Progressive Multiple Sequence Alignments from Triplets

Matthias Kruspe [a] and Peter F. Stadler [a,b,c]

[a]Bioinformatics Group, Department of Computer Science and Interdisciplinary Center for
Bioinformatics, University of Leipzig, Härtelstraße 16-18, D-04107 Leipzig, Germany
[b]Institute for Theoretical Chemistry, University of Vienna, Währingerstraße 17, A-1090 Wien,
Austria
[c]Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501

ABSTRACT

Motivation: The quality of progressive sequence alignments strongly depends on the accuracy of the individual pairwise alignment steps since gaps that are introduced at one step cannot be removed at later aggregation steps. Adjacent insertions and deletions necessarily appear in arbitrary order in pairwise alignments and hence form an unavoidable source of errors.

Idea: Here we present a modified variant of progressive sequence alignments that addresses both issues. Instead of pairwise alignments we use exact dynamic programming to align sequence or profile triples. This avoids a large fractions of the ambiguities arising in pairwise alignments. In the subsequent aggregation steps we follow the logic of the Neighbor-Net algorithm, which constructs a phylogenetic network by step-wisely replacing triples by pairs instead of combining pairs to singletons. To this end the three-way alignments are subdivided into two partial alignments, at which stage all-gap columns are naturally removed. This alleviates the "once a gap, always a gap" problem of progressive alignment procedures.

Results: The three-way Neighbor-Net based alignment program `aln3nn` is shown to compare favorably on both protein sequences and nucleic acids sequences. In the latter case one easily can include scoring terms that consider secondary structure features. Overall, the quality of resulting alignments in general exceeds that of `clustalw` alignments even though our software does not included heuristics for context dependent (mis)match scores.

Availability: Software is freely available for download from `http://www.bioinf.uni-leipzig.de/~matthias/aln3nn`.

Contact: {`matthias,studla`}`@bioinf.uni-leipzig.de`

1 INTRODUCTION

High quality multiple sequence alignments (MSAs) are a prerequisite for many applications in bioinformatics, from the reconstruction of phylogenies and the assessment of evolutionary rate variations to gene finding and phylogenetic footprinting. A large part of comparative genomics thus hinges on our ability to construct accurate MSAs. Since the multiple sequence alignment problem is NP hard (Wang & Jiang, 1994) with the computational cost growing exponentially with the number of sequences, it has been a long-standing challenge to devise approximation algorithms that are both efficient and accurate. These approaches can be classified into progressive, iterative, and stochastic alignment algorithms. The most widely used tools such as `clustalw` (Thompson *et al.*, 1994) and `pileup` utilize the progressive method that was at first introduced

in (Hogeweg & Hesper, 1984; Feng & Doolittle, 1987). This ansatz makes explicit use of the evolutionary relatedness of the sequences to build the alignment. The complete multiple sequence alignment of the given sequences is calculated from pairwise alignments of previous aligned sequences by following the branching order of a pre-computed "guide" tree, which reflects (at least approximately) the evolutionary history of the input sequences. It is reconstructed from pairwise sequence distances by some clustering method such as Neighbor-Joining (Saitou & Nei, 1987) or UPGMA (Sokal & Michner, 1958).

Progressive sequence alignments, while computationally efficient, suffer from two major shortcomings. First, they are of course not guaranteed to find the optimal alignment. Pairwise comparisons necessarily utilize only a small part of the information that is potentially available in the complete data set. In particular, the relative placement of adjacent insertions and deletions leads to score-equivalent alignments among which the algorithm chooses one by means of a pragmatic rule (e.g. "Always make insertions before deletions"). At a later aggregation step, when profiles are aligned to sequences or with each other, these alternative are no longer equivalent. Secondly, in contrast to other techniques, there is no mechanism to identify errors that have been made in previous steps and to correct them during later stages.

In this contribution we present a novel approach to progressive sequence alignment that alleviates both shortcomings at the expense of utilizing an exact algorithm to compute alignment of sequence and profile triples. Instead of using a single guide tree, we follow here the logic of phylogenetic networks as constructed by the Neighbor-Net algorithm (Bryant & Moulton, 2002) which calls for an aggregation step that constructs pairs from triples. As this requires us to subdivide 3-way alignments into pairs of alignments, it provides a chance for the removal of erroneously inserted gaps at later aggregation steps.

The contribution is organized as follows: In the following section we outline the algorithms aspects of our approach. Furthermore we describe a straightforward way of incorporating RNA secondary information. Section 3 summarizes benchmark data in comparison to other multiple alignment tools. We conclude with a brief discussion of future improvements.

2 METHODS

2.1 Dynamic Programming

The basic dynamic programming scheme for pairwise sequence comparison, known as the Needleman-Wunsch algorithm (Needleman & Wunsch, 1970) requires quadratic space and time. It easily translates to a cubic space and time algorithms for three sequences. Biologically plausible sequence alignment, however, require the use of non-trivial gap cost functions. While cubic time algorithms are available for arbitrary gap costs (Dewey, 2001), affine gap costs (with a much higher penalty for opening a new gap than for extending an existing one) in general yield good results already. In this contribution we also use an affine gap costs. Gotoh's algorithm solves this problem with quadratic CPU and memory requirements for two sequences (Gotoh, 1982). The same author also described a dynamic programming scheme for the alignment of three sequences with affine gap costs that requires $\mathcal{O}(n^3)$ time and space, which we use here with minor modifications.

Let A, B, and C denote the three sequences. We use A_i, B_j, and C_k to refer to the ith, jth, and kth position in A, B, and C, respectively, counting from 1. As usual, '-' denotes the gap character. Scores for the alignment of two or three non-gap characters are denoted by $S(\alpha, \beta)$ and $S(\alpha, \beta, \gamma)$, resp. Gap penalties are determined from gap open (g_o) and gap extensions (g_e) scores. The best score of the alignments of the prefixes A_i, B_j, and C_k is denoted by $M(i, j, k)$ if the residues (A_i, B_j, C_k) are aligned; $I_{xy}(i, j, k)$ the best score given that $(A_i, B_j, -)$ is the last column of the partial alignment, and $I_x(i, j, k)$ the best score given that the last column is of the form $(A_i, -, -)$. $I_{xz}(i, j, k)$, $I_{yz}(i, j, k)$, $I_y(i, j, k)$, and $I_z(i, j, k)$ are defined analogously. It is not hard to verify that these quantities must satisfy the recursions summarized in Fig. 1.

While the algorithm would in principle allow us to use arbitrary three residue substitution scores $S(a, b, c)$ as described by (Konagurthu *et al.*, 2004), we restrict ourselves to the sum-of-pairs model $S(a, b, c) = S(a, b) + S(a, c) + S(b, c)$. As is the case of pairwise sequence alignments, the recursions immediately generalize to alignments of profiles such that a single sequence becomes a special case of a profile. Match and gap scores are simply added up over all triples of sequences, one from each profile.

The resource requirements of this algorithm, in particular the cubic memory consumption, are acceptable only for relative small sequence lengths n even on modern workstations. Several approaches have been explored in the past to reduce the search space so that long sequences can be dealt with, see e.g. (Myers & Miller, 1988; Lipman *et al.*, 1989; Gupta *et al.*, 1995). We utilized here the divide-&-conquer approach described by (Stoye, 1997) to limit both space and time requirements. Input sequences that exceed a given threshold length l are subsequently subdivided into smaller sequences until the length criterion is fulfilled. The partial sequences are aligned separately and the emerging alignments are concatenated afterward. The result is an approximate solution of the global multiple sequence alignment problem. The methods described by (Myers & Miller, 1988; Lipman *et al.*, 1989) are known to produce optimal alignments but are much harder to implement.

2.2 Alignment order

The order in which sequences and profiles are aligned has an important influence on the performance of progressive alignment algorithms. In the programs such as `clustalw` or `pileup` that are

$$M(i,j,k) = \max \begin{cases} M(i-1, j-1, k-1) + S(A_i, B_j, C_k) \\ I_{xy}(i-1, j-1, k-1) + S(A_i, B_j, C_k) \\ I_{xz}(i-1, j-1, k-1) + S(A_i, B_j, C_k) \\ I_{yz}(i-1, j-1, k-1) + S(A_i, B_j, C_k) \\ I_x(i-1, j-1, k-1) + S(A_i, B_j, C_k) \\ I_y(i-1, j-1, k-1) + S(A_i, B_j, C_k) \\ I_z(i-1, j-1, k-1) + S(A_i, B_j, C_k) \end{cases}$$

$$I_{xy}(i,j,k) = \max \begin{cases} M(i-1, j-1, k) - g_o + S(A_i, B_j) \\ I_{xy}(i-1, j-1, k) - g_e + S(A_i, B_j) \\ I_{xz}(i-1, j-1, k) - g_o + S(A_i, B_j) \\ I_{yz}(i-1, j-1, k) - g_o + S(A_i, B_j) \\ I_x(i-1, j-1, k) - g_e + S(A_i, B_j) \\ I_y(i-1, j-1, k) - g_e + S(A_i, B_j) \\ I_z(i-1, j-1, k) - g_o + S(A_i, B_j) \end{cases}$$

$$I_{xz}(i,j,k) = \max \begin{cases} M(i-1, j, k-1) - g_o + S(A_i, C_k) \\ I_{xy}(i-1, j, k-1) - g_o + S(A_i, C_k) \\ I_{xz}(i-1, j, k-1) - g_e + S(A_i, C_k) \\ I_{yz}(i-1, j, k-1) - g_o + S(A_i, C_k) \\ I_x(i-1, j, k-1) - g_e + S(A_i, C_k) \\ I_y(i-1, j, k-1) - g_o + S(A_i, C_k) \\ I_z(i-1, j, k-1) - g_e + S(A_i, C_k) \end{cases}$$

$$I_{yz}(i,j,k) = \max \begin{cases} M(i-1, j-1, k) - g_o + S(B_j, C_k) \\ I_{xy}(i-1, j-1, k) - g_o + S(B_j, C_k) \\ I_{xz}(i-1, j-1, k) - g_o + S(B_j, C_k) \\ I_{yz}(i-1, j-1, k) - g_e + S(B_j, C_k) \\ I_x(i-1, j-1, k) - g_o + S(B_j, C_k) \\ I_y(i-1, j-1, k) - g_e + S(B_j, C_k) \\ I_z(i-1, j-1, k) - g_e + S(B_j, C_k) \end{cases}$$

$$I_x(i,j,k) = \max \begin{cases} M(i-1, j, k) - 2g_o \\ I_{xy}(i-1, j, k) - g_o - g_e \\ I_{xz}(i-1, j, k) - g_e - g_o \\ I_{yz}(i-1, j, k) - 2g_o \\ I_x(i-1, j, k) - 2g_e \\ I_y(i-1, j, k) - g_o - g_e \\ I_z(i-1, j, k) - g_e - g_o \end{cases}$$

$$I_y(i,j,k) = \max \begin{cases} M(i, j-1, k) - 2g_o \\ I_{xy}(i, j-1, k) - g_o - g_e \\ I_{xz}(i, j-1, k) - 2g_o \\ I_{yz}(i, j-1, k) - g_e - g_o \\ I_x(i, j-1, k) - g_o - g_e \\ I_y(i, j-1, k) - 2g_e \\ I_z(i, j-1, k) - g_e - g_o \end{cases}$$

$$I_z(i,j,k) = \max \begin{cases} M(i, j, k-1) - 2g_o \\ I_{xy}(i, j, k-1) - 2g_o \\ I_{xz}(i, j, k-1) - g_o - g_e \\ I_{yz}(i, j, k-1) - g_e - g_o \\ I_x(i, j, k-1) - g_o - g_e \\ I_y(i, j, k-1) - g_e - g_o \\ I_z(i, j, k-1) - 2g_e \end{cases}$$

Fig. 1. Dynamic programming recursions for three-way alignments with affine gap costs. The empty alignments are initialized as $M(0, 0, 0) = 0$ and $I_{..}(0, 0, 0) = 0$. The boundaries of the cubic tables are initialized using the recursions above with the understanding that alternatives with negative indices are ignored.
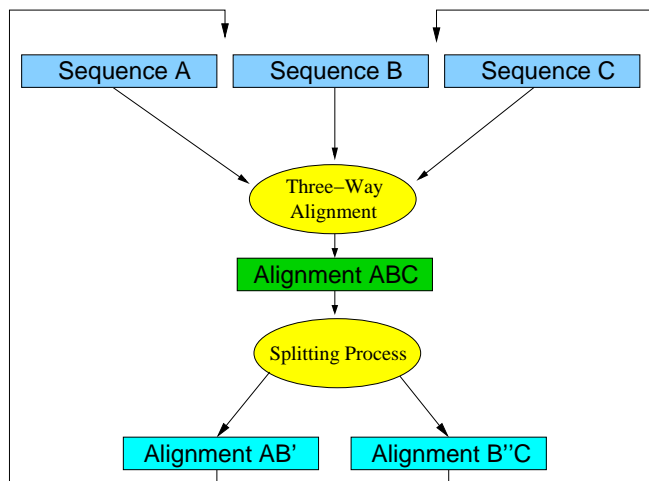
Fig. 2. The three sequences/alignments A, B, and C are aligned simultaneously resulting in the alignment ABC. This alignment is divided into the two new alignments AB and BC. Duplicated sequences in B are deleted. The process continues until all sequences or alignments are aligned.

based on pairwise alignments, binary guide trees are used to determine the alignment order. The input sequences form the leaves of this tree; each interior node corresponds to an alignment, so that the root of the guide tree represents the desired multiple alignment of all input sequences. Clustering procedures such as Neighbor-Joining (Saitou & Nei, 1987) or the simpler UPGMA approach are used to compute the guide tree from estimated distances between pairs of sequences.

Guide trees encapsulate at least an approximation to the phylogenetic relationships of the input sequences. Instead of a phylogenetic tree our tool uses a phylogenetic network to calculate the alignment order. The network is constructed using the Neighbor-Net (Nnet) approach, a distance based clustering algorithm that can be seen as a proper generalization of Neighbor-Joining (Bryant & Moulton, 2002). The Nnet algorithm can be described as follows: The input sequences are represented as nodes that are all disconnected in the beginning. In each aggregation step, Nnet selects two nodes using a specific selection criterion such as minimal distance. In contrast to Neighbor-Joining, the two nodes are not paired immediately. Instead, Nnet waits until a node has been paired up a second time. Then the corresponding three linked nodes are replaced by two new linked nodes. The entire procedure is repeated until only three active nodes are left. Then the agglomerated nodes are expanded to produce the planar splits graph that represents the desired phylogenetic network.

In our picture, every node agglomeration corresponds to a triplet alignment. The alignment order is therefore given by the order of node fusions. Nnet however replaces a triple by a pair. This suggests to split the three-way alignment again into a pair of alignments, see Fig. 2. In the Nnet, a node agglomeration occurs when one of the three involved nodes (B) has two neighbors, while the other two (A and C) have only a single one. We choose to split the alignment such the sequences contained in B are distributed between two subsets B' and B'' so as to maximize the scores of partial alignments AB' and $B''C$. In practice, we start with partial alignments AB and

BC obtained from ABC. Then each of the duplicated B sequences is removed from either AB or BC using a greedy rule, i.e., we remove the copy that yields the smaller average score contribution. Of course, other division strategies are conceivable. For example, one could subdivide the alignment along the longest internal edge of its Neighbor-Joining tree.

The division of the ABC alignment into AB' and $B''C$ may result in all-gap columns that are removed in order to recover valid MSAs. This constitutes a mechanism by which gaps introduced in early agglomeration steps can be removed again in later steps. This removal is guided by the increased amount of information that is implicit in profiles composed of a larger number of sequences. Our software keeps track of gaps that appear in intermediate alignments but that are not present in the final result to demonstrate that gap removal is not uncommon in practice.

2.3 Complexity

The dynamics programming algorithm for the three-way alignment requires $\mathcal{O}(n^3)$ space and time (where n is length of the input sequences). Thus the alignment of all N sequences takes $\mathcal{O}(Nn^3)$ time. If the divide-&-conquer approach with the cutoff length l is used, the complexity of the alignment of one triplet can be reduced to $\mathcal{O}(n^2 + l^3)$ space. This is the space needed to store the additional-cost matrices (see (Stoye, 1997)) plus the space required for aligning the remaining (sub)sequences of length at most l. The time complexity is given by $\mathcal{O}(n^2 + nl^2)$. The term n^2 results from the time that is needed to calculate the additional cost matrices plus the time to search for the optimal slicing positions. The term nl^2 comes from the alignment of the triplet itself. We assume for simplicity that all sequences have the same length $n = l \cdot 2^D$ ($D = 1, 2, ...$ is the number of dividing levels) and all slicing positions are located exactly at the midpoint of the (sub)sequences. The total time complexity of the alignment is therefore $\mathcal{O}(Nn^2 + Nnl^2)$. The determination of the alignment order runs in $\mathcal{O}(N^3)$ time and $\mathcal{O}(N)$ space. The calculation of the necessary pairwise distances takes $\mathcal{O}(N^2n^2)$ time and $\mathcal{O}(n^2)$ space.

The full source code of the program package is available free for academic users. The code will compile and run well on any machine with a full ANSI conforming C compiler and an installed Vienna RNA package. The source code and documentation is available from `http://www.bioinf.uni-leipzig.de/~matthias/aln3nn`.

2.4 Alignments of Structured RNAs

Recent discoveries of a large number of small RNAs with distinctive secondary structures has prompted the development of specialized multiple alignment programs for this class of molecules. Most of these approaches make explicit use of structural alignment techniques such as tree editing (MARNA (Siebert & Backofen, 2005)), tree alignments (RNAforrester (Höchsmann *et al.*, 2003)), or variants of the Sankoff algorithm (Sankoff, 1985) (foldalign (Hull Havgaard *et al.*, 2005), dynalign (Mathews & Turner, 2002), locarna (Will *et al.*, 2006)). In contrast, "structure enhanced" approaches utilize standard sequence alignment algorithms but incorporate modified match and mismatch scores designed to take structural information information into account (Bonhoeffer *et al.*, 1993). The STRAL program (Dalli *et al.*, 2006) recently has demonstrated that such "structure enhanced" perform comparable to true structural alignments in many cases. We have thus included in our
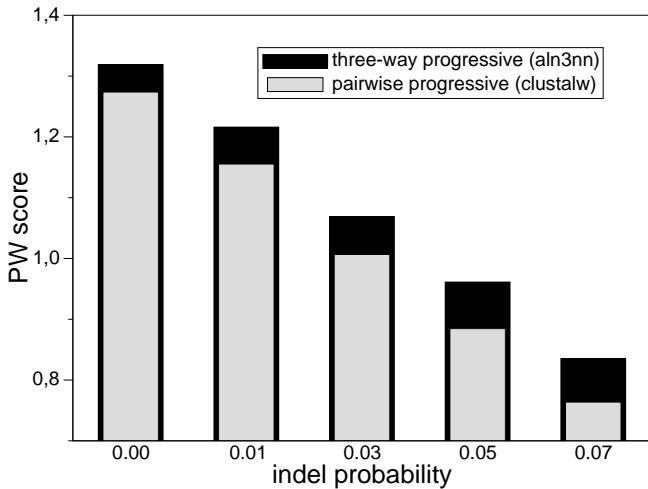
Fig. 3. Comparison of alignments scores of `aln3nn` with progressive pairwise alignments for simulated data sets for different in/del rates. Data are averages over 100 simulated sets of 10 related nucleotide sequences with an average length of 100nt. The sequences in each set are derived using `ROSE` from a randomly generated root sequence following the order of a given phylogenetic tree with arbitrary branch lengths using a constant mean substitution of 0.13. The folliowing scoring model was used: Match score 1.9, mismatch 0.0 (as in the IUB DNA scoring matrix used e.g. by `clustalw`, gap open 2.0, gap extensions 0.5.

software the possibility to use RNA secondary structure annotation as additional input with nucleic acid alignments.

We use McCaskill's algorithm (McCaskill, 1990) (implemented e.g. in the `Vienna RNA package` (Hofacker *et al.*, 1994, 2002)) to compute the matrix of equilibrium base pairing probabilities P_{ij} for each input sequence and derive for each sequence position the probabilities $p^1(i) = \sum_{j<i} P_{ij}$, $p^2(i) = \sum_{j>i} P_{ij}$, and $p^3(i) = 1 - p^1(i) - p^2(i)$ that sequence position i is paired with a position $j < i$, a position $j > i$, or that it remains unpaired, resp. The $p^x(i)$-values are used as structure annotation. For a pair of residues A_i and B_j in the annotated input sequences with define structural score contributions by

$$
\begin{aligned}
S_{\text{struct}}(A_i, B_j) = & \sqrt{p^1(A_i) \cdot p^1(A_j)} \\
& + \sqrt{p^2(A_i) \cdot p^2(A_j)} \\
& + \sqrt{p^3(A_i) \cdot p^3(A_j)}
\end{aligned} \tag{1}
$$

This rewards bases that share similar structural properties. The total (mis)match score is the weighted sum of the sequence score and the structure score using the equation

$$
S_{\text{final}}(A_i, B_j) = \psi \cdot S_{\text{seq}}(A_i, B_j) + (1 - \psi) \cdot S_{\text{struct}}(A_i, B_j) \tag{2}
$$

with a balance term ψ that measure the relative contribution of sequence and structure similarity. In the case of very similar sequence one should use $\psi \approx 1$ since inaccuracies in the structure prediction are more harmful than the extra information in this case. Conversely, very dissimilar sequences have to be aligned with a score dominated by the structural component.
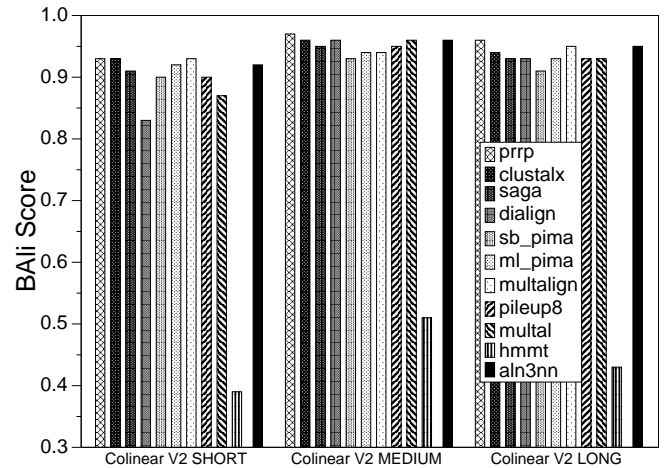
Fig. 4. Comparison of different alignment programs on several `BAliBASE` benchmark data sets.

# 3 RESULTS

3.1 Pairwise *versus* Tree-Way Alignments

In order to test whether the additional computational costs of explicit three-way alignments is worth while, we generated sets of artificial sequences using the `ROSE` package (Stoye *et al.*, 1998) and compared the quality of `aln3nn` alignments to standard progressive alignments of three sequences using `clustalw`. To this end we used the same scoring model in `aln3nn` and `clustalw` so that the resulting scores can be compared directly. We report the main pairwise alignment score divided by the length of the alignment as "*pw-score*". Figure 3 shows that, as expected, the alignment score decrease quickly with increasing in/del probabilities. At the same time, the advantage of the three-way alignments increases. Somewhat surprisingly, three-way alignments also provide a significant gain in alignment score even in the cases where the simulated data correspond to a correct alignment that is entirely gap free. This introduction of spurious gaps is well-known problem with pairwise nucleic acid alignments.

3.2 Protein Alignments

The `aln3nn` software is designed for the alignment of both amino acid and nucleic acid sequences. For proteins, the current implementation used three types of substitution matrices: BLOSUM, PAM and GONNET. The algorithm chooses the best suiting matrix of the given type according to sequence identity. The user can also specify a certain substitution matrix explicitly.

We used benchmark data sets from `BAliBASE` (Thompson *et al.*, 1999) to asses the quality of the `aln3nn` alignments. To assure statistical robustness, we utilized the median BAliBASE score for each sequence set as a measurement for alignment quality, Figure 4.

Our software does not employ any heuristic rules to alter scoring parameters based on local sequence context or properties of partial profiles. Nevertheless, `aln3nn` compares well with other common alignment programs, indicating that a simple affine scoring model is sufficient. Elaborate scoring heuristics thus essentially seem to compensate for the algorithmic shortcoming of MSAs based on initial pairwise alignments.
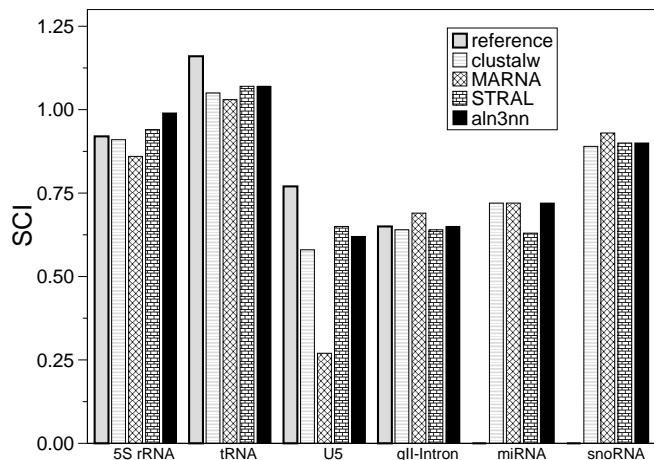
Fig. 5. The median SCI for approximately 100 alignments computed by `clustalw`, `STRAL`, `MARNA`, and `aln3nn` as well as hand-curated reference alignments for Group II introns, 5S rRNA, tRNA, U5 spliceosomal RNA. For the miRNA and snoRNA data set no reference alignment were available. The relative weight of sequence and structure scores is set to $\psi = 0.5$ for all data sets.

Fig. 6. Impact of the balancing parameter ψ on SCI mean values for different sets of RNA sequences. $\psi = 0$ fully weights the structure whereas $\psi = 1$ weights the sequence.

3.3 RNA Alignments

RNA sequences often evolve much faster than their secondary structure. This is true in particular for many of the non-coding RNA genes, including ribosomal RNAs, tRNAs, and spliceosomal RNAs. In these cases, alignment quality can be increased dramatically by including structural information.

In Fig. 5 we compare structure enhanced `aln3nn` alignments with pure sequence alignments (`clustalw`), pair-wise structure enhanced alignments (`STRAL`) and true structural alignments (`MARNA`) as well as the manually curated reference alignments for `Rfam` (v.5.0) (Griffiths-Jones *et al.*, 2005). We use six diverse families of RNA data sets from the `BRaliBase` that have been used in an extensive benchmark study of RNA multiple alignment algorithms (Gardner *et al.*, 2005): Group II introns, 5S rRNA, tRNA, U5 spliceosomal RNA, miRNA and snoRNA. For each family approximately 100 alignments had to be carried out, where each of them contain five sequences encompassing a range of sequence distances. As in (Gardner *et al.*, 2005), we used the structure conservation index (SCI) (Washietl *et al.*, 2005) to assess the quality of the calculated alignments. The SCI is defined as the ratio of consensus folding energy of a set of aligned sequences (calculated using the `RNAalifold` program (Hofacker *et al.*, 2002)) and average unconstrained folding energies of the individual sequences. The SCI is close to 0 for structurally divergent sequences and close to 1 for correctly aligned sequences with a common fold. Values larger one indicate a perfectly RNA structure which is additionally supported by compensatory as well as consistent mutations that preserve the common structure. The benchmark study (Gardner *et al.*, 2005) established that the SCI is an appropriate measure for RNA alignment quality when the sequences are known to have a common fold, since decreased values of the SCI can be attributed to alignment errors.

In all test cases, `aln3nn` produces high quality alignments that are at least competitive with the other methods using a fixed tradeoff
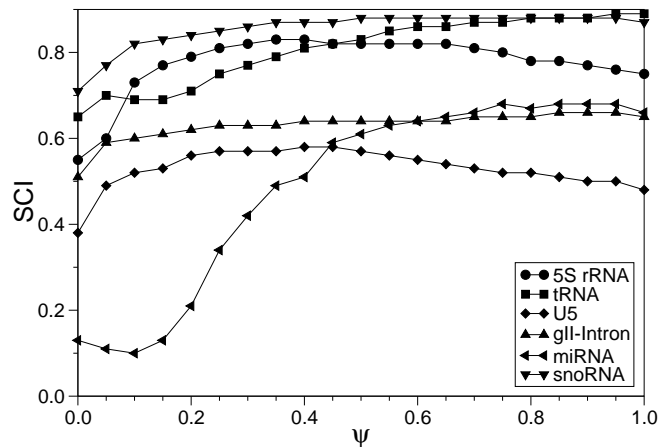
between sequence and structure scores of $\psi = 0.5$. Interestingly, `aln3nn` achieves significantly higher SCI values than even the reference alignment on the 5S rRNA data set.

Not surprisingly, the performance of structured enhanced alignments depends on the proper weighting of sequence and structure information. Figure 6 shows the influence of the parameter ψ on the SCI values for the given RNA sequences. As expected the SCI decreases if structural information is completely ignored ($\psi = 1$). On the other hand, ignoring the sequence information ($\psi = 0$) yields even worse results. The reason is that RNA secondary structure prediction has limited accuracy so that alignments based on predicted structures for individual sequences are based on very noisy data (Bonhoeffer *et al.*, 1993; Hofacker *et al.*, 2002). The impact of the ψ parameter varies between different RNA families. While alignments of group II introns and U5 spliceosomal RNAs are fairly robust against variations in ψ, we observe large variations for miRNA and 5S rRNA alignments.

3.4 Gap removals

The possibility to correct gaps that are introduced at early stages was a major motivation for developing `aln3nn`. We therefore investigated to what extent the algorithm actually utilizes this feature.

Table 1. Mean frequency f and standard deviation σ_f of correctly removed gap columns from the intermediate alignments after the division process.

RNA family	f	σ_f
Group II Intron	0.040	0.501
miRNA	0.152	1.445
5S rRNA	0.265	1.314
snoRNA	0.131	0.710
tRNA	0.197	2.068
U5	0.083	0.584

Table 1 shows the frequency f of gaps that are removed at intermediate division steps and that are not re-introduced at later stages. We find that in some data sets one fourth of the gaps in the early stages of the progressive alignment are later removed again. This observation emphasizes the fact that the "once a gap, always a gap" property of pair-wise progressive alignment algorithms is a major shortcoming.

4 DISCUSSION AND OUTLOOK

We have presented here a novel progressive alignment tool, `aln3nn`, that uses exact dynamic programming to construct three-way alignments of sequences and profiles and that uses a three-to-two aggregation procedure in the spirit of Neighbor-Net. A direct comparison of exact three-way alignments with progressive alignments of the same three sequences shows that the progressive approach leads to significantly suboptimal scores. The discrepancy increases with sequence diversity and in/del probability. While incurring significant additional computational costs compared to pair-wise, guide-tree based, approaches, `aln3nn` achieves competitive alignment accuracies on both protein and nucleic acid data on `BAliBASE` and `BRaliBase` benchmark data set. The software furthermore provides an option to compute structure enhanced RNA alignments.

Programs such as `clustalw` employ a variety of heuristic rules that introduce local modifications of the scoring scheme to (partially) compensate for problematic sections of intermediate alignments. In contrast, `aln3nn` achieves this encouraging performance without any heuristic modifications of the scoring schemes. This indicates that three-way alignments and the more sophisticated aggregation steps provide a significant advantage of pair-wise methods. In particular, we observe that the three-to-two aggregation step, with its division procedure, removed up to one fourth of the previously introduced gap characters, emphasizing that the inability to correct misplaced gaps is major shortcoming of traditional progressive alignment algorithms.

In its present implementation, `aln3nn` demonstrates that progressive alignment schemes can produce competitive high quality alignments even without sophisticated scoring functions. This leaves ample room for future improvements. In particular, one might want to include gap penalties that depend on local sequence context in particular in the intermediate profile alignment steps. The division-step for the three-way alignments could also be modified in several ways. A possible approach would infer a phylogenetic tree that is is then subdivided at the longest or the most central edge.

REFERENCES

Bonhoeffer, L. S., McCaskill, J. S., Stadler, P. F. & Schuster, P. (1993). RNA multi-structure landscapes. a study based on temperature dependent partition functions. *Eur. Biophys. J.*, **22**, 13–24.

Bryant, D. & Moulton, V. (2002). NeighborNet: An agglomerative method for the construction of planar phylogenetic networks. In *WABI '02: Proceedings of the Second International Workshop on Algorithms in Bioinformatics*. Springer-Verlag, London, UK, pp. 375–391.

Dalli, D., Wilm, A., Mainz, I. & Steger, G. (2006). STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time. *Bioinformatcs*, **22**, 1593–1599.

Dewey, T. G. (2001). A sequence alignment algorithm with an arbitrary gap penalty function. *J. Comp. Biol.*, **8**, 177–190.

Feng, D. & Doolittle, R. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.

Gardner, P., Wilm, A. & Washietl, S. (2005). A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Research*, **33**, 2433.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *J. Mol. Biol.*, **162**, 705–708.

Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S. & Bateman, A. (2005). Rfam: annotating noc-coding RNAs in complete genomes. *Nucleic Acid Research*, **33**.

Gupta, S., Kececioglu, J. & Schaffer, A. (1995). Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, **2**, 459–462.

Höchsmann, M., Töller, T., Giegerich, R. & Kurtz, S. (2003). Local similarity in RNA secondary structures. In *Proc of the Computational Systems Bioinformatics Conference, Stanford, CA, August 2003 (CSB 2003)*. pp. 159–168.

Hofacker, I., Fekete, M. & Stadler, P. (2002). Secondary structure prediction for aligned RNA sequences. *J. Mol. Evol.*, **319**, 1059–1066.

Hofacker, I., Fontana, W., Stadler, P., Bonhoeffer, L., Tacker, M. & Schuster, P. (1994). Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie*, **125**, 167–188.

Hogeweg, P. & Hesper, B. (1984). The alignment of sets of sequences and the construction of phylogenetic trees. an integrated method. *J. Mol. Evol.*, **20**, 175–186.

Hull Havgaard, J. H., Lyngsø, R., Stormo, G. D. & Gorodkin, J. (2005). Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics*, **21**, 1815–1824.

Konagurthu, A., Whisstock, J. & Stuckey, P. (2004). Progressive multiple alignment using sequence triplet optimization and three-residue exchange costs. *J. Bioinf. and Comp. Biol.*, **2**, 719–745.

Lipman, D., Altschul, S. & Kececioglu, J. (1989). A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences of the United States of America*, **86**, 4412–4415.

Mathews, D. H. & Turner, D. H. (2002). Dynalign: An algorithm for finding secondary structures common to two RNA sequences. *J. Mol. Biol.*, **317**, 191–203.

McCaskill, J. (1990). The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.

Myers, E. & Miller, W. (1988). Optimal alignemnts in linear space. *Bioinformatics*, **4**, 11–17.

Needleman, S. B. & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the aminoacid sequences of two proteins. *J. Mol. Biol.*, **48**, 443–452.

Saitou, N. & Nei, M. (1987). The neighbor-joining method: a new method, for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, **4**, 406–425.

Sankoff, D. (1985). Simultaneous solution of the RNA folding, alignment, and proto-sequence problems. *SIAM J. Appl. Math.*, **45**, 810–825.

Siebert, S. & Backofen, R. (2005). MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, **21**, 3352–3359.

Sokal, R. R. & Michner, C. D. (1958). A statistical method for evaluating systematic relationships. *Univ. Kans. Sci. Bull.*, **38**, 1409–1438.

Stoye, J. (1997). Multiple sequence alignment with the divide-and-conquer method. *Gene Combis*, **211**, 45–56.

Stoye, J., Evers, D. & Meyer, F. (1998). Rose: generating sequence families. *Bioinformatics*, **14**, 157–163.

Thompson, J., Higgins, D. & Gibson, T. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, **22**, 4673–4680.

Thompson, J., Plewniak, F. & Poch, O. (1999). BAliBASE: a benchmark alignment databse for the evaluation of multiple alignment programs. *Bioinformatcs*, **15**, 78–88.

Wang, L. & Jiang, T. (1994). On the complexity of multiple sequence alignment. *J. Comp. Biol.*, **1**, 337–348.

Washietl, S., Hofacker, I. & Stadler, P. (2005). Fast and reliable prediction of noncoding RNAs. *PNAS*, **102**, 2454–2459.

Will, S., Missal, K., Hofacker, I. L., Stadler, P. F. & Backofen, R. (2006). Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comp. Biol.*. Submitted.