR CF II COMPONENT
	CBP1	MRNA STABILITY	UNKNOWN
	PRP24	MRNA SPLICING	U4/U6 SNRNP PROTEIN
	STD1	GLUCOSE REPRESSION	MODULATOR OF GLUCOSE REPRESSION
	MGM1	MITO GENOME MAINT	DYNAMIN FAMILY PROTEIN
	PRP19	MRNA SPLICING	NON-SNRNP SPLICEOSOME COMPONENT
	SLS1	MITOCHONDRIAL METABOLISM	INTEGRAL MEMBRANE PROTEIN
E	TPI1	GLYCOLYSIS	TRIOSEPHOSPHATE ISOMERASE
	GPM1	GLYCOLYSIS	PHOSPHOGLYCERATE MUTASE
	PGK1	GLYCOLYSIS	PHOSPHOGLYCERATE KINASE
	TDH3	GLYCOLYSIS	GLYCERALDEHYDE-3-PHOSPHATE DEHYDROGENASE 3
	TDH2	GLYCOLYSIS	GLYCERALDEHYDE-3-PHOSPHATE DEHYDROGENASE 2
	ENO2	GLYCOLYSIS	ENOLASE II
	TDH1	GLYCOLYSIS	GLYCERALDEHYDE-3-PHOSPHATE DEHYDROGENASE 1
	FBA1	GLYCOLYSIS	ALDOLASE
	TKL1	PENTOSE PHOSPHATE CYCLE	TRANSKETOLASE
	PDC5	GLYCOLYSIS	PYRUVATE DECARBOXYLASE
	PDC6	GLYCOLYSIS	PYRUVATE DECARBOXYLASE 3
	PDC1	GLYCOLYSIS	PYRUVATE DECARBOXYLASE
	CDC19	GLYCOLYSIS	PYRUVATE KINASE
	HXK2	GLYCOLYSIS	HEXOKINASE II
	TYE7	GLYCOLYSIS	BASIC H-L-H TRANSCRIPTION FACTOR
	PFK1	GLYCOLYSIS	PHOSPHOFRUCTOKINASE
	ACS2	ACETYL-COA BIOSYNTHESIS	ACETYL-COENZYME A SYNTHETASE
F	DBI56	UBIQUINONE BIOSYNTHESIS	METHYLTRANSFERASE
	MSS51	MRNA SPLICING	UNKNOWN
	MSF1	PROTEIN SYNTHESIS	MITOCHONDRIAL PHENYLALANYL-TRNA SYNTHETASE SUBUNIT
	MRPS5	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL S5
	MSM1	PROTEIN SYNTHESIS	MITOCHONDRIAL METHIONYL-TRNA SYNTHETASE
	COX10	RESPIRATION	CYTOCHROME-C OXIDASE ASSEMBLY
	MSK1	PROTEIN SYNTHESIS	MITOCHONDRIAL LYSYL-TRNA SYNTHETASE
	MEF1	PROTEIN SYNTHESIS	TRANSLATION ELONGATION FACTOR G, MITOCHONDRIAL
	NONE	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL LARGE SUBUNIT
	MRPL35	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL L35
	MRP7	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL LARGE SUBUNIT
	NONE	RESPIRATION	MITOCHONDRIAL ACIDIC MATRIX PROTEIN
	COX17	RESPIRATION	CYTOCHROME OXIDASE ASSEMBLY
	MRP17	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL SMALL SUBUNIT
	MRPL6	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL L6
	MRPL8	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL L8
	RML2	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL L2
	MRPL13	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL L13
	MRPL9	PROTEIN SYNTHESIS	RIBOSOMAL PROTEIN, MITOCHONDRIAL L9

Algorithmics of Cluster/TreeView

This agglomerative clustering algorithm uses the average-linkage method of Sokal and Michener. Each cluster is assigned a gene-expression profile by, for each probe averaging all of the values of genes in its cluster. The profile of a merged cluster is determined by averaging the two component clusters weighted by the number of non-missing values in each cluster.

These clusters are displayed by permuting the rows of the matrix to reflect the structure of the tree/dendogram. A heuristic solution for this TSP-like problem is used to order the genes so as to help visualize the clusters.

Note that there are 2^{n-1} ways to permute the n leaves of any binary tree while leaving a non-intersecting drawing. A polynomial-time dynamic programming can be used to find the best possible ordering:
Let $C[r, i, j]$ be the cost of the cheapest way to permute the subtree with root r such that gene i is on the left and gene j is on the right. Then

$$C[r, i, j] = \min_{k,l} C[child_1, i, k] + C[child_2, l, j] + cost(k, l)$$

SBH: The Vision

The original proposed application for hybridization arrays was *de novo* DNA sequencing.

The idea was to build an array with a large number of short probes, hybridize the target against it, and read off the sequence by knowing which subsequences were and *were not* present.

Shotgun sequencing gives you longer strings which are present in the sequence, but no information as to what is not present.

These extra combinatorial constraints should be very helpful in unraveling the sequence.

SBH: The Reality

Important classes of Affymetrix arrays are used to *resequence* variants of known sequences.

Single nucleotide polymorphisms (SNPs) are common one-base DNA differences among members of a given species.

Such minor differences in coding sequences may or may not have important consequences for particular diseases.

Much of the genetic variation among humans consists of SNPs.

The Affymetrix HIV-1 array attempts to determine the exact strain of the virus.

Each of the first $9,718 - 24$ starting positions in the sequence of HIV-1 defines a particular 25-mer.

For each of these 25-mers, we can construct four 25-mer probes by trying all possibilities (A,C,G,T) of the 13th base. The correct one should clearly have the strongest signal.

Such a probe set will have trouble if there are more than one SNP in a 25-mer, however.

SBH: The Theory

Applying SBH to *de novo* sequencing is a different problem, because we do not have any information as to what k-mers should be in the sequence.

Thus we might construct an array $C(k)$ asking *all* k-mers for the largest possible k.

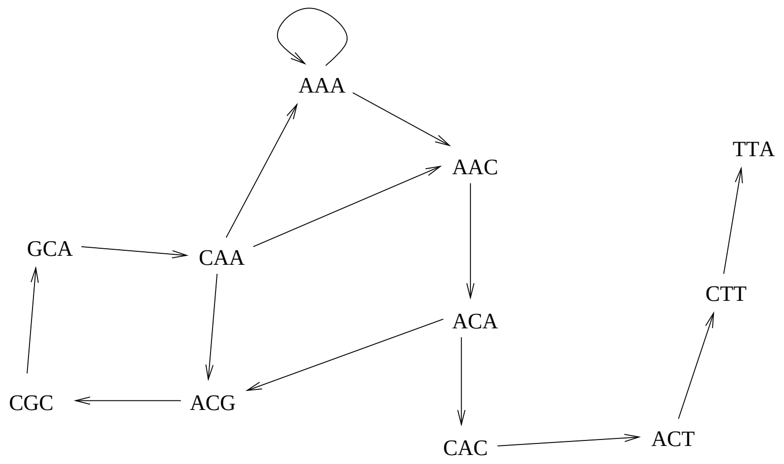
Asking *all* $4^{20} \approx$ one trillion 20-mers is clearly impossible with current arrays sizes.

Such classical hybridization arrays up to all $4^{10} \approx$ one million 10-mers have been built by Affymetrix.

How can we reconstruct a sequence by knowing which k-mers are and *are not* in the sequence?

Hamiltonian Paths

Suppose that exactly the following 3-mers occur in the unknown sequence: *AAA*, *AAC*, *ACA*, *CAC*, *CAA*, *ACG*, *CGC*, *GCA*, *ACT*, *CTT*, *TTA*.



We can construct a graph where the vertices correspond to positive k-mers, with a directed edge between two k-mers which overlap in $k-1$ positions.

We seek a path that visits each vertex at least (and hopefully at most) once to define a sequence consistent with the data.

This is the *Hamiltonian Path* problem, so it is NP-complete to find such a tour.

Interpreting the Spectrum

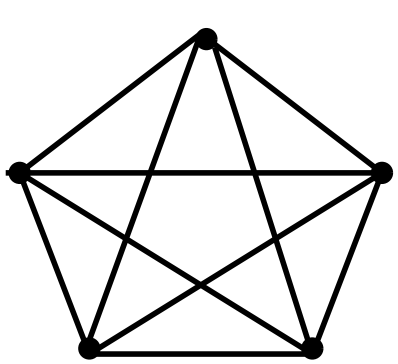
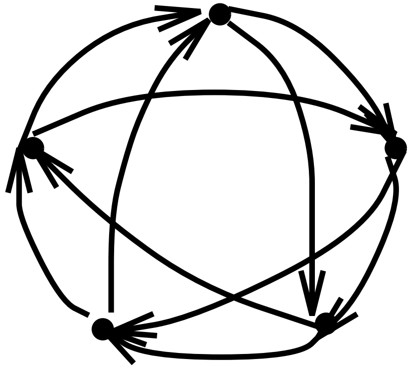
Multiple Hamiltonian paths means that the sequence reconstruction is not uniquely defined by its *spectrum*. Frequency counts for each oligo can help resolve ambiguity.

The length of the oligos, k, must be large enough to span all repeats if the sequence is to be reconstructed unambiguously.

This formulation is general enough to be extended to reconstruction in the presence of hybridization errors by appropriately weighting edges and asking for a minimum cost tour.

Eulerian Paths

An *Eulerian path/cycle* visits every *edge* of a graph exactly once.

A connected, directed graph has an Eulerian cycle iff every vertex has indegree equal to its out-degree.
Thus there is an efficient algorithm to find Eulerian cycles in graphs.

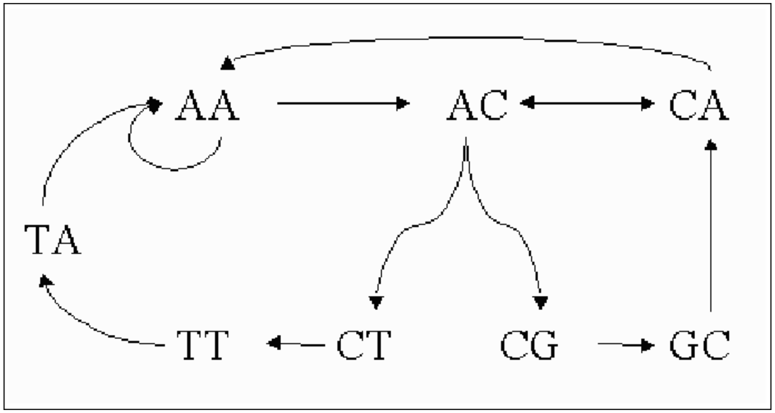
Note that any pair of cycles in an Eulerian cycle can be exchanged to create a different cycle. Thus one can't make a mistake by extracting a cycle.

de Bruijn Graphs

There is a clever way to reduce the SBH reconstruction problem to finding an Eulerian cycle on a subgraph of the *de Bruijn graph* of order k.

We will represent every k-mer as an *edge* of the graph.

The vertices will represent all $(k-1)$-mers, with k-mer S being the edge between the vertex defined by the first $(k-1)$ characters of S and the last $(k-1)$ characters of S.



This only works because the strings restrict the set of possible Hamiltonian cycle graphs to highly structured examples.

Resolving Power

The sequence a^{k+1} cannot be distinguished from a^{k+2} using just k-mer probes.

No sequence of length greater than $4^k + k$ can be unambiguously sequenced with $C(k)$, since (by the pigeonhole principle) at least one fragment will repeat.

Expected resolving lengths are even shorter, *even assuming no hybridization errors*:

Fragment	k	Probes	$4^{k/2}$
80	7	16,384	128
180	8	65,536	256
260	9	262,144	512
560	10	1,048,576	1024
1300	11	4,194,304	2048
2450	12	16,777,216	4096

We can estimate the resolving fragment length by picking the k such that the probability of any k-mer repeating ($\approx n/4^k$) is sufficiently small. For $k = (\lg_4 n)/2$ this probability goes to $1/n$.

From an *information theory* perspective, an experiment which maximizes resolving power should have about half positive and half negative probes – but almost all probes will be negative for any given sequence.

SBH: The Variants

A large variety of variations on the basic SBH technology have been proposed with the goal of increasing the size of the fragments which can be reconstructed.

In *positional* SBH, we also get information about roughly where each fragment exists on the final sequence through auxiliary data.

In *interactive* SBH, we use ink-jet technology to design new arrays in response to previous results to quickly converge on the correct sequence.

Alternate chip designs using *universal bases* which hybridize at any base can theoretically increase resolving power using fixed length probes with “don’t care” characters.