

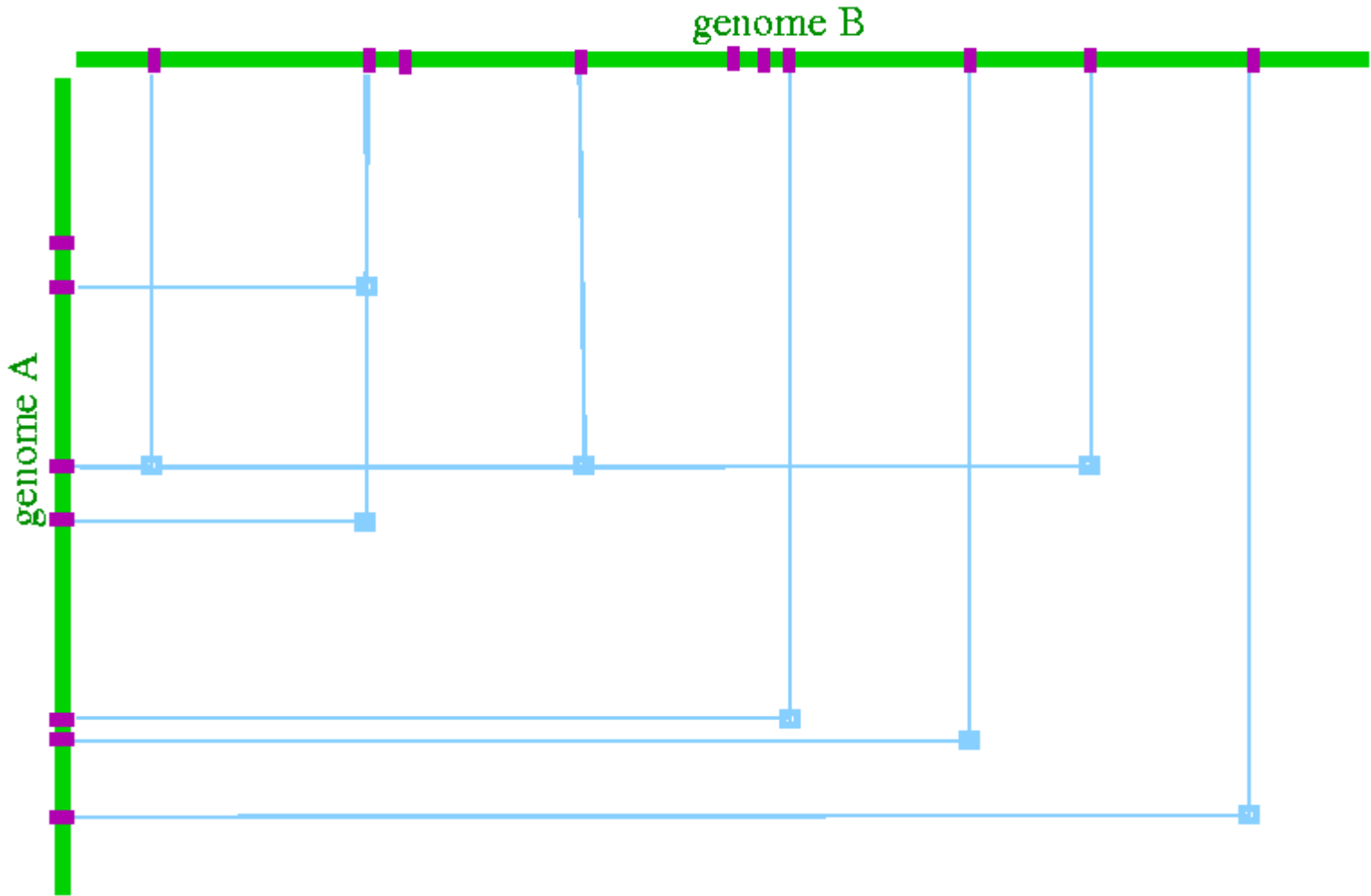
*(problems in)*  
Whole genome alignment  
and  
Mass spectrometry

Ross A. Lippert

# What I will talk about

- Whole genome alignment
  - For translations or comparisons
  - Finding “trustworthy” matches
    - Repeats
    - Calibration
  - Chaining
- Fast peptide identification
  - The nature of the data
  - The nature of the queries
  - The problem of short-sightedness

# Whole Genome Alignment



# Whole Genome Alignment

- Why we care?
  - Annotation translation
  - Cross species comparison
- How do we do it?
  - Seed selection
  - Extend/Filling-in
- Statistical observations
  - Indications for future analysis
- Techniques for Calibration

# Feature translation

- A growing number of genome assemblies
  - HGP drafts of human
  - Celera's private/public/semi-public drafts of human
  - Other mammalian genomes
  - Identification of SNPs and causes of phenotypic differences
- A large and growing annotation
  - Draft dependant
  - Species dependant

***Annotations must be translated  
between drafts***

# The Synteny Problem

- Between distant species can reveal function
  - Conservation reveals selective pressure
- Between near species
  - Conservation reveals evolutionary history
- Between similar or the same species
  - Recent events in subpopulations
  - Phenotypic differences

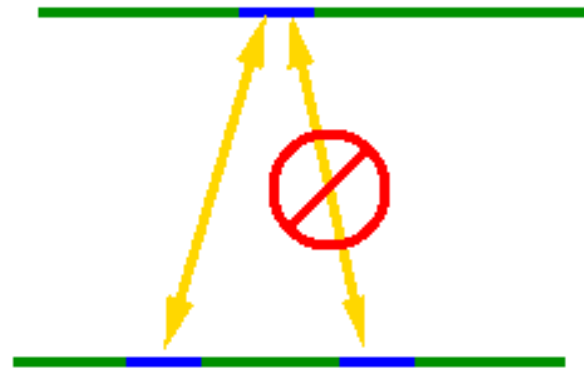
# What else does this?

- MUMmer
  - Delcher, et al [1999]
- AVID
  - Bray, Dubchak, Pachter [2003]
- BLASTZ
  - Schwartz, et al [2003]
- LAGAN
  - Brudno, et al [2003]

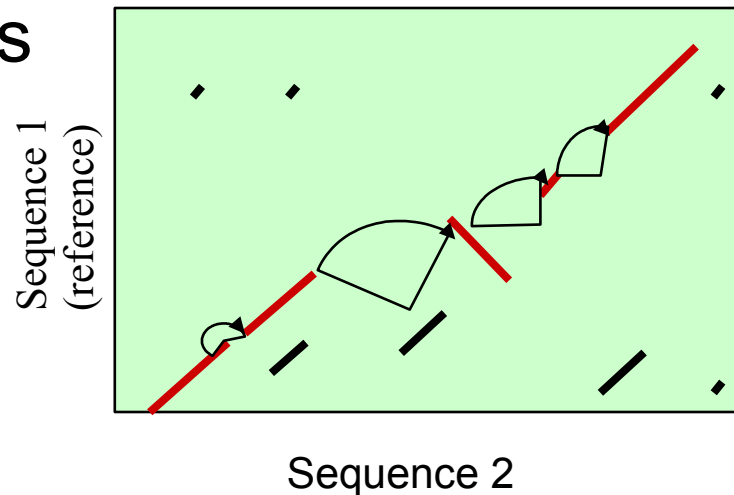
All major players use a *seed and extend* approach

# Seed and Extend

- Find matches of high confidence **seeds**
  - Repeats are a liability

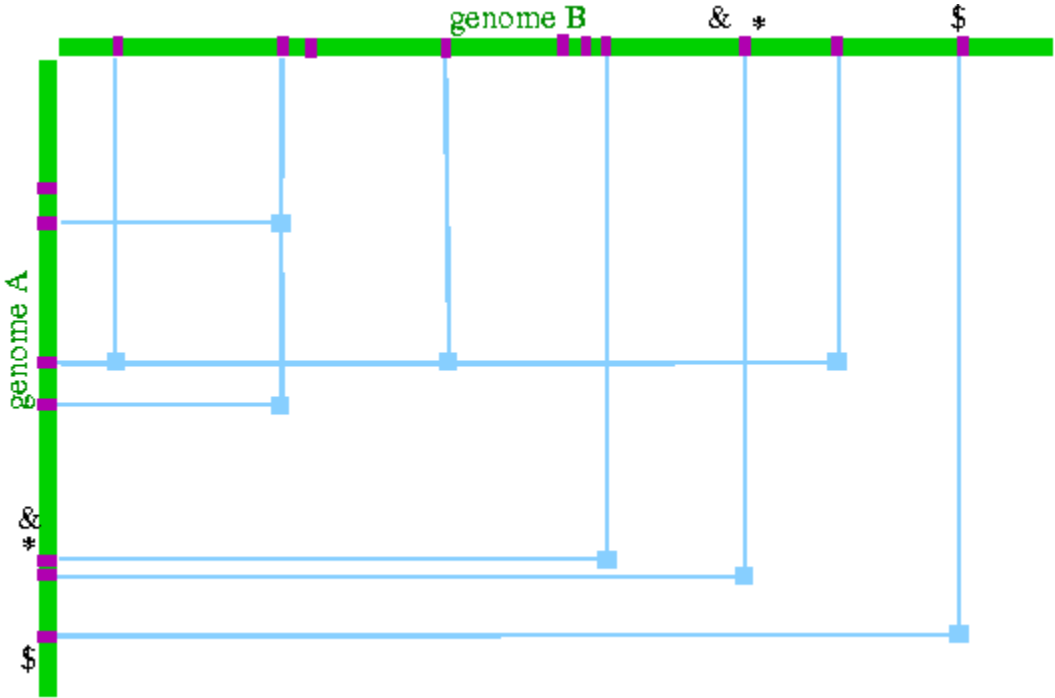


- Extend/Fill-in matches to get alignment

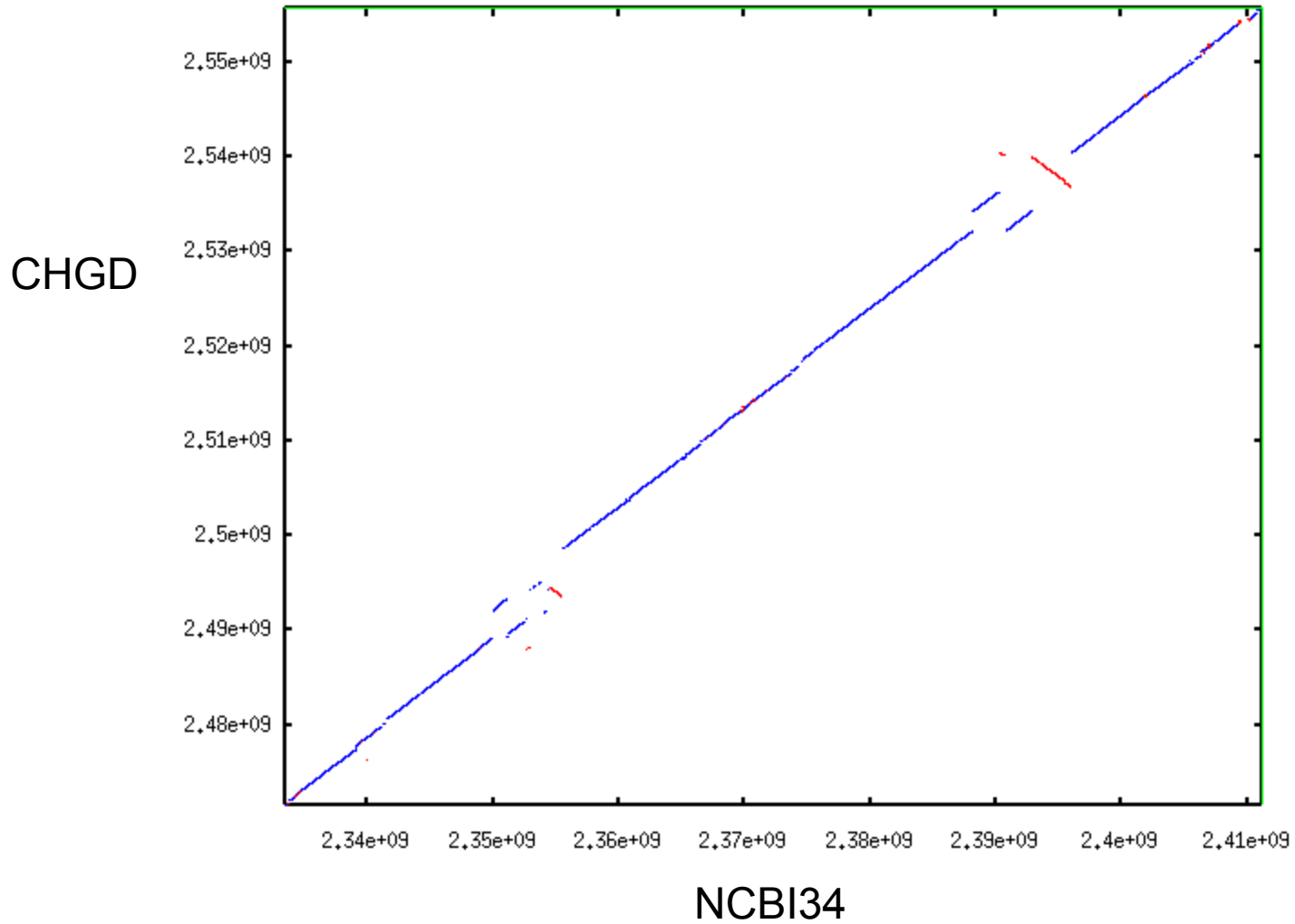


# Seeds have their own utility

Just the seeds themselves can be used for feature propagation

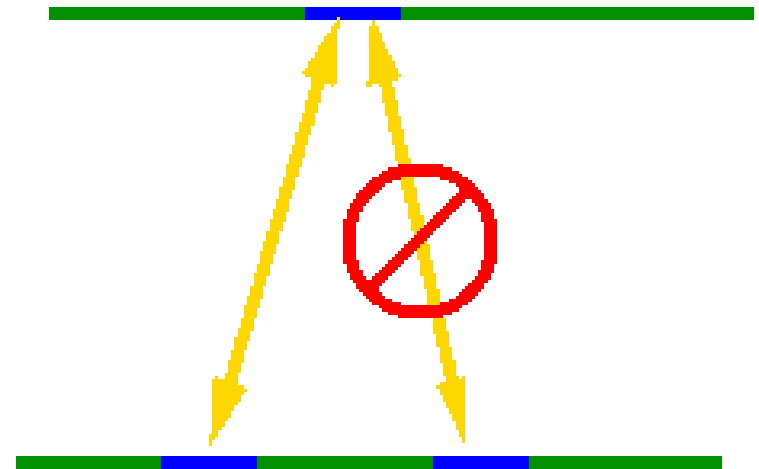


# 50-unique Chrom 17



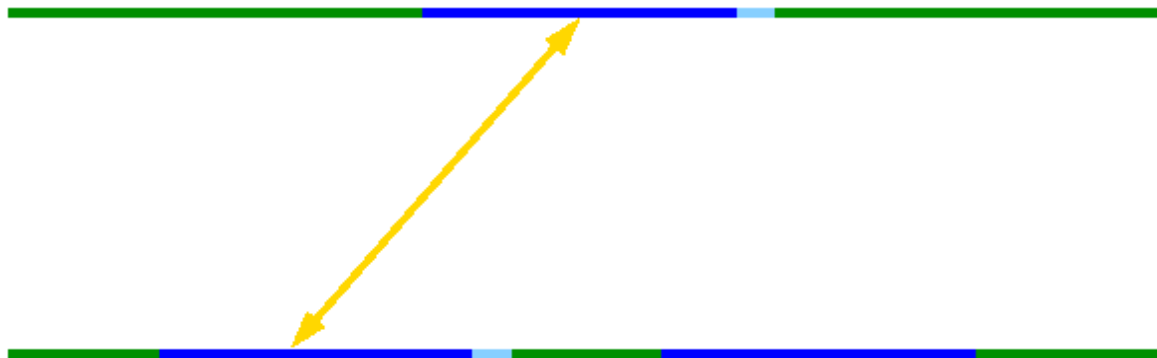
# What is a good seed?

- A really confident match
  - High similarity
    - Exact or nearly exact
  - Significance
    - Length of match (random string model)?
    - No competition: No other match looks like it



# Seeding with Exact Matches

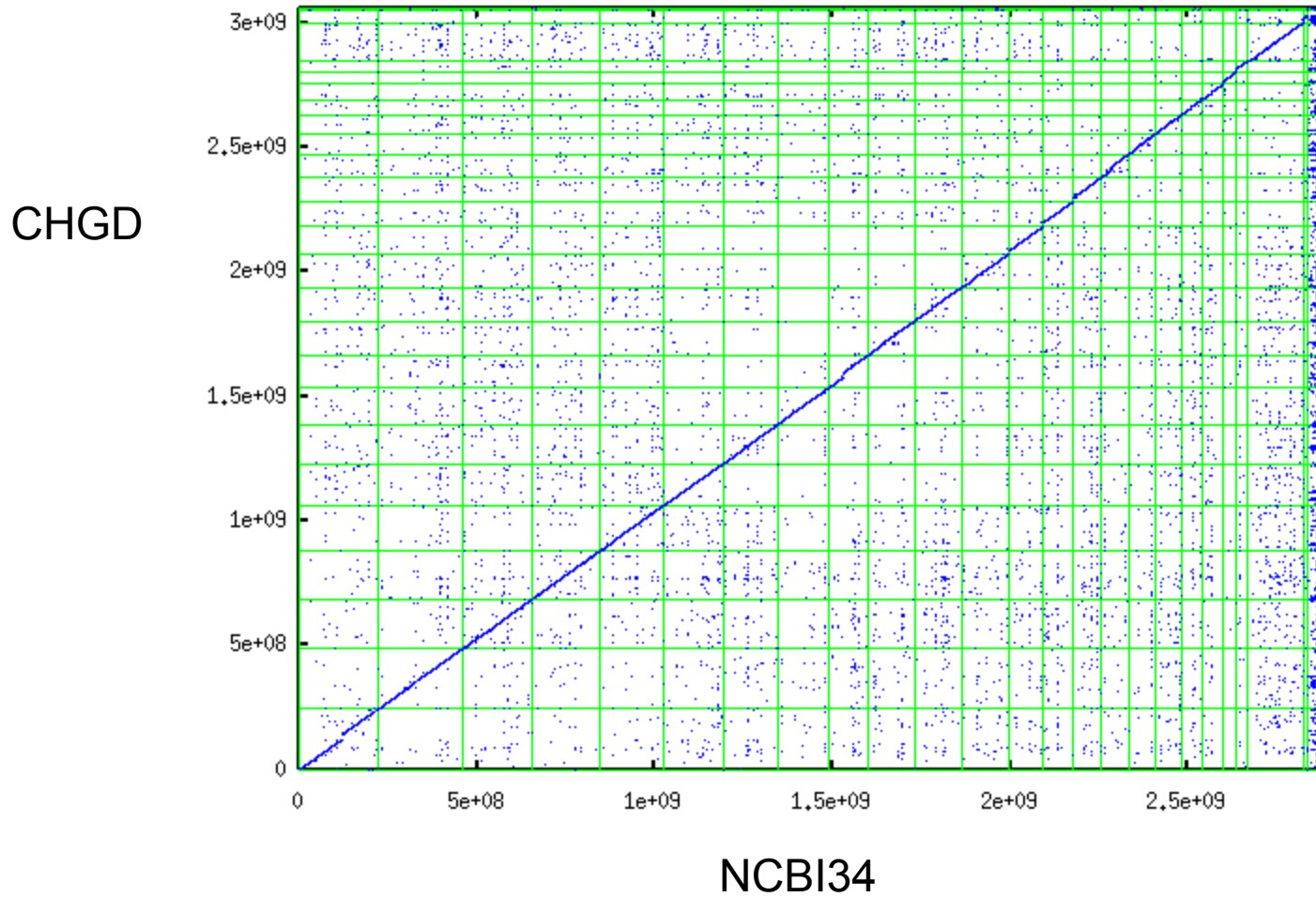
- How about MUMs?
  - M Exact matches
  - U Unique (no competition?)
  - M Maximal (flanked by mismatches)
- The U in MUM is weak
  - matches are unique, any sub-piece might repeat.
  - MUM might be a common repeat with slightly better flanking match.



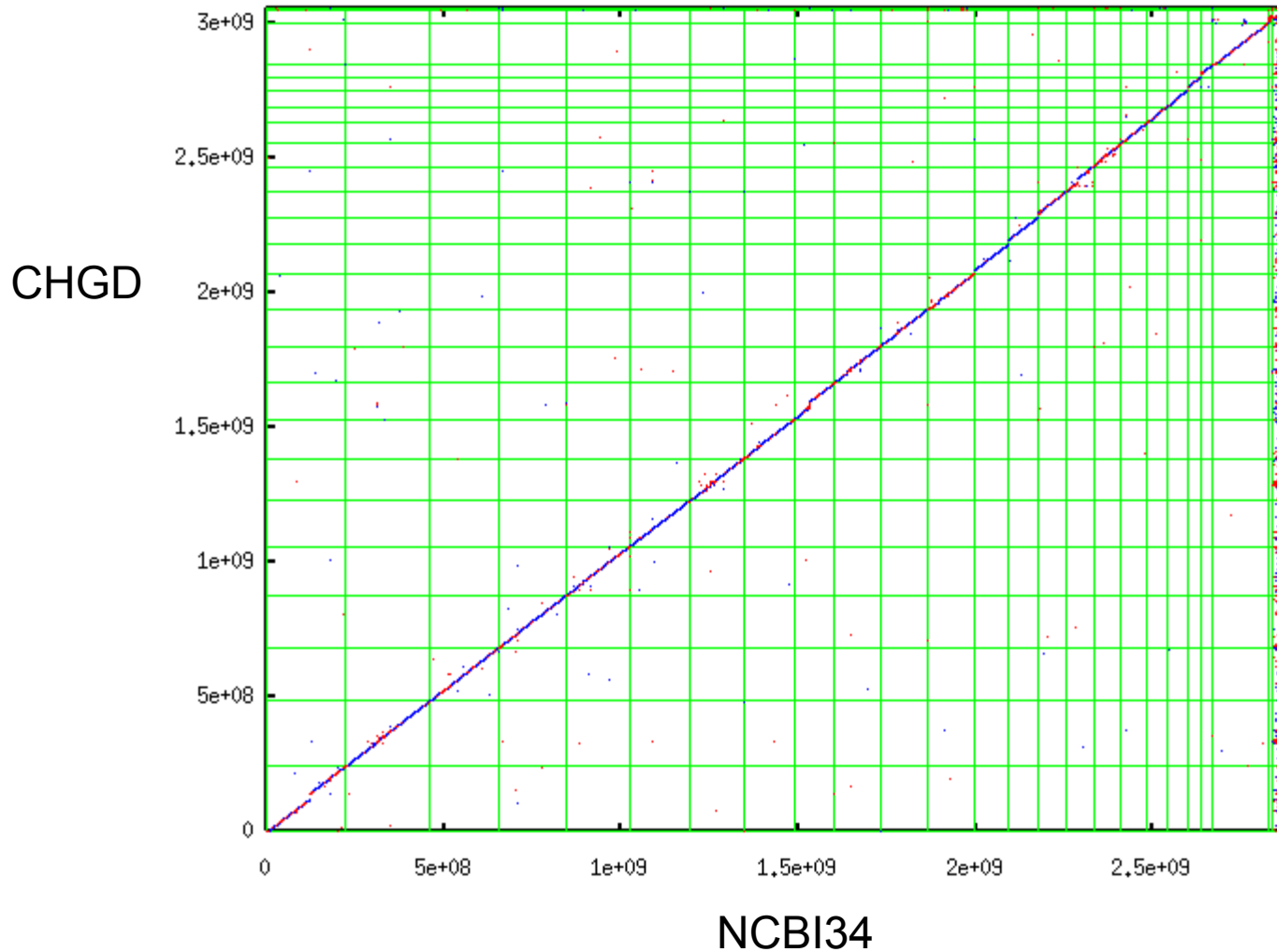
# Experiment NCBI31 & CHGD

- To get a bird's eye view of the match landscape
- Match NCBI31 to a Celera variant
- See how filtration affects the picture
  - All matches are  $> 300$  in length
  - Most long matches are spurious

# Max Unique Matches

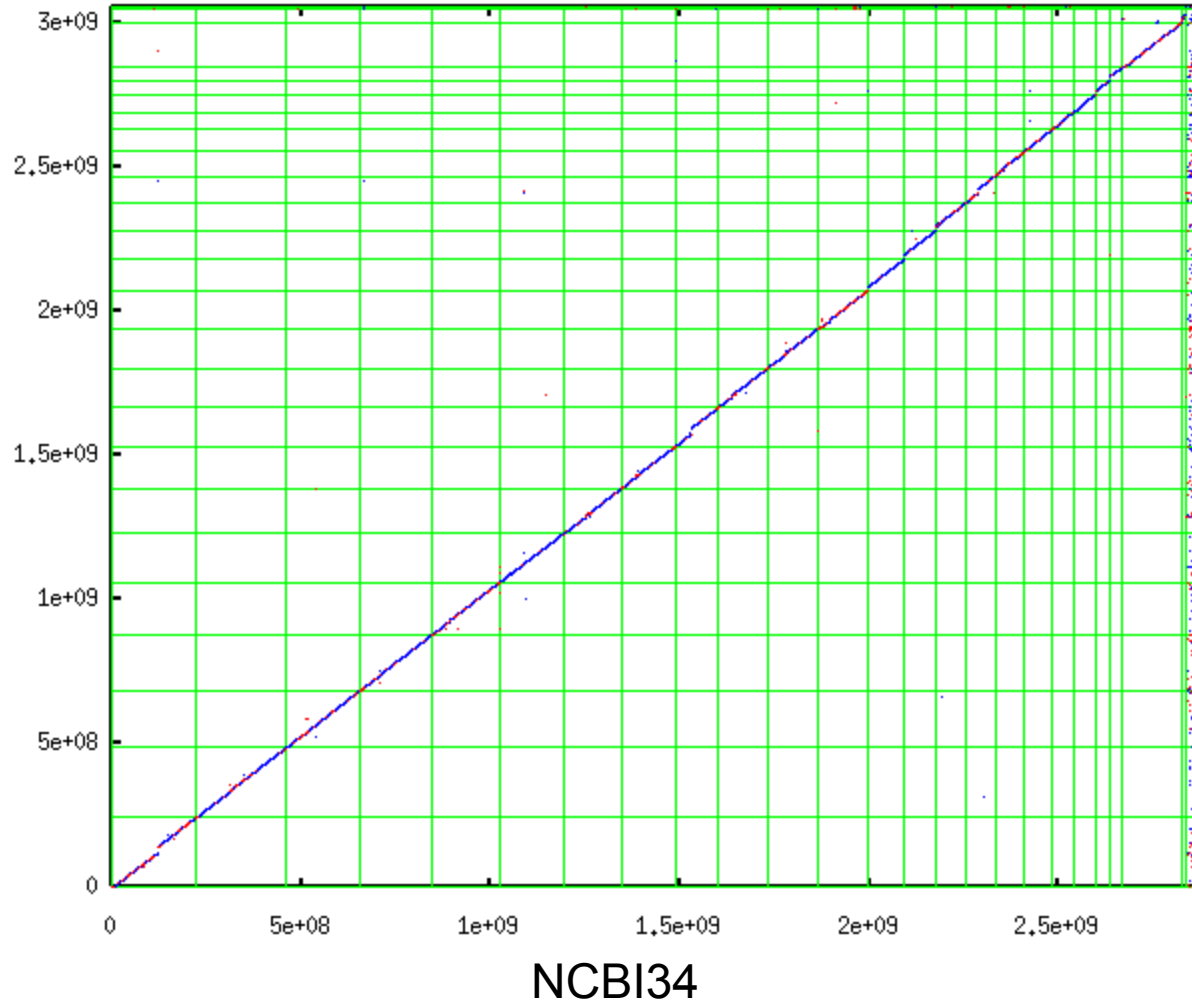


# Containing a 250 unique piece



# Containing a 50 unique piece

CHGD

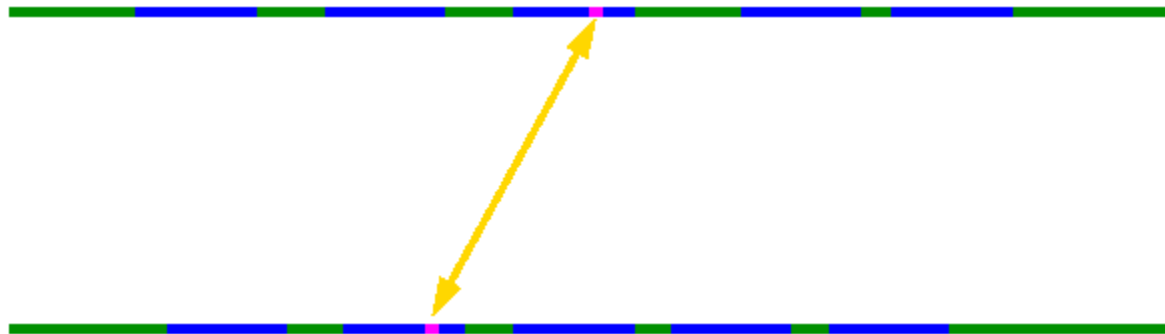


# Short matches

Why short matches?

- Short unique words less likely to occur in repetitive sequence
- Short bi-unique matches might be significant

short unique matches have their problems in repetitive regions.



# What to use for seeds?

- MUMmer
  - Maximal unique matches which are non-overlapping (greedy).
- AVID
  - Maximal exact matches filtered by length
  - **RepeatMasker** on input
- BLASTZ
  - High scoring local alignments of length  $>11$  matches.
- LAGAN
  - Length  $>6$  matches within a fixed window
  - **RepeatMasker** on input

# Modified (de-repeated) Matches

- $M$  and contains one UM of length =  $k$ 
  - Call this the  $k$ -core condition
  - Produces a subset of the MUMs
- $M$  and *tilled* by UM's of length =  $k$ 
  - Call this the  $k$ -tilled condition
  - Same as taking all biunique  $k$ -mer matches and doing trivial concatenation

What condition and what  $k$  do we pick?

How does quality vary with  $k$  and conditions?

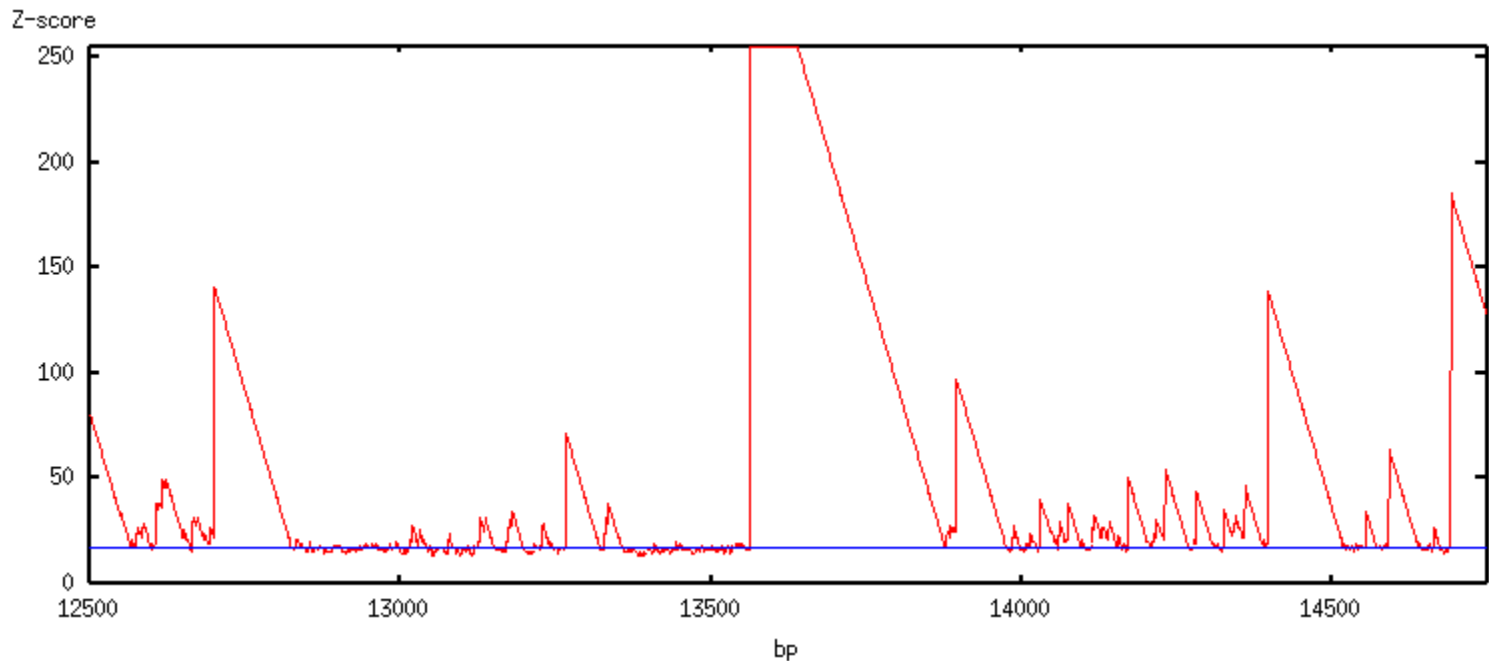
# A novel approach

- Compute a robust annotation for repetition
  - Length of longest exact repeat
  - Precomputed per genome
- Construct seeds by combining match and repetition information
  - Postcomputed per pair of matching genomes

Lippert, Florea, Zhao, Mobarry, Istrail. *Finding Anchors for Genomic Sequence Comparison*. RECOMB 2004

# Repetitiveness anotation

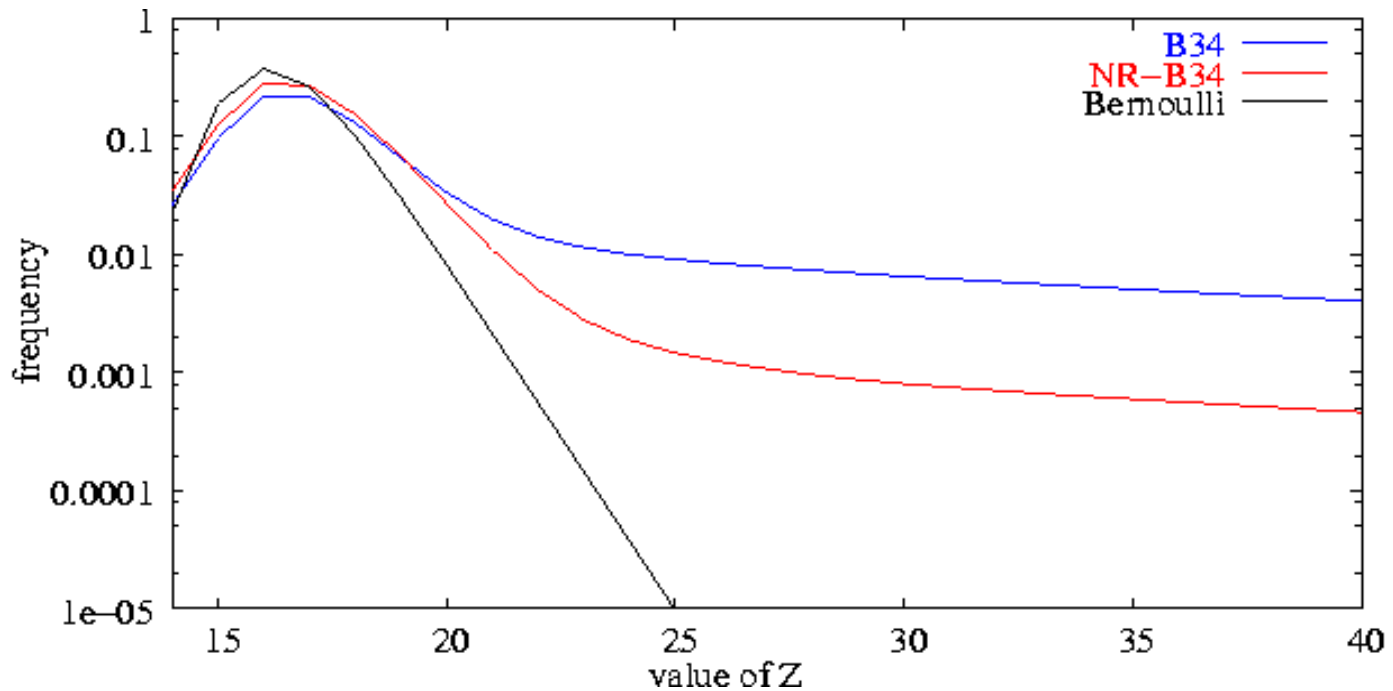
- We can annotate a genomic position with the length of longest exact repetition (length =  $Z$ ) starting there.



Non-repetitive regions should look like a random model (in blue)

# Z-score distribution

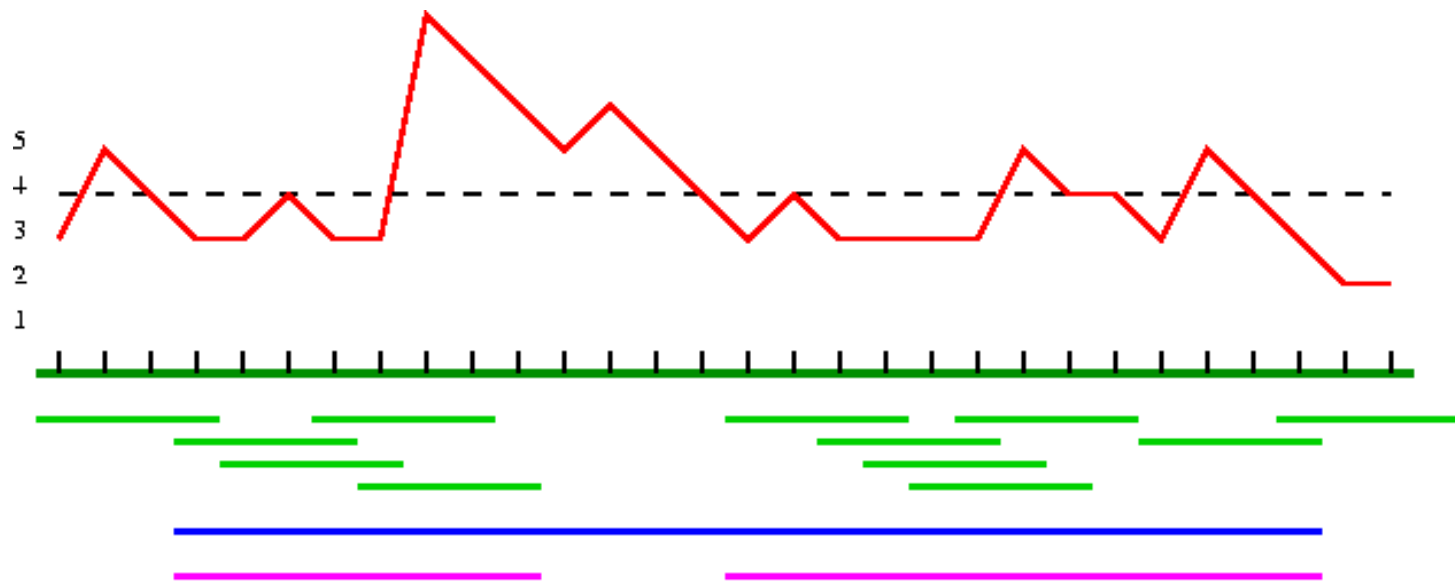
Bernoulli string model and empirical



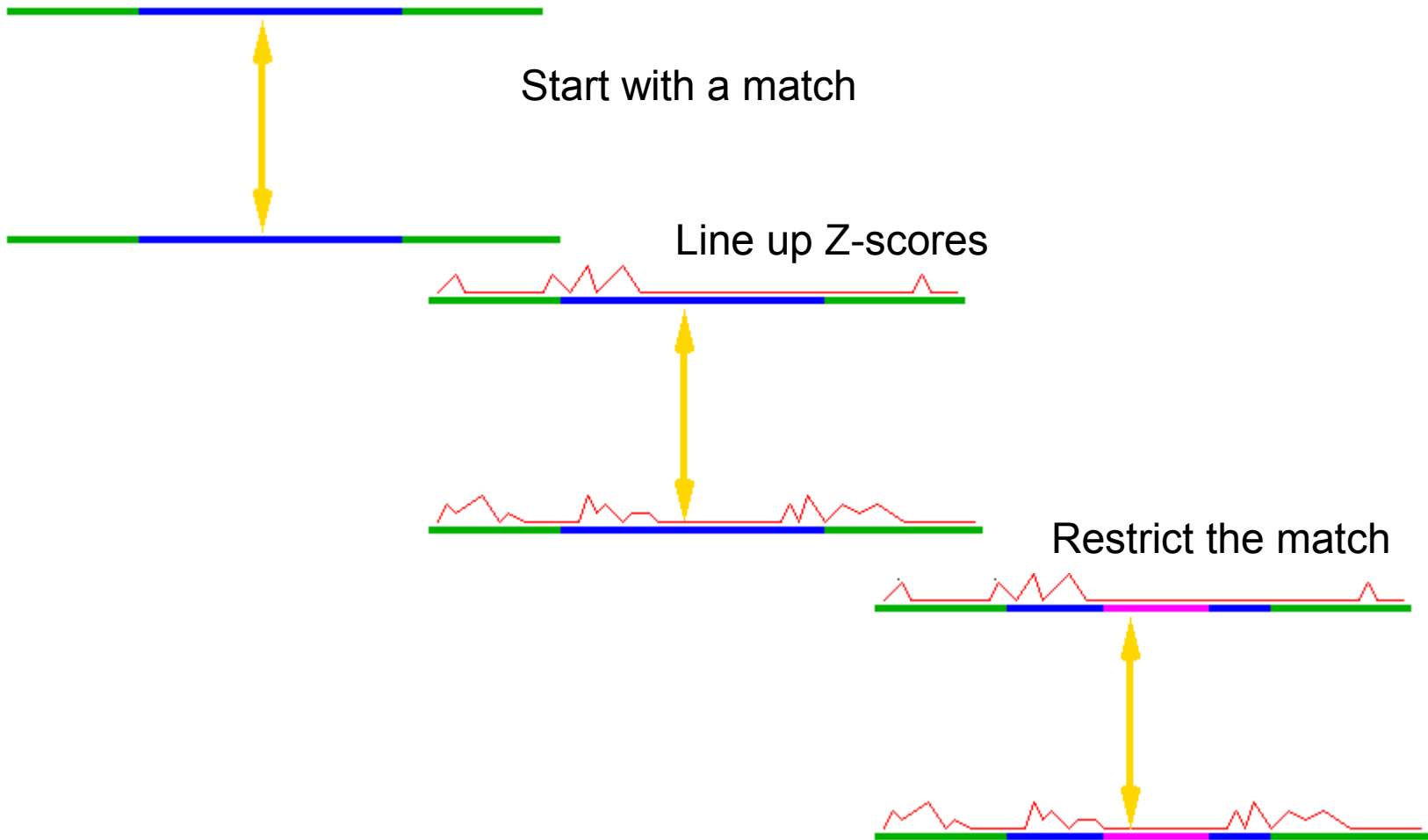
Heavy tail even when RepeatMasked

# Example (4-unique filter)

- Red = Z-score
- Green = sequence and substrings
- Blue = interesting region
- Magenta = restriction to unique 4-mers



# Filtering Max Matches



# Computational issues

- Computing maximal unique matches is a big compute
- Compute Z-scores per genome is about as big
- Match restriction is a small compute (by 25x roughly)

Flexibility/Calibration: vary  $k$  and uniqueness conditions

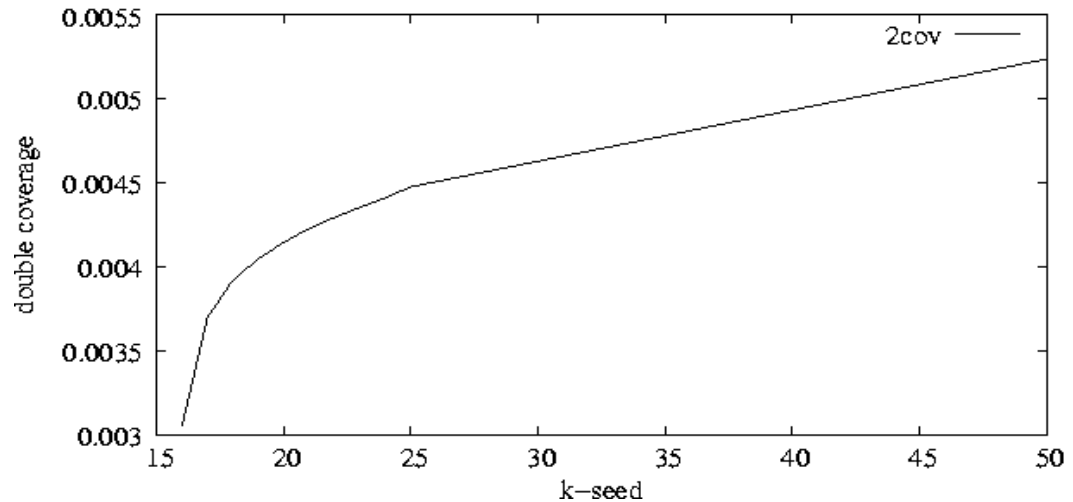
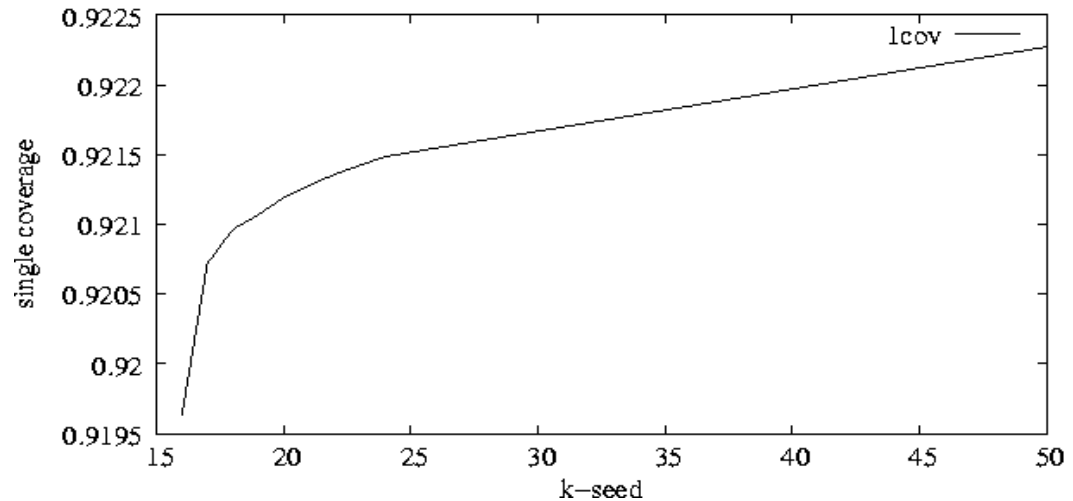
# Varying k

K-core filtering

For  $K \geq 20$  the coverage accessed is insensitive

Most maximal matches of interest contain a short bi-unique word

Odds of a multiple assignment are less than half a percent

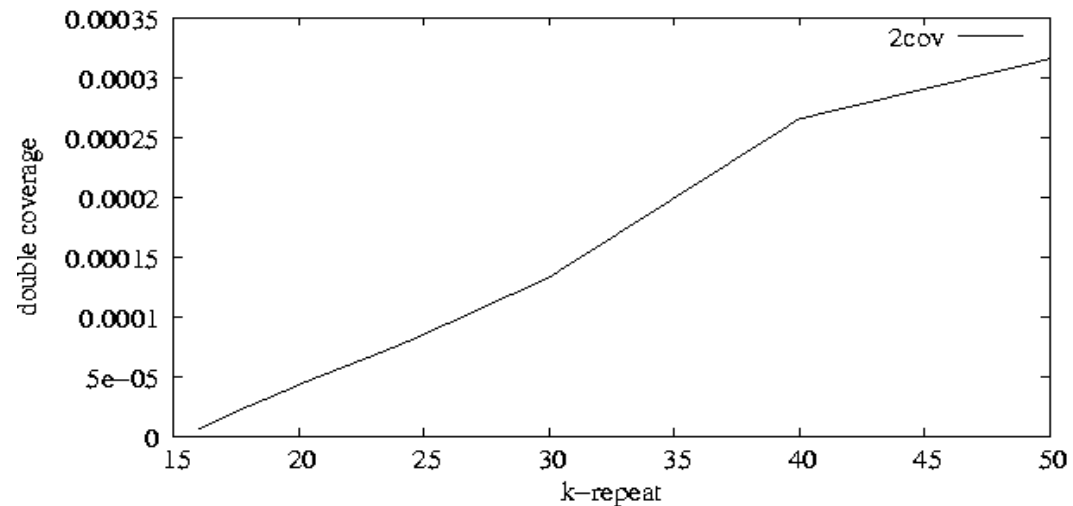
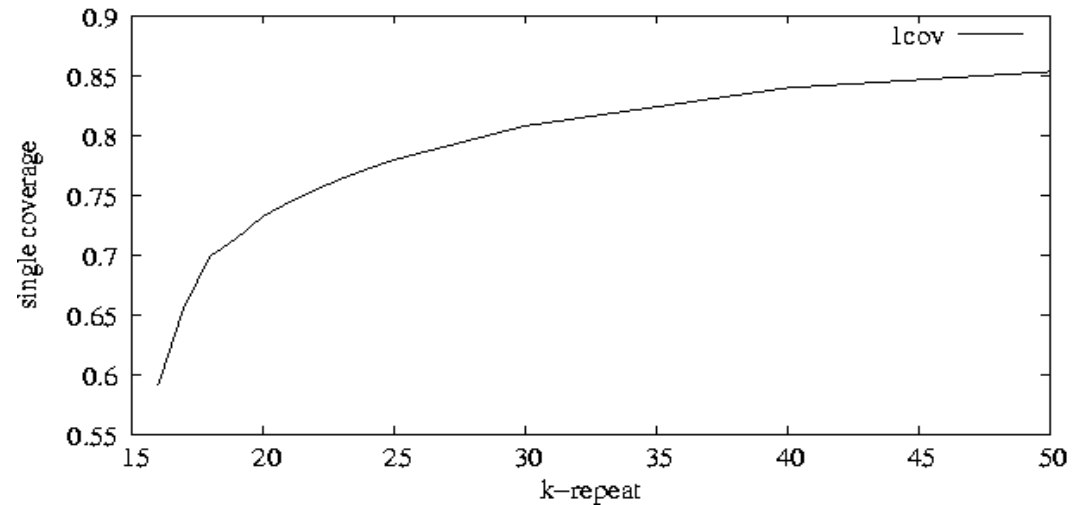


# Varying k

## K-tiled filtering

demanding that every base be assigned by a containing bi-unique match of length k is more stringent

double coverage decreases to the order of 0.01% and is roughly linear in k

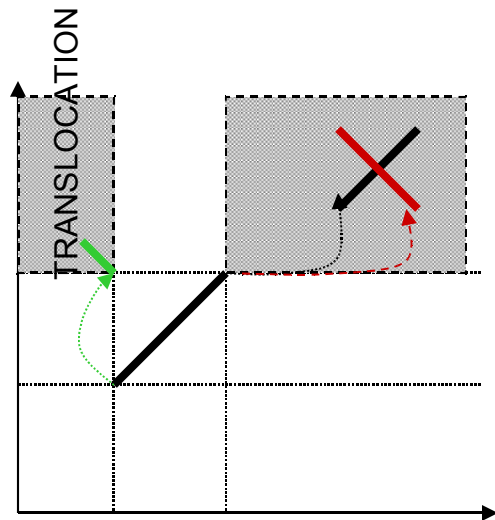


# Further filtering by context

- “Word counting” filters degrade as species are more distant
  - Exact match lengths decrease
  - Significance again hard to determine
- Extrinsic filtering methods can supplement
  - Good matches should occur in blocks or ***chains***
  - Match significance based on chain score
- Sparse dynamic programming can effectively compute the chains

# Match filtering by context: chaining

**Goal:** select and order a subset of matches to guide the alignment of the sequences.



Chaining graph  $(V, E, S)$ :

**Vertices** = set of matches

$$V = \{f_i\}_{i=1, M}, f_i = (b_i^1, e_i^1, b_i^2, e_i^2, ori)$$

**Edges** = precedence constraints ( $\rightarrow$ )

- match order
- match orientation
- distance between matches

**Scores**

- positive match scores
- penalties for edges

# Dynamic programming formulation

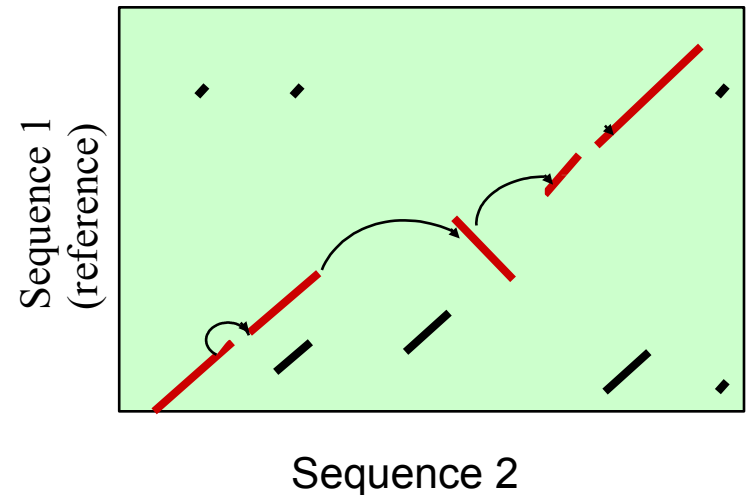
Determine one global or several local maximal scoring paths, or match *chains*, in the (V,E,S) graph.

Let  $P = f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow \dots \rightarrow f_m$  be a chain,  $f_i = (b^1_i, e^1_i, b^2_i, e^2_i, ori)$  matches

$$\begin{aligned} \text{score}(P) &= \sum_{i=1,m} \text{score}(f_i) - \sum_{i=2,m} \text{connect}(f_{i-1}, f_i) = \\ &= M * \sum_{i=1,m} \text{length}(f_i) \\ &\quad - O * \sum_{i=2,m} \text{reversal}(f_{i-1}, f_i) \\ &\quad - R * \sum_{i=2,m} \text{translocation}(f_{i-1}, f_i) \\ &\quad - D * \sum_{i=2,m} \text{diag\_diff}(f_{i-1}, f_i) \end{aligned}$$

Score( $f_i$ ) = maximal score of a path ending in  $f_i$

Score( $f_i$ ) = max (0, score( $f_i$ ) + max <sub>$f_{i-1} \rightarrow f_i$</sub>  (Score( $f_{i-1}$ ) - connect( $f_{i-1}, f_i$ )))



# Sparse Dynamic Programming

- Input: a collection of matches  $(score_i, x_i, y_i)$
- Output: a maximum chain,  $i_1, i_2, i_3, \dots$  which maximizes a total score
  - Total =  $s_{i_1} + s_{i_2} + \dots$  - Penalties
  - Penalties based on jumps between adjacent matches, e.g. on  $(x_{i_2} - x_{i_1})$  and  $(y_{i_2} - y_{i_1})$
- Zhang, et al first reported the use of k-d trees
- k-d trees answer spatial queries quickly, e.g. give me all points within  $D$  of  $(x, y)$
- k-d trees avoid unnecessary computes by using distance-based bounds

# Short unique matches don't survive chaining

% of bp in matches covered in a heaviest weight path through biunique k-mers, broken down by chromosome: (k=20,50,250)

1	5.8	72.6	86.7	13	5.0	78.3	90.7
2	5.9	74.9	87.3	14	6.0	77.0	92.2
3	5.2	72.2	84.9	15	6.2	68.7	83.0
4	4.3	74.8	88.1	16	6.0	64.1	77.6
5	5.4	74.9	87.9	17	5.8	56.4	72.5
6	5.5	76.0	89.6	18	6.2	77.7	89.1
7	5.4	71.6	86.7	19	5.6	59.0	87.9
8	5.3	74.0	86.2	20	7.4	78.1	92.4
9	6.1	73.8	87.9	21	6.7	80.3	92.9
10	6.3	74.3	88.5	22	7.3	65.8	84.6
11	6.0	71.2	85.7	X	4.1	68.6	85.6
12	4.7	66.8	80.1	Y	0.2	17.7	29.3

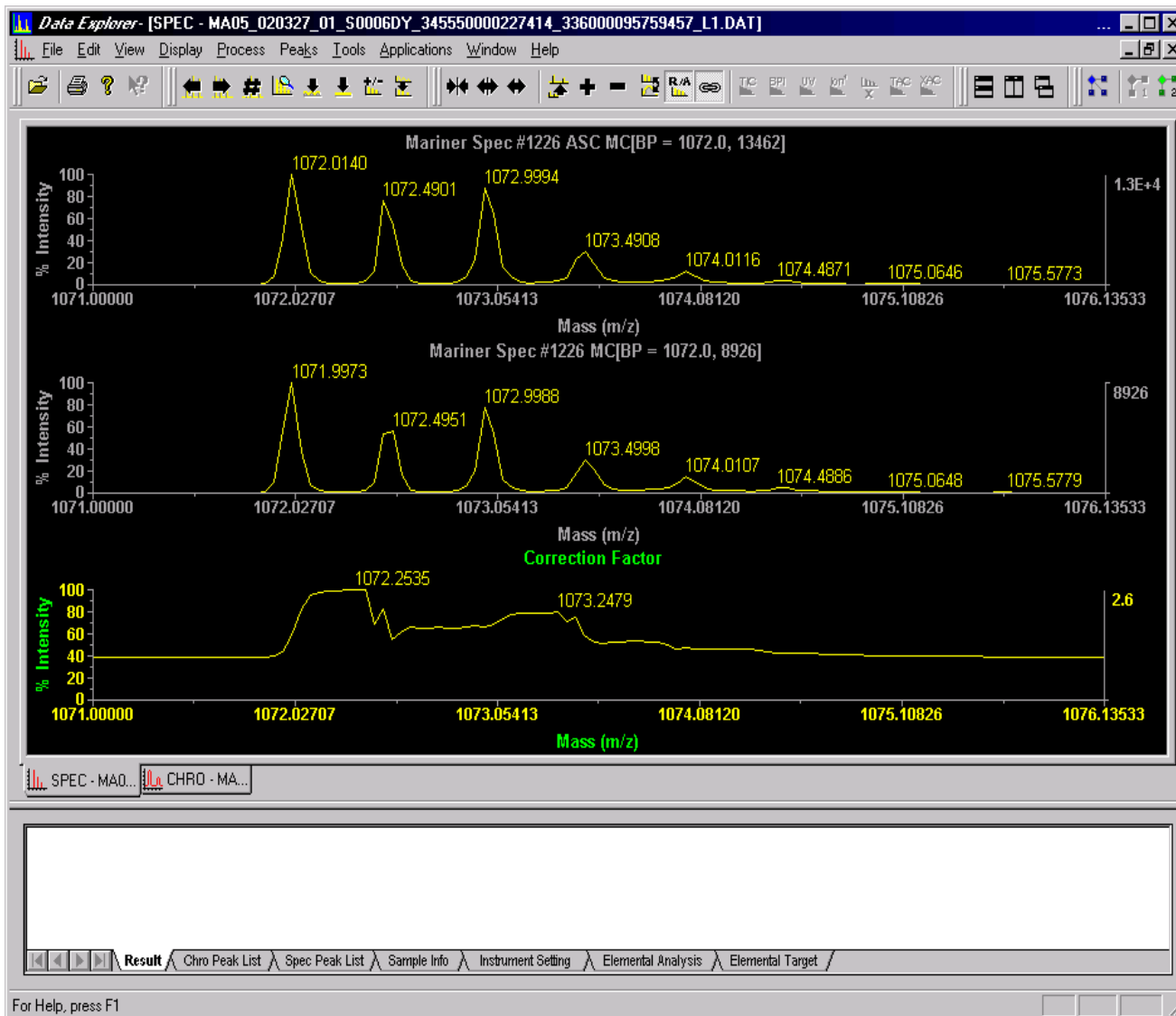
# Genome comparison wrap-up

- Finding good matches is a question of significance
  - Most short matches are not significant
  - Most long matches are not significant
  - Repetition and chaining context determine significance
- Comparison parameter selection can be independent of computational task
  - There is no *magic* k-mer
  - a posteriori computation of anchors and chaining is computationally inexpensive
- All calibration can be done dynamically

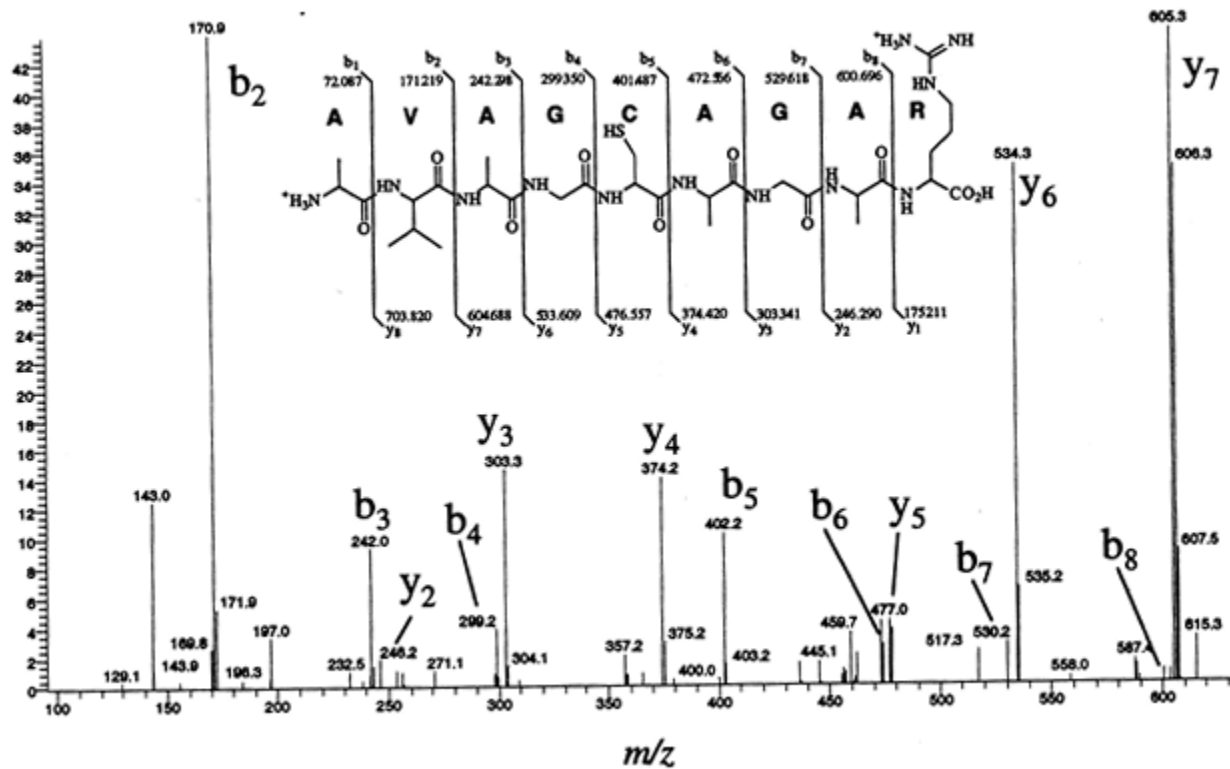
# Generating Peptide Candidates from Sequence Databases for Protein Identification



# Spectrum Data



# Masses coming from prefixes and suffixes



# LC/MS/MS for Protein Id

- Suitable for complex mixtures (multiple stages of separation and selection)
- 1 experiment produces 1000's of MS/MS spectra with parent peptide masses
- 100's of proteins identified by a single experiment

**-High-throughput protein identification**

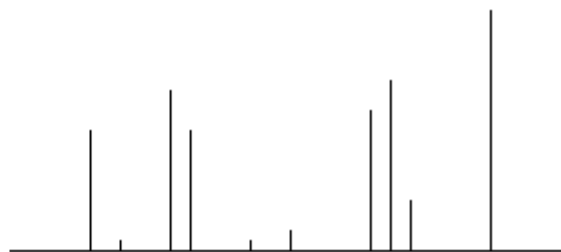
# Sequence Database Search

**Input:** Set of MS/MS spectra with parent peptide masses

**Output:** Peptide sequence for each spectrum

```
>108_LYCES (Q43495) Protein 108 precursor
MASVKSSSSSSSSSFISLLLLLIVLQSQVIECQPQQSCTASLTGLNVCAPFLVPGSP
TASTECCNAVQSIHDCMCNTMRIAQAQIPACNLPPPLSCSAN
>10KD_VIGUN (P18646) 10 kDa protein precursor (Clone PSAS10)
MEKKSIAGLCFLFLVLFVAQEVVVQSEAKTCENLVDTYRGPCFTTGSCDDHCKNKEHLLS
GCRDDVRCWCTRNC
>110KD_PLAKN (P13813) 110 kDa antigen (PK110) (Fragment)
FNSNMLRGSVCEEDVSLMTSIDNMIIEIDFYEKEIYKGSHTGGVVIKGM DYDLEDDENDED
EMTEQMVEEVADHITQDMIDEVAHHVLDNITHDMAHMEIIVHGLSGDVTQIKEIVQKVN
AVEKVKHIVETEETQKTVEPEQIEETQNTVEPEQTEETQKTVEPEQTEETQNTVEPEQIE
ETQKTVEPEQTEEAQKTVEPEQTEETQKTVEPEQTEETQKTVEPEQTEETQKTVEPEQTE
ETQKTVEPEQTEETQKTVEPEQTEETQKTVEPEQTEETQNTVEPEPTQETQNTVEP
```

SLMTSI  
SLMTS I  
SLMT SI  
SLM TSI  
SL MTSI  
S LMTSI



1. Generate peptide candidates from a protein or genomic sequence database
2. Score and rank the peptide candidates

# Peptide Candidate Generation

**Input:** Sequence  $\sigma$  (length  $n$ ),  
from alphabet  $\mathbf{A}$   
mass  $\mu(a)$  for  $a$  in  $\mathbf{A}$   
Query masses  $M_1, \dots, M_k$

**Output:** All (distinct) pairs of query masses  $i$  and substrings  $\omega$ :  $\sum_{j=1}^{|\omega|} \mu(\omega_j) = M_i$

```
>108_LYCES (Q43495) Protein 108 precursor
MASVKSSSSSSSSSFISLLLLLIVIVLQSQVIECQPQQSCTASLTGLNVCAPFLVPGSP
TASTECCNAVQSIHDCMCNTMRIAQAQIPACNLPLSCSAN
>10KD_VIGUN (P18646) 10 kDa protein precursor (Clone PSA510)
MEKKSIALGLCFLFLVLFVAQEVVVQSEAKTCENLVDITYRGPCFTTGSCDDHCKNKEHLLS
GCRDDVRCWCTRNC
>110KD_PLAKN (P13813) 110 kDa antigen (PK110) (Fragment)
FNSNMLRGSVCEEDVSLMTSIDNMIEEIDFYEKEIYKGSHTGGVVKGM DYDLEDDENDED
EMTEQMVEEVADHITQDMIDEVAHHVLDNITHDMAHMEEIVHGLSGDVTQIKEIVQKVVN
AVEKVKHIVETEETQKTVEPEQIEETQNTVEPEQTEETQKTVEPEQTEETQNTVEPEQIE
ETQKTVEPEQTEEAQKTVEPEQTEETQKTVEPEQTEETQKTVEPEQTEETQKTVEPEQTE
ETQKTVEPEQTEETQKTVEPEQTEETQKTVEPEQTEETQNTVEPEPTQETQNTVEP
```

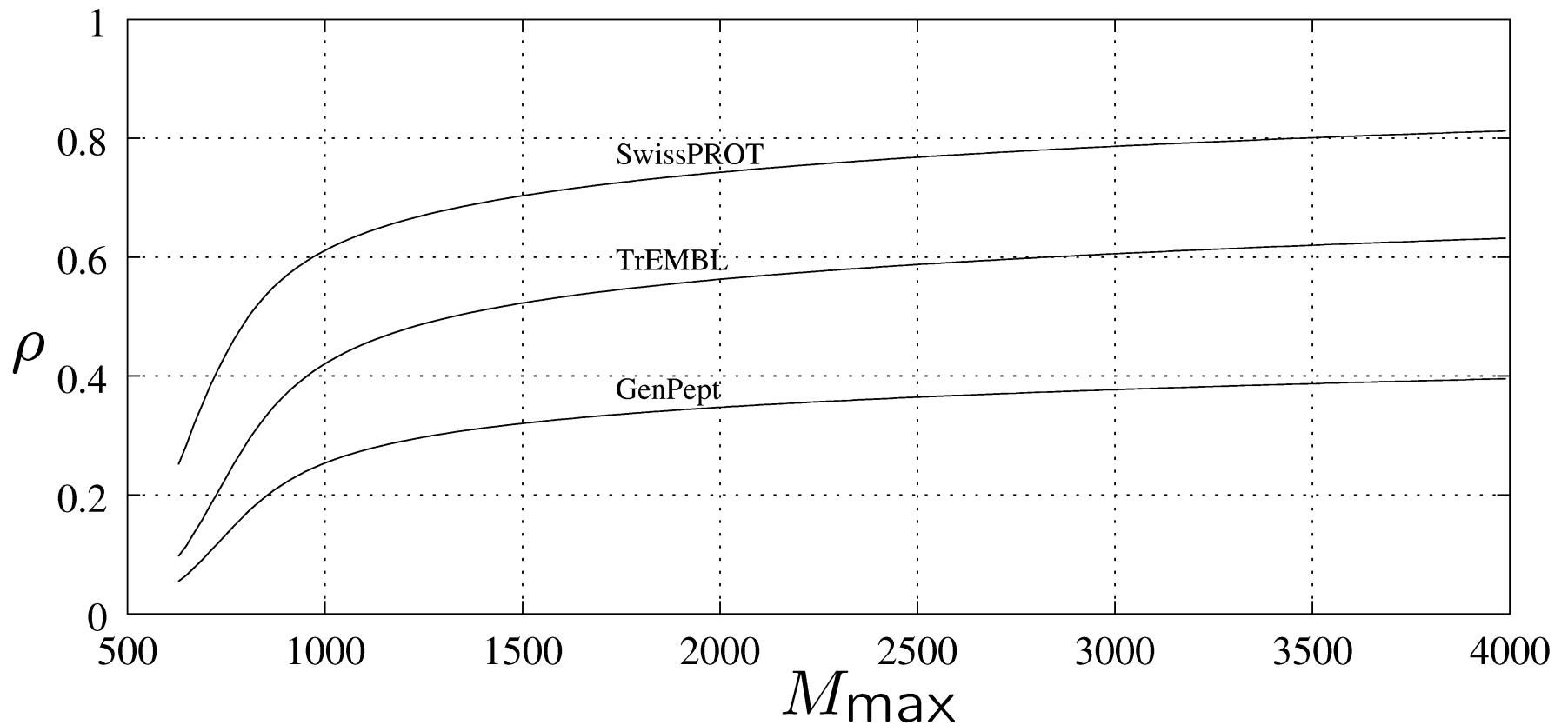
M=Mass( )



# Linear Scan

- Simple to implement
- Easy to track protein context (e.g. Trypsin digest)
- Poor data locality
- Redundant candidates
- String scanning problem

# Substring Density (uniqueness probability)

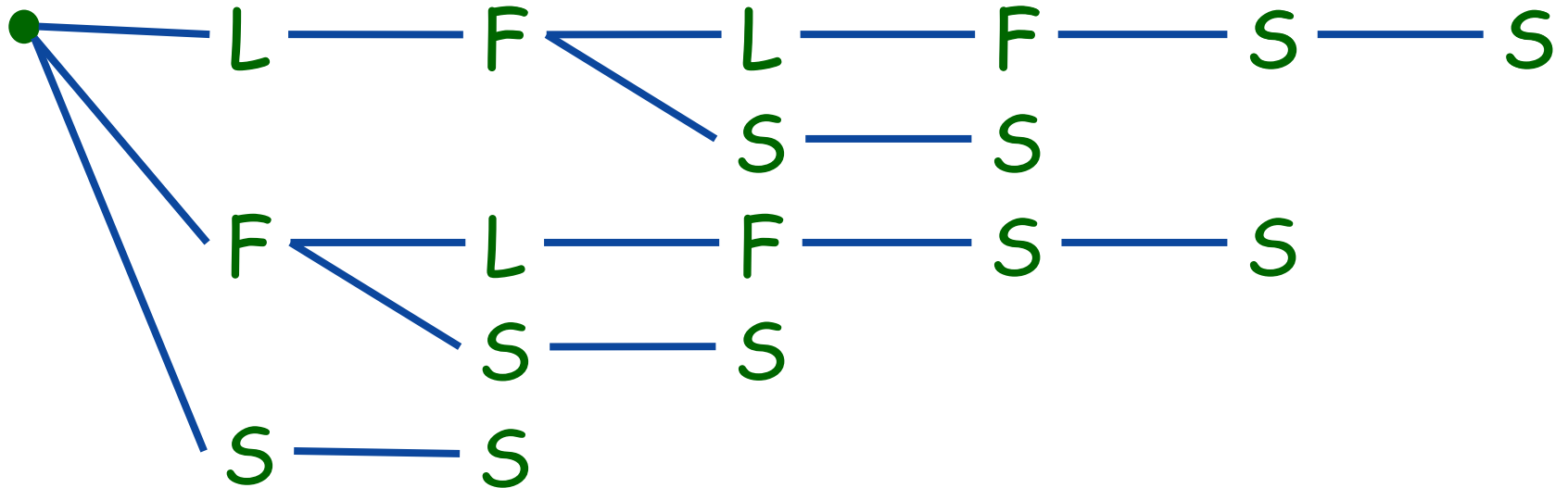


# Redundant Candidate Elimination

- Should **avoid repeat scoring** of the same peptide candidate
- Want to **avoid generating redundant candidates**
- Non-redundant sequence databases contain some **substring redundancy**

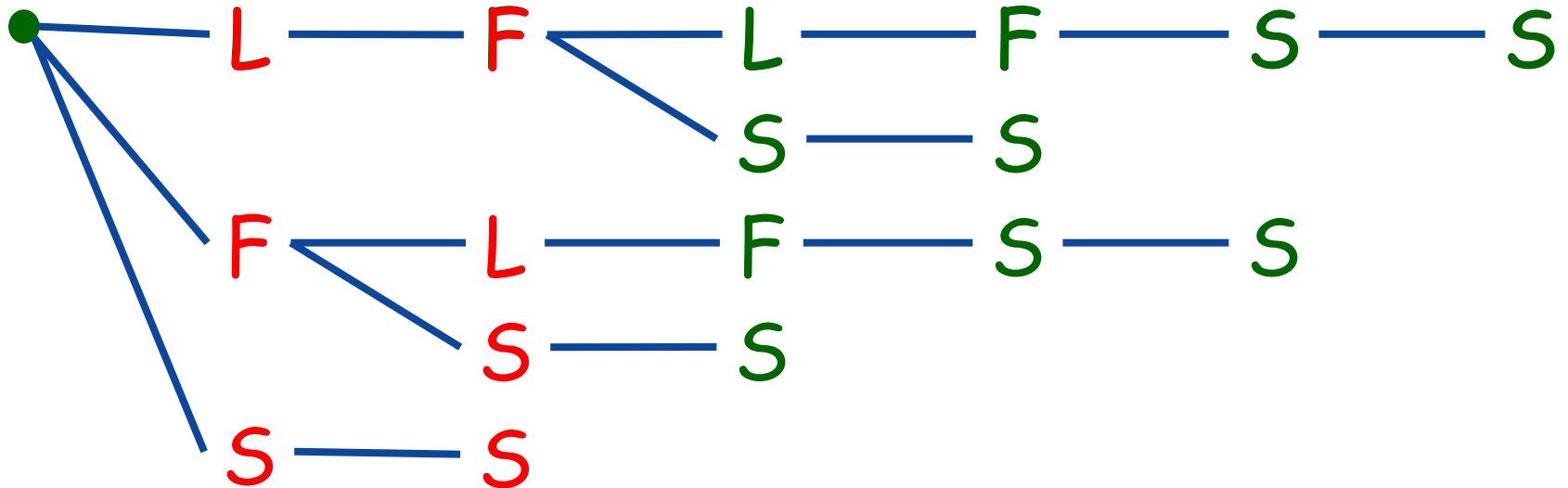
# Redundant Candidate Elimination

- Suffix trees represent all distinct substrings of a string.



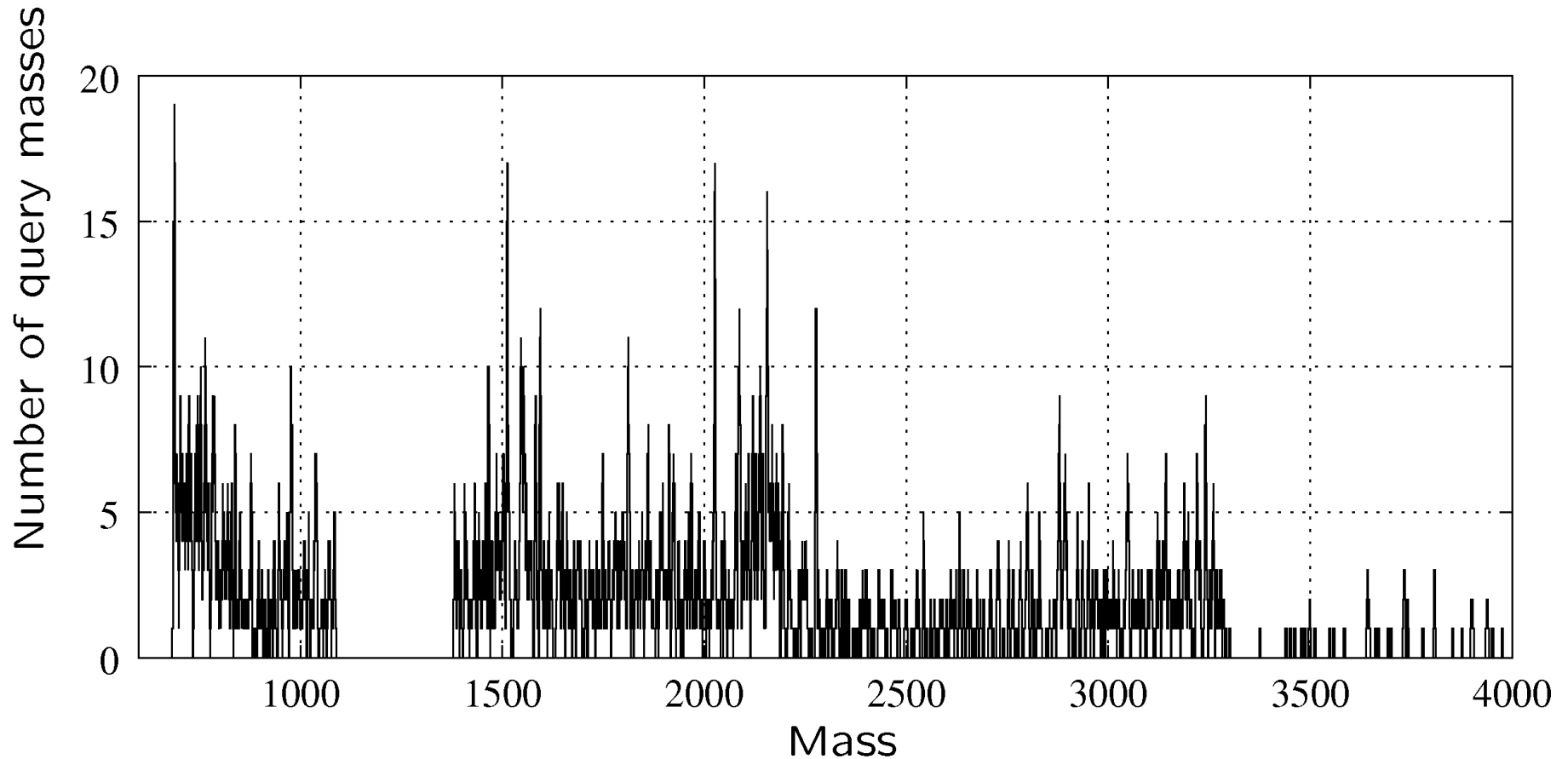
# Redundant Candidate Elimination

- Suffix trees represent all distinct substrings of a string.



Edwards, Lippert. *Sequence database compression for peptide identification from tandem mass spectra*. WABI 2004

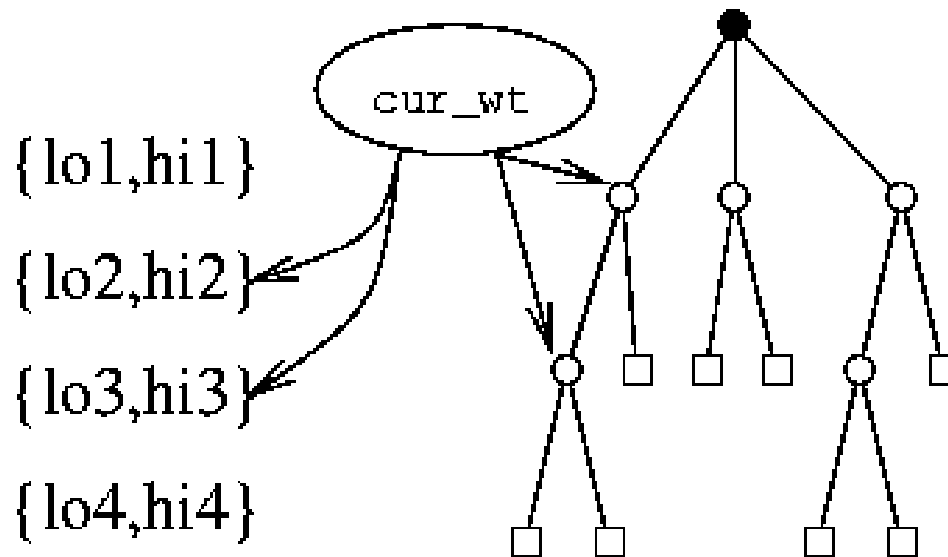
# Overlap Plot from a typical LC/MS/MS Experiment



Queries tend to be redundant too

# Simultaneous scanning

- The set of all mass intervals are scanned simultaneously



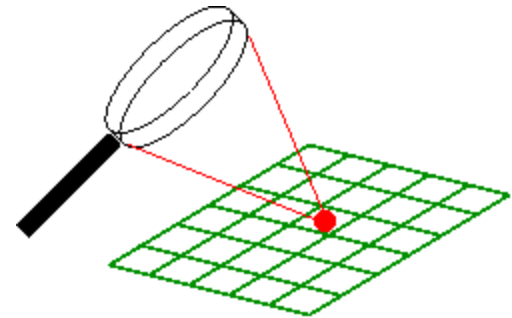
- Still easy to track protein context
- Better data locality (if done with a suffix array or the raw string)

Edwards, Lippert. *Generating peptide candidates from amino-acid sequence databases for protein identification via mass spectrometry*. WABI 2002

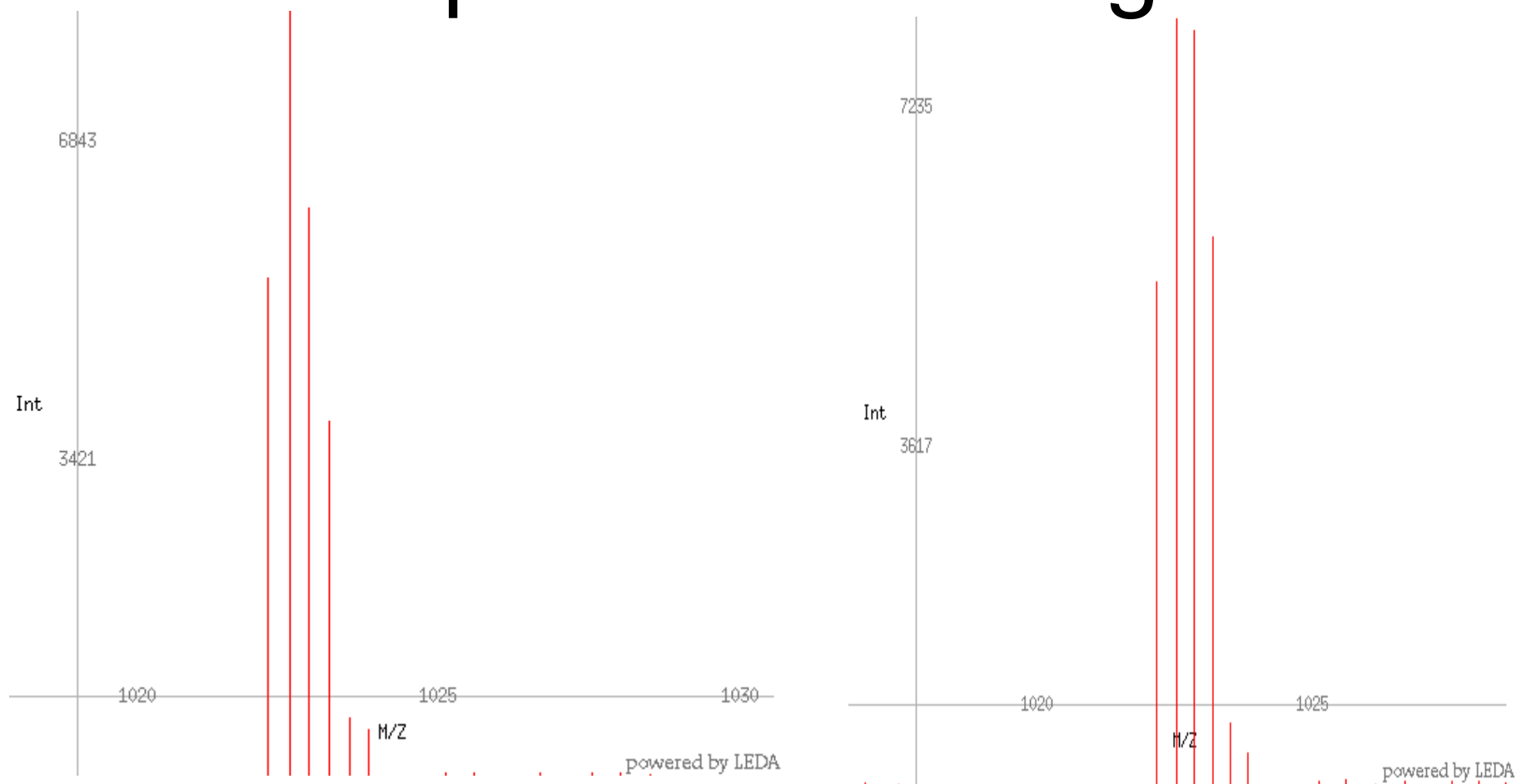
# Observations

- Peptide candidate generation is a **core issue**.
- We can eliminate **substring redundancy**.
- More importantly, peptide candidate generation is also an **interval lookup** problem.

***Single candidate search is an example of myopia....***



# Spectral matching

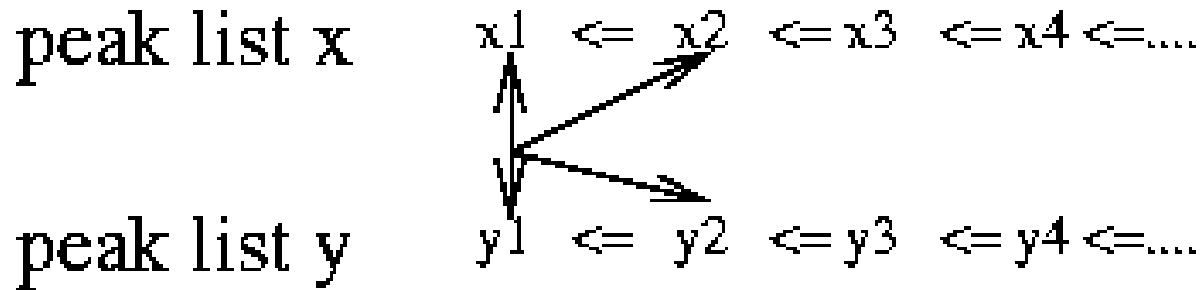


Triage step: match peaks against previously identified empirical or theoretical peaks (by shared peak count)

# A myopic solution

*Computing the largest intersections of a bunch of “fingerprint” sets is actually a pretty common problem.*

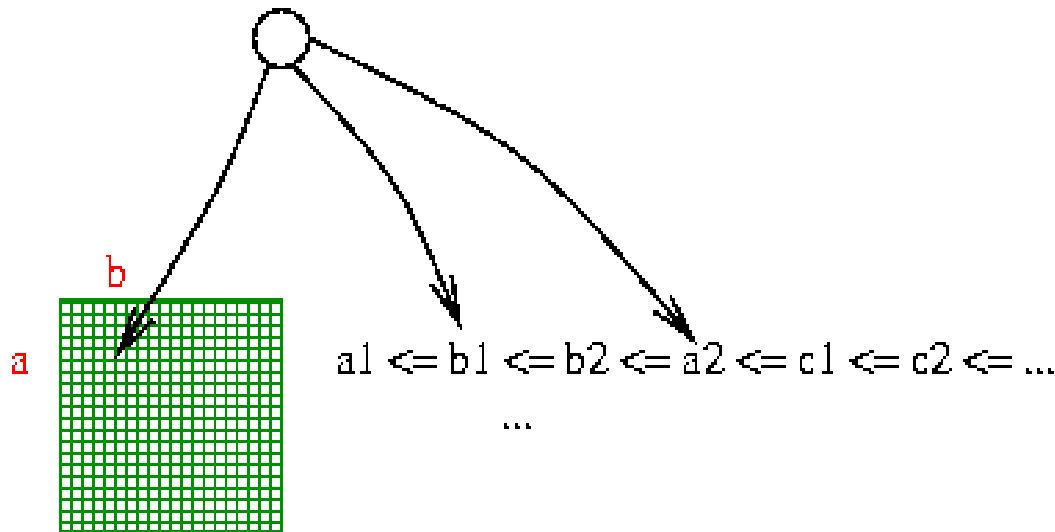
Assemble peak lists into “feature vectors”



With some effort one discovers sorting: shared peak counting becomes a kind of merge sort

# The aggregate solution is more efficient

Find all the counts of a block



Solving the same problem thousands of times gives a thousand-fold opportunity for improvement

# Conclusions

- Redundancy is in the nature of our data AND our queries
- High throughput scenarios demand an understanding of the big picture
- Aggregated solutions exploit more context