Chapter 4

Database Similarity Searching

Overview

1. Introduction
2. Introduction to Biological Databases
3. Pairwise Sequence Alignment
4. Database Similarity Searching
5. Multiple Sequence Alignment
6. Profiles and Hidden Markov Models
7. Protein Motifs and Domain Prediction
8. Gene Prediction
9. Promoter and Regulatory Element Prediction
10. Phylogenetics Basics
11. Phylogenetic Tree Construction Methods and Programs
12. Protein Structure Basics
13. Protein Structure Visualization, Comparison and Classification
14. Protein Secondary Structure Prediction
15. Protein Tertiary Structure Prediction
16. RNA Structure Prediction
17. Genome Mapping, Assembly and Comparison
18. Functional Genomics
19. Proteomics

Database similarity searching

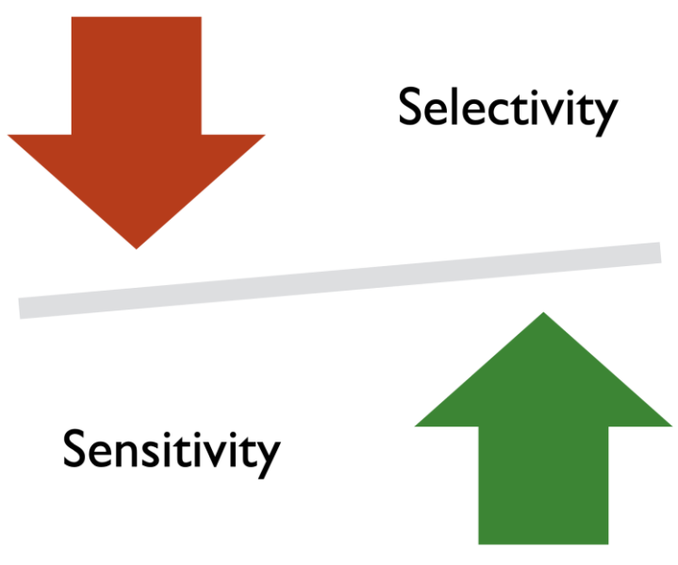
- Arguably the main application of pairwise alignment
- Typically: submission of a *query sequence* to a web-based server (*e.g.* Entrez or EBI)
- Server carries out an automated comparison of the query sequence to <u>every</u> sequence in the database to find the closest match(es)
- Example: finding related sequences to help assign putative functions to the query sequence

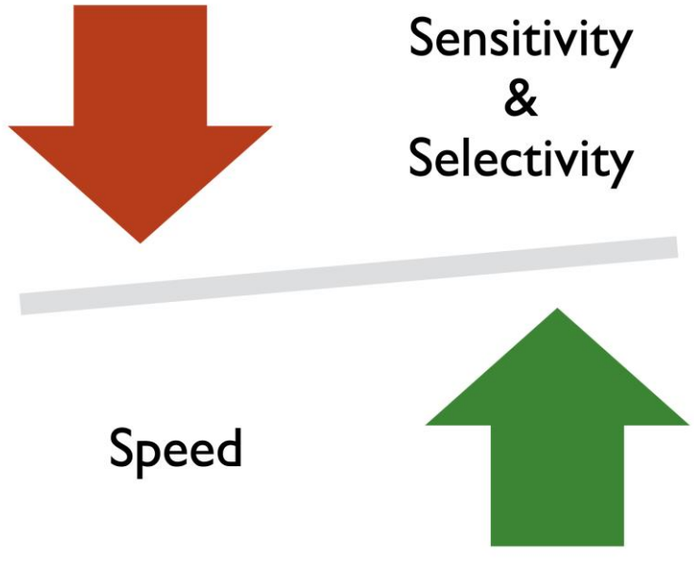
The requirements of database searching

1. Sensitivity: the ability to find as many correct hits (*true positives*) as possible
2. Selectivity (= specificity): the ability to exclude incorrect hits (*false positives*)
3. Speed (dynamic programming methods are usually too slow)

Database searching involves trade-offs

- Increase in sensitivity usually associated with decrease in selectivity and *vice versa*
- Increase in speed often comes at the cost of lowered sensitivity and selectivity
- Some kind of compromise between the three criteria has to be found





Computer-assisted search procedures

Two fundamentally different kinds of search algorithm exist (in general, not only with respect to sequence searches):

- *Exhaustive kind*: examination of all mathematical combinations (*e.g.* dynamic programming)
- *Heuristic kind*: strategy to find a near-optimal solution with no guarantee to find the best solution

Heuristic searches are often needed to finish within a realistic time frame without sacrificing too much of the accuracy of the result

Heuristic database searching

Heuristic algorithms are faster because they examine only a fraction of the possible alignments examined in regular dynamic programming

Two major heuristic algorithms in use for performing database searches: *BLAST* and *FASTA*:

- Typically 50–100 times faster than dynamic programming
- Only limited sacrifice of sensitivity and selectivity
- Both BLAST and FASTA essentially work by initially identifying short sequence segments rather than by attempting to immediately align full-length sequences
- Therefore, a large part of the “search space” is eliminated very quickly

Principles behind BLAST and FASTA

Both methods use a heuristic "word method" for fast pairwise sequence alignment:

- Find short stretches (= *words*) of identical or nearly identical letters in two sequences
- Basic assumption: two related sequences must have at least one word in common
- After identification of word matches a longer alignment is obtained by extension of similarity region from words
- Finally, adjacent high-scoring regions are joined into a full alignment

Basic Local Alignment Search Tool (BLAST)

- Developed by Stephen Altschul in 1990
- Objective: finding high-scoring ungapped segments
- Underlying idea: existence of high-scoring ungapped segments above a given threshold indicates pairwise similarity beyond random chance

BLAST

- Each word is typically 3 residues for protein sequences and 11 residues for DNA/RNA sequences
- *Seeding*: generation of list that includes every possible word extracted from query sequence
- Search sequence database for occurrence of these words and identify database sequences containing matching words
- Matching words are scored using a substitution matrix (BLOSUM), the word is considered match if above threshold
- Pairwise alignment is extended in both directions until score drops below a threshold

BLAST

- Resulting contiguous aligned segment pair without gaps is called "high-scoring segment pair" (HSP) = "maximum scoring pairs"
- More recent improvement of BLAST: gaps are now allowed during the extension step

BLAST: a simple example

1. Query: **MRDPYNKLIS**
2. Scan every three residues to be used in searching BLAST word database.
3. Assuming one of the words finds matches in the database.

Query	**PYN**	**PYN**	**PYN**	**PYN**	**...**
Database	**PYN**	**PFN**	**PFQ**	**PFE**	**...**

4. Calculate sums of match scores based on BLOSUM62 matrix.

Query	**PYN**	**PYN**	**PYN**	**PYN**	**...**
Database	**PYN**	**PFN**	**PFQ**	**PFE**	**...**
Sum of score	**20**	**16**	**10**	**10**	**...**

5. Find the database sequence corresponding to the best word match and extend alignment in both directions.

Query	**M**	**R**	**D**	**PYN**	**K**	**L**	**I**	**S**	
Database	**M**	**H**	**E**	**PYN**	**D**	**V**	**P**	**W**	
	←				→				
	extension to left				extension to right				

6. Determine high scored segment above threshold (22).

Query	**M**	**R**	**D**	**PYN**	**K**	**L**	**I**	**S**	
Database	**M**	**H**	**E**	**PYN**	**D**	**V**	**P**	**W**	
	5	**0**	**2**	**20**	**-1**	**1**	**-3**	**-3**	
									HSP, total score 24

Figure 4.1: Illustration of the BLAST procedure using a hypothetical query sequence matching with a hypothetical database sequence. The alignment scoring is based on the BLOSUM62 matrix (see Chapter 3). The example of the word match is highlighted in the box.

Variants of BLAST

- BLASTN: nucleotide query against a nucleotide sequence database
- BLASTP: protein query against a protein sequence database
- BLASTX: uses nucleotide sequences as queries and translates them in all six reading frames to produce translated protein sequences, which are used to search a protein sequence database

Variants of BLAST

- TBLASTN: protein query against a nucleotide sequence database with the sequences translated in all six reading frames
- TBLASTX: uses nucleotide sequences translated in all six reading frames to search against a nucleotide sequence database that has all the sequences translated in six frames

All of the above are available on the Entrez BLAST server: https://blast.ncbi.nlm.nih.gov/Blast.cgi

Advantages of using protein sequences in searches

- 20 amino acids *vs* 4 nucleotides (so with amino acids, each letter/symbol contains more information)
- Amino acid substitution matrices take into account differences in substitution likelihood
- However: consideration of all possible reading frames is computationally demanding (esp. TBLASTX)

False positives and statistical significance

- Problem: if the database is large enough, a number of unrelated sequences will align well with the query (*i.e.* with high scores), simply by chance (*false positives*)
- Therefore, we need some kind of indicator to establish how significant database search results really are
- *Cf* the *P*-value that indicates the chance of a pairwise sequence alignment arising by chance

The *E*-value

- ***E*-value** (expectation value): the expected number of (false) positives if we were to align an equally long random sequence against an equally large database of random sequences
- The closer the *E*-value is to zero, the more significant the match
- The *E*-value is approximated by (Altschul 1990):

$$E = m \times n \,/\, 2^{S'}$$

with:

m = length of the query sequence
n = total number of residues in the database
S' = *bit score* (depends on the alignment score, S)

The bit score

- **Bit score (*S´*)**: scaling of the ("raw") alignment score from BLAST (and BLOSUM):

$$S' = (\lambda \times S - \ln K) / \ln 2$$

with:

S = raw alignment score

λ = "Gumbel distribution constant"

K = constant that depends on the method / scoring matrix

The constants (λ and K) can be obtained empirically by doing many database searches with random sequences

Interpretation of *E*-values

- $E < 10^{-50}$: extremely high confidence that the database match is a result of homologous relationships
- $10^{-50} < E < 0.01$: match can be considered a result of homology
- $0.01 < E < 10$: match is considered not significant, but may hint at a remote homology
- $10 < E$: either unrelated sequences or extremely distant relationships

Interpretation of bit scores

- Normally not the most useful statistic to look at, as it does not take into account database size and query length (which are very important for assessing the significance of a hit)
- However, for average-length proteins and typical current databases, a bit score of 50 is almost always significant
- A bit score of 40 is only significant in searches of relatively small protein databases
- Usefulness: bit score is a normalised alignment score that allows direct comparisons between hits from different databases and/or scoring matrices

E-value vs bit score

***E*-value** normally the most useful metric but depends on the database size

- Credibility of the sequence match decreases as the database grows, *i.e.* one may "lose" previously detected homologs as the database is enlarged

Bit score is independent of query sequence length, database size and scoring matrix used

- Alternative statistical indicator
- Provides a constant statistical indicator that makes it possible to directly compare alignment quality of hits from multiple searches, carried out in different ways

Low complexity regions (LCRs)

- LCRs are regions that contain highly repetitive residues (repeats or segments characterised by a small number of heavily over-represented residue types)
- Example: “GGGGGGSSGGGGSGGGDGGGGGG”
- LCRs make up around 15% of all protein sequence information in public databases
- LCRs cause fortuitous database matches and lead to artificially high alignment scores for evolutionarily unrelated sequences

How to deal with low complexity regions

- **Masking**: filtering out problematic regions in both the query and database sequence to improve the signal-to-noise ratio:
  - **Hard** masking: replacement of LCR sequences with an ambiguity character such as N for nucleotide residues or X for amino acid residues, that are completely ignored by the BLAST program
  - **Soft** masking: converting the problematic sequences to lower case letters ignored in construction of the word dictionary, but used in word extension and optimization of alignments

Identifying low complexity regions

SEG-filtering

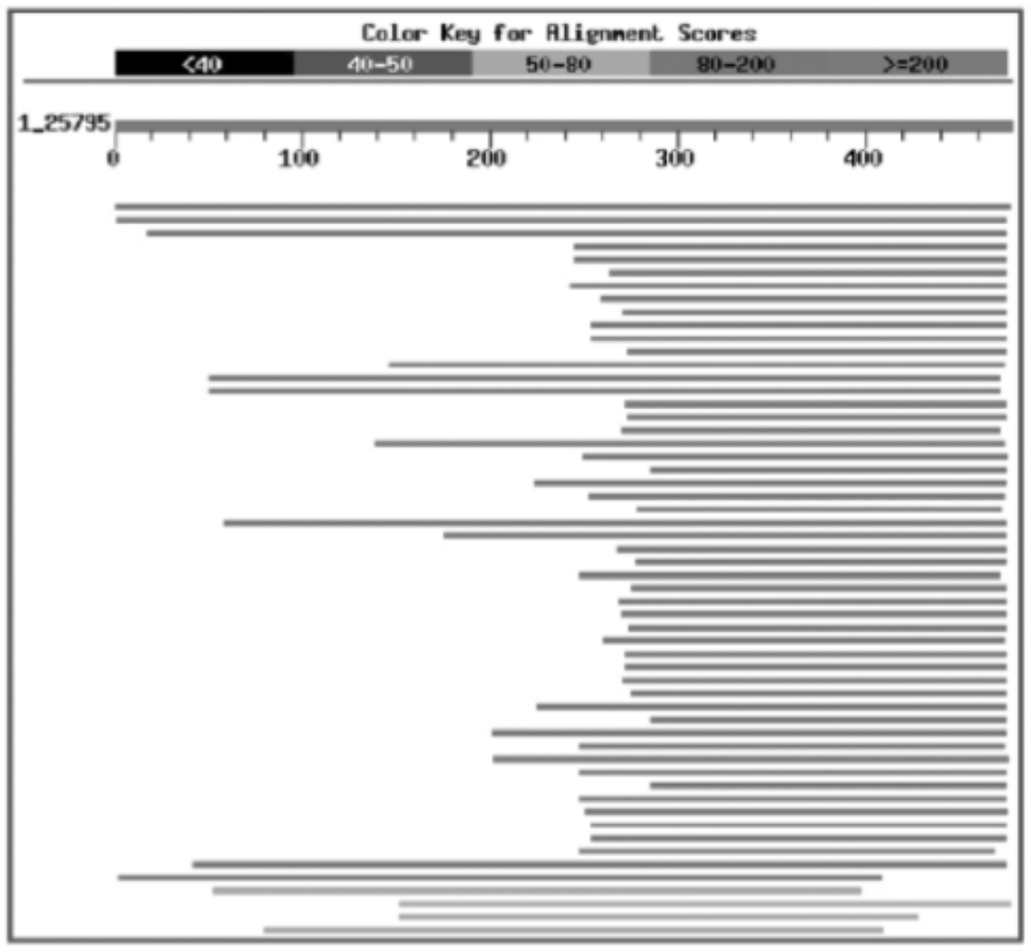
- Identifies LCRs by comparing residue frequencies of a certain region with average residue frequencies in the database
- This is integrated into BLAST web-based program

RepeatMasker

- LCRs are identified by built-in library of repetitive elements using Smith-Waterman algorithm
- http://www.repeatmasker.org

BLAST output

Graphical overview



Matching list

Sequences producing significant alignments:		Score (bits)	E Value
gi|22958938|ref|ZP_00006599.1|	COG3920: Signal transduction...	896	0.0
gi|22968827|ref|ZP_00016409.1|	COG3920: Signal transduction...	390	e-107
gi|39933087|ref|NP_945363.1|	putative signal transduction h...	365	e-100
gi|17935877|ref|NP_532667.1|	two component sensor kinase [A...	175	2e-42
gi|15889280|ref|NP_354961.1|	AGR_C_3616p [Agrobacterium tum...	175	2e-42
gi|31322739|gb|AAP22926.1|	CheS3 [Rhodospirillum centenum]	158	2e-37
gi|16126793|ref|NP_421357.1|	sensor histidine kinase, putat...	157	5e-37
gi|16127400|ref|NP_421964.1|	sensor histidine kinase, putat...	155	1e-36
gi|15966187|ref|NP_386540.1|	HYPOTHETICAL PROTEIN [Sinorhiz...	155	2e-36
gi|16264804|ref|NP_437596.1|	putative two-component sensor ...	152	2e-35
gi|2808506|emb|CAA12536.1|	ExsG protein [Sinorhizobium meli...	151	2e-35
gi|13476692|ref|NP_108261.1|	two-component, sensor histidin...	149	9e-35
gi|16127278|ref|NP_421842.1|	sensor histidine kinase, putat...	149	1e-34
gi|17939110|ref|NP_535898.1|	two component sensor kinase [A...	147	4e-34
gi|13473179|ref|NP_104746.1|	hypothetical protein [Mesorhiz...	147	6e-34
gi|16119758|ref|NP_396464.1|	AGR_pAT_788p [Agrobacterium tu...	147	6e-34
gi|13488521|ref|NP_109528.1|	sensory transduction histidine...	146	1e-33
gi|16125089|ref|NP_419653.1|	sensor histidine kinase, putat...	145	1e-33
gi|22957499|ref|ZP_00005199.1|	COG3920: Signal transduction...	145	2e-33

BLAST output

Alignment output

- header
- statistics
- alignment

```
>gi|22968827|ref|ZP_00016409.1|  COG3920: signal transduction histidine kinase [Rhodospirillum
          rubrum]
         Length = 489

 Score =  377 bits (968), Expect = e-103
 Identities = 235/484 (48%), Positives = 306/484 (63%), Gaps = 14/484 (2%)

Query: 3   PAEIDELRRRLHEAEETLKAIRQGDVDALVVGASDDTDVYVIGGDPDICRSFLDMMEIGA 62
           P  + ELRRRL EAEETL AIR+G+VDALV+G     +V+ IGGD +  R+F++ M+ GA
Sbjct: 4   PVVLSELRRRLAEAEETLNAIREGEVDALVIGEGGVDEVFAIGGDTESYRTFMEAMDTGA 63

Query: 63  AALDNTGRVLYANAVLADLVGRPLPELEGMRL-----SELTGDPAXXXXXXXXXXXXXXI 117
           AA+D  GRVLYAN+ L  L+  PLP L+G  L     +    +                I
Sbjct: 64  AAVDEDGRVLYANSALCRLIDHPLPTLQGKPLVSFFDARAAAEIGQMVGKTANQREKVEI 123

Query: 118 PLGVAGAER-QVMLSCGK-LRLGTVSGHAVTFTDFTEQLAAERSRQNEKAALAIIACANE 175
            L  A  +  QV L   K +RLG V GHAVTFTD TE++ +E + + E+ A AIIA ANE
Sbjct: 124 SLKDAATKMAQVFLVSAKPVRLGLVQGHAVTFTDLTERVRSETAERAERIAAAIIASANE 183

Query: 176 PVFVCDTLGLITHXXXXXXXXXXXXXRPLSEVMDLSVGDGTGLLTLGEIVAQATEGIP 235
            V VCD +G+ITH               +   + + L+  D   L++ G ++  A  G
Sbjct: 184 IVVVCDRVGNITHANSAASAIYDGDLIGKMPEDAIPLTFTDAPDLMSGGALIDLALNGQA 243

Query: 236 VQGIEAVAAEGTPF--YLISAAPLQVPGEAVSGCVITMVDLSQRKAAERHQQLLLRELDH 293
            QGIEA+A        YLISAAPLQV  + +SGCV+TMVDLSQRKAAE  Q LL+RELDH
Sbjct: 244 RQGIEAIATRAPKVKDYLISAAPLQVTEDQISGCVLTMVDLSQRKAAEMQQLLLMRELDH 303

Query: 294 RVKNTLALVMSISRRTMHSEETLEGYQKAFTARIQALAATHNLLADKSWSDISIRDVLVR 353
           RV+NTLALV+SIS RT+ +E+TL+G+ +AFT RI  LAATH+LLA + W+ +S+ D++
Sbjct: 304 RVRNTLALVLSISNRTLSNEDTLQGFHQAFTQRIHGLAATHSLLAKQGWTKLSLMDIVRA 363

Query: 354 ELAPYNEGFSQRILVEVPDVEIEPRSAIALGLVIHELATNATKYGSLSTPEGQ--VRVRG 411
           ELAPY E    R+ +E  +V + PR+AIALGL+ HELATNA KYG+LS   G   V VRG
Sbjct: 364 ELAPYVETDGTRLRLEGGEVALIPRAAIALGLIFHELATNAVKYGALSREGGHVLVAVRG 423

Query: 412 LPGADEPADVVCLEWLERGGPPVSEPTRSGFGQTVIRHAFAYAEGGGAEVSFEPDGVRCR 471
            P AD  A  V  +W+E GGP VS P R GFG TVI H+ AY+  GG ++SF P+GV C
Sbjct: 424 -PTADGAAMKV--DWVESGGPNVSPPQRKGFGHTVISHSLAYSSKGGTDLSFPPEGVICA 480

Query: 472 VSVP 475
           + +P
Sbjct: 481 LRIP 484
```

\+ : nonidentical but similar residues

\- : gap

X : LCR

FASTA

- FASTA (= FAST-ALL):
  Lipman & Pearson, 1985 (older than BLAST)
- https://www.ebi.ac.uk/jdispatcher/sss/fasta
- Uses a “hashing” strategy to find matches for a short stretch of identical residues with a length of *k* (= *ktuples* or *ktups*)
- Typically ktup is composed of 2 residues for protein sequences and 6 residues for DNA sequences
- Construction of lookup table containing ktup matches for the two sequences under consideration

FASTA

- The positional difference (= *offset*) for each ktup match (difference between the position of the ktup in the first sequence and its position in the second sequence) is recorded in the lookup table
- Ktup matches that have the same offset value are linked to reveal gap-less alignable regions (these correspond to a diagonal line in a dot matrix)
- The aligned regions are then scored using a substitution matrix and the highest-scoring segments selected

FASTA

- Segments near the same diagonal are connected to form a single alignment, allowing the introduction of gaps
- Gapped alignment is refined using the Smith-Waterman algorithm to produce final alignment
- Statistical evaluation as in BLAST (*E*-values and bit scores)

FASTA: a simple example

1. Given two amino acid sequences for comparision:

sequence 1 **AMPSDGL**
sequence 2 **GPSDNAT**

2. Construct a hashing table:

amino acid	sequence position		offset
	seq 1	seq 2	
A	1	6	-5
D	5	4	1
G	6	1	5
L	7	-	-
M	2	-	-
N	-	5	-
P	3	2	1
S	4	3	1
T	-	7	-

3. Identify residues with the same offset values (highlighted in grey).

4. Find the matching word of three residues in the order of 3, 4 and 5 in one sequence and 2, 3,and 4 in the other.

5. This allows establishment of alignment between the two sequences.

sequence 1 **AMPSDGL-**
| | |
sequence 2 **-GPSDNAT**

Figure 4.3: The procedure of ktup identification using the hashing strategy by FASTA. Identical offset values between residues of the two sequences allow the formation of ktups.

FASTA: joining short ungapped alignments

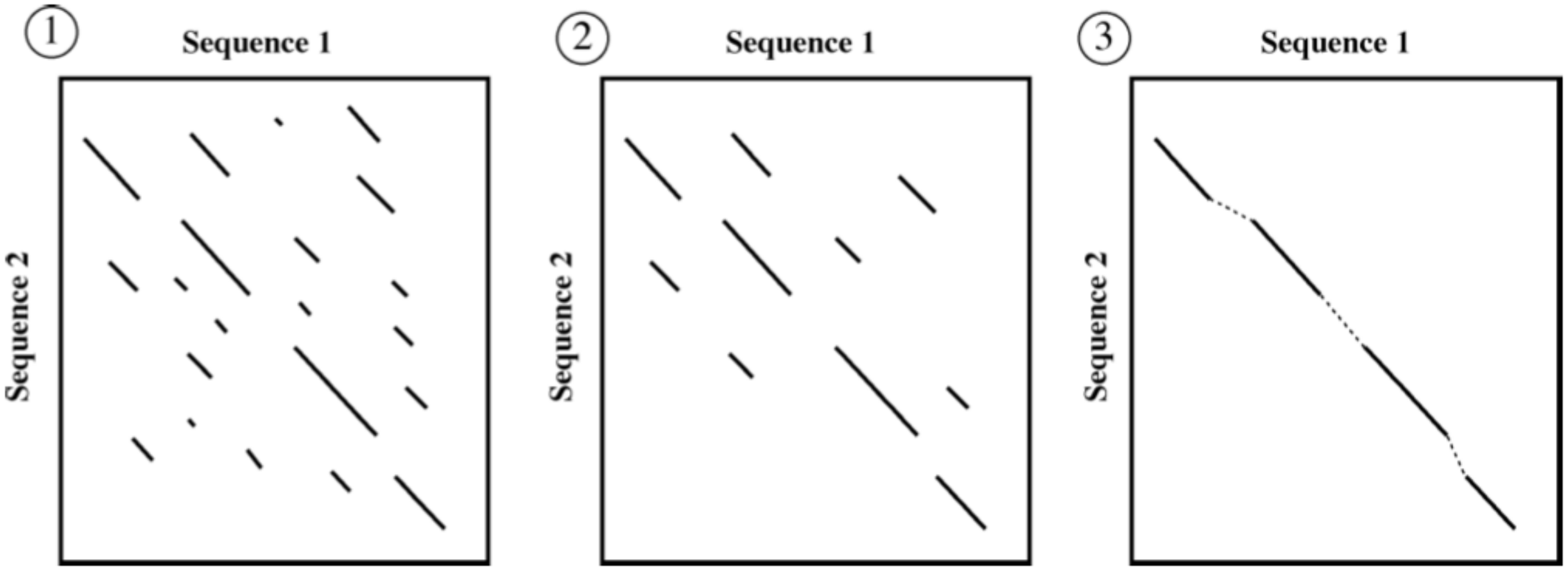


Figure 4.4: Steps of the FASTA alignment procedure. In step 1 (*left*), all possible ungapped alignments are found between two sequences with the hashing method. In step 2 (*middle*), the alignments are scored according to a particular scoring matrix. Only the ten best alignments are selected. In step 3 (*right*), the alignments in the same diagonal are selected and joined to form a single gapped alignment, which is optimized using the dynamic programming approach.

FASTA *Z*-score

Z-score: additional statistical parameter that describes the number of standard deviations from the mean score for the database search:

- 15 < *Z*: match can be considered extremely significant, with certainty of a homologous relationship
- 5 < *Z* < 15: sequence pair highly probable homologs
- *Z* < 5 : relationships less certain

Comparison of FASTA and BLAST

Major difference:

- BLAST uses a substitution matrix to find matching words
  - FASTA identifies identical matching words using the hashing procedure with a smaller window size (tuples)
- BLAST: faster, higher specificity
- FASTA: more sensitive with a better coverage of homologs

Database searching with the Smith-Waterman method

- Rarely used, because it is too slow ("computationally expensive")
- However: BLAST can miss up to 30% of truly significant hits for some protein families
- Parallel implementations of the Smith-Waterman algorithm can speed things up (to some extent):

➢ ScanPS
http://www.ebi.ac.uk/scanps

➢ ParAlign
http://www.paralign.org

2024: both of these servers no longer online!