Computational Biology

Lectures

part III Sequence analysis of nucleic acids and proteins

Sequence analysis

- The aim of computational sequence analysis is to get higher structural and functional information from nucleotid or amino acid sequence data

Major difference between physics and biology

- Computation in biology usually involves the processing of emperical knowledge rather than solving first principle equations

Basic idea sequence analysis

- Use emperical knowledge from molecular biology: when 2 molecules share similar sequences, they will have similar 3D structure and biological function.

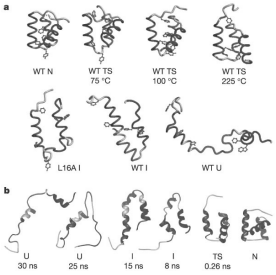
Biological problem vs method in computer science I

- Similarity search: pairwise sequence alignment, data base search for similar seqs, multiple seqs alignment,phylogenetic tree, protein 3D structure alignment – Dynamic programming, Simulated annealing, Genetic algorithms, Neural networks, Hierarchical clustering method
- Structure / function prediction, ab initio prediction :RNA sec. Struct pred, RNA 3D struct pred, protein 3D struct - Dynamic programming, Simulated annealing, Genetic algorithms, Neural networks, Molecular Dynamics

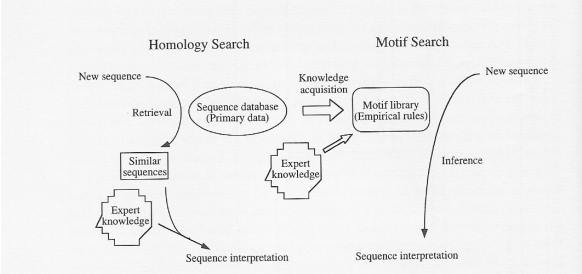
Biological problem vs method in computer science II

- Structure / function prediction, knowledge based prediction: motif extraction, coding region pred., protein secundary struct., protein 3D struct. Pred – discriminant analysis, neural networks, Hidden Markov Model, formal grammar
- Molecular classification: superfamily classification, ortholog/paralog grouping of genes, 3D fold classification- Hierarchical cluster analysis, neural network

Unfolding and folding of the engrailed homeodomain protein
(Molecular Dynamics simulations, Nature 421:863-7, 2003)



Homology search vs motif search



Amino acids

Ala	A	Alanine
Arg	R	Arginine
Asn	N	Asparagine
Asp	D	Aspartic acid
Cys	C	Cysteine
Gln	Q	Glutamine
Glu	E	Glutamic acid
Gly	G	Glycine
His	H	Histidine
Ile	I	Isoleucine
Leu	L	Leucine
Lys	K	Lysine
Met	M	Methionine
Phe	F	Phenylalanine
Pro	P	Proline
Ser	S	Serine
Thr	T	Threonine
Trp	W	Tryptophan
Tyr	Y	Tyrosine
Val	V	Valine
Asx	B	Asn or Asp
Glx	Z	Gln or Glu
Sec	U	Selenocysteine
Unk	X	Unknown

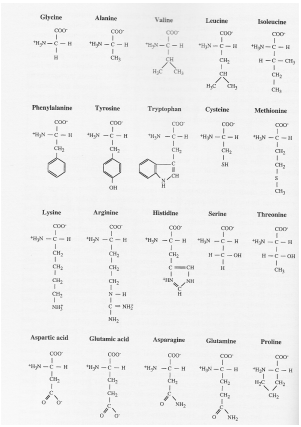
Amino acid index

- Based on physiochemical and biochemical properties of individual amino acids
- Amino acids can be classified into hydrophobic or hydrophylic, charged or uncharged, with or without OH group etc
- Derived property 20 numerical values, the amino acid index
- Simularity relations between amino acids: amino acid mutation matrix or amino acid similarity score 20 x 20

PAM-250 mutation matrix

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	2																			
Arg	−2	6																		
Asn	0	0	2																	
Asp	0	−1	2	4																
Cys	−2	−4	−4	−5	12															
Gln	0	1	1	2	−5	4														
Glu	0	−1	1	3	−5	2	4													
Gly	1	−3	0	1	−3	−1	0	5												
His	−1	2	2	1	−3	3	1	−2	6											
Ile	−1	−2	−2	−2	−2	−2	−2	−3	−2	5										
Leu	−2	−3	−3	−4	−6	−2	−3	−4	−2	2	6									
Lys	−1	3	1	0	−5	1	0	−2	0	−2	−3	5								
Met	−1	0	−2	−3	−5	−1	−2	−3	−2	2	4	0	6							
Phe	−4	−4	−4	−6	−4	−5	−5	−5	−2	1	2	−5	0	9						
Pro	1	0	−1	−1	−3	0	−1	−1	0	−2	−3	−1	−2	−5	6					
Ser	1	0	1	0	0	−1	0	1	−1	−1	−3	0	−2	−3	1	2				
Thr	1	−1	0	0	−2	−1	0	0	−1	0	−2	0	−1	−3	0	1	3			
Trp	−6	2	−4	−7	−8	−5	−7	−7	−3	−5	−2	−3	−4	0	−6	−2	−5	17		
Tyr	−3	−4	−2	−4	0	−4	−4	−5	0	−1	−1	−4	−2	7	−5	−3	−3	0	10	
Val	0	−2	−2	−2	−2	−2	−2	−1	−2	4	2	−2	2	−1	−1	−1	0	−6	−2	4

Amino acids



Sequence alignment I

- The problem of obtaining the best sequence alignment is equivalent to optimize a given score function that represents overall similarity and that is computed both from the similarity measure of amino acids or nucleotids and the penalties for insertions and deletions.

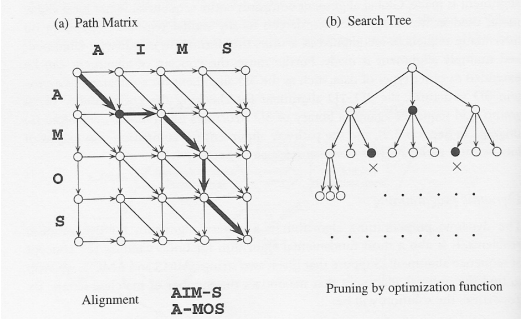
Sequence alignment II

- Global alignment compares enire sequences
- Local alignment (homology searches) locally similar regions

Sequence alignment III (not only about comparing sequences)

- Compare two 3D protein structures
- Pathway alignment (comparing two graphs)
- 3D-1D alignment, a sequence comparison to a library of 3D structures

Pairwise sequence alignment by the dynamic programming algorithm



Constructing a score function

- For nucleotid sequences fixed weights for matches /mismatches of the 4 nucleotids
- For amino acid sequences more elaborate substitution matrix is required

For score function in amino acid sequence use PAM-250 mutation matrix (alternative BLOSUM matrix)

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	2																			
Arg	−2	6																		
Asn	0	0	2																	
Asp	0	−1	2	4																
Cys	−2	−4	−4	−5	12															
Gln	0	1	1	2	−5	4														
Glu	0	−1	1	3	−5	2	4													
Gly	1	−3	0	1	−3	−1	0	5												
His	−1	2	2	1	−3	3	1	−2	6											
Ile	−1	−2	−2	−2	−2	−2	−2	−3	−2	5										
Leu	−2	−3	−3	−4	−6	−2	−3	−4	−2	2	6									
Lys	−1	3	1	0	−5	1	0	−2	0	−2	−3	5								
Met	−1	0	−2	−3	−5	−1	−2	−3	−2	2	4	0	6							
Phe	−4	−4	−4	−6	−4	−5	−5	−5	−2	1	2	−5	0	9						
Pro	1	0	−1	−1	−3	0	−1	−1	0	−2	−3	−1	−2	−5	6					
Ser	1	0	1	0	0	−1	0	1	−1	−1	−3	0	−2	−3	1	2				
Thr	1	−1	0	0	−2	−1	0	0	−1	0	−2	0	−1	−3	0	1	3			
Trp	−6	2	−4	−7	−8	−5	−7	−7	−3	−5	−2	−3	−4	0	−6	−2	−5	17		
Tyr	−3	−4	−2	−4	0	−4	−4	−5	0	−1	−1	−4	−2	7	−5	−3	−3	0	10	
Val	0	−2	−2	−2	−2	−2	−2	−1	−2	4	2	−2	2	−1	−1	−1	0	−6	−2	4

Score function

- Equation of the score matrix D

$$D_{i,j} = \max(D_{i-1,j-1} + w_{s(i),t(j)}, D_{i-1,j} + d, D_{i,j-1} + d)$$

Evaluating score function

- Start at upper left corner path matrix
- Apply the score function
- Final value score function, optimal value score function
- In each step 3 possibilities are selected, trace back all steps starting at lower right -> overall alignment

Evaluating score function II

- The number of operations is proportional to the size of nxm matrix, this is an $O(n^2)$ algorithm

Score function with gap penalty constant (global alignment) I

$$D_{i,j} = \max(D_{i-1,j-1} + w_{s(i),t(j)}, D_{i-1,j} + d, D_{i,j-1} + d)$$
$$D_{0,0} = 0$$
$$D_{i,0} = id, 1 \le i \le n$$
$$D_{jd} = jd, 1 \le j \le m$$

d Constant gap penalty

The algorithm is now $O(n^2)$

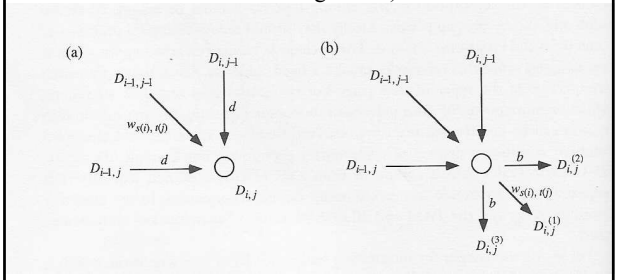
Score function with length dependent gap penalty (global alignment) II

$$D_{i,j} = \max(D_{i-1,j-1} + w_{s(i),t(j)}, \max(D_{i-k,j} + d_k), \max(D_{i,j-k} + d_k))$$

d_k Is the weight for a gap with length k

The algorithm is now $O(n^3)$

Methods for computing the optimal score in the dynamic programming algorithm (a. gap penalty is constant; b. gap penalty is a linear function of the gap length), Needleman and Wunsch algorithm)



Score function with length dependent gap

$$D_{i,j}^{(1)} = \max(D_{i-1,j-1}, D_{i-1,j}, D_{i,j-1}) + w_{s(i),t(j)}$$

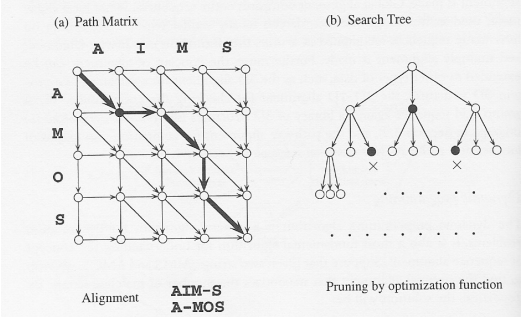
$$D_{i,j}^{(2)} = \max(D_{i-1,j-1} + a, D_{i-1,j}, D_{i,j-1} + a) + b$$

$$D_{i,j}^{(3)} = \max(D_{i-1,j-1} + a, D_{i-1,j} + a, D_{i,j-1}) + b$$

Algorithm is now $O(n^2)$

Note: horizontal and vertical scores are stored separately

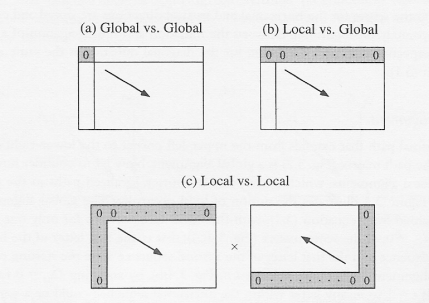
Global alignment



Local alignment

- Corresponds to shorter localized paths in the path matrix

Global and local optimality in pairwise sequence alignment



Local alignment

- Useful for detecting multiple matches within a horizontal sequence consisting of multiple domains, each of which is similar to the vertical sequence (in global alignment only one match would be detected)

Example why local alignment: rice genome I

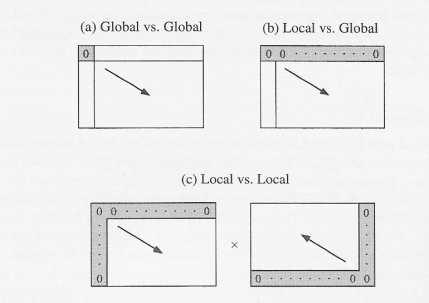
- Goff, S. A. et al. A draft sequence of the rice genome (Oryza sativa L. ssp japonica). Science 296, 92 - 100 (2002).

That means rice probably has more genes than we do, despite having only one-seventh as much DNA.One reason for this difference is that plants' genes seem to be on average much shorter than mammals'. A rice gene is usually about 4,500 DNA letters long. The average human gene probably stretches to over 30,000 letters.

Example why local alignment: rice genome II

- Plants are also prone to copying genes, chromosomes, and sometimes their entire genome. This might be a way for them to generate genetic diversity. Mammals can rearrange individual genes to make several different sorts of protein. Plants show much less of this, seeming to rely on a larger number of shorter genes.

local alignment (c)



Local alignment

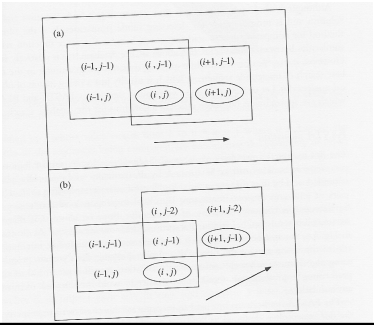
- Elements $w_{s,t}$ can be positive negative (favourable and unfavourable substitutions)
- by modifying score function optimal paths are not entered in path matrix when the element becomes negative, separates local clusters of favourable regions
- Smith Waterman local alignment algorithm trace back procedure, starts at matrix element with maximum alignment score

Score function with gap penalty constant (local alignment) I

$$D_{i,j} = \max(D_{i-1,j-1} + w_{s(i),t(j)}, D_{i-1,j} + d, D_{i,j-1} + d, 0)$$

d Constant gap penalty

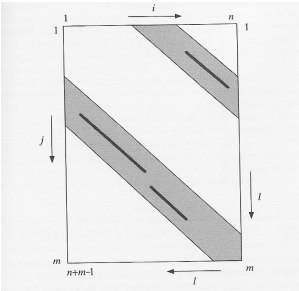
Sequential and parallel processing of the path matrix



FASTA and BLAST

- FASTA a computer program based on the method of W. Pearson and D. Lipman (Proc. Natl. Acad. Sci. USA 85, 2444-2448, 1988) to search similarities between a sequence and a database with sequences. Originally used for the alignment of amino acid sequences.
- BLAST (Basic Local Alignment Tool) from the National Center for Biotechnology Information (NCBI), BLAST is used to scan nucleotide or amino acid databases (for example GenBank and SWISS-PROT) for ``hits''

Dynamic programming: dot matrix



Dynamic programming: dot matrix II

- The area of different stretches is small compared to rest of the matrix: dynamic programming involves a great deal of unproductive computing!
- FASTA was designed to use this property of the dot matrix and to limit search area of dynamic programming

FASTA algorithm I

- Suppose you can specify position diagonal with parameter l. The values l are between

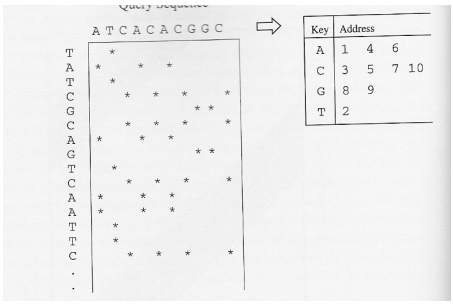
$$1 \le l \le n + m - 1$$

- Possible insertions deletions may shift diagonal
- Limited areas (shaded) are searched by the dynamic programming algorithm

FASTA algorithm II

- In FASTA do an initial search for candidate diagonals
- Use a hash table (lookup table) to get rapid access to data items

FASTA algorithm III: hashing technique for rapid access data items



BLAST algorithm I

- BLAST makes use of the biological observation that cores of conserved segments are often represented as segment motifs
- BLAST is a heuristic algorithm, it incorporates good guesses based on knowledge how sequences are related
- BLAST uses statistics on local alignments in random sequences

Standard genetic code: codon->amino acid

	U		C		A		G		
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	U
	UUC	Phe	UCC	Ser	UAC	Tyr	UGC	Cys	C
	UUA	Leu	UCA	Ser	UAA	*Stop*	UGA	*Stop*	A
	UUG	Leu	UCG	Ser	UAG	*Stop*	UGG	Trp	G
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	U
	CUC	Leu	CCC	Pro	CAC	His	CGC	Arg	C
	CUA	Leu	CCA	Pro	CAA	Gln	CGA	Arg	A
	CUG	Leu	CCG	Pro	CAG	Gln	CGG	Arg	G
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	U
	AUC	Ile	ACC	Thr	AAC	Asn	AGC	Ser	C
	AUA	Ile	ACA	Thr	AAA	Lys	AGA	Arg	A
	AUG	Met, *Start*	ACG	Thr	AAG	Lys	AGG	Arg	G
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	U
	GUC	Val	GCC	Ala	GAC	Asp	GGC	Gly	C
	GUA	Val	GCA	Ala	GAA	Glu	GGA	Gly	A
	GUG	Val	GCG	Ala	GAG	Glu	GGG	Gly	G

BLAST algorithm II

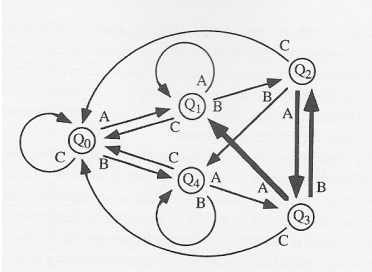
- The optimal ungapped alignment between 2 sequences is called Maximum Scoring Pair (MSP).
- In practice High Scoring Segment Pairs (HSP) are used
- The expected frequency E of observing an HSP with score S of 2 random sequences of length n and m is given by (K and lambda are the Karlin-Altschul parameters):

$$E = Knm(\exp(-\lambda S))$$

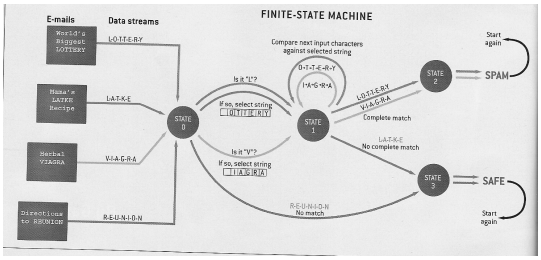
BLAST algorithm III

- Do a systematic search for conserved words. A word is a tuple of letters (default length W=3 for amino acids, W=11 for nucleotids)
- Decompose query into words of length W, create a list of these words and similar words
- Similar words are collected from all other combinations of W-tuples when the score for ungapped alignment with the original word exceeds a given threshold T
- Any word in the list found in the data base with sequences can become a core segment to start local alignment

Finite state automaton for pattern matching



Finite state automaton for pattern matching II (Scient.Am. January issue, 2006)



Finite state machine

- Finite state machines process data streams by matching each input character simultaneously against many different characters indicative of a word W that are stored in memory (not an evaluation word by word).

Statistical significance of similarities

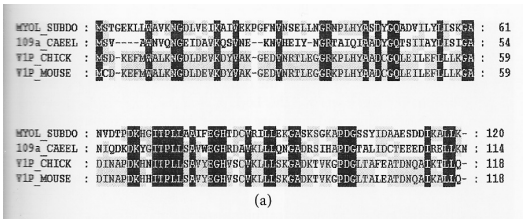
- In BLAST the expected frequency E of observing an HSP with score S of 2 random sequences of length n and m is given by (K and lambda are the Karlin-Altschul parameters):

$$E = Knm(\exp(-\lambda S))$$

- The Z value can be computed (S is observed score, mu is mean and delta is standard deviation):

$$Z = \frac{S - \mu}{\delta}$$

Multiple alignment: example (MYOL_SUBDO:sponge myotrophin, 109a_CAEL: Caenorhabdites, VIP_CHICK, VIP_MOUSE)



(a)

Multiple sequence alignment

- Multiple sequence alignment can be used to investigate global similarities and evolutionary relationships within a sequence family
- For multiple alignment you need to generalize your optimization algorithms for global and local alignments

Multiple sequence alignment

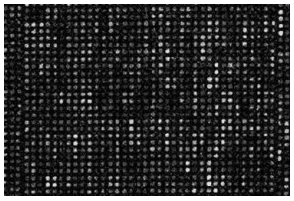
- Merge results of pair wise alignments (disadvantage: not most optimal alignment, gaps are likely to be inserted at different locations)
- Tree-based progressive alignment. First a tree is constructed using a hierarchical clustering of set of sequences.

Intermezzo: Cluster analysis

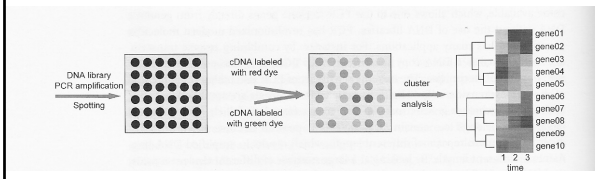
Intermezzo: Cluster analysis I

- Clustering algorithms are a general group of tools from multivariate statistics. They are used to group data objects according to their pair wise similarity with respect to a set of characteristics measured on these objects.

Microarrays I



Microarrays II



Intermezzo: Cluster analysis II

- Similarity (or dissimilarity) is defined numerically by a (dis)similarity function
- Consider n objects with p different characteristics measured on the objects, the profile of object i is a p-dimensional vector:

$$x_i = (x_{i1},\ldots,x_{ip})$$

Intermezzo: Cluster analysis III

- Similarity measure:

$$d_q(x_n, x_m) = \left(\sum_{i=1}^{p} \left|x_{ni} - x_{mi}\right|^q\right)^{\frac{1}{q}}$$

- Note q=1 Manhattan distance, q=2 Euclidean distance
- Other similarity measures are Pearson's or Spearman's correlation coefficient, scoring function (sequences)

Intermezzo: Cluster analysis IV

- Missing value problem, suppose there is at least one missing value k and Euclidean distance
- Without missing value:

$$d_2^2(x_n, x_m) = \sum_{i=1}^{p} (x_{ni} - x_{mi})^2$$

- With missing value:

$$d_2^2(x_n, x_m) = \frac{p}{p-k} \sum_{i=1}^{p} (x_{ni} - x_{mi})^2$$

Intermezzo: Cluster analysis V

- Data transformation: transform data vectors, one possible transformation: divide every component by its Euclidean norm:

$$x = (x_1,\ldots,x_p)^T$$

$$x'_j = \frac{x_j}{\|x\|}$$

Intermezzo: Cluster analysis VI

- Select a similarity measure
- Select a data transformation
- Select a clustering method: two classes the hierarchical and the partitioning methods

Intermezzo: Cluster analysis VII

- Partitioning methods try to find the best partition given a fixed number of clusters (method not discussed in detail)
- Hierarchical methods calculate a full series of partitions starting from n clusters, each of which contains one object, and ending with one cluster that contains all n objects

Intermezzo: Cluster analysis, hierarchical clustering VIII

$x_1,..,x_n$ p-dimensional data points (n objects with p characteristics), d is a dissimilarity measure

1. **For v=n start with finest partition**
2. **Calculate a new partition by joining two clusters That minimize d**
3. **Update the distances of the remaining clusters and the joined cluster**
4. **Stop if v=1, I.e., all data points are in one cluster, otherwise repeat steps 1-3**

Intermezzo: Cluster analysis, hierarchical clustering IX

Several cluster dissimilarity measures are in use:

1. Single linkage:

$$d(C_k^{(v)}, C_l^{(v)}) = \min x_i \in C_k^{(v)}, x_j \in C_l^{(v)} d'(x_i, x_j)$$

2. Complete linkage:

$$d(C_k^{(v)}, C_l^{(v)}) = \max x_i \in C_k^{(v)}, x_j \in C_l^{(v)} d'(x_i, x_j)$$

3. Average linkage:

$$d(C_k^{(v)}, C_l^{(v)}) = \frac{1}{\left|C_k^{(v)}\right\| C_l^{(v)}\right|} \sum_{x_i \in C_k^{(v)} x_j \in C_l^{(v)}} d'(x_i, x_j)$$

Intermezzo: Cluster analysis, hierarchical clustering X

- Update procedure: computing distance between a new (merged) cluster and all other clusters:

$$d(C_m^{(v-1)}, C_k^{(v)} \cup C_l^{(v)}) = \alpha_k d(C_m^{(v)}, C_k^{(v)}) + \alpha_l d(C_m^{(v)}, C_l^{(v)}) + \beta d(C_k^{(v)}, C_l^{(v)}) + \gamma \left| d(C_m^{(v)}, C_l^{(v)}) - d(C_m^{(v)}, C_k^{(v)}) \right|$$

Intermezzo: Cluster analysis, hierarchical clustering XI

- Update procedure parameters:

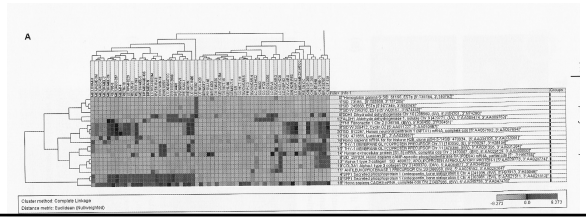
Method	α_i $(i = k, l)$	β	γ
Single linkage	0.5	0	–0.5
Complete linkage	0.5	0	0.5
Average linkage	$\frac{\lvert C_i^{(v)}\rvert}{\lvert C_k^{(v)}\rvert + \lvert C_i^{(v)}\rvert}$	0	0

Intermezzo: Cluster analysis XII, problem 1 many different choices

- Select a similarity measure
- Select a data transformation
- Select a clustering method: two classes the hierarchical and the partitioning methods
- Hierarchical method: single, complete, average linkage

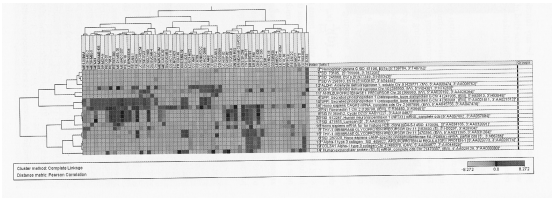
Intermezzo: Cluster analysis, hierarchical clustering XIII

- Problem 2:different similarity measures produce different dendrograms! Example a Euclidean distance



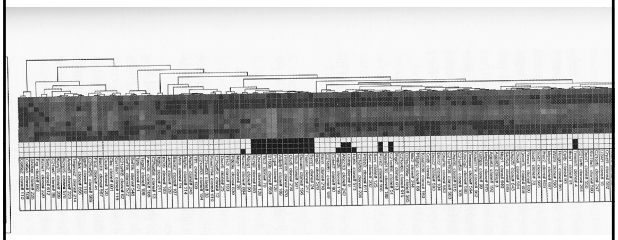
Intermezzo: Cluster analysis, hierarchical clustering XIV

- Problem 2:different similarity measures produce different dendrograms! Example b Pearson correlation



Intermezzo: Cluster analysis, hierarchical clustering XV

- Problem 3: false joining of data points



End Intermezzo: Cluster analysis

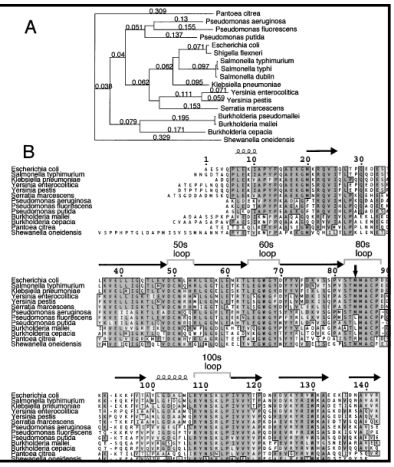
Multiple sequence alignment

- Merge results of pair wise alignments (disadvantage: not most optimal alignment, gaps are likely to be inserted at different locations)
- Tree-based progressive alignment. First a tree is constructed using a hierarchical clustering of set of sequences.

Multiple sequence alignment

- Tree-based progressive alignment. First a tree is constructed using a hierarchical clustering of set of sequences.

Dendrogram based on cluster analysis of a number of sequences (Eggers et al., Biochem. J. 379:107-118, 2004)



Multiple sequence alignment, other methods

- Multiple sequence alignment using simulated annealing (Kim et al., Bioinformatics 10:419-426, 1994)
- Multiple sequence alignment using genetic algorithms (Notredame et al., Nucleic Acids Research 24:1515-1524, 1996)

Intermezzo: Simulated annealing

Intermezzo: Simulated annealing I, Metropolis algorithm

- Using the metropolis algorithm one can generate a homogeneous Markov chain $\{X_n\}$ that steps through a state space, such that the points X_n are distributed according a required probability function $p(x)$

Intermezzo: Simulated annealing II, Metropolis algorithm

- Random walk with a transition probability Γ_{ij} being the probability to get from ``state'' X_i to X_j such that the distribution of the points (or states) converges to $p(x)$
- Detailed balance condition:
  $p(X_i)\Gamma_{ij} = p(X_j)\Gamma_{ji}$
- For example:
  $\Gamma_{ij} = \min\left[1, p(X_j)/p(X_i)\right]$

Intermezzo: Simulated annealing III, Metropolis algorithm

Algorithm 1.6 Metropolis algorithm.

choose trial position $X_t = X_i + \delta_i$ with δ_i random over $[-\delta, \delta]$
calculate $\Gamma_{ij} = \min[1, p(X_{i+1})/p(X_i)]$
if $\Gamma_{ij} = 1$ **then**
  accept move and put $X_{i+1} = X_t$
end if
if $\Gamma_{ij} < 1$ **then**
  generate random number R (uniform on $[0,1]$)
  if $R \leq \Gamma_{ij}$ **then**
    accept move and put $X_{i+1} = X_t$
  else
    put $X_{i+1} = X_i$
  end if
end if

Intermezzo: Simulated annealing IV, temperature dependent Metropolis algorithm

- Temperature dependent transition probability

$$\Gamma_{ij}(T) = G_{ij}(T)\min\left[1, \frac{p(j,T)}{p(i,T)}\right]$$

- G_{ij} is probability of generating state j if system is in state i
- $\min\left[1, \frac{p(j)}{p(i)}\right]$ is probability of accepting new state j

Intermezzo: Simulated annealing V, temperature dependent Metropolis algorithm

- Probability in physics of finding a system in a certain state at a certain temperature T is given by the Boltzmann equation:

$$p(i,T) = c\exp(-\frac{E(i)}{k_B T})$$

- $E(i)$ is the energy of the system at state i, k_B is Boltzmann constant, c normalizing constant

Intermezzo: Simulated annealing VI, temperature dependent Metropolis algorithm

- Consider a system with 4 states, certain cost values (column 2), c=1

		Probability at		
state	cost	T=100	T=1	T=0.01
1	1	0.251	0.225	≈ 0
2	2	0.248	0.0826	≈ 0
3	0	0.253	0.610	≈ 1
4	2	0.248	0.0826	≈ 0

Intermezzo: Simulated annealing VII, temperature dependent Metropolis algorithm

- Temperature dependent transition probability

$$\Gamma_{ij}(T) = G_{ij}(T)\min\left[1, \exp\left(\frac{(E(j)-E(i))}{k_B T}\right)\right]$$

- G_{ij} is probability of generating state j if system is in state i

Intermezzo: Simulated annealing VII, temperature dependent Metropolis algorithm

Propose a new state j with probability G_{ij} (based on the current state i)
Calculate the energies of the state points i and j :
$E(i)$ and $E(j)$
Calculate $A_{ij} = min[1, \exp(-\frac{(E(j)-E(i))}{k_B T})]$
if A_{ij} is 1 then the proposed state is accepted.
else Take a uniformly distributed random number, R, $0 \leq R < 1$.
  If $A_{ij} \geq R$ then the proposed state is accepted as the new state
  If $A_{ij} < R$ the old situation is reused as the new state.

Intermezzo: Simulated annealing VIII

```
Put the system in a random state
Start with a high temperature
While the stop criterion is not met :
    Make a Markov chain with the Metropolis algorithm
    Lower the temperature
End of the loop
```

Intermezzo: Simulated annealing IX, cooling schedule

- 1. Constant cool rate c:

$$T' = cT$$

- 2. Dependent on $\sigma(T)$ the standard deviation in the value of the cost function of the current chain, δ controls how much the probability functions may differ between two chains:

$$T' = T\left[1 + \frac{\exp(1+\delta)T}{3\sigma(T)}\right]^{-1}$$

End Intermezzo: Simulated annealing

Multiple sequence alignment, other methods

- Multiple sequence alignment using simulated annealing (Kim et al., Bioinformatics 10:419-426, 1994)

Multiple sequence alignment using simulated annealing: generating new states, the swap operation

```
                    ++
                  *******
MKQIGGA--MGSLA-   MKQIGGA--MGSLA-
MKK---IGGATGALG   MKKIGGA---TGALG
```

(a) (b)

Multiple sequence alignment using simulated annealing: solution sets

- An alphabet is a finite set of characters and null `-'
- A sequence is a finite string of characters
- A pseudo-alignment of the sequence $a_1, a_2, .., a_n$ is a set of pseudo-sequences $a'_1, a'_2, .., a'_n$ and the removal of the nulls from a'_i generates a_i
- A null column is a column whose elements are nulls
- A character column is a column such that at least one the elements is a character
- An alignment is a pseudo-alignment whose columns are only character columns

Multiple sequence alignment using simulated annealing:length of the pseudo-alignment

- Let l_{a_i} be a length of a sequence a_i The range of the length l of an alignment of the sequences $a_1, a_2, .., a_n$ can be calculated by the definition of an alignment. The range l is:

$$l_{\min} = \max(l_{a_i}) \le l \le l_{\max} = \sum_{i=1}^{n} l_{a_i}$$

Multiple sequence alignment using simulated annealing:cost (``energy'') function

- Let S_l be the set of all alignments with the same set of sequences and each alignment in the set has its own cost and the length of each alignment is l. Then the multiple sequence alignment problem is defined as finding the alignment with a smallest cost in the solution set:

$$S_{l_{\min}}^{l_{\max}} = \bigcup_{l_{\min} \le l \le l_{\max}} S_l$$

Multiple sequence alignment using simulated annealing:cost (``energy'') function

- Null columns do not affect the cost function
- Character columns do affect the cost function
- But you are not allowed to delete null columns during annealing process

Multiple sequence alignment using simulated annealing:``temperature''

- At final temperature the probability of accepting a new state with higher cost is:

$$e^{-\Delta E / T_f} = \varepsilon$$

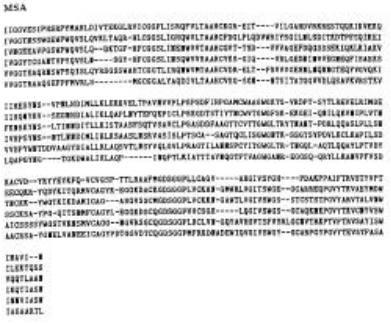
- Where $0 < \varepsilon < 1$
- Constant for reducing temperature $\gamma = (T_f / T_1)^{1/k}$

  k is total number of iteration steps in the annealing process

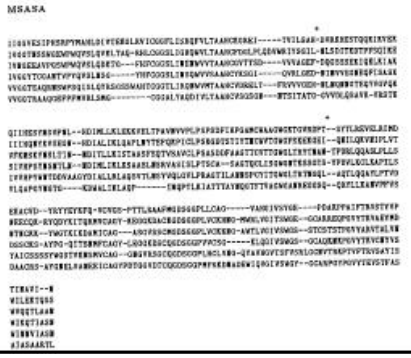
Multiple sequence alignment using simulated annealing:the algorithm

```
begin
  current_pseudo_alignment ← Output generated from the heuristic algorithm;
  T ← T_initial;
  E_current ← C(current_pseudo_alignment) where C is a cost function;
  final_pseudo_alignment ← current_pseudo_alignment;
  while (T > T_final)
    x ← random{column(1..l)};
    for each sequence i in the current_pseudo_alignment
      if column x is a null, then apply the swap operation;
      if column x is a character, then  do nothing;
    end for;
    E_new ← C(new_pseudo_alignment);
    if Metropolis conditions are satisfied then
      current_pseudo_alignment ← new_pseudo_alignment;
      E_current ← E_new;
      if (E_new < E_min) then
        E_min ← E_new;
        final_pseudo_alignment ← current_pseudo_alignment;
      end if;
    end if;
    T ← γ · T, 0 < γ < 1;
  end while;
  Remove null columns from the final_pseudo_alignment;
  Return final_alignment and minimal cost E_min as an optimal alignment;
end
```

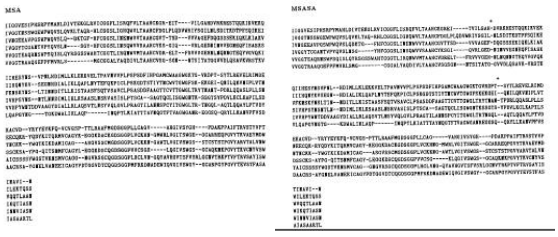
Multiple sequence alignment using dynamic programming:alignment of rat mast proteinase II, human plasma kaliikrin, bovine chymotrypsin, bovine trypsin, pig elastase and Streptomyces griseus trypsin



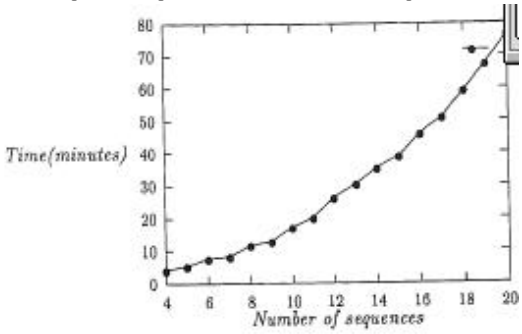
Multiple sequence alignment using simulated annealing:alignment of rat mast proteinase II, human plasma kaliikrin, bovine chymotrypsin, bovine trypsin, pig elastase and Streptomyces griseus trypsin



Multiple sequence alignment MSA vs MSASA:alignment of rat mast proteinase II, human plasma kaliikrin, bovine chymotrypsin, bovine trypsin, pig elastase and Streptomyces griseus trypsin



Multiple sequence alignment using simulated annealing: required computation time vs number of sequences



Multiple sequence alignment using simulated annealing: comparison computation time dynamic programming (MSA) and simulated annealing (MSASA)

Table 1. Comparison of computation time and score in MSA and MSASA

Sequences	MSA		MSASA		
	score	time	score	time	iteration (million)
4	21244	20 sec	21244	5 min 40 sec	1.3
5	35853	48 min 54 sec	35845	10 min 26 sec	2.0
6	54054	3 hour 18 min 37 sec	54050	16 min 40 sec	2.0

Multiple sequence alignment, other methods

- Multiple sequence alignment using simulated annealing (Kim et al., Bioinformatics 10:419-426, 1994)
- Multiple sequence alignment using genetic algorithms (Notredame et al., Nucleic Acids Research 24:1515-1524, 1996)

Intermezzo: genetic algorithms

Intermezzo: genetic algorithms I, general properties

- In GA solutions are coded in data structures (the ``chromosomes''), usually strings
- In the GA the algorithm starts with a population with random states
- For every individual the ``fitness'' (cost function) can be determined
- Higher fitness indicates a more optimal solution
- A higher fitness gives an individual a higher probability to reproduce and to be selected and to enter a next generation

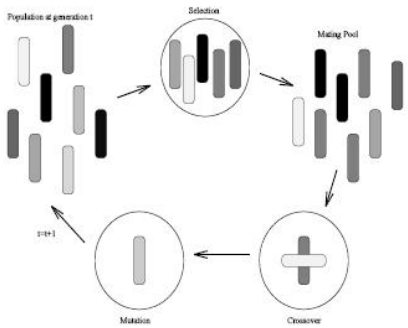
Intermezzo: genetic algorithms II, general properties

- Genetic algorithms form another class of stochastic solutions to optimization problems.
- Genetic algorithms more actively incorporate drastic changes as well as small modifications
- A drastic change is called a crossover and a small change a mutation, based on superficial resemblance to the actual genetic process in biology
- While simulated annealing has a sound basis in theory and is related to a physical phenomenon, genetic algorithms do have only a superficial resemblance to real biology

Intermezzo: genetic algorithms III, general properties

- Coding ``individuals'' (solutions) in bit strings
- How to define the ``objective function'' (cost function or energy function) and to determine the ``fitness'' of an individual

Intermezzo: genetic algorithms IV



Intermezzo: genetic algorithms V, basic operators

- Reproduction (fitness determination, selection method)
- Crossover (recombination operator)
- Mutation (recombination operator)

Intermezzo: genetic algorithms VI,

- A commonly used objective-to-fitness transformation:

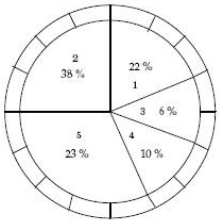
$$if\,(G(x) < C_{\max}) < F(x) = C_{\max} - G(x)$$
$$else F(x) = 0$$

- G(x) is the objective function, F(x) fitness function, $C_{\max}$ a largest value of G(x)

Intermezzo: genetic algorithms VII, 3 commonly used selection methods:

- Roulette wheel selection
- Stochastic remainder selection
- Tournament selection

Intermezzo: genetic algorithms VIII, roulette wheel selection



Arbitrary individuals and fitness values

Individual:	bitstring:	fitness:	selection prob.:
1	001010	119	0.22
2	011101	211	0.38
3	101101	34	0.06
4	000011	56	0.10
5	100001	128	0.23
Total		548	1.0

Intermezzo: genetic algorithms IX, stochastic remainder sampling

- Selection probability p multiplied by the population size N
- Individuals with $p > 1/N$ will have an expected number above 1 and an above average fitness
- Individuals below average fitness will have an expected number smaller than 1

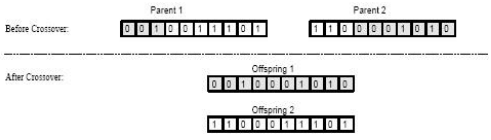
Intermezzo: genetic algorithms X, stochastic remainder sampling , construction mating pool

- 1. Population will be filled with an integer part number of copies of every individual
- 2. Remainder will be filled using the fractional part in a probabilistic process

Intermezzo: genetic algorithms XI, tournament selection

- K strings are picked at random. These individuals will compete to get into the mating pool. The winner of the tournament will be the individual with the highest fitness. The process is repeated until the pool is completely filled . The size of the tournament K varies between 2 and N (size of the population)

Intermezzo: genetic algorithms XII, crossover operator



A possible value for the crossover operator :

$0.6 \leq p_c \leq 0.9$

Intermezzo: genetic algorithms XII, mutation operator

A possible mutation value p_m is 1/l, where l is the length of the bit string

Intermezzo: genetic algorithms XIII, pseudocode

```
1. Initialize population with random solutions
2. Evaluate each individual, i.e.  calculate its
   fitness
3. Choose individuals, using some selection
   scheme, for the mating pool
4. Produce offspring for new population using
   crossover and mutation
5. Calculate convergence criterion
     If converged goto step 6
     Else goto step 2
6. Output best individual
```

Intermezzo: genetic algorithms IX, example (on blackboard)

Intermezzo: genetic algorithms X, theory

- Building block hypothesis, schemata theory (Holland et al.)
- Population of N strings, the number of building blocks (schemata) that are usefully processed is about N^3

End Intermezzo: genetic algorithms

Multiple sequence alignment, other methods

- Multiple sequence alignment using simulated annealing (Kim et al., Bioinformatics 10:419-426, 1994)
- Multiple sequence alignment using genetic algorithms (Notredame et al., Nucleic Acids Research 24:1515-1524, 1996)

Multiple sequence alignment using genetic algorithms, objective function (cost function)

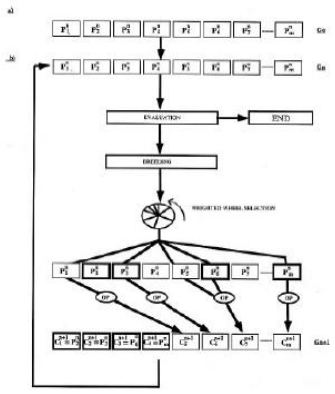
$$alignment\_\cos t = \sum_i \sum_j W_{i,j} \cos t(A_i, A_j)$$

Where cost is the alignment score between two aligned sequences A_i and A_j, $W_{i,j}$ is their weight. related to their similarity to other pairs. The alignment score is based on the cost of aligned residues (based for example on the PAM matrix) and the cost of gaps.

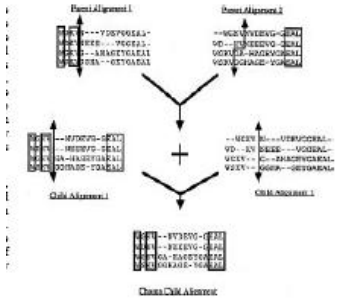
Multiple sequence alignment using genetic algorithms, pseudocode

Initialisation	1. create G_0
Evaluation	2. evaluate the population of generation n (G_n)
	3. if the population is stabilised then END
	4. select the individuals to replace
	5. evaluate the expected offspring (EO)
Breeding	6. select the parent(s) from G_n
	7. select the operator
	8. generate the new child
	9. keep or discard the new child in G_{n+1}
	10. goto 6 until all the children have been successfully put into G_{n+1}
	11. n = n+1
	12. goto EVALUATION
End	13. end

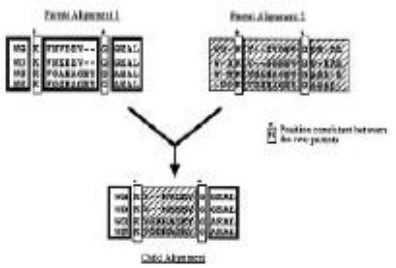
Multiple sequence alignment using genetic algorithms, layout of the algorithm



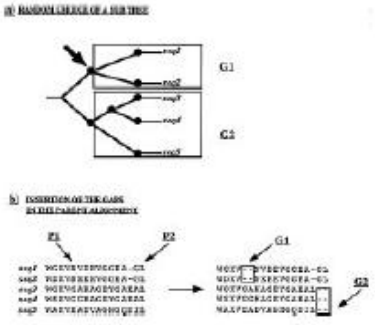
Multiple sequence alignment using genetic algorithms, one point crossover operator



Multiple sequence alignment using genetic algorithms, uniform crossover operator



Multiple sequence alignment using genetic algorithms, gap insertion



Multiple sequence alignment using genetic algorithms, structural alignment

- To asses the biological accuracy 13 test cases are selected: sequences with known three-dimensional structures, for which a structural alignment is available
- Protein structural alignment is a form of alignment which tries to establish equivalences between two or more protein structures based on their fold. In contrast to simple structural superposition, where at least some equivalent residues of the two structures are known, structural alignment requires no a priori knowledge of equivalent positions. (Wikipedia)

Multiple sequence alignment using genetic algorithms, dynamic programming vs GA

Test case	Nseq	Length	MSA score	MSA versus structure (%)	CPU-time	SAGA score	SAGA versus structure (%)	CPU-time
Cyt c	6	129	1 051 257	74.26	7	1 051 257	74.26	960
Ger	8	60	371 875	75.05	3	371 650	82.00	75
Ac protease	5	183	379 997	80.10	13	379 997	80.10	331
S protease	6	280	574 884	91.00	184	574 884	91.00	3500
Chtp	6	247	111 924	*	4525	111 579	*	3542
Dfr ecoli	4	189	171 979	82.03	5	171 975	82.50	411
Sbt	4	296	271 747	80.10	7	271 747	80.10	210
Globin	7	167	659 036	94.40	7	659 036	94.40	330
Plasto	5	132	236 343	54.03	22	236 195	54.03	510

Multiple sequence alignment using genetic algorithms, clustering vs GA

Test case	Nseq	Length	CLUSTAL W score	CLUSTAL W versus structure (%)	CPU-time	SAGA score	SAGA versus structure (%)	CPU-time
Igb	32	144	31 812 824	55.86	60	31 417 736	55.97	41 135
Ac Protease2	10	186	10 514 101	41.02	16	10 393 145	43.50	12 256
S Protease2	12	281	16 354 800	64.37	21	16 282 179	66.18	20 537
Globin2	12	171	5 249 682	94.90	18	5 233 058	94.01	2538

Multiple sequence alignment using genetic algorithms, example

Multiple sequence alignment, conclusion I

- Dynamic programming gives rigorous solutions for the multiple sequence alignment of a low (up to 8) number of sequences based on the primary structure of the protein
- Tree-based approach (hierarchical clustering) can be used for the multiple sequence alignment of a large number of sequences. Clustering can have certain problems (false fusion of branches), you only do a pair-wise comparison and not a multi-parameter optimization

Multiple sequence alignment, conclusion II

- Simulated annealing can be used for the multiple sequence alignment of a larger number of sequences. SA has a sound basis in theory but can computationally very expensive
- Genetic algorithms can be used for the multiple sequence alignment of a larger number of sequences, involves many heuristics, many parameters, there is no real sound basis in theory. The crossover operator may have some resemblance with the actual biology process

Computational Biology

Lectures

End part III Sequence analysis of nucleic acids and proteins