Znajdowanie sekwencji w bazach danych

Bartek Wilczyński

24. kwietnia 2018

Organizacyjne

- Materiały do dzisiejszego wykładu:
  `http://www.mimuw.edu.pl/~dojer/wobm/alg-heuryst.pdf`

How much similarity is there?

Many important genes are conserved between distantly related species

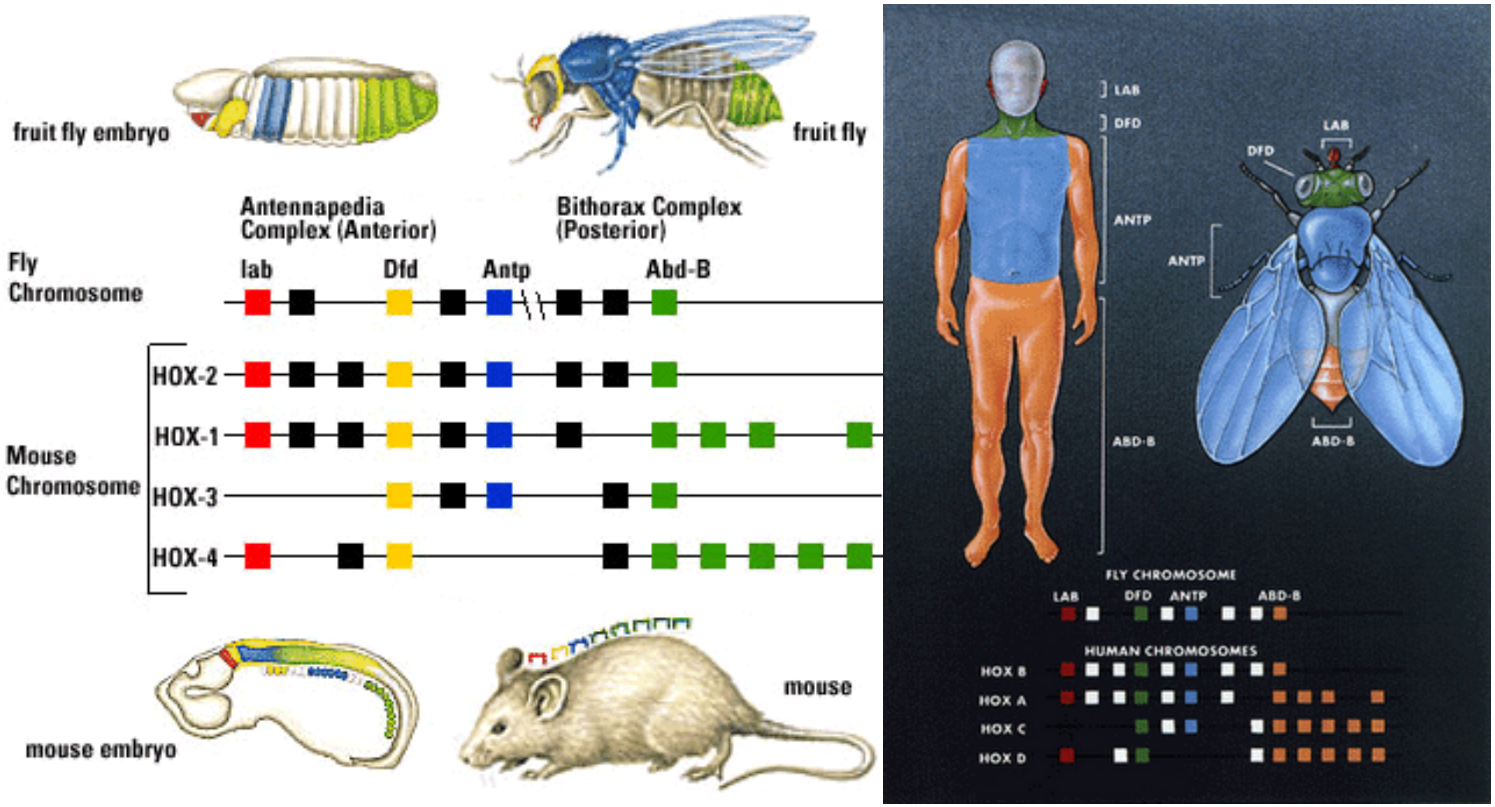


image sources biologycorner.org and ilbiologista.blogspot.com

Sequencing efforts are getting cheaper...

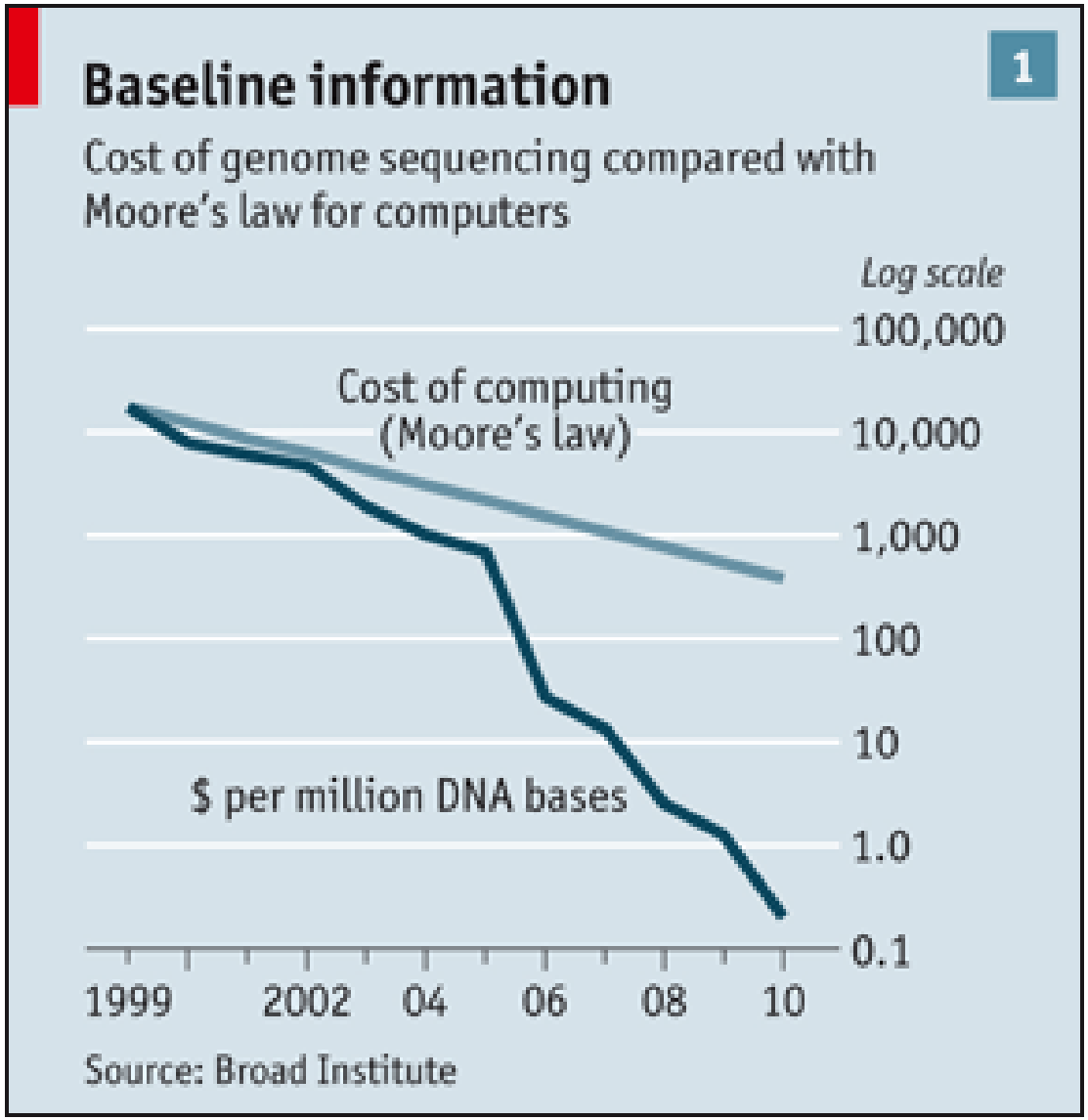


image source the economist

...and grow exponentially...

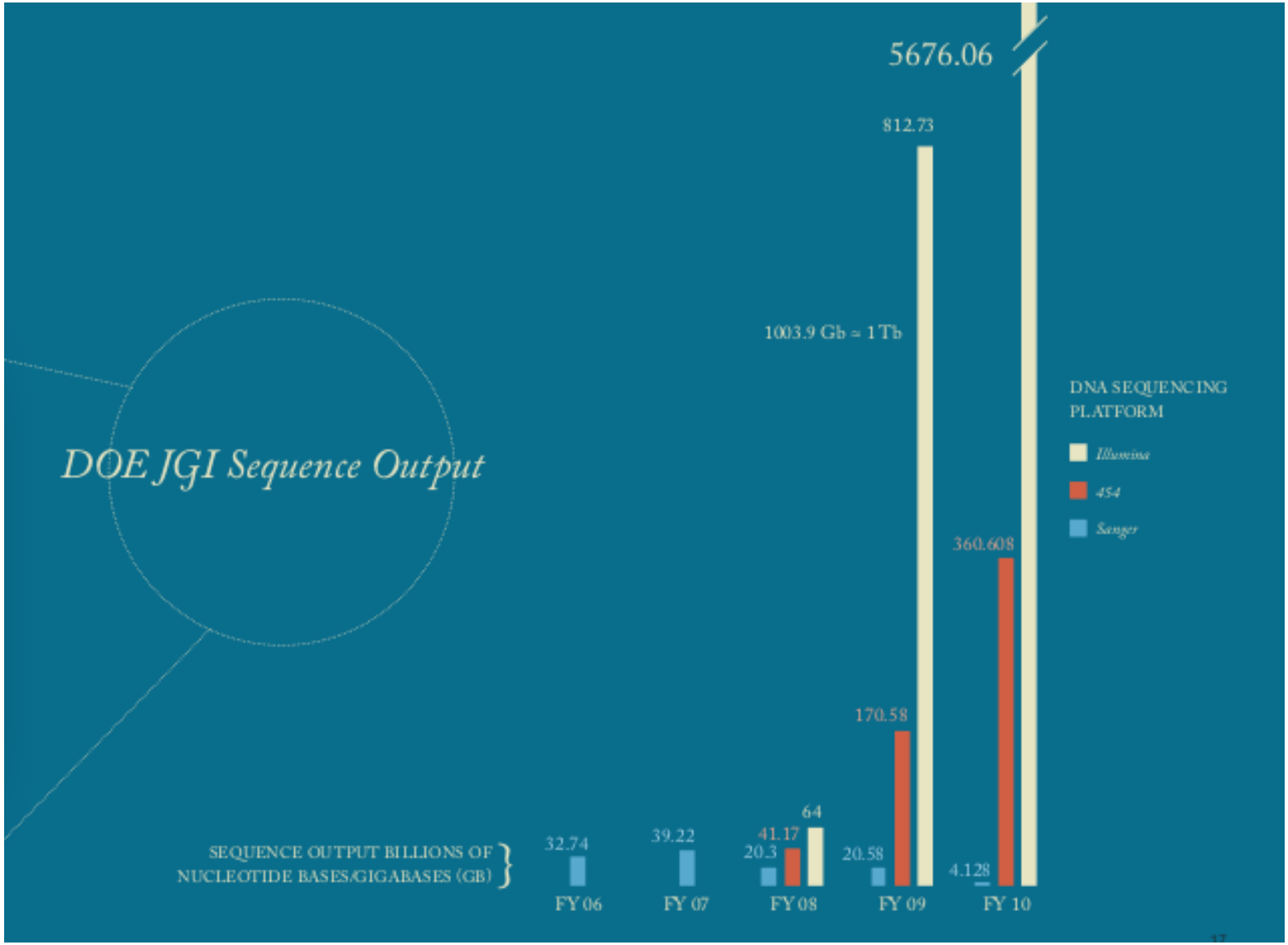
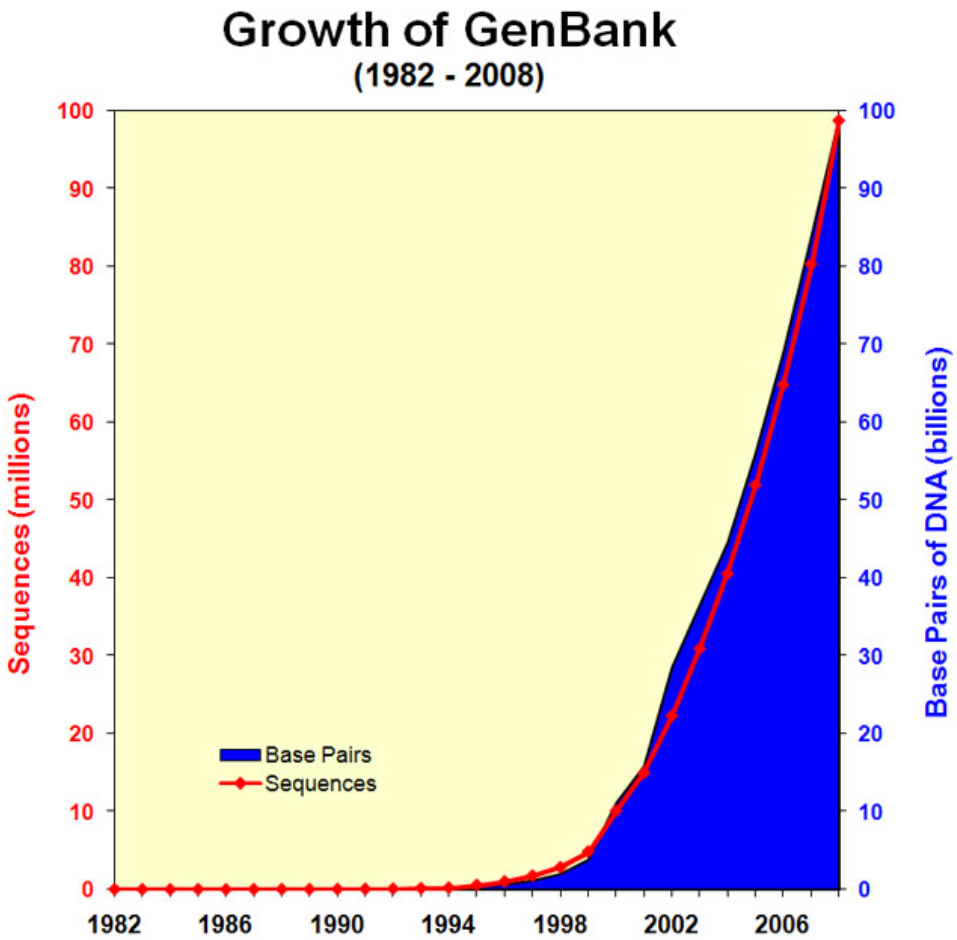


image source JGI annual report 2010

...and so do the databases of known sequences



Genbank non-redundant nucleotide count is now $\geq 10^{11}$ and sequence count $\geq 10^{8}$. image source NIH NCBI release notes

So what's the problem?

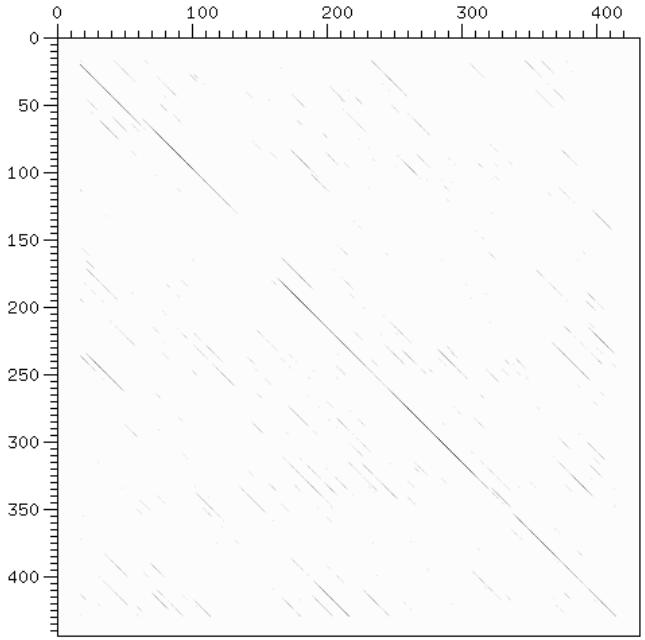
- Indeed, we can find similar sequences by comparing them with local sequence alignment methods
- Such algorithms run in $\mathcal{O}(n \cdot m)$ time scale
- How much would a Smith-Waterman analysis of a single new sequence (1000bp) against genbank take?
- How long for a genome with 10 thousand genes?
- How long for the JGI annual throughput?
- Can we wait that long?
- Can it be done faster?
- What assumptions do we need to make?

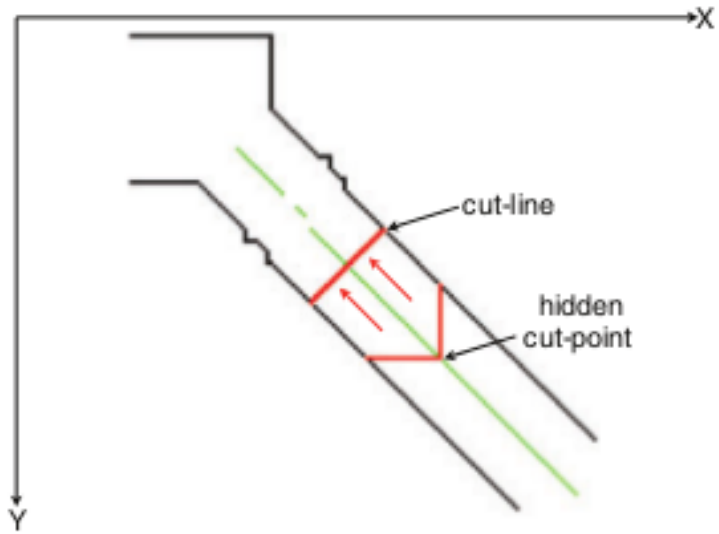
How to define our problem?

- We are looking only for **similar** sequences in the database, so most of our work with S-W algorithm is comparing sequences which will not show up in the result
- Can we tell if a sequence is *not-similar* more quickly than S-W?
- We need to define a meaningful way of specifying our definition of *not-similar*
- **We need an algorithm that can reject bad alignments based on a meaningful and computable criteria**

First idea: global alignments

Good global alignments reside close to the diagonal





- Restricting to search within fixed distance from diagonal brings our computing time to almost linear
- but **not for local alignments**

image source: pecan algorithm

Second idea: FASTA matching short exact matches

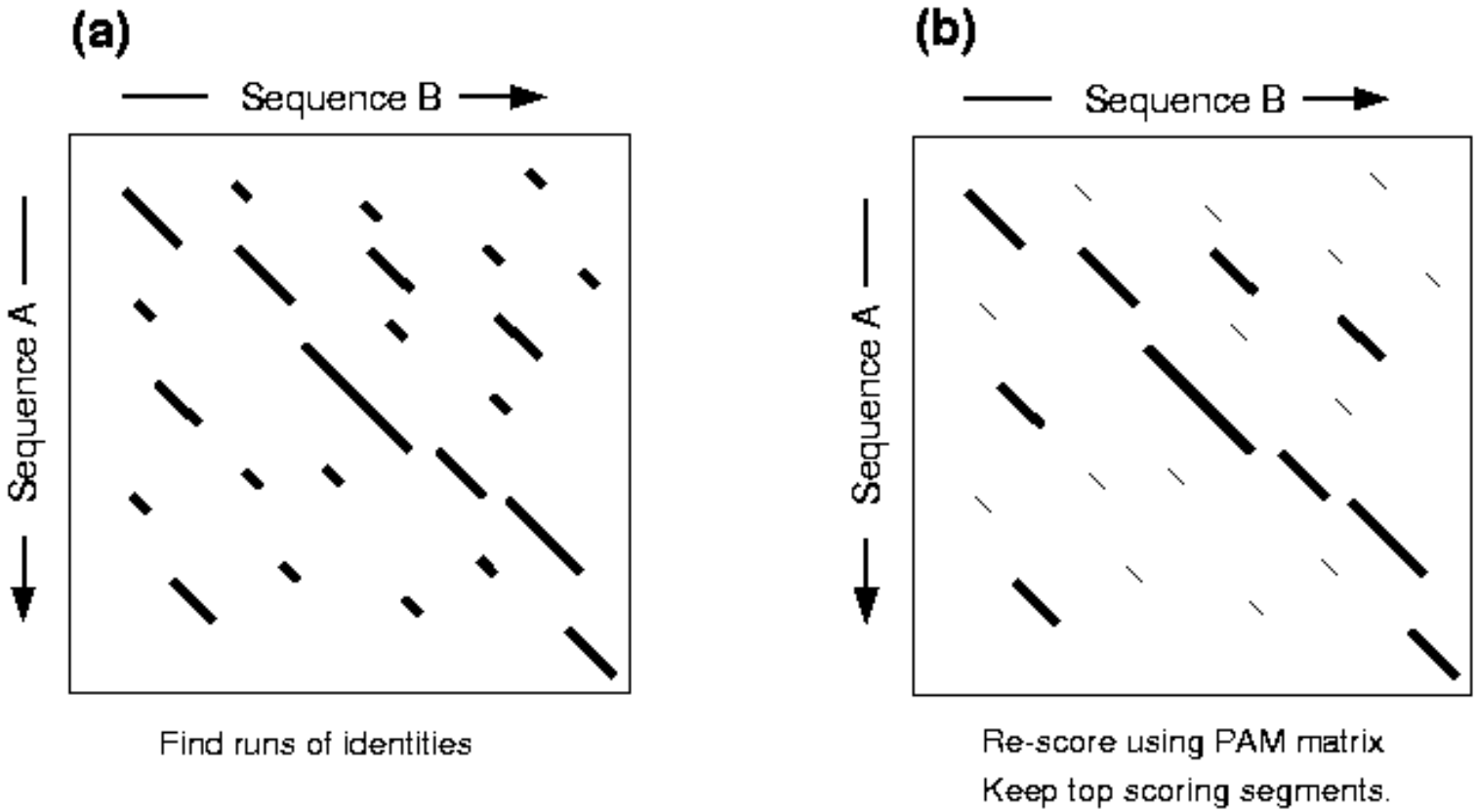


image source GJ Barton

Third idea: FASTA merging short matches to find the right diagonal

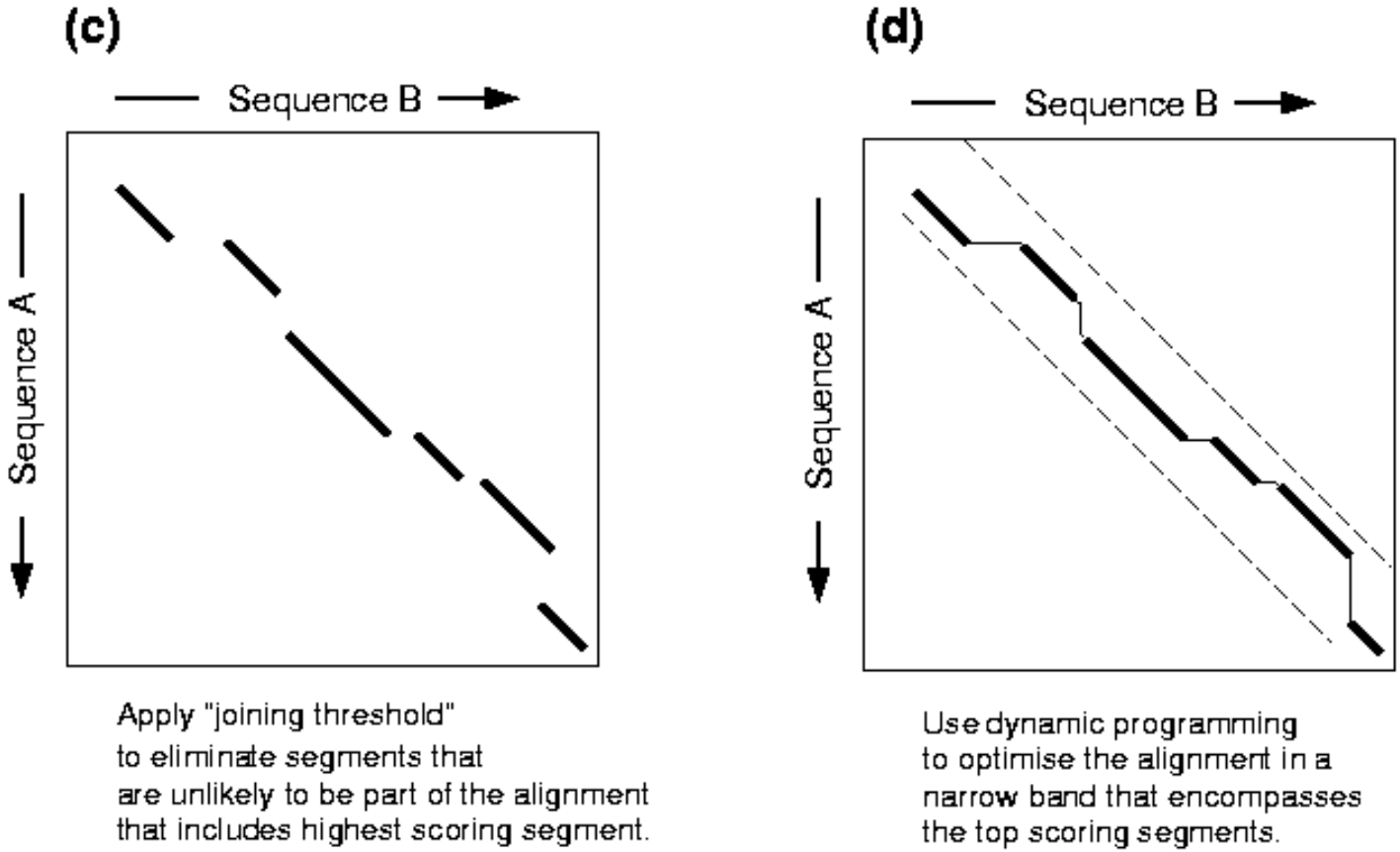
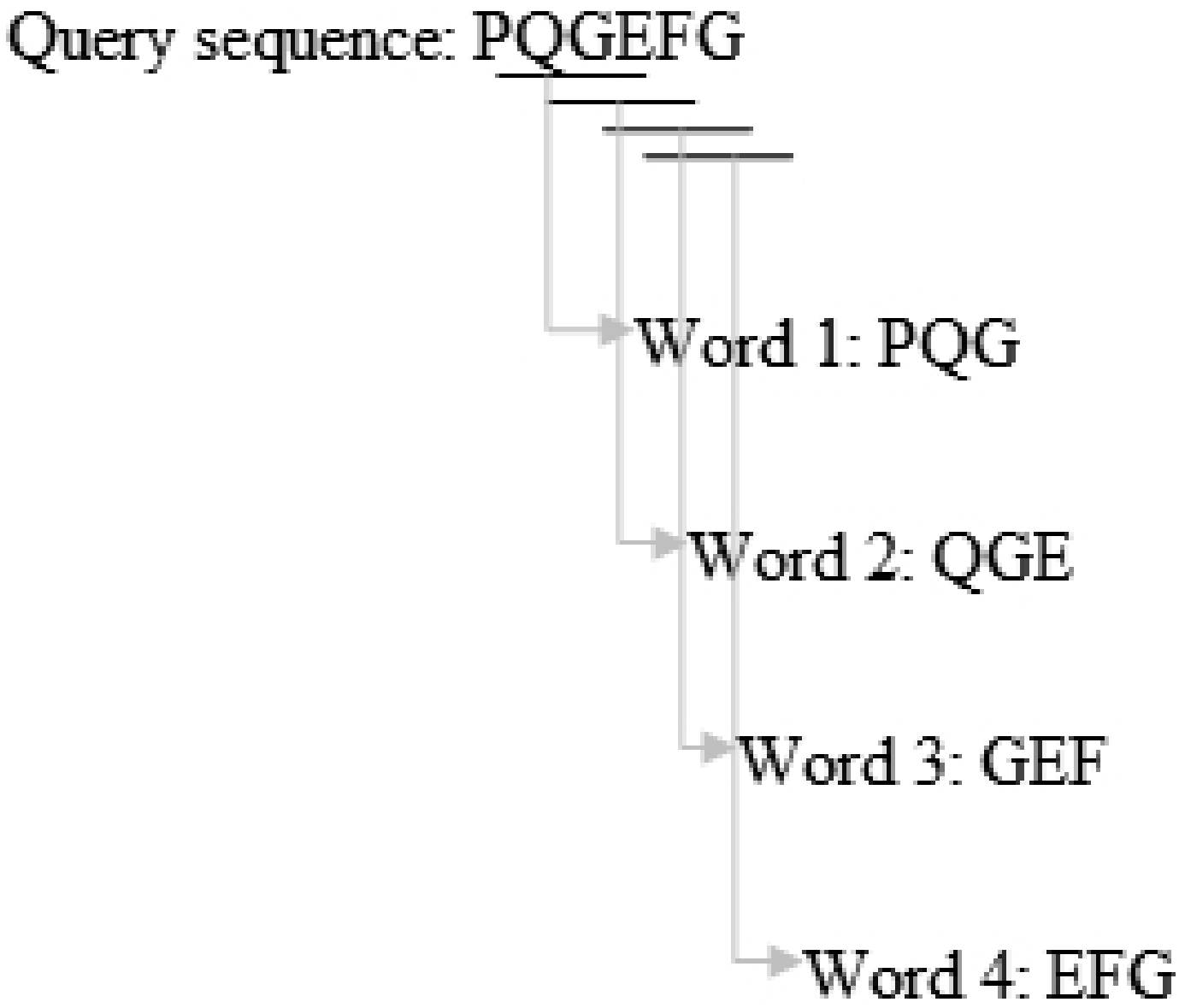


image source GJ Barton

Fourth idea: BLAST *hashing* words *similar* to query



Idea 3': BLAST extending words to segments

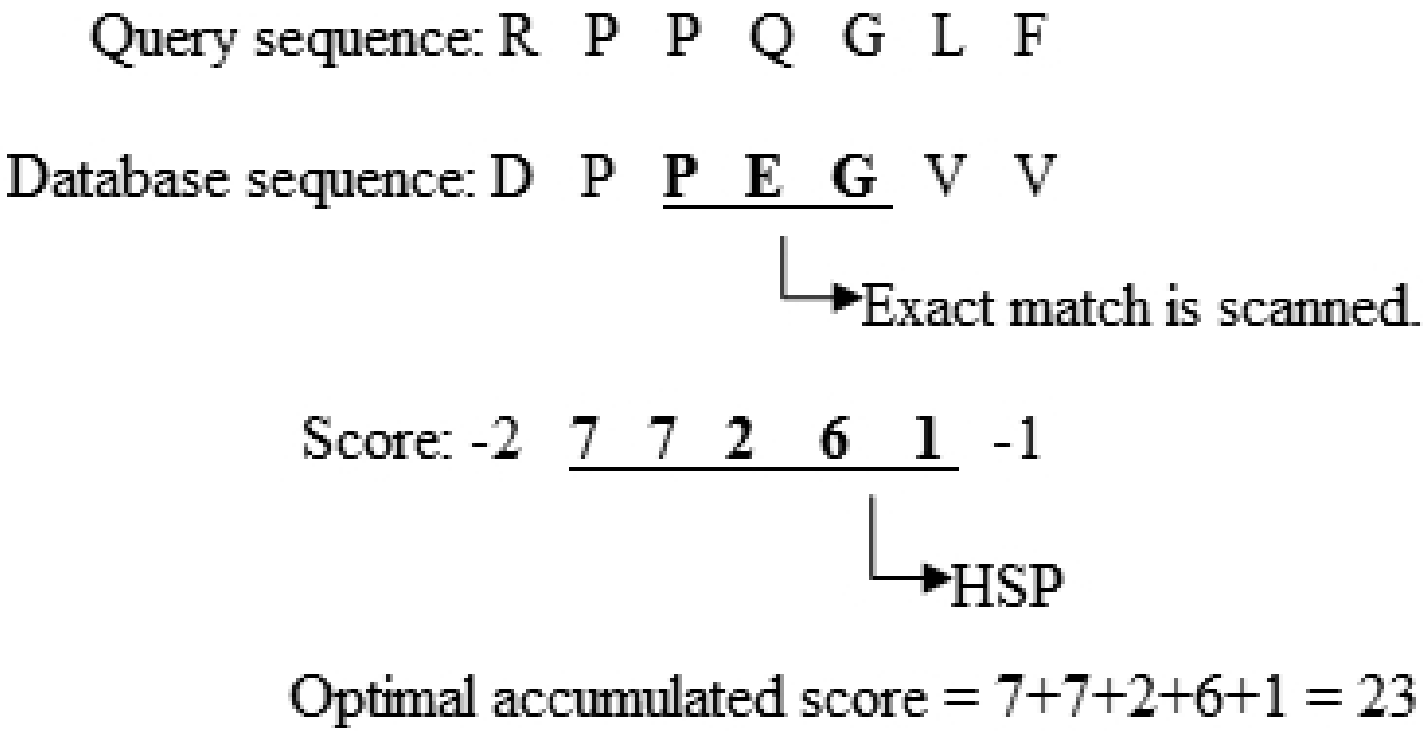


image source wikipedia

Idea 3'': BLAST high scoring segment pairs (HSP)

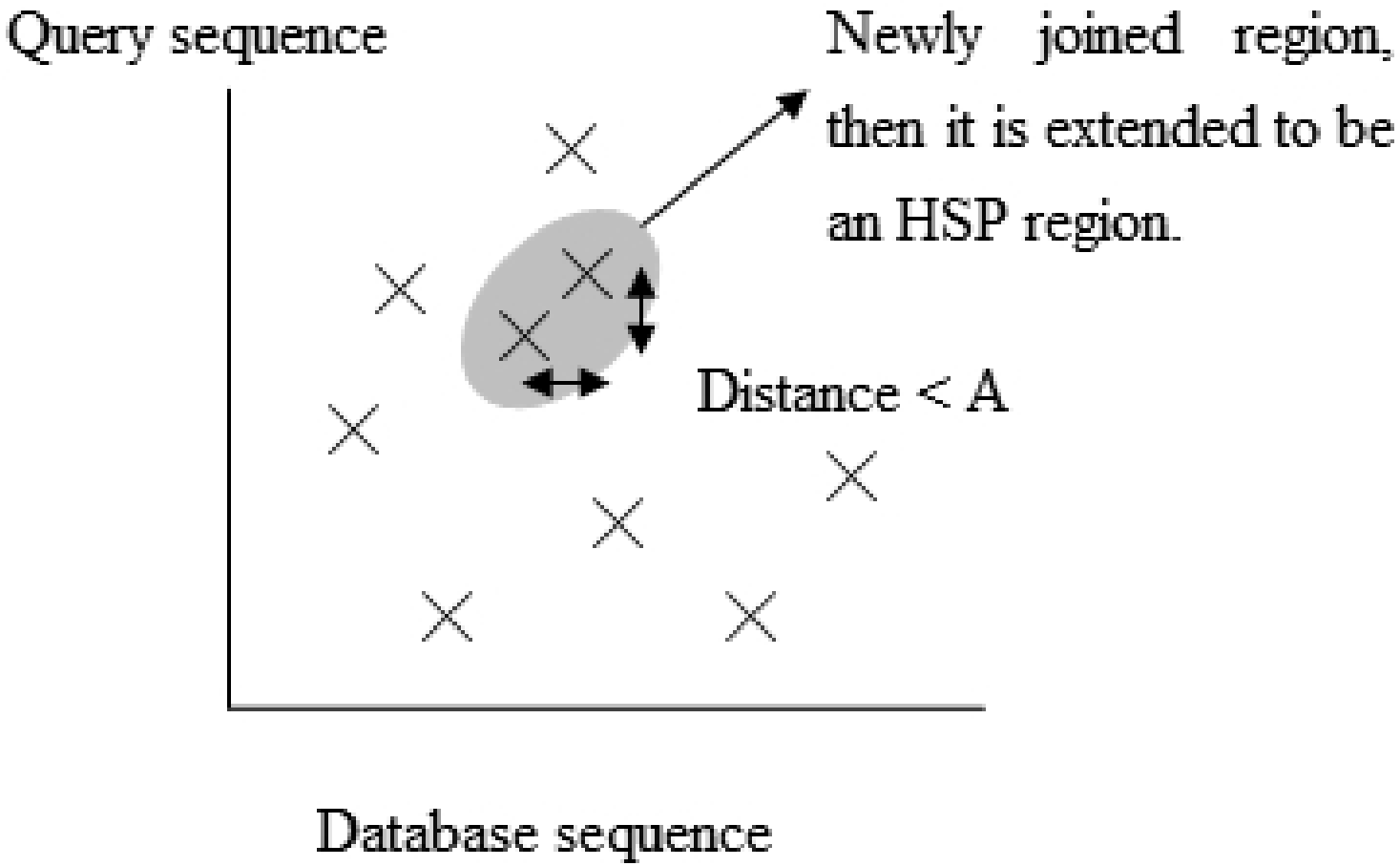


image source wikipedia

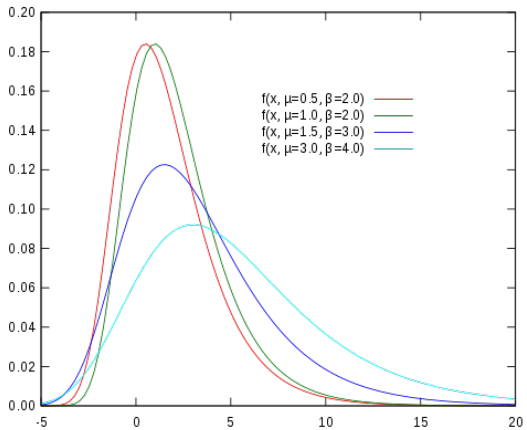
Idea 5: BLAST computing significance of an HSP

- Assume that you found an HSP, is it worth keeping it in the result?
- Behave like a collector: it's only worth keeping if it is rare
- Formally, we want matches which are **ulikely to occur by random** in similar situations (defined by size and composition of the query and database)
- In statistics, we are performing **hypothesis testing**: under **null hypothesis**, there are no matching sequences in the database
- We are interested in the probability of observing a given score (or higher) under assumption of the null model

Idea 5: BLAST computing significance of an HSP

- We cannot really estimate this probability by Monte-Carlo (data is too large for large-scale sampling)
- It is assumed, that it should follow the extreme value distribution (Gumbel distribution)

$$p(s \geq x) = 1 - \exp(-e^{-\lambda(x-\mu)}), \mu = \frac{log(Km'n')}{\lambda}$$



parameters K and λ can be estimated from data, then the E-value is computed $E = pD$, where D is the number of sequences in the database (similar to Bonferroni correction)

image source wikipedia

BLAST: summary

- Very fast algorithm
- Rather complicated heuristic approach
- Many very specialized variants (blastn,blastp, blastx, psi-blast, etc.)
- Does not attempt to find a global alignment, but rather generate a number of *significant* predictions
- Computing e-values and bit-scores (e-value normalized for m and n) is a very important feature