Practical Multiple Sequence Alignment

Tobias Rausch and Knut Reinert

Abstract Multiple sequence alignment as a means of comparing DNA, RNA, or amino acid sequences is an essential precondition for various analyses, including structure prediction, modeling binding sites, phylogeny, or function prediction. This range of applications implies a demand for versatile, flexible, and specialized methods to compute accurate alignments. This chapter summarizes the key algorithmic insights gained in the past years to facilitate an easy understanding of the current multiple sequence alignment literature and to enable the readers to use and apply current tools in their own research.

1 History of the Problem

The problem of comparing multiple sequences is a long-standing brainteaser of molecular biology. The research was sparked by a simple insight: Weak and faint biologically important sequence similarities vanish in a pairwise alignment but stand out in a multiple sequence alignment (MSA). For the last 20 years, the driving force behind MSA is the assumption that sequence similarity or sequence conservation implies structural, evolutionary, or functional correspondence. In other words, biologically important residues or nucleotides are assumed to be less likely to mutate than unimportant ones. In a MSA, we thus rewrite the sequences in such a way that conserved residues or nucleotides appear in the same column (see Table 1). MSA problems are characterized by (1) the number of sequences, (2) the length of the sequences, (3) the alphabet of the sequences (usually DNA, RNA, or amino acids), and (4) the relatedness of the sequences. Here, relatedness refers to both

Tobias Rausch
Freie Universität, e-mail: rausch@inf.fu-berlin.de

Knut Reinert
Freie Universität, e-mail: reinert@inf.fu-berlin.de

L.S. Heath and N. Ramakrishnan (eds.), *Problem Solving Handbook in Computational Biology and Bioinformatics*, DOI 10.1007/978-0-387-09760-2_2,

the divergence of the sequences and whether the sequences are globally or locally related. Many applications rely on accurate MSAs. The most prominent ones are phylogeny, functional predictions, domain identification, modeling binding sites, sequence consensus, and structure prediction (see Table 1 and Figure 1). With the benefit of hindsight, it is obvious that MSA algorithms and applications mutually fueled each other. Applications and progress in sequencing technologies created a continuous demand for new and more efficient alignment algorithms. In return, the progress in algorithms opened up unforeseen possibilities in terms of applications. For years, for instance, research has focused on sequence comparisons where the order of characters in the sequences is preserved. This colinearity condition was in fact the defining property of an alignment. In recent years, however, with an increasing number of genomic sequences at hand, sequence comparison involves the identification of the classical alignment operations, namely substitutions, deletions and insertions *and* more complex operations such as transpositions, translocations, duplications and inversions. This chapter is an attempt to capture the essential developments in the past years, starting from the first programs developed in the late 1980s to the first genome aligners of recent years.

HBA_HUMAN	.MVLS**PADKTNVKAAWGKVGAHAGEYGAEALERMFLSPTTKTY**FPHF.DLSH
HBB_HUMAN	MVHLT**PEEKSAVTALWGKV..NVDEVGGEALGRLLVVYPWTQRFFESF**GDLST
HBA_HORSE	.MVLS**AADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTY**FPHF.DLSH
HBB_HORSE	.VQLS**GEEKAAVLALWDKV..NEEEVGGEALGRLLVVYPWTQRFFDSF**GDLSN
MYG_PHYCA	.MVLS**EGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEK**FDRFKHLKT
LGB2_LUPLU	MGALT**ESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKDLF**SFLKGTSE
HBA_HUMAN	.....GS**AQVKGHGKKVADALTNAVAHVDD...M..PNALSALSDLHAH**KLRVD
HBB_HUMAN	**PDAVMGNPKVKAHGKKVLGAFSDGLAHLDN...L..KGTFATLSELHC**DKLHVD
HBA_HORSE	.....GS**AQVKAHGKKVGDALTLAVGHLDD..L..PGALSNLSDLHAH**KLRVD
HBB_HORSE	**PGAVMGNPKVKAHGKKVLHSFGEGVHHLDN..L..KGTFAALSELHC**DKLHVD
MYG_PHYCA	**EAEMK**ASEDLKKHGVTVLTALGAILKKKGH...H..**EAELKPLAQSHAT**KHKIP
LGB2_LUPLU	VPQ..NN**PELQAHAGKVFKLVYEAAIQLQVT**GVVVTD**ATLKNLGSVHVS**K.GVA
HBA_HUMAN	P**VNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR**.....
HBB_HUMAN	P**ENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH**.....
HBA_HORSE	**PVNFKLLSHCLLSTLAVHLPNDFTPAVHASLDKFLSSVSTVLTSKYR**.....
HBB_HORSE	P**ENFRLLGNVLVVVLARHFGKDFTPELQASYQKVVAGVANALAHKYH**.....
MYG_PHYCA	**IKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG**
LGB2_LUPLU	**DAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDA**...

Table 1 MSA of 6 globin sequences: Human hemoglobin subunit alpha (UniProt accession: P69905), human hemoglobin subunit beta (P68871), horse hemoglobin subunit alpha (P01958), horse hemoglobin subunit beta (P02062), sperm whale myoglobin (P02185) and European yellow lupin leghemoglobin-2 (P02240). The helix secondary structure annotation from UniProt is shown in bold font.

Throughout the history of MSA, one can distinguish two types of algorithms, optimal ones and heuristics. The former algorithms compute an optimal alignment with respect to some scoring function such as the sum of pairs score. The latter algorithms compute an alignment based on some kind of biologically sound procedure

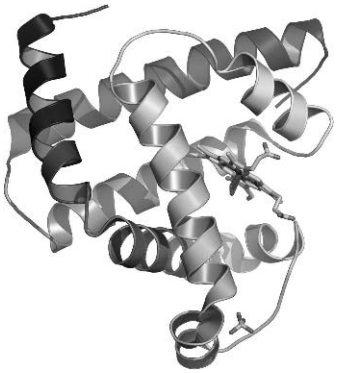

Fig. 1 A 3D model showing the helical domains of myoglobin.

such as progressive alignment. Both classes of algorithms are reviewed in Section 2. The first optimal methods could align three sequences simultaneously using standard dynamic programming [32, 56]. A few years later, the program MSA [35, 52] could align up to eight sequences of average protein length by using a clever bounding technique for the dynamic programming lattice. Time and space was further reduced using the A* algorithm [51, 72] and (partly heuristic) divide and conquer techniques [73]. Besides bounding techniques for the dynamic programming formulation, other algorithms used an alignment graph or trace graph [43, 77]. This alignment graph was used in an integer linear programming (ILP) formulation [71] extended by various branch-and-cut techniques [2, 3, 4].

Computing an optimal alignment for an arbitrary number of sequences is, however, NP-complete using the sum of pairs score [93]. That is why a vast number of heuristics has been developed enabling the alignment of more sequences of greater length. Heuristic methods were difficult to compare in the beginning but gained enormous leverage with the advent of protein benchmark data sets of sometimes manually refined MSAs such as BAliBASE [87, 89], PREFAB [24], OXBENCH [67], SABmark [92] and IRMBASE [85]. For protein alignments, these benchmarks are the de facto standard for judging the performance of individual methods. The first heuristic *progressive* aligner was published in 1987 [27] followed by a great variety of other heuristics, most prominently the Clustal series of programs [38, 47, 88]. More recently, the progressive alignment paradigm has been extended using approaches outlined in the next section, such as consistency [20, 59], refinement [24, 41], and segmentation [70, 85]. The increasing number of genomic sequences stimulated the development of genome aligners or genome comparison tools in the past 10 years. The MUMmer series of programs [18, 19, 46] remarkably pioneered this research area, but lately a number of other interesting anchor-based alignment tools appeared [11, 16].

2 Algorithm Description

Over the years, numerous research projects have contributed to steady progress in the area of MSA. Despite such a long history, methods are still far away from being optimal in a biological sense. The main obstacles are (1) that we still lack a precise mathematical formulation of such a biologically optimal alignment and (2) that the problem is already NP-hard if we use a very simplified formulation such as alignment score maximization. This very question of finding an alignment of maximum score has driven the field significantly in the past years and many sequence based methods, both heuristics and optimal ones, have been developed to solve this problem. The nuts and bolts of these methods are described in depth in Section 2.1. Recently, the sequence based methods have been complemented by methods that go beyond the raw sequence data. These structure based methods use a great variety of structure prediction methods and databases with structural information. The goal is either to substantiate a possibly weak signal of sequence similarity or to identify novel domains where conservation only manifests itself on a structural level. The basics of these methods are investigated in Section 2.2. The predominant representation of an alignment is the well-known alignment matrix. An example is shown on the right in Figure 2. Based upon that matrix, we can formally define the properties of a multiple alignment of a set $\mathrm{S} = \{S^0, S^1, \ldots, S^{n-1}\}$ of n sequences.

- $S^i \in \mathrm{S}$ is a string over the finite ordered alphabet Σ that is $S^i \in \Sigma^*$. Σ is, for instance, the DNA or amino acid alphabet. Each string S^i is a sequence of letters $s^i_0 s^i_1 \ldots s^i_{|S^i-1|}$ of length $|S^i|$ where $s^i_u \in \Sigma$.
- The alphabet $\tilde{\Sigma} = \Sigma \cup \{-\}$ is the extended alphabet including a gap character '$-$'.

A multiple alignment A of the strings in S is an $n \times l$ matrix consisting of n strings $\tilde{S}^0, \tilde{S}^1, \ldots, \tilde{S}^{n-1} \in \tilde{\Sigma}^*$ such that

- The strings $\tilde{S}^0, \tilde{S}^1, \ldots, \tilde{S}^{n-1}$ are of length l.
- The string $\tilde{S}^i$ with gaps removed is equal to S^i.
- The matrix entry a^i_u in row i and column u is either from the alphabet Σ or equal to the gap character '$-$': $\quad a^i_u \in \tilde{\Sigma} \quad \forall\, 0 \leq i < n,\ 0 \leq u < l$
- No column consists entirely of gap characters. This implies:

$$\max_{i=0,\ldots,n-1} |S^i| \leq l \leq \sum_{i=0,\ldots,n-1} |S^i|$$

Alternatively, one can think of an alignment as a path through an n-dimensional hypercube as shown in Figure 2.

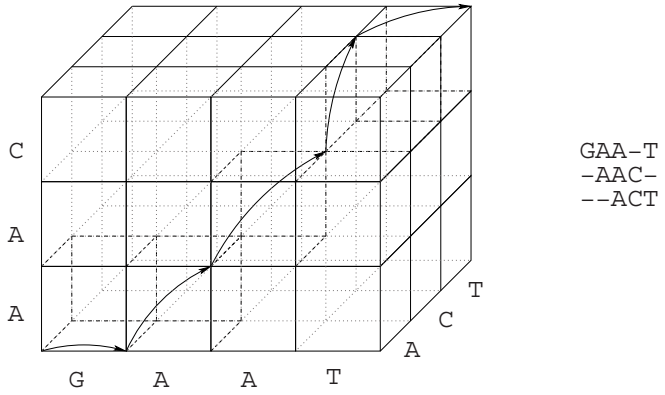

Fig. 2 The MSA path in a 3-dimensional lattice corresponding to the alignment shown on the right.

2.1 Sequence based methods

Given a number of different multiple alignments for a set of sequences, we somehow need a quantitative measure to decide which one is the best. For sequence based methods the ubiquitously used measure is the sum of pairs multiple alignment score, which is an extension of the pairwise alignment score to more than 2 sequences. The score of a multiple alignment is simply the sum over all alignment scores of each and every possible induced pairwise alignment. This notion can be formalized using (1) the projection A^I of a multiple alignment onto a set of sequences and (2) the definition of pairwise alignment scores. A pairwise alignment projection is a mere selection of 2 distinct rows in a given alignment and a subsequent removal of all columns containing only gaps. For example, the projection $A^{\{0,1\}}$ of the alignment in Figure 2 results in the pairwise alignment:

$$\begin{array}{cccccc} G & A & A & - & T \\ - & A & A & C & - \end{array}$$

A projection A^I for an index set $I \subset \{0,\ldots,n-1\}$ is obtained from A by

1. Selecting row i in A if and only if $i \in I$.
2. Deleting column u in A^I if and only if column u contains only gap characters.

This formulation respects the requirement that $A^{\{i\}} = S^i$. Similarly, one can project the path through the n-dimensional space onto a subspace as shown in Figure 3. The most common pairwise scoring function uses linear gap costs. Linear gap costs penalize a gap of length γ with a cost of $g + e \cdot (\gamma - 1)$ where g is the constant gap opening penalty, e is the constant gap extension penalty and $g \leq e$ with $g, e \leq 0$. If $g = e$ the number of gap openings is irrelevant and such gap costs are called constant hereafter. Using linear gap costs, the score of a pairwise alignment is

$$Score(\mathrm{A}^{\{i,j\}}) = \Big(\sum_{\substack{u=0,\ldots,\tilde{l}-1 \\ a_u^i \neq -;\ a_u^j \neq -}} \delta(a_u^i, a_u^j) \Big) + g \cdot \#\text{GapOpen} + e \cdot \#\text{GapExtension}$$

where $\tilde{l}$ is the length of the projected alignment and δ a scoring function or substitution matrix for all pairs of characters $a_u^i, a_u^j \in \Sigma$. The BLOSUM [37] and PAM [17] matrices are commonly used substitution matrices for protein alignments. For DNA alignments, most tools use a simple match / mismatch scoring function. The alignment

G A T A T A − − T
− A T G T A C C −

evaluated with linear gap costs (gap opening penalty $g = -4$, gap extension penalty $e = -1$) and a scoring function defined by a match score of $\delta(x,x) = 4$ and a mismatch score of $\delta(x,y) = -2$ results in a total score of $14 + (-4) \cdot 3 + (-1) \cdot 1 = 1$. Finally, the sum of pairs multiple alignment score SP_{Score} can be simply defined as

$$SP_{Score}(\mathrm{A}) = \sum_{0 \leq i < j < n} Score(\mathrm{A}^{\{i,j\}})$$

For the sake of completeness, one should note that, besides the sum of pairs score, other quantitative alignment quality measures are available, most notably the weighted sum of pairs score, the tree alignment score, and the consensus score [34]. In this chapter, we focus on the well-established sum of pairs score. For the heuristic algorithms, all of these measures became less important with the publication of reference MSA benchmarks such as BAliBASE [87] but research devoted to exact algorithms relies on such quantitative measures.

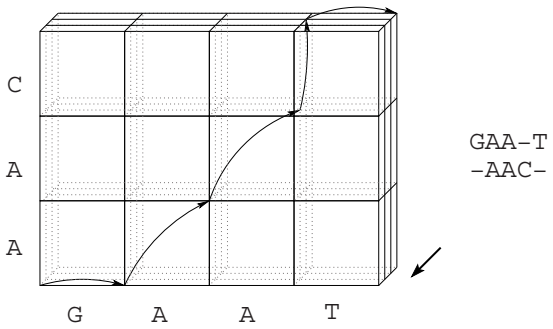

Fig. 3 A projection of a 3-dimensional lattice to a 2-dimensional matrix corresponding to the projection of an alignment of 3 sequences onto a subset of 2 sequences.

2.1.1 Exact algorithms

The multiple alignment score maximization problem can be solved optimally using either natural extensions of the dynamic programming algorithm [35, 51, 52, 72, 73] or exact algorithms based on graph-theoretic models [2, 3, 4, 71].

Dynamic programming

The dynamic programming recursion to compute the optimal pairwise alignment between sequence $S^0 = s_0^0 s_1^0 \ldots s_{|S^0-1|}^0$ and sequence $S^1 = s_0^1 s_1^1 \ldots s_{|S^1-1|}^1$ is

$$M_{u,v} = \max \begin{cases} M_{u-1,v-1} & + \delta(s_u^0, s_v^1) \\ M_{u-1,v} & + \delta(s_u^0, -) \\ M_{u,v-1} & + \delta(-, s_v^1) \end{cases}$$

where $M_{u,v}$ is the 2-dimensional dynamic programming matrix and δ is the scoring function. For a constant gap penalty $g = e$ and the Blosum62 substitution matrix one could define δ as $\delta(s_u^0, s_v^1) = \text{Blosum}_{62}(s_u^0, s_v^1)$ and $\delta(s_u^0, -) = \delta(-, s_v^1) = e$. The extension to 3 sequences involves two changes. First, a 3-dimensional dynamic

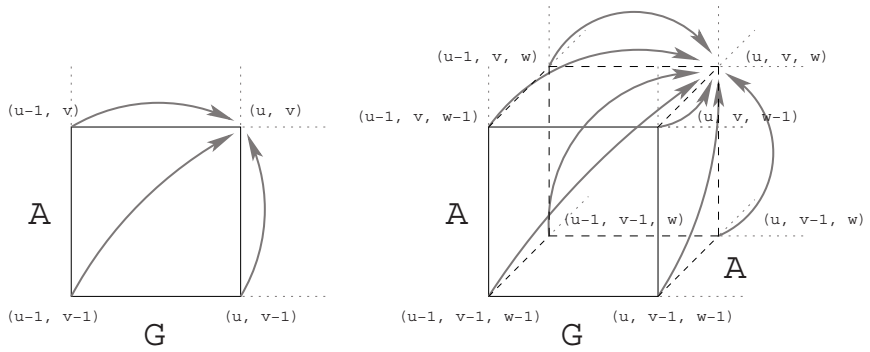

Fig. 4 In each cell of the dynamic programming matrix / cube $(2^n - 1)$ predecessor have to be evaluated where n is the number of sequences.

programming hypercube has to be computed and second, for each entry we have to evaluate $(2^n - 1) = (2^3 - 1) = 7$ predecessors as shown in Figure 4. The recursion is

$$M_{u,v,w} = \max \begin{cases} M_{u-1,v-1,w-1} & + \tilde{\delta}(s_u^0, s_v^1, s_w^2) \\ M_{u,v-1,w-1} & + \tilde{\delta}(-, s_v^1, s_w^2) \\ M_{u-1,v,w-1} & + \tilde{\delta}(s_u^0, -, s_w^2) \\ M_{u-1,v-1,w} & + \tilde{\delta}(s_u^0, s_v^1, -) \\ M_{u,v,w-1} & + \tilde{\delta}(-, -, s_w^2) \\ M_{u-1,v,w} & + \tilde{\delta}(s_u^0, -, -) \\ M_{u,v-1,w} & + \tilde{\delta}(-, s_v^1, -) \end{cases}$$

For the sum of pairs score with constant gap costs, $\tilde{\delta}$ can be defined in terms of δ as $\tilde{\delta}(a,b,c) = \delta(a,b) + \delta(b,c) + \delta(a,c)$ with $a,b,c \in \tilde{\Sigma}$ and $\delta(-,-) = 0$. This can be extended to higher dimensions d. As in the pairwise case, the key idea is that larger alignments are constructed from already computed subsolutions. Any $M_{u,v,\ldots,z}$ is the best score of aligning the prefixes $s_0^0 s_1^0 \ldots s_u^0$, $s_0^1 s_1^1 \ldots s_v^1$, …, $s_0^{n-1} s_1^{n-1} \ldots s_z^{n-1}$. In addition, the optimal alignment can be retrieved through the standard traceback operations extended to the d-dimensional hypercube. Note that it is also possible to apply Gotoh's algorithm [31] for linear gap costs to more than two sequences. Similar to the pairwise case, we then require additional hypercubes for the best gapped alignment in each dimension. The size of the hypercube is exponential in the number of sequences $O(\prod_{i=0}^{n-1} |S^i|)$. For each cell of this hypercube, $(2^n - 1)$ predecessor cells have to be evaluated. Thus, the time complexity is $O((2^n - 1) \cdot \prod_{i=0}^{n-1} |S^i|)$ if and only if the computation of the δ function is constant $O(1)$. This is roughly $O((2\tilde{n})^n)$ where $\tilde{n}$ is the average sequence length. Bounding techniques try to minimize the actually computed hypercube alignment space by using lower and upper bounds [35, 51, 52, 72] or a combination of an exact algorithm with a heuristic divide and conquer approach [73].

Graph based models

An alignment can be visualized as an alignment graph of sequence segments as shown in Figure 5. The alignment edges of this graph represent matches or possible mismatches. Gaps are implicitly represented by the topology of the graph. For instance, a vertex without any outgoing edge is aligned to gaps in all other sequences. An alignment graph can be easily converted into an alignment matrix using standard graph algorithms, namely connected components and topological sort [14]. If we allow arbitrary alignment edges as shown in Figure 5 an actual alignment can only realize a subset of the given edges. This subset is called a trace [77]. The graph can be extended by edge weights that capture some kind of quantitative measure of alignment quality. A possible measure is, for instance, the pairwise BLOSUM score of two aligned segments. Given such a problem, the trace problem can be rephrased as a maximum weight trace problem and also naturally extended to multiple sequences [42, 43]. The graph formulation translates easily into an integer linear program, giving rise to the possibility of applying techniques from combinatorial optimization such as branch-and-cut [3, 4]. The alignment graph can be extended with gap arcs to incorporate positional gap penalties [71]. Recently, a Lagrangian

approach was proposed to solve the integer linear programming formulation more efficiently [2].

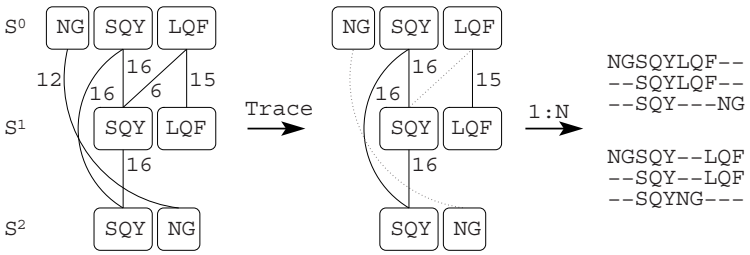

Fig. 5 An alignment graph with arbitrary alignment edges (left), its heaviest trace (middle) and its conversion to an alignment matrix (right). The graph does not impose an order on adjacent indels and hence, there is a 1:N relation between alignment graphs and alignment matrices.

2.1.2 Heuristic algorithms

In practice, optimal methods are only practical for a few, relatively short sequences. Hence, the development of fast and accurate heuristics for MSA problems is a very active research field. In this chapter, we can only review the most important heuristics, which is first and foremost the progressive alignment strategy [27].

Progressive alignment

A sound multiple alignment of n sequences should induce $(\frac{n\cdot(n-1)}{2})$ projected pairwise alignments that are as close as possible to optimal pairwise alignments. Unfortunately, pairwise alignments may be incompatible as shown in Figure 6. Progressive alignment resolves these inconsistencies in a greedy manner. The multi-

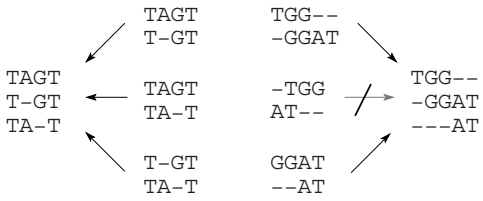

Fig. 6 A set of pairwise alignments that are compatible (left) or incompatible (right).

ple alignment is started from the most similar pair and then gradually, the other less similar sequences are added to the growing alignment. The intuitive assumption is

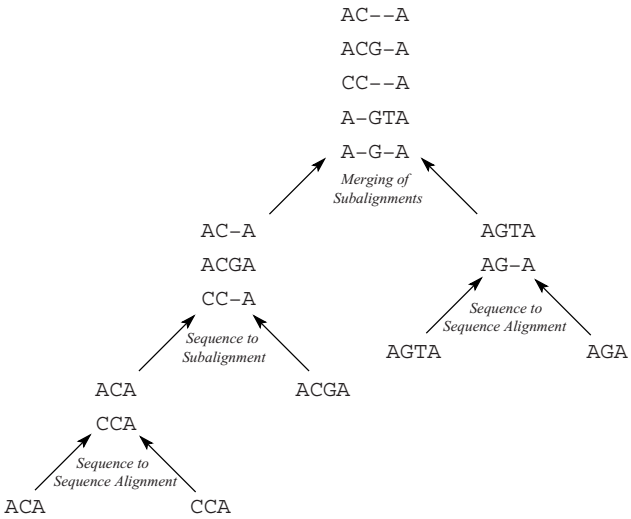

Fig. 7 The progressive alignment greedily builds a final alignment along the guide tree using a given method to merge subalignments.

that a pairwise alignment of closely related sequences is more to be trusted than an alignment of distantly related sequences [27]. The method thus requires 2 things. First, a binary tree, called a guide tree, that indicates when every sequence (a leaf of the tree) is merged into a growing multiple alignment and second, a means of aligning already finished subalignments with another sequence or another subalignment. The later situation arises if the progressive alignment is started from multiple seeding alignments as shown in Figure 7.

The guide tree can be obtained in 2 steps. First, a distance score between all pairs of sequences is computed, and, second, the phylogenetic tree is reconstructed [28] using clustering methods such as UPGMA [82] or neighbor-joining [76]. Several distance measures for two sequences are based upon simple similarity scores. Examples are the percent identity between two sequences or the fractional number of common k-mers where a k-mer is a contiguous substring of length k. For large alphabets, the percent identity and the number of common k-mers are less applicable, unless the sequences are closely related or both measures are applied over a compressed alphabet [23]. More precise measures are based upon pairwise global or local alignment scores [58, 81], which are usually normalized by alignment length. UPGMA is, besides neighbor-joining, a widely used distance based tree reconstruction method. The algorithm requires a set of n elements (e.g. sequences) and all pairwise distances $d_{i,j}$. Initially, each element is in its own group, and, thus, the sequences are the leaves of the tree. The algorithm proceeds in 4 steps:

1. Select the minimum distance $d_{i,j}$.
2. Create a new group u that joins i and j.
3. Compute the distances $d_{k,u}$ of any group k to the new group u.

4. Remove i, j from the set of elements and go to 1.

The UPGMA algorithm reconstructs the correct tree only for ultrametric distances. Such distances imply that all sequences have evolved from a common ancestor at a constant rate. This assumption is, in general, not true, and, thus, UPGMA is not used very often in phylogenetic studies. It is, however, widely used in progressive alignment tools because some authors argue [24] that a reliable evolutionary tree is not as important as a tree that guarantees that the subalignments with the fewest differences are merged first. In Step (3) the new distance $d_{k,u}$, from any group k to the new group u that joined i and j, can be computed using different methods:

1. Single linkage clustering: $d_{k,u} = \min(d_{k,i}, d_{k,j})$
2. Complete linkage clustering: $d_{k,u} = \max(d_{k,i}, d_{k,j})$
3. Average linkage clustering: $d_{k,u} = \frac{d_{k,i}+d_{k,j}}{2}$
4. Weighted average linkage clustering: $d_{k,u} = \frac{n_i \cdot d_{k,i}+n_j \cdot d_{k,j}}{n_i+n_j}$

In the last method n_i and n_j are the number of elements in group i and j, respectively. The neighbor joining method is a different method to reconstruct a tree. It has a higher time complexity of $O(n^3)$, where n is the number of sequences, compared to $O(n^2)$ for the UPGMA algorithm. The guide tree obtained with that method is, however, regarded as a better evolutionary tree because the neighbor joining method does not assume a molecular clock. The idea of the method is to start with a star tree and then to gradually group pairs of sequences so that the overall tree length is minimized.

Aligning the children of an internal node in the guide tree either involves an ordinary sequence alignment or an alignment of subalignments. In the latter case, one possible objective is to optimize the already mentioned sum of pairs multiple alignment score.

$$SP_{Score}(\mathrm{A}) = \sum_{0 \le i < j < n} Score(\mathrm{A}^{\{i,j\}})$$

Using linear gap costs, an optimal merging of subalignments is NP-complete [44, 53]. However, in Kececioglu and Starrett [44], the authors propose an algorithm that is exact and quite fast in practice. Other methods favor speed over optimality and use approximations of gap opening counts [45]. More often, however, practical tools use their own way of merging subalignments with quite distinct objective functions [26]. These methods are usually subsumed under the generic term profile-profile alignments. A profile of a multiple alignment A of length l is a $|\tilde{\Sigma}| \times l$ matrix P, where $\mathrm{P}_{a,u}$ is the frequency of character $a \in \tilde{\Sigma}$ in column u of A.

A	G	C	T
A	G	C	C
A	–	C	A
C	G	C	A

P	1	2	3	4
A	0.75	0	0	0.5
C	0.25	0	1.0	0.25
G	0	0.75	0	0
T	0	0	0	0.25
–	0	0.25	0	0

Assuming constant gap costs, a string $S = s_0 s_1 \cdots s_{|S|-1}$ can be quickly aligned to a profile with a standard pairwise dynamic programming algorithm. Only the scoring

function δ has to be adapted.

$$\delta_{\text{New}}(s_w, u) = \sum_{a \in \tilde{\Sigma}} \mathrm{P}_{a,u} \cdot \delta(s_w, a)$$

In this case, δ_{New} scores a column u against a character $s_w \in \Sigma$. The δ function has to be extended to handle the special case of scoring a gap character against another gap character.

$$\delta(a,b) = \begin{cases} \text{Blosum}_{62}(a,b) & \text{if and only if } a,b \in \Sigma \\ e & \text{if and only if } a = \text{"}-\text{"} \text{ or } b = \text{"}-\text{"} \\ 0 & \text{if and only if } a = b = \text{"}-\text{"} \end{cases}$$

Note that in a projected alignment gap columns are removed and hence, the score for two aligned gaps is set to 0. For instance, a final string to profile alignment of the string $S = ACCA$ can be scored as shown below, assuming $\delta(x,x) = 4, \delta(x,y) = -3, \delta(x,-) = \delta(-,x) = -2$ and $\delta(-,-) = 0$.

A	G	–	C	T
A	G	–	C	C
A	–	–	C	A
C	G	–	C	A
A	–	C	C	A

P	1	2	–	3	4
A	0.75	0		0	0.5
C	0.25	0		1.0	0.25
G	0	0.75		0	0
T	0	0		0	0.25
–	0	0.25	1.0	0	0

S	A	–	C	C	A
δ_{New}	2.25	−1.5	−2	4	0.5

Hence, the score of the full string to the profile alignment is 3.25. Note that non-linear or constant gap penalties simplify the sum of pairs score of a multiple alignment A of length l to

$$SP_{Score}(\mathrm{A}) = \sum_{0 \le i < j < n} Score(\mathrm{A}^{\{i,j\}}) = \sum_{i,j} \sum_{u=0}^{l-1} \delta(\tilde{s}_u^i, \tilde{s}_u^j) = \sum_{u=0}^{l-1} \sum_{i,j} \delta(\tilde{s}_u^i, \tilde{s}_u^j)$$

The last equality stems from the independence of the alignment columns using the δ scoring function with constant gap penalties. Using dynamic programming, the optimal string to profile alignment can be found in quadratic time $O(|\tilde{\Sigma}| \cdot l \cdot |S|)$ where l is the length of the profile, $|S|$ the length of the sequence and $|\tilde{\Sigma}|$ a small constant, e.g., 5 for the DNA alphabet or 21 for the amino acid alphabet. Similarly, a profile-profile alignment can be carried out. The only difference is an extra sum over the alphabet $\tilde{\Sigma}$.

$$\delta_{\text{New}}(u,w) = \sum_{a \in \tilde{\Sigma}} \sum_{b \in \tilde{\Sigma}} \mathrm{P}_{a,u} \cdot \mathrm{P}_{b,w} \cdot \delta(a,b)$$

Numerous other profile-profile column scoring functions have been published [24, 26, 41].

In summary, an optimal merging of subalignments with linear gap costs $g + e \cdot (\gamma - 1)$ is NP-complete. A merging with $g = e$ remains polynomial because gap opening counts are irrelevant. In this case, each column can be treated as a meta-character in an extended alphabet. Given a scoring function for such meta-characters, the problem is to find an alignment of two strings of meta-characters, which is clearly solvable with a pairwise dynamic programming algorithm.

Consistency and refinement

The choice of the binary tree and the method to merge subalignments has great influence on the final alignment. Once a new sequence is added to the growing alignment all the aligned characters and inserted gaps are fixed ("Once a gap, always a gap." [27]). But this is also true for alignment errors: once made they are preserved and they may even cause new alignment errors in the subsequent progressive steps. There are two strategies, "consistency" and "refinement", to handle alignment errors; one aims to prevent errors and the other one aims to correct errors [94]. The prevention approach tries to substantiate pairwise alignments by multiple sequence information. That is, it tries to make pairwise alignments consistent with all the other sequences, and, hence, the name consistency [33, 59]. The refinement approach takes a possibly erroneous alignment, iteratively splits this alignment into two subalignments and merges these alignments together again. These methods, thus, iteratively "refine" or "realign" a given alignment. In other publications authors sometimes use the term "Iterative Alignment" to describe such techniques [66].

Although current algorithms use slightly different means of consistency, the basic idea is always the same: the confidence of aligning substrings of a pair of sequences S^0 and S^1 is greater the more intermediate sequences S^i support this alignment. In other words, the alignments $S^0 \leftrightarrow S^i$ and $S^i \leftrightarrow S^1$ induce a putative transitive alignment $S^0 \leftrightarrow S^1$ that is either consistent or inconsistent with a precomputed alignment of S^0 and S^1. If it is consistent, greater confidence in the alignment of these substrings of S^0 and S^1 is established, and the scores are somehow increased. In an alignment graph, this consistency extension or triplet extension corresponds to a search for three-way cliques (see Figure 8).

The refinement approach [24, 41] splits a full alignment randomly or following a deterministic order into subalignments and then merges these subalignments using, for example, profile to profile alignment methods. Random cutting usually halts if no improvement in alignment score is observed during the last iterations.

Anchor-based alignment

Even the heuristic progressive alignment becomes prohibitively expensive when aligning *genomic* DNA sequences. In these cases, any approach involving a full pairwise dynamic programming is impossible. Nevertheless, genome alignments or genome comparisons are more important than ever before because of several

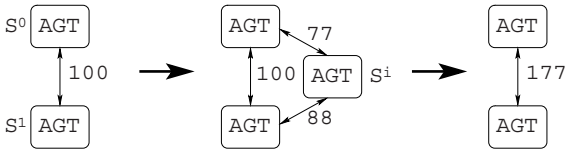

Fig. 8 A possible means of consistency extension: Every supported alignment is increased by the minimum of the two connecting edges.

vertebrate genomes at hand and thousands of on-going sequencing projects. The applications are numerous, ranging from the comparison of different assemblies, annotation tasks, regular elements identification, and phylogenetic studies to analyzing principal questions addressing mechanisms of genome evolution. Almost all genome aligners make use of the same strategy: anchor-based alignment or synonymously seeded alignment. Anchor-based alignment has three steps: (1) the computation of small segment matches of high similarity shared by multiple sequences, (2) the ordering of these segment matches into a collinear chain of non-overlapping segment matches (the fixed alignment anchors) and (3) closure of gaps between the anchors. The sole purpose of Step 1 and Step 2 is to abandon a large chunk of the possible alignment space as shown in Figure 9. Only small indels are allowed within the anchors and thus, full dynamic programming is only required between the anchors. Some programs also try to extend anchors first to the left and right to further reduce the search space. Note that Step 1 does not yet imply colinearity as shown in Figure 9. The initial segment matches can be, for example, maximal unique or exact matches [46], maximal multiple exact matches [39] or exact or hashed *k*-mers [12, 85]. Segment matches are optionally extended, and, finally, the quality of a segment match is assessed using some weight function. Chaining algorithms [1, 57] can be applied to compute the heaviest (best) collinear chain of these segment matches. The resulting list of anchors is refined by applying the above procedure iteratively (e.g. by using a smaller *k*-mer) or by filling the gaps between the anchors using more sensitive approaches such as pairwise dynamic programming. Since genomic rearrangements such as transposition, duplication, or inversion are rather likely, novel methods try to cover at least some of these operations, for example, by computing only local chains [16, 61].

Others

Another option for aligning large sequences is to consider gapless sequence segments instead of single characters. The segment matches can be derived from pairwise alignments, BLAST [5] matches or any other local comparison tool. The local alignments are usually scored and subdivided into gapless segment matches to simplify the subsequent alignment. The problem with these methods is, however, that segment matches might overlap and intersect each other. SeqAn::T-Coffee [70] refines the set of segment matches so that all parts of all segment matches can be

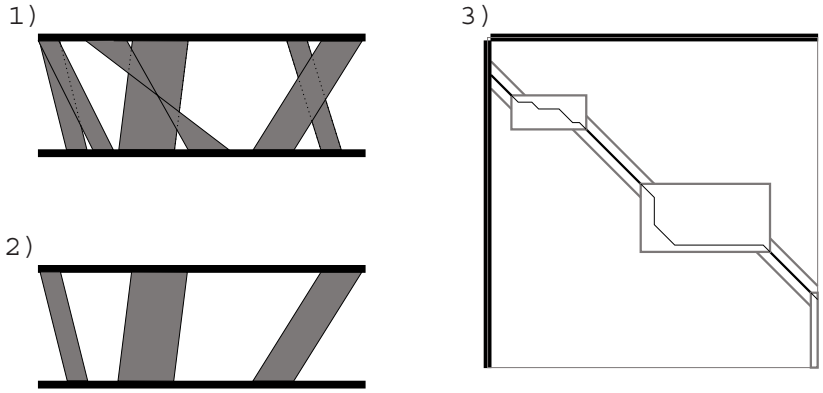

Fig. 9 Anchor-based alignment: (1) computation of initial segment matches, (2) collinear chaining of non-overlapping segment matches and (3) dynamic programming to close the alignment gaps.

used. In the subsequent progressive alignment, the program makes use of the alignment graph introduced in Section 2.1.1 and shown in Figure 5. The DIALIGN series of programs [55, 84, 85] takes a different approach by leaving the set of segment matches unchanged. This implies that overlapping segment matches involving the same pair of sequences must be greedily resolved. The objective function of DIALIGN is to find a consistent, maximum score subset of segment matches.

POA [50] uses partial order graphs to represent multiple sequence alignments. Each individual sequence is a trivial partial order graph where each character is a node connected to the subsequent node for the following character. The final partial order graph is obtained by successively aligning such a trivial graph to the growing partial order graph MSA. The key idea is that aligned nodes are merged to a new node whereas the graph bifurcates for unaligned regions. Thus in comparison to the segment based aligners, POA does not reduce the complexity depending on the length of the sequences but on the number of sequences.

2.2 Methods using structure and sequence homologs

The improvements in de-novo structure prediction methods and the growth of sequence and structural databases opened up new possibilities to extend sequence based alignment methods. These extended methods tend to deliver more accurate alignments on standard benchmarks, especially in the twilight zone of highly diverged sequences with less than 20% identity. Three combinable techniques are in common use: homology extension [40, 64, 80, 95], secondary structure prediction [63, 64, 79, 95], and the use of a known 3D structure [60, 65].

Homology extension augments the raw sequence information using database searches such as PSI-BLAST [6]. Given such a set of retrieved database homologs, a profile can be built for each input sequence. The profiles can then be readily used

in progressive alignment, as outlined in Section 2.1.2. The use of profiles turned out to be beneficial because profiles differentiate between conserved and variable sites.

Predicted or known secondary structures can further improve the alignment quality because, in most cases, structure is more conserved than sequence information. Structural elements can be predicted, for instance, with PSIPRED [54] and many other tools [75]. The pairwise sequence alignment is then carried out under structural constraints. For instance, one could add a simple secondary structure weight function to the profile to profile alignment that indicates if the two corresponding structural elements at a given position match or mismatch.

Similarly, a known 3D structure eases the alignment of highly diverged sequences. Methods such as SAP [86] employ a double dynamic programming algorithm to compute a structural alignment. The time complexity is, however, $O(\tilde{n}^4)$ where $\tilde{n}$ is the average sequence length. Hence, structure based methods are usually significantly slower than sequence based heuristics. Results are, however, highly accurate because the structural constraints are of great value to build the final sequence alignment. The consistency-based methods usually employ these constraints during the consistency extension. That is, the weights of aligned substrings are adapted depending on intermediate sequences *and* structural information.

3 Available Implementations

In Table 2, we compile a list of current multiple sequence alignment tools. Given the plethora of available tools, this list is necessarily incomplete but should include most of the frequently used programs. Online web servers hosting the different alignment algorithms are frequently available, except for the genome aligners. Nevertheless, we restrained ourselves from providing web addresses of these servers because they tend to change frequently and can be easily found online by searching the name of the tool and the word ”alignment”. It is hard to recommend a specific tool because none of them is superior in all cases. For a classical protein alignment of less than a hundred sequences the most accurate aligners are probably MAFFT, ProbCons, MUSCLE, T-Coffee and SeqAn::T-Coffee. For huge numbers of sequences, MAFFT and MUSCLE seem to scale the best. For long sequences, the previously mentioned aligners tend to have memory problems due to the pairwise dynamic programming, except for MAFFT and if long segments are provided DIALIGN-TX and SeqAn::T-Coffee. ABA seems to be an interesting choice if there are repeated or shuffled elements. Genome aligners are still in its infancy but MUMmer is certainly the most widely used tool for all kinds of genomic analyses.

Category	Name	Method	Protein / DNA
Sequence-based exact	LASA [2]	Lagrangian ILP approach	Both
	MSA [52]	Bounded dynamic programming	Both
Sequence-based heuristic	ABA [68]	A-Bruijn alignment	Both
	AMAP [78]	Sequence annealing	Both
	CLUSTAL W [88]	Progressive alignment	Both
	DIALIGN-T [85]	Segment-based alignment	Both
	DIALIGN-TX [84]	Progressive, segment-based	Both
	Kalign [48]	Progressive alignment	Both
	POA [50]	Partial order alignment	Protein
	MAFFT [41]	Progressive with refinement	Both
	MUSCLE [24]	Progressive with refinement	Both
	ProbCons [20]	Progressive with consistency	Protein
	SeqAn::T-Coffee [70]	Progressive, segment-based	Both
	T-Coffee [59]	Progressive with consistency	Both
Sequence-based meta-alignment	M-Coffee [91]	Progressive with consistency	Both
	SeqAn::T-Coffee [70]	Progressive, segment-based	Both
Using secondary structure and database homologs	MUMMALS [63]	Progressive with consistency	Protein
	PRALINE [79]	Progressive alignment	Protein
	PROMALS [64]	Progressive with consistency	Protein
	SPEM [95]	Progressive with consistency	Protein
Using 3D structure	3D-Coffee [91]	Progressive with consistency	Protein
	Expresso [7]	Progressive with consistency	Protein
	PROMALS3D [65]	Progressive with consistency	Protein
Genome aligners	M-GCAT [90]	Anchor-based alignment	DNA
	Mauve [16]	Anchors, local collinear blocks	DNA
	MGA [39]	Anchor-based, chaining	DNA
	Mulan [61]	Anchor-based alignment	DNA
	Multi-LAGAN [11]	Anchor-based alignment	DNA
	MUMmer [46]	Anchor-based, suffix-tree	DNA
	TBA [10]	Anchor-based alignment	DNA

Table 2 Available MSA programs, categorized according to the used information sources (sequence / structure), the nature of the algorithm (exact / heuristic) and the ability to align genomic sequences. The method column highlights only the *predominant* technique. Thus, a progressive aligner using refinement might also use some kind of consistency extension.

4 Advanced Topics

This chapter focused on the classical alignment problems and described the predominant algorithms in detail. This demanded excluding some other topics. Most importantly, we could not touch some algorithms for finding conserved motifs in multiple sequences. Even if sequences look completely unrelated, they still might share a conserved pattern such as a regulatory binding site. This sort of local multiple sequence alignment problem is addressed in numerous programs such as the Gibbs Sampler [49] or MEME [8].

Related RNA sequences quite often have low sequence but high structural sequence similarity. An RNA sequence folds onto itself and the pairwise interactions induced by such a fold are characteristic for related RNA sequences, e.g. the

cloverleaf shape of transfer-RNA. Hence, an accurate multiple sequence alignment of RNA sequences either requires (1) a sequence alignment followed by a folding step, (2) a de-novo folding followed by an alignment of fixed structures or (3) a simultaneous sequence-structure alignment. Similar to BAliBASE, RNA alignment algorithms can be evaluated using BRAliBase [30].

Another special problem occurs in genome assemblies. Genome assemblers usually follow a three phase methodology: an overlap phase, a layout phase and a consensus phase. In the consensus phase, the problem is to compute a multi-read alignment given a large set of reads and their approximate layout positions. That is, numerous short sequences overlap only a few bases and global relatedness cannot be assumed. Likewise a local approach is not enough, since all reads must be placed in a multi-read alignment to retrieve a consensus sequence. Tools from the AMOS and SeqAn library [21, 69, 83] address this kind of alignment problem.

Finally, we did not address the various alignment formats because Fasta seems to be the de facto standard. If you do need another format such as MSF, PIR, or Phylip, it is very likely that there exists already a converter in the EMBOSS [74] suite of programs. Similarly, we did not mention alignment viewers and editors. Interesting options are, for instance, Jalview [13] and SEAVIEW [29] or sequence logo generators [15].

5 Exercises

1. Calculate the sum of pairs score of the following MSA. (a) Assume a match score of 4, a mismatch score of -3, a gap extension score of -2 and a gap opening score of -4. (b) Recalculate the sum of pairs score for constant gap costs of -2.

```
A A T G
A – T G
– – T G
```

2. The EMBOSS explorer is a graphical user interface to a number of bioinformatics tools from the EMBOSS project (http://emboss.sourceforge.net/). One of these programs, “fneighbor”, computes a phylogeny from a distance matrix by neighbor-joining or UPGMA

   `http://emboss.bioinformatics.nl/cgi-bin/emboss/fneighbor`

   Use the Phylip distance matrix below to compute a phylogeny using both methods and compute the tree manually using UPGMA and weighted average linkage clustering.

```
  6
A 0.0000 0.5000 0.4000 0.7000 0.6000 0.8000
B 0.5000 0.0000 0.7000 1.0000 0.9000 1.1000
C 0.4000 0.7000 0.0000 0.7000 0.6000 0.8000
D 0.7000 1.0000 0.7000 0.0000 0.5000 0.9000
E 0.6000 0.9000 0.6000 0.5000 0.0000 0.8000
F 0.8000 1.1000 0.8000 0.9000 0.8000 0.0000
```

3. Get the sequences shown in Table 1 from UniProt (http://www.uniprot.org/). Add other globin sequences and align all of them using different multiple sequence alignment programs. Compare the alignments using Jalview or any other alignment viewer of your choice.
4. How many distinct pairwise comparisons can be done for n sequences?
5. Given n sequences, what is the minimum height of a perfectly balanced binary guide tree? What is the height of a guide tree that never requires a merging of two subalignments but only sequence to sequence or subalignment to sequence alignments?
6. Build a multiple sequence alignment of the following sequences using MUSCLE and MAFFT. Do you get an alignment where all the characters are matched, that is, all columns contain only one specific DNA nucleotide and possibly gaps? If this is not the case build such an alignment yourself. Why do the programs fail to deliver such an alignment?

```
CTTCGCGTCATCATCACT
CTTGAGTCATCATCACC
TCATCATCACTTGA
TCATCATCACCTCGGA
```

7. Implement the exact multiple alignment algorithm using dynamic programming for n DNA sequences. Assume constant gap penalties and the sum of pairs score with a constant match and mismatch score.

6 Further Reading

A number of review articles cover certain aspects of multiple sequence alignment in more depth than the preceding chapter. For the following list of topics, we can point the reader to interesting articles: (1) computational methods for genomic alignments [9], (2) accurate protein sequence alignments for divergent protein sequences [62], (3) evaluation of parameter choices in progressive alignment methods [94], (4) algorithms for multiple string comparisons [36], and (5) two more program-centered multiple sequence alignment review articles [25, 66]. Another in-depth discussion of multiple sequence alignment problems can be found in the book ”Biological sequence analysis” [22]. Besides introducing probabilistic alignment algorithms, the book provides extensive information on Profile HMMs to repre-

sent sequence families and search databases for new family members. It also covers stochastic context-free grammars (SCFGs) for RNA structure analysis.

References

1. Abouelhoda, M.I., Ohlebusch, E.: Multiple genome alignment: Chaining algorithms revisited. In: Proc. 14th Annual Symposium on Combinatorial Pattern Matching, Lect. Notes Comput. Sci., pp. 1–16 (2003)
2. Althaus, E., Canzar, S.: Bioinformatics research and development, chap. LASA: A tool for non-heuristic alignment of multiple sequences, pp. 489–498. Springer (2008)
3. Althaus, E., Caprara, A., Lenhof, H.P., Reinert, K.: Multiple sequence alignment with arbitrary gap costs: Computing an optimal solution using polyhedral combinatorics. Bioinformatics **18 Suppl 2**, S4–S16 (2002)
4. Althaus, E., Caprara, A., Lenhof, H.P., Reinert, K.: A branch-and-cut algorithm for multiple sequence alignment. Math. Programm. **105**, 387–425 (2006)
5. Altschul, S.F., Gish, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. **215**(3), 403–410 (1990)
6. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. Nucleic Acids Res. **25**(17), 3389–3402 (1997)
7. Armougom, F., Moretti, S., Poirot, O., Audic, S., Dumas, P., Schaeli, B., Keduas, V., Notredame, C.: Expresso: Automatic incorporation of structural information in multiple sequence alignments using 3D-Coffee. Nucleic Acids Res. **34**, W604–608 (2006)
8. Bailey, T.L., Williams, N., Misleh, C., Li, W.W.: MEME: Discovering and analyzing DNA and protein sequence motifs. Nucleic Acids Res. **34**(suppl 2), W369–373 (2006)
9. Blanchette, M.: Computation and analysis of genomic multi-sequence alignments. Annu. Rev. Genomics Hum. Genet. **8**(1), 193–213 (2007)
10. Blanchette, M., Kent, W.J., Riemer, C., Elnitski, L., Smit, A.F., Roskin, K.M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E.D., Haussler, D., Miller, W.: Aligning multiple genomic sequences with the threaded blockset aligner. Genome Res. **14**(4), 708–715 (2004)
11. Brudno, M., Do, C.B., Cooper, G.M., Kim, M.F., Davydov, E., Green, E.D., Sidow, A., Batzoglou, S.: LAGAN and Multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA. Genome Res. **13**, 721–731 (2003)
12. Buhler, J.: Efficient large-scale sequence comparison by locality-sensitive hashing. Bioinformatics **17**(5), 419–428 (2001)
13. Clamp, M., Cuff, J., Searle, S.M., Barton, G.J.: The Jalview Java alignment editor. Bioinformatics **20**(3), 426–427 (2004)
14. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to algorithms. MIT Press, Cambridge, MA (2001)
15. Crooks, G.E., Hon, G., Chandonia, J.M., Brenner, S.E.: WebLogo: A sequence logo generator. Genome Res. **14**(6), 1188–1190 (2004)
16. Darling, A.C., Mau, B., Blattner, F.R., Perna, N.T.: Mauve: Multiple alignment of conserved genomic sequence with rearrangements. Genome Res. **14**(7), 1394–1403 (2004)
17. Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C.: A model of evolutionary change in proteins. In: M.O. Dayhoff (ed.) Atlas of Protein Structure, vol. 5(Suppl. 3), pp. 345–352. National Biomedical Reasearch Foundataion, Silver Spring, Md. (1979)
18. Delcher, A.L., Kasif, S., Fleischmann, R.D., Peterson, J., White, O., Salzberg, S.L.: Alignment of whole genomes. Nucleic Acids Res. **27**(11), 2369–2376 (1999)
19. Delcher, A.L., Phillippy, A., Carlton, J., Salzberg, S.L.: Fast algorithms for large-scale genome alignment and comparison. Nucleic Acids Res. **30**(11), 2478–2483 (2002)
20. Do, C.B., Mahabhashyam, M.S., Brudno, M., Batzoglou, S.: ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome Res. **15**, 330–340 (2005)

21. Döring, A., Weese, D., Rausch, T., Reinert, K.: SeqAn - An efficient, generic C++ library for sequence analysis. BMC Bioinformatics **9**, 11 (2008)
22. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge University Press (1998)
23. Edgar, R.C.: Local homology recognition and distance measures in linear time using compressed amino acid alphabets. Nucleic Acids Res. **32**(1), 380–385 (2004)
24. Edgar, R.C.: MUSCLE: Multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. **32**(5), 1792–1797 (2004)
25. Edgar, R.C., Batzoglou, S.: Multiple sequence alignment. Curr. Opin. Struct. Biol. **16**(3), 368 – 373 (2006)
26. Edgar, R.C., Sjolander, K.: A comparison of scoring functions for protein sequence profile alignment. Bioinformatics **20**(8), 1301–1308 (2004)
27. Feng, D.F., Doolittle, R.F.: Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol. **25**, 351–360 (1987)
28. Fitch, W.M., Margoliash, E.: Construction of phylogenetic trees. Science **155**(760), 279–84 (1967)
29. Galtier, N., Gouy, M., Gautier, C.: SEAVIEW and PHYLO WIN: Two graphic tools for sequence alignment and molecular phylogeny. Comput. Appl. Biosci. **12**(6), 543–548 (1996)
30. Gardner, P.P., Wilm, A., Washietl, S.: A benchmark of multiple sequence alignment programs upon structural RNAs. Nucleic Acids Res. **33**(8), 2433–2439 (2005)
31. Gotoh, O.: An improved algorithm for matching biological sequences. J. Mol. Biol. **162**(3), 705–708 (1982)
32. Gotoh, O.: Alignment of three biological sequences with an efficient traceback procedure. J. Theor. Biol. **121**(3), 327–37 (1986)
33. Gotoh, O.: Consistency of optimal sequence alignments. Bull. Math. Biol. **52**, 509–525 (1990)
34. Gotoh, O.: Multiple sequence alignment: Algorithms and applications. Adv. Biophys. **36**, 159–206 (1999)
35. Gupta, S.K., Kececioglu, J.D., Schffer, A.A.: Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. J. Comput. Biol. **2**, 459–472 (1995)
36. Gusfield, D.: Algorithms on strings, trees, and sequences: Computer science and computational biology. Cambridge University Press, New York, NY, USA (1997)
37. Henikoff, S., Henikoff, J.G.: Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. U.S.A. **89**(22), 10,915–10,919 (1992)
38. Higgins, D.G., Sharp, P.M.: CLUSTAL: A package for performing multiple sequence alignment on a microcomputer. Gene **73**(1), 237–244 (1988)
39. Hohl, M., Kurtz, S., Ohlebusch, E.: Efficient multiple genome alignment. Bioinformatics **18**(suppl 1), S312–320 (2002)
40. Katoh, K., Kuma, K., Toh, H., Miyata, T.: MAFFT version 5: Improvement in accuracy of multiple sequence alignment. Nucleic Acids Res. **33**(2), 511–518 (2005)
41. Katoh, K., Misawa, K., Kuma, K., Miyata, T.: MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform. Nucleic Acids Res. **30**, 3059–3066 (2002)
42. Kececioglu, J.D.: Exact and approximation algorithms for DNA sequence reconstruction. Ph.D. thesis, University of Arizona, Tucson, AZ, USA (1992)
43. Kececioglu, J.D.: The maximum weight trace problem in multiple sequence alignment. In: Proc. 4th Annual Symposium on Combinatorial Pattern Matching, Lect. Notes Comput. Sci., pp. 106–119. Springer-Verlag, London, UK (1993)
44. Kececioglu, J.D., Starrett, D.: Aligning alignments exactly. In: Proc. 8th Annual International Conference on Research in Computational Molecular Biology, RECOMB, pp. 85–96. ACM, New York, NY, USA (2004)
45. Kececioglu, J.D., Zhang, W.: Aligning alignments. In: Proc. 9th Annual Symposium on Combinatorial Pattern Matching, Lect. Notes Comput. Sci., pp. 189–208. Springer Verlag (1998)
46. Kurtz, S., Phillippy, A., Delcher, A., Smoot, M., Shumway, M., Antonescu, C., Salzberg, S.: Versatile and open software for comparing large genomes. Genome Biol. **5**(2), R12 (2004)

47. Larkin, M.A., Blackshields, G., Brown, N.P., Chenna, R., McGettigan, P.A., McWilliam, H., Valentin, F., Wallace, I.M., Wilm, A., Lopez, R., Thompson, J.D., Gibson, T.J., Higgins, D.G.: Clustal W and Clustal X version 2.0. Bioinformatics **23**(21), 2947–2948 (2007)
48. Lassmann, T., Sonnhammer, E.: Kalign - An accurate and fast multiple sequence alignment algorithm. BMC Bioinformatics **6**(1), 298 (2005)
49. Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F., Wootton, J.C.: Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. Science **262**(5131), 208–214 (1993)
50. Lee, C., Grasso, C., Sharlow, M.F.: Multiple sequence alignment using partial order graphs. Bioinformatics **18**(3), 452–464 (2002)
51. Lermen, M., Reinert, K.: The practical use of the A* algorithm for exact multiple sequence alignment. J. Comput. Biol. **7**, 655–671 (2000)
52. Lipman, D.J., Altschul, S.F., Kececioglu, J.D.: A tool for multiple sequence alignment. Proc. Natl. Acad. Sci. U.S.A. **86**, 4412–4415 (1989)
53. Ma, B., Wang, Z., Zhang, K.: Alignment between two multiple alignments. In: Proc. 14th Annual Symposium on Combinatorial Pattern Matching, Lect. Notes Comput. Sci., *Lect. Notes Comput. Sci.*, vol. 2676, pp. 254–265. Springer (2003)
54. McGuffin, L.J., Bryson, K., Jones, D.T.: The PSIPRED protein structure prediction server. Bioinformatics **16**(4), 404–405 (2000)
55. Morgenstern, B., Frech, K., Dress, A., Werner, T.: DIALIGN: Finding local similarities by multiple sequence alignment. Bioinformatics **14**(3), 290–294 (1998)
56. Murata, M., Richardson, J.S., Sussman, J.L.: Simultaneous comparison of three protein sequences. Proc. Natl. Acad. Sci. U.S.A. **82**(10), 3073–3077 (1985)
57. Myers, G., Miller, W.: Chaining multiple-alignment fragments in sub-quadratic time. In: Proc. 6th Annual ACM-SIAM Symposium, pp. 38–47. Soc. Ind. Appl. Math., Philadelphia, PA, USA (1995)
58. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. **48**, 443–453 (1970)
59. Notredame, C., Higgins, D., Heringa, J.: T-Coffee: A novel method for fast and accurate multiple sequence alignment. J. Mol. Biol. **302**, 205–217 (2000)
60. O'Sullivan, O., Suhre, K., Abergel, C., Higgins, D.G., Notredame, C.: 3DCoffee: Combining protein sequences and structures within multiple sequence alignments. J. Mol. Biol. **340**(2), 385 – 395 (2004)
61. Ovcharenko, I., Loots, G.G., Giardine, B.M., Hou, M., Ma, J., Hardison, R.C., Stubbs, L., Miller, W.: Mulan: Multiple-sequence local alignment and visualization for studying function and evolution. Genome Res. **15**(1), 184–194 (2005)
62. Pei, J.: Multiple protein sequence alignment. Curr. Opin. Struct. Biol. **18**(3), 382 – 386 (2008)
63. Pei, J., Grishin, N.V.: MUMMALS: Multiple sequence alignment improved by using hidden Markov models with local structural information. Nucleic Acids Res. **34**, 4364–4374 (2006)
64. Pei, J., Grishin, N.V.: PROMALS: Towards accurate multiple sequence alignments of distantly related proteins. Bioinformatics **23**, 802–808 (2007)
65. Pei, J., Kim, B.H., Grishin, N.V.: PROMALS3D: A tool for multiple protein sequence and structure alignments. Nucleic Acids Res. **36**(7), 2295–2300 (2008)
66. Pirovano, W., Heringa, J.: Multiple sequence alignment. Methods Mol. Biol. **452**, 143–61 (2008)
67. Raghava, G.P., Searle, S., Audley, P., Barber, J., Barton, G.: OXBench: A benchmark for evaluation of protein multiple sequence alignment accuracy. BMC Bioinformatics **4**(1), 47 (2003)
68. Raphael, B., Zhi, D., Tang, H., Pevzner, P.: A novel method for multiple alignment of sequences with repeated and shuffled elements. Genome Res. **14**(11), 2336–2346 (2004)
69. Rausch, T., Emde, A.K., Reinert, K.: Robust consensus computation. BMC Bioinformatics **9**(Suppl 10), P4 (2008)
70. Rausch, T., Emde, A.K., Weese, D., Döring, A., Notredame, C., Reinert, K.: Segment-based multiple sequence alignment. Bioinformatics **24**(16), i187–192 (2008)
71. Reinert, K.: A polyhedral approach to sequence alignment problems. Ph.D. thesis, Universität Saarbrücken (1999)

72. Reinert, K., Lenhof, H.P., Mutzel, P., Mehlhorn, K., Kececioglu, J.: A branch-and-cut algorithm for multiple sequence alignment. In: Proc. 1st Annual International Conference on Research in Computational Molecular Biology, RECOMB, pp. 241–249 (1997)
73. Reinert, K., Stoye, J., Will, T.: An iterative method for faster sum-of-pairs multiple sequence alignment. Bioinformatics **16**(9), 808–814 (2000)
74. Rice, P., Longden, I., Bleasby, A.: EMBOSS: The european molecular biology open software suite. Trends Genet. **16**(6), 276 – 277 (2000)
75. Rost, B.: Review: Protein secondary structure prediction continues to rise. J. Struct. Biol. **134**(2-3), 204 – 218 (2001)
76. Saitou, N., Nei, M.: The neighbor-joining method: A new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**, 406–425 (1987)
77. Sankoff, D., Kruskal, J.B.: Time warps, string edits, and macromolecules: The theory and practice of sequence comparison. Addison-Wesley, Reading, MA (1983)
78. Schwartz, A.S., Pachter, L.: Multiple alignment by sequence annealing. Bioinformatics **23**, e24–29 (2007)
79. Simossis, V.A., Heringa, J.: PRALINE: A multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. Nucleic Acids Res. **33**, W289 (2005)
80. Simossis, V.A., Kleinjung, J., Heringa, J.: Homology-extended sequence alignment. Nucleic Acids Res. **33**(3), 816–824 (2005)
81. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. **147**(1), 195–197 (1981)
82. Sokal, R.R., Michener, C.D.: A statistical method for evaluating systematic relationships. Univ. Kansas Sci. Bull. **38**, 1409–1438 (1958)
83. Sommer, D., Delcher, A., Salzberg, S., Pop, M.: Minimus: A fast, lightweight genome assembler. BMC Bioinformatics **8**(1), 64 (2007)
84. Subramanian, A., Kaufmann, M., Morgenstern, B.: DIALIGN-TX: Greedy and progressive approaches for segment-based multiple sequence alignment. Algorithms Mol. Biol. **3**(1), 6 (2008)
85. Subramanian, A., Weyer-Menkhoff, J., Kaufmann, M., Morgenstern, B.: DIALIGN-T: An improved algorithm for segment-based multiple sequence alignment. BMC Bioinformatics **6**(1), 66 (2005)
86. Taylor, W.: Protein structure comparison using iterated double dynamic programming. Protein Sci. **8**(3), 654–665 (1999)
87. Thompson, J., Plewniak, F., Poch, O.: BAliBASE: A benchmark alignment database for the evaluation of multiple alignment programs. Bioinformatics **15**, 87–88 (1999)
88. Thompson, J.D., Higgins, D.G., Gibson, T.J.: CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. **22**, 4673–4680 (1994)
89. Thompson, J.D., Koehl, P., Ripp, R., Poch, O.: BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark. Proteins **61**, 127–136 (2005)
90. Treangen, T., Messeguer, X.: M-GCAT: Interactively and efficiently constructing large-scale multiple genome comparison frameworks in closely related species. BMC Bioinformatics **7**(1), 433 (2006)
91. Wallace, I.M., O'Sullivan, O., Higgins, D.G., Notredame, C.: M-Coffee: Combining multiple sequence alignment methods with T-Coffee. Nucleic Acids Res. **34**, 1692–1699 (2006)
92. Walle, I.V., Lasters, I., Wyns, L.: SABmark - A benchmark for sequence alignment that covers the entire known fold space. Bioinformatics **21**(7), 1267–1268 (2005)
93. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. J. Comput. Biol. **1**, 337–348 (1994)
94. Wheeler, T.J., Kececioglu, J.D.: Multiple alignment by aligning alignments. Bioinformatics **23**, 559–568 (2007)
95. Zhou, H., Zhou, Y.: SPEM: Improving multiple sequence alignment with sequence profiles and predicted secondary structures. Bioinformatics **21**(18), 3615–3621 (2005)