TR-07-07

An Exact Mathematical Programming Approach to Multiple RNA Sequence-Structure Alignment

Markus Bauer, Gunnar W. Klau, and Knut Reinert

March 2007

Freie Universität Berlin

AN EXACT MATHEMATICAL PROGRAMMING APPROACH TO MULTIPLE RNA SEQUENCE-STRUCTURE ALIGNMENT

MARKUS BAUER, GUNNAR W. KLAU, AND KNUT REINERT

Abstract. One of the main tasks in computational biology is the computation of alignments of genomic sequences to reveal their commonalities. In case of DNA or protein sequences, sequence information alone is usually sufficient to compute reliable alignments. RNA molecules, however, build spatial conformations—the secondary structure—that are more conserved than the actual sequence. Hence, computing reliable alignments of RNA molecules has to take into account the secondary structure. We present a novel framework for the computation of exact multiple sequence-structure alignments: We give a graph-theoretic representation of the *sequence-structure* alignment problem and phrase it as an *integer linear program.* We identify a class of constraints that make the problem easier to solve and relax the original integer linear program in a *Lagrangian* manner. Experiments on a recently published benchmark show that our algorithms has a comparable performance than more costly dynamic programming algorithms, and outperforms all other approaches in terms of solution quality with an increasing number of input sequences.

1. Motivation

Recent advances in modern molecular biology would have been impossible without the application of sophisticated algorithmic and mathematical modelling techniques. Some of the most eminent examples are the determination of the genomic sequences of human and fruit fly [1, 44] that marked a milestone in modern biology. Besides that, biologists use programs like BLAST [40] as an everyday tool to find similar sequences in large databases.

Advanced combinatorial optimization entered the field around the mid 1990s when Kececioglu introduced the notion of a *maximum trace* [28], and has been extended to various fields in subsequent years [2, 4, 6, 11, 29, 32, 38]. The interested reader is referred to [21] where the authors give a survey on combinatorial optimization problems appearing in computational biology.

Sequence analysis of proteins, RNA, and DNA is still the core application in computational biology. The human genome, for example, can be seen as an approximately three billion character long string over the four-letter DNA alphabet $\Sigma = \{A, G, C, T\}$. The first step in almost every analysis is the computation of an alignment of two sequences in order to detect their commonalities: a *pairwise sequence alignment* of sequences a and b denotes

	global:	local:
`ACGTCGCG`	`-ACGTCGCG`	`CGCG`
`GACCG`	`GAC----CG`	`C-CG`

Figure 1. Given the two input sequences to the left, one possible global alignment (aligning the entire sequences) is shown in the middle, whereas the right-hand side shows one possible local alignment (aligning two subsequences).

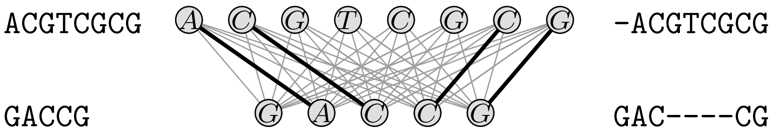


Figure 2. Alignment graphs. Vertices correspond to characters in a sequence, solid lines to alignments of characters: Given the input sequences on the left, we construct a complete bipartite graph. The subset of edges shown in bold represent the alignment on the right side.

an arrangement of a and b such that identical or similar characters are written in one column. This is accomplished by inserting a so called gap character, usually "-", into the sequences. Scores for pairs of symbols express the benefit or penalty for aligning these two symbols. The seminal paper of Needleman and Wunsch described an algorithm to compute an optimal global alignment of two strings [35], which has been subsequently modified to detect locally similar subsequences [42]. Figure 1 shows an illustration for both global and local sequence alignment.

A different way to model sequences and alignments is by weighted graphs: We set the nucleotides as the nodes in the graph, and we insert edges between every node from the first to the second sequence. The edge weights correspond to the score of aligning the first to the second nucleotide. An alignment is then a *non-crossing matching of maximum weight* in a bipartite graph. See Fig. 2 for an illustration.

Although the variety and applications of alignment problems tremendously increased over the years, the core algorithms are largely based on *dynamic programming* (DP). In [38] the authors describe the first graph-theoretical formulation for the NP-hard problem of aligning multiple sequences and solve it exactly using branch-and-cut.

Another important class of molecules in the cell are *RNA*s. In recent years, they have gained more and more attention. Unlike previously thought, RNA molecules perform important catalytic functions in the cell, that is, RNA itself is able to trigger or inhibit functions in the cell [33]: this discovery contradicts the traditional model in molecular biology, where these functional activities have been attributed exclusively to proteins.

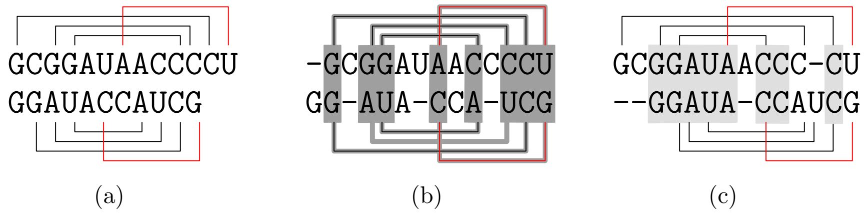


Figure 3. Given two RNA sequences with their corresponding secondary structure (a), alignment that maximizes sequences and structure score (in grey) (b), alignment maximizing sequence score alone (in light grey) (c).

From an algorithmic point of view, the algorithms for DNA still work in case of RNA sequences, the only difference is that the four-letter alphabet Σ contains a U instead of the T, but it has been shown that the sequence alone does not carry all information to compute reliable alignments. An RNA sequence folds back onto itself and forms hydrogen bonds between pairs of (G, C), (A, U), and (G, U). These bonds lead to the distinctive *secondary structures* of an RNA sequence. Figures 4 and 5 show common representations of small toy examples of RNA sequences together with their secondary structure.

In the course of evolution, RNA sequences mutate at a much higher rate than the structure that they are forming, following the *structure-function* paradigm: RNA molecules with different sequences but same or similar secondary structure are likely to belong to the same functional family, in which the secondary structure is conserved by selective pressure. This in turn means that the computation of reliable alignments must take structural information into account. For example Figure 3 shows two possible alignment of two RNA sequences and structures, where the first maximizes the structural similarity and the second maximizes the sequence similarity.

Figure 3 also contains a so called *pseudoknot* depicted by the red line crossing the other lines in the secondary structure. Pseudoknots do occur naturally in some classes of RNA families. Their presence or absence in the corresponding computational models plays an important role for the computational complexity of the corresponding optimization problems. Allowing pseudoknots makes the problems computationally hard [20]. Hence, most approaches assume a pseudoknot-free, *nested* structure as their input. A nested structure can be drawn as an outer-planar graph in its circular representation (see Fig. 5 on the right side for an illustration): Nested structures allow a straightforward decomposition of the entire structure into smaller substructures leading to polynomial time algorithms using *dynamic programming*. In addition it is well known that the multiple alignment problem is NP-hard [45] even without considering secondary structure.

```
GAUAGGCAUGUAGCCUAUCGCC
(((((((.....)))))))...
```

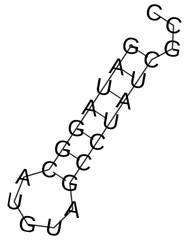


Figure 4. Two ways to depict an RNA sequence and corresponding secondary structure. Left the bracket notation in which pairing brackets indicate base pairs. Right an alternative way to represent the structure using a graph.

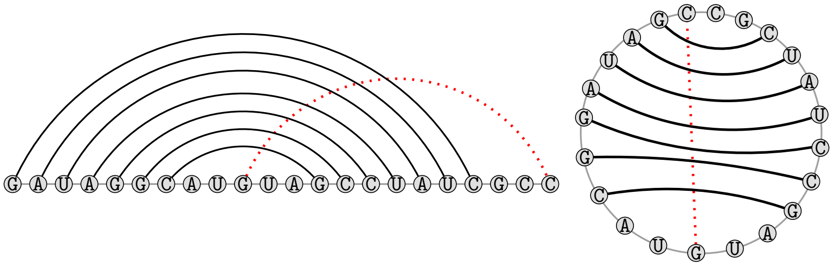


Figure 5. Graph-based representations of RNA structures. The left side shows the standard graph representation, whereas on the right side a circular graph representation is given. Adding the dotted red edge yields a pseudoknot, i.e. crossing base pairs, in the secondary structure.

Subsuming the above introductory discussion, we aim at solving the *sequence-structure alignment* problem: Given two or more RNA sequences, we aim at computing the optimal multiple sequence-structure alignment.

More specifically, let A denote an alignment of the sequences. We define by $s_S(A)$ the sequence score of alignment A, whereas $s_P(A)$ denotes the score of structural features that are realized by the alignment A. We aim at maximizing the combined sequence-structure score, that is an alignment A^* that maximizes $s_S(A^*) + s_P(A^*)$. Figure 3 gives a toy example showing two sequences—with their corresponding secondary structure—and two possible alignments, one maximizing the score of sequence and structure, and the other one maximizing just the sequence score alone. We will elaborate on this in Sect. 3.

2. Previous Work

Sankoff described the first algorithm for the simultaneous alignment of sequence and folded structure in his seminal paper [39]: the original dynamic programming algorithm takes $\mathcal{O}(n^{3k})$ and $\mathcal{O}(n^{2k})$ in time and space, where

k is the number of sequences and n their maximal length. This makes the algorithm applicable only to short sequences even in the pairwise case. Consequently, light-weight implementations were subsequently developed that restricted the original recursions in various ways, like banding [25], or by keeping some aligned positions a priori fixed [16, 24]. Bafna et al. [5] give recursions for the simultaneous alignment of sequence and structure that build the basis for subsequent work [23, 43].

In [47] the authors gave an alternative model for comparing RNA sequences. They view the nested structures as a tree and compute the minimal number of node operations (node substitution, node insertion, and node deletion) to transform one tree into the other. Along these lines the authors of [27] propose an alternative view by introducing the alignment of trees.

The authors of [26] introduce so called *edit operations* on RNA structures to transform on structure into the other. A cost function gives the score for each edit operation: the goal is then to find a series of operations of minimal cost to transform one RNA structure into the other.

Evans presented the model of an *arc-annotated sequence* in [18] and reduces the computation of sequence-structure alignments to the computation of the *longest arc-preserving common subsequence.* The authors of [15] present a novel computational model for aligning multiple RNA structure based on the notion of a *linear* graph.

Reinert et al. [32] gave a different approach for comparing RNA structures: they phrase their graph-based model as an *integer linear program* and solve it afterwards by branch-and-cut. They are able to align RNA sequences with known structure to those of unknown structure by maximizing the sequence and structure score. Their approach allows for pseudoknots and is able to tackle problem instances with a sequence length of approximately 1400 bases. However, for problems of that size their algorithm already needs prohibitive resources. Lancia and coworkers developed a branch-and-cut algorithm [30] that is similar to [32] for the related problem of aligning contact maps. In subsequent work [11] they introduced *Lagrangian relaxation* to the field of computational biology: Their formulation is based on previous work in the field of quadratic programming problems like the Quadratic Knapsack Problem [12] or the Quadratic Assignment Problem [13].

In [6] the authors adapt the Lagrangian relaxation formulation to the problem of aligning two RNA structures: Their implementation yields an algorithm that is an order of magnitude faster than the algorithm from [32] for solving the same instances with respect to the same objective function. Along these lines, [7] describes an initial integer linear programming formulation for solving multiple RNA structures simultaneously. Althaus et al. [3] give a formulation for aligning multiple sequences with arbitrary gap costs which also contains extensive polyhedral studies about facet-defining inequalities.

In this paper we present a graph-based model that unifies the formulations given in [7] and [3] for the simultaneous alignment of multiple RNA structures. Here, we concentrate on a sound description of our mathematical basis, a first formulation for multiple structural RNA alignments including arbitrary gap costs in the graph-based framework, and our algorithmic contribution. In a companion paper, we focus on the application and comparison of our new method to state-of-the art tools [8].

Section 3 describes the graph-based model, in Sect. 4 we give an integer linear programming (ILP) formulation for our model and show how we find (near-)optimal solutions using *Lagrangian relaxation*. Section 5 is a summary of computational results on the recently published benchmark set BRAliBase [46]. We show that with an increasing number of input sequences, our approach outperforms all the traditional DP based algorithms in terms of the quality solution.

3. Graph-theoretic framework

We first give some basic definitions that we use throughout the rest of the paper. Afterwards, we describe our graph-theoretical model, which is based on the formulations given in [6] and [3].

3.1. **Basic Definitions.**

Definition 1. Let Σ be some alphabet excluding the gap character "-", and let $\hat{\Sigma} = \Sigma \cup \{\text{-}\}$. Given a set S of k strings $s^1, \ldots, s^k$ over Σ, we call $A = (\hat{s}^1, \ldots, \hat{s}^k)$ a *multiple alignment* of the sequences in S if and only if the following conditions are satisfied: (a) The sequences $\hat{s}^i$, $1 \leq i \leq k$, are over the alphabet $\hat{\Sigma}$, (b) all sequences $\hat{s}^i$ have the same length $|A|$, (c) sequence $\hat{s}^i$ without "-" corresponds to s^i, for $1 \leq i \leq k$, and (d) there is no index j such that $\hat{s}^i_j =$ "-", $1 \leq i \leq k$. By s^i_j we refer to the jth character in sequence s^i. We define $M_i(j)$ as the mapping of s^i_j to its position in the alignment, and by $M_i^{-1}(j)$ the mapping from the position in the alignment to the actual position in the sequence. If $\hat{s}^i_j \neq$ "-" and $\hat{s}^l_j \neq$ "-", $1 \leq j \leq |A|$, then we say that $s^i_{M_i^{-1}(j)}$ is aligned to $s^l_{M_l^{-1}(j)}$, and to a gap otherwise.

Alphabets commonly used in computational biology are the four letter alphabet $\Sigma = \{A, G, C, T\}$ or $\Sigma = \{A, G, C, U\}$ in case of DNA or RNA sequences, respectively. We define a scoring function $\sigma : \hat{\Sigma} \times \hat{\Sigma} \rightarrow \mathbb{R}$ that represents the benefit of aligning the two characters. Usually, pairs of identical characters receive a high score, whereas different characters get a low score (or even a negative score in case of gap characters).

We can extend the score definition to alignments:

Definition 2. Given a set S of k strings $s^1, \ldots, s^k$, an alignment A consisting of strings $\hat{s}^1, \ldots, \hat{s}^k$, and a scoring function σ, the *sum-of-pairs* (SPS)

```
AAAAAA  AAAAAA  AAAAAA
AAA     A-A-A-  AAA---
  (a)     (b)     (c)
```

FIGURE 6. Given the sequences from (a), a linear gap function would assign the same gap score to the alignment of (b) and (c). The beginning of a gap, however, should be penalized higher compared to subsequent gap characters, and therefore the alignment of (c) is biologically more accurate.

score of A is defined by

$$\mathrm{SPS}(A,\sigma) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{l=1}^{|A|} \sigma(\hat{s}_l^i, \hat{s}_l^j) \ .$$

Intuitively speaking, the sum-of-pairs score adds up all scores of pairs of aligned characters in the alignment A. Usually, we are interested to find an *optimal multiple sequence alignment* under the scoring function σ.

Definition 3. Given a scoring function σ and a set S of sequences, we aim at computing an alignment A^* with

$$\mathrm{SPS}(A^*,\sigma) = \max_{A \in \mathcal{A}} \mathrm{SPS}(A,\sigma) \ ,$$

where $\mathcal{A}$ is the set of all possible multiple alignments for S. We call A^* an *optimal multiple sequence* alignment of S under the scoring function σ.

In this score model gaps are not explicitly modelled and inherently present by the alignment of a gap character to a non-gap character. Hence it is not possible to penalize different numbers of consecutive gaps differently. For example a gap of length three—aligning three 'A's to three '-'—is scored the same as three individual gaps of a single 'A' aligned to a single '-' (see Fig. 6 (b) and (c)).

Unfortunately this is not desirable. Biological findings motivate a different gap model: the begin of a gap should be penalized higher compared to subsequent gap characters. This leads to *affine gap costs* that score a gap of length x by $a + (x-1)b$, where $a > b$ are the *gap open* and *gap extension* penalties. Using this model would clearly favor the single gap (Fig. 6 (c)) over the three individual gaps (Fig. 6 (b)).

Motivated by this discussion we introduce the following score which models gaps explicitly and hence can assign affine gaps costs (or any other gap cost) to the gaps in an alignment. Mind that by using an explicit gap model the scores for aligning a character to a gap character might have to be modified accordingly.

Definition 4. Given a set S of k strings $s^1, \ldots, s^k$, an alignment A consisting of strings $\hat{s}^1, \ldots, \hat{s}^k$, and the set $G(A)$ containing all gaps of alignment

A. Let σ be a sequence scoring function, and a gap penalty function γ, then the *gapped sum-of-pairs* (GSPS) score of A is defined by

$$\mathrm{GSPS}(A, \hat{\sigma}, \gamma) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{l=1}^{|A|} \sigma(\hat{s}_l^i, \hat{s}_l^j) + \sum_{g \in G(A)} \gamma(g) \ .$$

Note that γ assigns negative scores to gaps in the alignments, since we want to penalize the occurrence of gaps in alignments.

As described in Sect. 1 in case of RNA molecules sequence alignments are in general not sufficient enough to build reliable alignments. Therefore, in addition to the gaps, one has to incorporate structural information. This leads to the notion of *annotated sequences*.

Definition 5. Let $s = s_1, \ldots, s_n$ be a sequence of length n over the alphabet $\Sigma = \{A,\ C,\ G,\ U\}$. A pair (s_i, s_j) is called an *interaction* if $i < j$ and nucleotide i interacts with j. In most cases, these pairs will be (G, C), (A, U), or (G, U). The set p of interactions is called the *annotation* of sequence s. Two interactions (s_k, s_l) and (s_m, s_o) are said to be *inconsistent*, if they share one base; they form a *pseudoknot* if they "cross" each other that is if $k < m < l < o$ or $m < k < o < l$. A pair (s, p) is called an *annotated sequence*. Note that a structure where no pair of interactions is inconsistent with each other forms a valid secondary structure of an RNA sequence, possibly with pseudoknots.

Definition 6. Given a sequence alignment $A = (\hat{s}^1, \ldots, \hat{s}^k)$ of k sequences, consider two annotated sequences (s^i, p^i) and (s^j, p^j). We call two interactions $(s_k^i, s_l^i) \in p^i$ and $(s_m^j, s_n^j) \in p^j$ a *structural match* if s_k^i is aligned with s_m^j and s_l^i is aligned with s_n^j. Two structural matches (s_k^i, s_l^i), (s_k^j, s_l^j) and (s_m^i, s_n^i), (s_m^j, s_n^j) are *inconsistent* if $k = m$, $l = m$, $k = n$, or $k = n$. We define a scoring function $\tau : \Sigma^4 \to \mathbb{R}$ that assigns a score to quadruples of characters representing the benefit of matching the two interactions.

In other words, in case of a structural match of two interactions, their "left" and "right" endpoints are aligned by A. Two structural matches are *inconsistent*, if they share an aligned column: In case of RNA sequences, we allow each nucleotide to be paired with at most one other nucleotide, inconsistent matches represent pairings with two or more nucleotides which we do not allow in case of RNA sequences.

This leads to the definition of *sequence-structure alignments* of RNA structures.

Definition 7. Given a set S of k strings $s^1, \ldots, s^k$ and an alignment A consisting of strings $\hat{s}^1, \ldots, \hat{s}^k$. Let $G(A)$ be the set of all gaps of A, and let σ, τ, γ be functions for scoring sequence, structural matches, and gaps. Then the

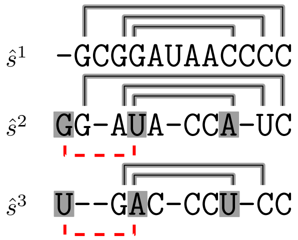

$\hat{s}^1$ -GCGGAUAACCCC
$\hat{s}^2$ GG-AUA-CCA-UC
$\hat{s}^3$ U--GAC-CCU-CC

Figure 7. Realized structural matches are highlighted with grey edges: the structural match $(\hat{s}^2_1, \hat{s}^2_5)$, $(\hat{s}^3_1, \hat{s}^3_5))$ (the red dotted edge) is inconsistent with the structural match $(\hat{s}^2_5, \hat{s}^2_{10})$, $(\hat{s}^3_5, \hat{s}^3_{10})$.

gapped structural sum-of-pairs score of A is defined by $\mathrm{GSSPS}(A, \sigma, \tau, \gamma) =$

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \left(\sum_{l=1}^{|A|} \sigma(\hat{s}^i_l, \hat{s}^j_l) + \sum_{l=1}^{|A|-1} \sum_{m=l+1}^{|A|} \tau(\hat{s}^i_l, \hat{s}^j_l, \hat{s}^i_m, \hat{s}^j_m) \right) + \sum_{g \in G(A)} \gamma(g)$$

where we do not score inconsistent structural features, that is we ensure that every base realizes at most one structural match.

Figure 7 gives an illustration for the definitions given above. Analogously to the optimal sequence alignment problem, we consider the *optimal sequence-structure alignment* of RNA structures:

Definition 8. Given scoring functions σ, τ, and γ for scoring sequence, structural matches and gaps. Let set S of k sequences $s^1, \ldots, s^k$. We aim at computing an alignment A^* with

$$\mathrm{GSSPS}(A^*, \sigma, \tau, \gamma) = \max_{A \in \mathcal{A}} \mathrm{GSSPS}(A, \sigma, \tau, \gamma) \ ,$$

where $\mathcal{A}$ is the set of all possible multiple alignments for S. We call A^* the *optimal multiple sequence-structure* alignment of S.

3.2. Graph-Theoretical Model for Structural RNA Alignment.

Basic Model. We are given a set of k annotated sequences $\{(s^1, p^1), \ldots, (s^k, p^k)\}$ and model the input as a structural graph $G_S = (V, L)$. The set V denotes the vertices of the graph, in this case the bases of the sequences, and we write v^i_j for the jth base of the ith sequence. The set L contains undirected *alignment edges* between vertices of two different input sequences (for sake of better distinction called *lines*). A line $l \in L$ with $l = (v^i_k, v^j_l), i \neq j$ represents the alignment of the k-th character in sequence i with the l-th character in sequence j. The set L^{ij} represents all lines between sequences i and j. We address the *source node* and *target node* of line l by $s(l)$ and $t(l)$, respectively (i.e., for $l = (v^i_k, v^j_l)$ we have $s(l) = v^i_k$ and $t(l) = v^j_l$). The graph G_S is a k-partite graph.

We extend the original graph $G_S = (V, L)$ by the edge set F to model the annotation of the input sequences in our graph. Consequently, we have

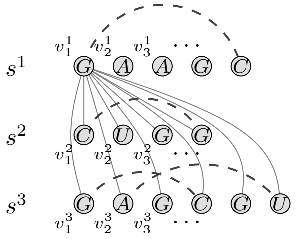


Figure 8. Basic graph model of three annotated sequences containing lines (grey solid lines) and interaction edges (bold dotted edges).

interaction edges between vertices of the same sequence, i.e., an edge (v^i_k, v^i_l) representing the interaction between vertices v^i_k and v^i_l. Figure 8 illustrates these definitions.

Consecutivity and Gap Arcs. In addition to the undirected alignment and interaction edges we augment the graph by the set D of directed arcs representing *consecutivity* of characters within the same string. We have an arc that runs from every vertex to its "right" neighbor, i.e., $D = \{(v^i_j, v^i_{j+1}) \mid 1 \leq i \leq k,\ 1 \leq j < |s^i|\}$.

At this point, gaps are not represented in our graph model. Hence, we introduce the edge set G: for each pair of sequences (i, j) we have an edge a^{ij}_{kl} from v^i_k to v^i_l representing the fact that no character of the substring $s^i_k \cdots s^i_l$ is aligned to any character of the sequence j, whereas s^i_{k-1} (if $k - 1 > 1$) and s^i_{l+1} (if $l + 1 \leq |s^j|$) are aligned with some characters in sequence j. We say that $v^i_k, \ldots, v^i_l$ are *spanned* by the gap arc a^{ij}_{kl}. The entire set G is partitioned into distinct subsets G^{ij} with $i, j = 1, \ldots, k,\ i \neq j$, and $G^{ij} = \{a^{ij}_{lm} \in G \mid 1 \leq l \leq m \leq |s^i|\}$. Intuitively spoken, for each sequence i we have $k - 1$ arcs between each pair of nodes (v^i_k, v^i_l) in order to represent gaps between the actual sequence and the remaining $k - 1$ sequences.

Two gap arcs a^{ij}_{kl}, $a^{ij}_{mn} \in G^{ij}$, w.l.o.g. $k < m$, are *in conflict* with each other if $\{k, \ldots, l + 1\} \cap \{m, \ldots, n\} \neq \emptyset$, that is we do not allow overlapping or even touching gap arcs. This is intuitively clear, because we do not want to split a longer gap into two separate gaps; as a result there has to be at least one aligned character between two realized gap arcs. The set $\mathcal{C}$ codes for the collection of all maximal sets of pairwise conflicting gap arcs. Finally, we define $G^{ij}_{v^i_k \leftrightarrow v^i_l}$ as the set of gap arcs that span the nodes $v^i_k \cdots v^i_l$. See Fig. 9 for an illustration.

Mixed Cycles. A *mixed path* in the graph G_S is an alternating sequence $v_1, e_1, v_2, e_2, \ldots$ of vertices $v_i \in V$ and lines or edges $e_i \in L \cup D$. It is a *mixed path* if it contains at least one arc in D and one line in L. A mixed path is called a *mixed cycle* if the start and end vertex are the same. A mixed cycle represents an ordering conflict of the letters in the sequences. In the two-sequence case a mixed cycle represents lines crossing each other. The

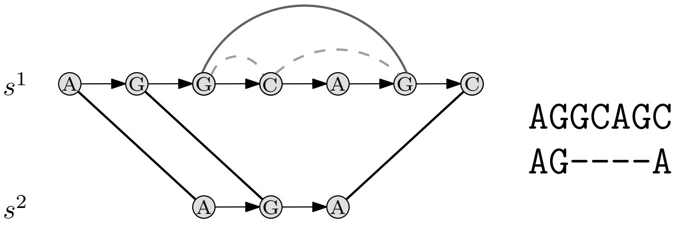


FIGURE 9. A longer gap cannot be split into two shorter gaps: the two dashed gap edges are in conflict with each other and are replaced by the solid gap edge spanning the two shorter gap edges.

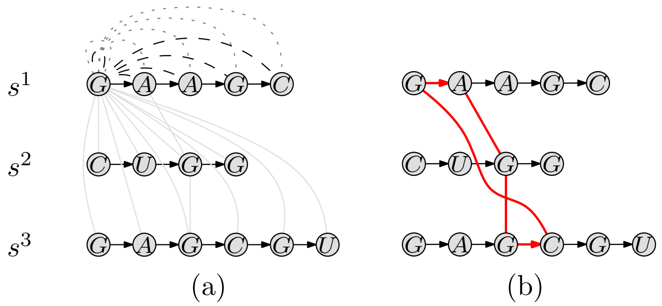


FIGURE 10. (a) Basic graph model augmented by gap edges (interaction edges are not displayed) , (b) showing an instance of a mixed cycle.

set of all mixed cycles is denoted by $\mathcal{M}$. A subset $\mathcal{L} \subseteq L$ corresponds to an *alignment* of the sequences $s^1, \dots s^k$ if $\mathcal{L} \cup A$ does not contain a mixed cycle. In this case, we use the term alignment for $\mathcal{L}$.

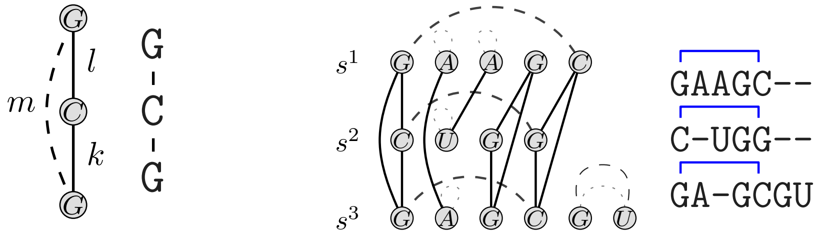
Interaction Match. Two interaction edges $o = (v^i_k, v^i_l) \in p^i$ and $p = (v^j_m, v^j_n) \in p^j$ form an *interaction match* if there exist two lines $e = (v^i_k, v^j_m)$ and $f = (v^i_l, v^j_n)$ such that e and f do not cross each other. A subset $\mathcal{L} \subset L$ *realizes* the interaction match (e, f) if $e, f \in \mathcal{L}$. Observe that the definition of an interaction match is a graph-theoretical reformulation for a structural match as defined in Sect. 3.1. The set I codes all interaction matches of L.

Gapped Structural Trace. A triple $(\mathcal{L}, \mathcal{I}, \mathcal{G})$ with $\mathcal{L} \subseteq L$, $\mathcal{I} \subseteq I$, and $\mathcal{G} \subseteq G$ denotes a valid *gapped structural trace* if and only if the following constraints are satisfied:

(1) For $i, j = 1, \cdots, k, i \neq j$ we define $\mathcal{L}^{ij} = L^{ij} \cap \mathcal{L}$: Then, for $l = 1, \cdots, |s^i|$ the vertex v^i_l is incident to exactly one alignment edge $e \in \mathcal{L}^{ij}$ or spanned by a gap arc $g \in \mathcal{G}^{ij}$.

(2) An alignment edge l can realize at most one single interaction match (l, m).

(3) There is no mixed cycle $M \in \mathcal{M}$ such that $M \cap L = M$.



(a) (b)

FIGURE 11. (a) Transitive edges must also be realized: If k and l are part of the alignment, then m has to be realized as well. (b) Example of a valid gapped structural trace of three annotated sequences. Three interaction matches are conserved by the alignment.

(4) There are no two gaps arcs $a^{ij}_{kl}, a^{ij}_{mn} \in \mathcal{G}$ such that a^{ij}_{kl} is in conflict with a^{ij}_{mn}.

(5) Given $\mathcal{L}$, then we denote by $H(\mathcal{L})$ the convex hull of $\mathcal{L}$. Then

$$H(\mathcal{L}) = \mathcal{L}$$

must hold true. This makes sure that alignment $\mathcal{L}$ also realizes all *transitive* edges induced by $\mathcal{L}$: See Fig. 11(a) for an illustration.

See Fig. 11(b) for an illustration of a gapped structural trace.

We assign positive weights w_l and w_{ij} to each line l and each interaction match (i, j), respectively, representing the benefit of realizing the line or the match. Although we are able to set each weight independently, line weights are usually given by empirically derived mutation score matrices where $\sigma(s^i_k, s^j_l)$ gives a high value in case of identical (or similar) characters. We assign scores to interaction edges by calculating *base pair probabilities* [34]. The base pair probability $\mathrm{bpp}(v^i_k, v^i_l)$ gives the probability that nucleotides s^i_k and s^i_l fold onto each other. To use the probabilities in an additive scoring scheme, we have to transform the probabilities logarithmically, i.e. the actual score p^i_{kl} for an interaction between s^i_k and s^i_l is given by

$$p^i_{kl} = \lg\left(\frac{\mathrm{bpp}(v^i_k, v^i_l)}{p_{\min}}\right)$$

where lg and $p_{\min}$ are the natural logarithm and the minimal probability that we consider. The weight $w_{\hat{i}\hat{j}}$ for an interaction match of lines $\hat{i} = (v^i_k, v^j_m)$ and $\hat{j} = (v^i_l, v^j_n)$ is then given by $w_{\hat{i}\hat{j}} = p^i_{kl} + p^j_{mn}$, i.e. the sum of the scores of the realized interaction edges.

Note that since each interaction edge occurs in two interaction matches (m, l) and (l, m) we divide the weight of these edges by two. Finally, we assign negative weights to gap edges a^{ij}_{kl} representing the gap penalty for aligning substring $s^i_k, \cdots, s^i_l$ with gap characters in sequence j.

3.3. **Complexity.** Jiang and Wang showed [45] that computing an optimal multiple sequence alignment is NP-hard. Along these lines, the authors of [17] prove that the problem remains NP-hard for different scoring functions.

The complexity of sequence-structure alignments depend on the input of the problem and on the actual model one is using: pairwise sequence-structure alignments of RNA structures—as defined in Sect. 3.1— where pseudoknots are not allowed can be solved in polynomial time [5]. The authors of [20] show that computing the *maximal contact map overlap*—a similar problem to RNA structures—is NP-hard in the pairwise case. As a byproduct they state that the computation of the maximal contact map overlap, where every node has a maximum degree of 1, is already NP-hard. Unfortunately, this problem corresponds exactly to the sequence-structure alignment of RNA structures in our model. Hence, computing sequence-structure alignments of two RNA structures of arbitrary structure, i.e. with pseudoknots, is already NP-hard in the pairwise case.

In [18] Evans gave an NP-hardness prove for the computation of the longest arc-preserving common subsequence. Along these lines, Blin and coworkers give several NP-completeness proofs [9, 10] for variants of the arc-annotated sequence model.

Computing sequence-structure alignments in the general edit-model of [26] turns out to be MAXSNP-hard, even if we do not allow crossing interactions. If one limits the number of edit-operations by choosing appropriate costs per edit operations, the authors give polynomial time algorithms based on dynamic programming.

4. Integer Linear Program and Lagrangian Relaxation

4.1. **Integer Linear Program.** Given the graph-theoretical model it is straightforward to transform it to an *integer linear program* (ILP). We associate binary variables with each line, interaction match, and gap edge, and model the constraints of a valid gapped structural trace by adding constraints to the linear program.

The handling of lines and gap edges is straightforward: We associate a x and z variable to each line and gap edge, respectively. We set $x_l = 1$ if and only if line $l \in L$ is part of the alignment $\mathcal{L}$, and $z_a = 1$ if and only if gap edge $a \in G$ is realized.

Interaction matches, however, are treated slightly differently: Instead of assigning an ILP variable to each interaction edge, we split an interaction match (l, m) into two separate *directed interaction matches* (l, m) and (m, l) that are detached from each other. A directed interaction match (l, m) is *realized* by the alignment $\mathcal{L}$ if $l \in \mathcal{L}$. We then have $y_{lm} = 1$ if and only if the directed interaction match (l, m) is realized (note again that y_{lm} and y_{ml} are distinct variables). Figure 12 gives an illustration of the variable splitting. Note that this does not change the underlying model, it just makes the ILP formulation more convenient for further processing.

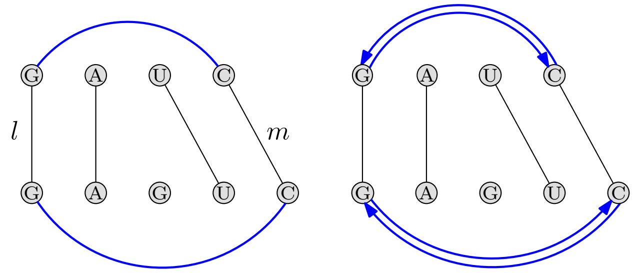


FIGURE 12. One interaction match is split into two *directed* interaction matches.

Splitting interaction matches has first been proposed by Caprara and Lancia in the context of contact map overlap [11], whereas the process of splitting variables has already proven useful in the context of the Quadratic Knapsack Problem [12].

(1)

$$\max \sum_{l \in L} w_l x_l + \sum_{g \in G} w_g z_g + \sum_{l \in L} \sum_{m \in L} w_{lm} y_{lm}$$

(2)

$$\text{s. t.} \sum_{l \in L \cap M} x_l \leq |L \cap M| - 1 \qquad \forall M \in \mathcal{M}$$

(3)

$$x_l + x_k - x_m \leq 1 \qquad \forall (l,k,m) \in L, \quad (x_l, x_k, x_m) \text{ forming a cycle}$$

(4)

$$\sum_{a \in C} z_a \leq 1 \qquad \forall C \in \mathcal{C}$$

(5)

$$\sum_{l \in L^{ij}} x_l + \sum_{a \in G^{ij}_{s(l) \leftrightarrow s(l)}} z_a = 1 \qquad 1 \leq i,j \leq k, i \neq j$$

(6)

$$\sum_{l,m \in L} y_{lm} \leq x_l \qquad \forall\, l \in L$$

(7)

$$y_{lm} = y_{ml} \qquad \forall\, l,m \in L$$

(8)

$$x, y, z \in \{0, 1\}$$

Definition 9. We call the ILP containing (1)—(8) the *master ILP*.

Note that we set the weights w_l, w_g, and w_{lm} for $l,m \in L$ and $g \in G$ as described in Sect. 3.2, and therefore we have $w_g < 0, g \in G$.

Lemma 4.1. *A feasible solution to the ILP* (1)–(8) *corresponds to a valid gapped structural trace and vice versa.*

Proof. We first prove that a feasible solution $(\hat{x}, \hat{y}, \hat{z})$ of the ILP describes a valid multiple gapped structural trace.

Let $\hat{\mathcal{L}} = \{l \in L \mid \hat{x}_l = 1\}$. Observe that constraints (2) guarantee that $\hat{\mathcal{L}}$ does not contain mixed cycles. If $\hat{\mathcal{L}}$ generated a mixed cycle M, then $|\hat{\mathcal{L}} \cap M| = |M|$. But this would contradict (2) that $\sum_{l \in \hat{\mathcal{L}} \cap M} x_l \leq |\hat{\mathcal{L}} \cap M| - 1$. Furthermore, there cannot be lines $k, l \in \hat{\mathcal{L}}$ such that there exists a line m that is induced by k and l, i.e. m is the transitive edge induced by k and l. If this was the case, we have a sum of 2, contradicting constraints (3).

Constraints (4) guarantee that there are no mutually crossing gap edges: Assume there exists two gap edges a_{kl}^{ij} and a_{mn}^{ij} that cross each other. Consequently, they are in the same set $C \in \mathcal{C}$ of conflicting gap edges contradicting that the sum of (4) is constrained by 1.

Equality (5) guarantees that every node is incident to exactly one alignment edge or spanned by exactly one gap edge. If a node was not incident to any line or gap edge, we had a sum of 0. There cannot be any node incident to a line and spanned by a gap edge, because this implies a sum of 2.

Finally, a line cannot realize more than one directed interaction match, otherwise this violates constraints (6).

To complete the proof, we have to show that a valid gapped structural trace represents a feasible solution to the ILP. Given $(\mathcal{L}, \mathcal{I}, \mathcal{G})$ with $\mathcal{L} \subseteq L$, $\mathcal{I} \subseteq I$, and $\mathcal{G} \subseteq G$ that form a valid multiple gapped structural trace. Set the values of the $\hat{x}$, $\hat{y}$, and $\hat{z}$ variables in correspondence if the respective edges are part of $\mathcal{L}, \mathcal{I}$, or $\mathcal{G}$. □

Definition 10. We call the relaxed ILP consisting of (1)—(8) without (7) the *slave ILP*.

Lemma 4.2. *The slave ILP is equivalent to the multiple sequence alignment problem with arbitrary gap costs.*

Proof. The key observation is that after the removal of constraints (7), variables y_{lm} appear only in constraints (6), each variable x_l associated with a set of y_{lm}, the set of outgoing interaction matches that l can realize.

Hence, we have to distinguish two cases, depending on whether a line l is part of an alignment or not. First, assume $x_l = 0$. In this case, as a consequence of (6), all y_{lm} must be zero as well, and due to (5) there has to be a $g_a \in G^{ij}$ with $g_a = 1$ (remember that a vertex is either incident to an alignment edge or to a gap arc). Hence, the contribution of line l to the objective function is less then zero.

If, however, a line $l = (v_k^i, v_l^j)$ is part of an alignment, its maximal contribution to the score is given by solving

$$p_l := \max\ w_l + \sum_{m \in L} w_{lm} y_{lm} + \sum_{a \in G^{ij}_{s(l) \leftrightarrow s(l)}, G^{ji}_{t(l) \leftrightarrow t(l)}} w_a z_a \tag{9}$$

$$\text{s. t. } \sum_{m \in L} y_{lm} \leq 1 \tag{10}$$

$$\sum_{a \in G^{ij}_{s(l) \leftrightarrow s(l)}, G^{ji}_{t(l) \leftrightarrow t(l)}} z_a = 0 \tag{11}$$

$$x, y, z \in \{0, 1\} \tag{12}$$

Inequality (10) states that we can choose only one single interaction match. According to the objective function (9) it is clear that this will be the one with the largest weight w_{lm}. Furthermore, there cannot be a gap arc that spans vertex v^i_k or v^j_l, since otherwise constraints (11) would be violated. This ILP (for each line l) is easily solvable by just selecting the most profitable outgoing interaction match $(l, \hat{m})$ such that l and $\hat{m}$, which can be done in linear time. Therefore, the profit a line can possibly achieve is solely computed by considering the weight of line l and the best directed interaction match $(l, \hat{m})$ that line l can realized, *i.e.* $p_l = w_l + w_{l\hat{m}}$.

In the second step, we compute the optimal overall profit by solving the ILP consisting of the remaining constraints:

$$\begin{aligned}
\max\ & \sum_{l \in L} p_l x_l + \sum_{g \in G} w_g z_g \\
\text{s. t. } & \sum_{l \in L \cap M} x_l \leq |L \cap M| - 1 && \forall M \in \mathcal{M} \\
& x_l + x_k - x_m \leq 1 && \forall (l, k, m) \in L, \quad (x_l, x_k, x_m) \text{ forming a cycle} \\
& \sum_{a \in C} z_a \leq 1 && \forall C \in \mathcal{C} \\
& \sum_{l \in L^{ij}} x_l + \sum_{a \in G^{ij}_{s(l) \leftrightarrow s(l)}} z_a = 1 && \forall i, j \in 1, \cdots, k, i \neq j \\
& x, z \in \{0, 1\}
\end{aligned}$$

The remaining ILP only considers x and z variables, because due to the case distinction described above the values of the y variables depend on the value of the corresponding x variables. Then, the remaining constraints corresponds to the *multiple sequence alignment* formulation given in [3].

Let (x^*, z^*) be the solution to this problem. We claim that an optimal solution of the relaxed problem is given by (x^*, y^*, z^*) by setting $y^*_{lm} = x^*_m y_{l\hat{m}}$ (remember that $y_{l\hat{m}}$ is the highest scoring directed interaction match that l can realized), and by setting the x and z variables according to the solution of the multiple sequence alignment problem. First, it is easy to see that (x^*, y^*, z^*) is indeed a feasible solution of the relaxed problem, since (x^*, z^*) represent a valid alignment (with arbitrary gap costs) and our choice of y^* does not violate the restrictions given in (6). To see that (x^*, y^*, z^*) is optimal, observe that its value is given by

$$\begin{aligned}\sum_{l\in L} p_l x_l^* + \sum_{g\in G} w_g z_g^* &= \sum_{l\in L}(w_l + w_{l\hat{m}})x_l^* + \sum_{g\in G} w_g z_g^* \\ &= \underbrace{\sum_{l\in L} w_l x_l^* + \sum_{g\in G} w_g z_g^*}_{\text{optimal solution for MSA}} + \underbrace{\sum_{l\in L}\sum_{m\in L} w_{lm} y_{lm}^*}_{\text{optimal solution for } y_{l\hat{m}} \text{ due to (9)—(12)}}\end{aligned}$$

For the sake of contradiction, assume that there exists a valid solution $(\bar{x}^*, \bar{y}^*, \bar{z}^*)$ that has a higher objective function value than (x^*, y^*, z^*). (x^*, z^*) and $(\bar{x}^*, \bar{z}^*)$ differ in at least one position, and both form valid alignments (we have to consider only x and z variables, because the values of y follow from the choice of x). If, however, $(\bar{x}^*, \bar{z}^*)$ forms a valid sequence alignment, we would have found it in the first place, because we are computing *optimal* multiple sequence alignments.

□

4.2. **Lagrangian Relaxation.** Obviously we have not yet solved the master ILP, since we dropped equalities (7). Instead of just dropping them, we relax the master ILP in a *Lagrangian* fashion: We move the dropped constraints into the objective function and assign a penalty term—the *Lagrangian multiplier*—to each dropped constraint. The multipliers represent a penalty to objective function in case the dropped constraint is not satisfied.

Moving constraints (7) into the objective function yields the *Lagrangian dual*

$$\max \sum_{l\in L} w_l x_l + \sum_{g\in G} w_g z_g + \sum_{l\in L}\sum_{m\in L} w_{lm} y_{lm} + \sum_{l\in L}\sum_{m\in L} \lambda_{lm}(y_{lm} - y_{ml})$$

which can then be reformulated to

$$\max \sum_{l\in L} w_l x_l + \sum_{g\in G} w_g z_g + \sum_{l\in L}\sum_{m\in L} (w_{lm} + \lambda_{lm}) y_{lm}$$

Note that, according to Lemma 4.2, we can solve instances of the Lagrangian problem by solving a multiple sequence alignment problem with arbitrary gap costs where the profits of the interaction matches are coded in the weights of the lines.

The task is now to find Lagrangian multipliers that provide the best bound to the original problem. We do this by employing iterative *subgradient optimization* as proposed by Held and Karp [22]. This method determines the multipliers of the current iteration by adapting the values from the previous iteration.

More formally, we set $\lambda^1_{lm} = 0, \forall m, l \in L$ and

$$\lambda^{i+1}_{lm} = \begin{cases} \lambda^i_{lm} & \text{if } s^i_{lm} = 0 \\ \lambda^i_{lm} - \gamma_i & \text{if } s^i_{lm} = 1 \\ \lambda^i_{lm} + \gamma_i & \text{if } s^i_{lm} = -1 \end{cases}$$

$$\text{where} \qquad s^i_{lm} = y^*_{lm} - y^*_{ml} \quad \text{and} \quad \gamma_i = \mu \frac{v_U - v_L}{\sum\limits_{l,m \in L} (s^i_{lm})^2} \quad .$$

Here, μ is a common adaption parameter and v_U and v_L denote the best upper and lower bounds, respectively. A fundamental result [37] states that for $\lim\limits_{i \to \infty} \gamma_i = 0$ and $\sum_{i=1}^{\infty} \gamma_i = \infty$ the value of v_U always converges to the optimal value of the Lagrangian dual.

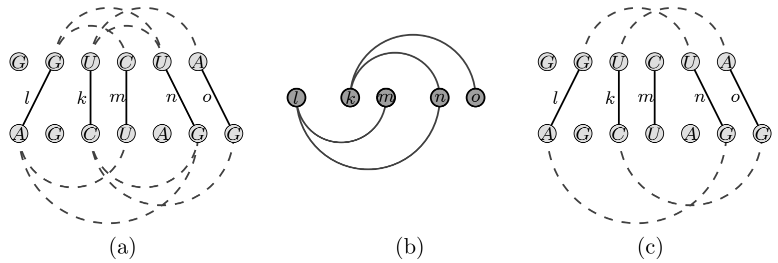
In each iteration of the subgradient optimization procedure we get a value for the Lagrangian dual. Given this series $(v^1, v^2, \dots, v^n)$ we can set v_U to $\min\{v^i \mid 1 \leq i \leq n\}$, the lowest objective function value of the Lagrangian dual solved so far. To obtain a high lower bound is more involved and we show in Sect. 4.3 how to use the information computed in the Lagrangian problem in order to deduce a good feasible solution.

In our computational experiments we also tried more advanced methods to solve the Lagrangian dual, for example *bundle methods* [31]. However, currently the described subgradient optimization yields better bounds than bundle methods.

Note that unless the lower and the upper bound, v_L and v_U coincide, we have not found the provable optimal solution. Even if we had already found the optimal value v^* of the Lagrangian dual, the solution corresponding to v^* is not necessarily a valid solution in the primal problem. Our experiments, however, show that in case of instances that share medium or high structural similarity, the lower and upper bound often coincide yielding provable optimal solution for our original problem. If, however, the two bounds do not match, an incorporation of the Lagrange bounds into a branch-and-bound framework is straightforward.

4.3. **Computing a Feasible Solution.** A solution (x^*, y^*, z^*) of the Lagrangian dual yields a multiple alignment $\mathcal{L}$ (represented by x^*) plus some information about interaction matches coded by the y^*-values; see Figure 13 (a). If for all lines l and m the equation $y^*_{lm} = y^*_{ml}$ holds, then the solution is a feasible multiple structural alignment, and we have found the optimal solution to the original problem. Otherwise, some pairs y^*_{lm} and y^*_{ml} contradict each other. For a valid secondary structure, however, we have to ensure that $y^*_{lm} = y^*_{ml}$ for all pairs of $l, m \in L$.

The set of lines and gap edges that constitute the alignment is fixed: the problem is is to find a subset $\hat{I}$ of interaction edges of maximum weight



(a) (b) (c)

FIGURE 13. Given the alignment $\mathcal{L} = (l, k, m, n, o)$, we have different possibilities to augment the alignment with structural matches. Creating an interaction matching graph (b) and calculating a general matching of maximum weight yields the structural completion of $\mathcal{L}$ (c).

such that the structural information for each sequence is valid, that is, each base is paired with at most one other base. Figure 13 (a) illustrates the problem: Given the alignment $\mathcal{L} = (l, k, m, n, o)$, we have different possibilities to augment $\mathcal{L}$ by structural matches: We can for example either realize the structural match (l, m) or (l, n), but not both. Realizing both interaction matches yields an invalid secondary structure. We therefore define the problem of finding the *structural completion* of an alignment $\mathcal{L}$.

Definition 11. Given an alignment $\mathcal{L}$ and a set $\mathcal{I}$ of interaction matches that $\mathcal{L}$ realizes. Find a subset $\hat{\mathcal{I}} \subseteq \mathcal{I}$ such that $\hat{\mathcal{I}}$ forms a valid secondary structure of maximal weight on $\mathcal{L}$. We call $\hat{\mathcal{I}}$ the *structural completion* of $\mathcal{L}$.

We can formulate this problem as a *general weighted matching problem* in an auxiliary graph M_S, the *interaction matching graph*: $M_S = (V, E)$ where the set V and E constitute vertices and edges, respectively. We have $V = (\hat{v}_1, \ldots, \hat{v}_{|\mathcal{L}|})$ where $\hat{v}_i$ corresponds to the ith element of $\mathcal{L}$. We insert an edge $e_i = (\hat{v}_i, \hat{v}_j)$ if and only there exists a pair of interaction edges (v^i_k, v^i_l) and (v^j_m, v^j_n) whose endpoints are adjacent to a pair $(o, p) \in \mathcal{L} \times \mathcal{L}$ (see Fig. 13 (b)). The weight of edge e_i is given by the weight of the two interaction edges (v^i_k, v^i_l) and (v^j_m, v^j_n).

Lemma 4.3. *A matching of maximum weight in the interaction matching graph M_S corresponds to the structural completion of $\mathcal{L}$.*

Proof. The equivalence follows directly from the construction of M_S and the definition of a matching. □

5. Computational Results

Note that, according to Lemma 4.2, solving an instance of the Lagrangian problem corresponds to the computation of an exact multiple sequence alignment problem with arbitrary gap costs. Although the problem is NP-hard, the branch-and-cut algorithm of [3] is able to solve medium-sized instances within reasonable time. Our experiments, however, showed that with an increasing number of iterations, the average computation time per instance grows significantly due to the adaption of the Lagrangian multiplier in the relaxed problem.

We therefore constrained ourselves to the computation of sequence-structure alignments of two sequences, because pairwise sequence alignments can be computed in $\mathcal{O}(nm)$—with n and m being the sequence lengths—independent from the values of the Lagrangian multipliers. In practice, the length of RNA sequences rarely exceeds 1500 nucleotides, yielding fast computations of a pairwise sequence alignment.

Furthermore, for the fast computation of multiple sequence-structure alignments, we computed all pairwise alignments and used the tool T-COFFEE [36] to heuristically infer a multiple sequence-structure alignment based on the pairwise information. Although this approach does not compute true multiple sum-of-pairs sequence-structure alignments, the performance on real-world instances is very good, as we will show in the following.

We evaluated the performance of our implementation—called Lara (*La*grangian *R*elaxed *A*lignments)—on a recently published benchmark set for RNA sequence-structure alignments called BRAliBase [46]: the benchmark contains approx. 18900 high-quality sequence-structure alignments containing either 2, 3, 5, 7, 10, or 15 input sequences. For the case of two input sequences, we computed sequence-structure alignments as described in Sect. 4, whereas for multiple sequence-structure alignments we resorted to the T-COFFEE approach as described above.

We compared our algorithm to three other sequence-structure alignment programs: FoldalignM [43] which is based on a variant of the Sankoff algorithm, Marna [41], and Stral [14]. These programs have time requirements of $\mathcal{O}(nm\Delta^2)$, $\mathcal{O}(n^2m^2)$, and $O(nm)$, with n and m being the sequence lengths, and Δ being a FoldalignM-specific parameter. Additionally, we took Muscle to compare our alignments with a program that is pure sequence-based.

The authors of [19] showed that structure-based alignment programs produce significantly better alignments compared to sequence-based programs if the sequence similarity drops below approx. $50-60$ percent. For our tests, we therefore excluded all instances that had a pairwise sequence similarity greater than 50 percent.

Figure 14 shows the results of our experiments: the x-axis denotes the pairwise sequence similarity of the input instances, whereas the y-axis gives the Compalign score of the computed alignment: The Compalign score codes the degree of similarity to a given reference alignment as given by the percentage of columns that are identically aligned as in the reference alignment. A value of 1 states that the reference and test alignment are the same, whereas 0 denotes that no column was correctly aligned with respect to the reference alignment. Hence, the higher the value is, the bigger is the similarity of an alignment to the reference alignment.

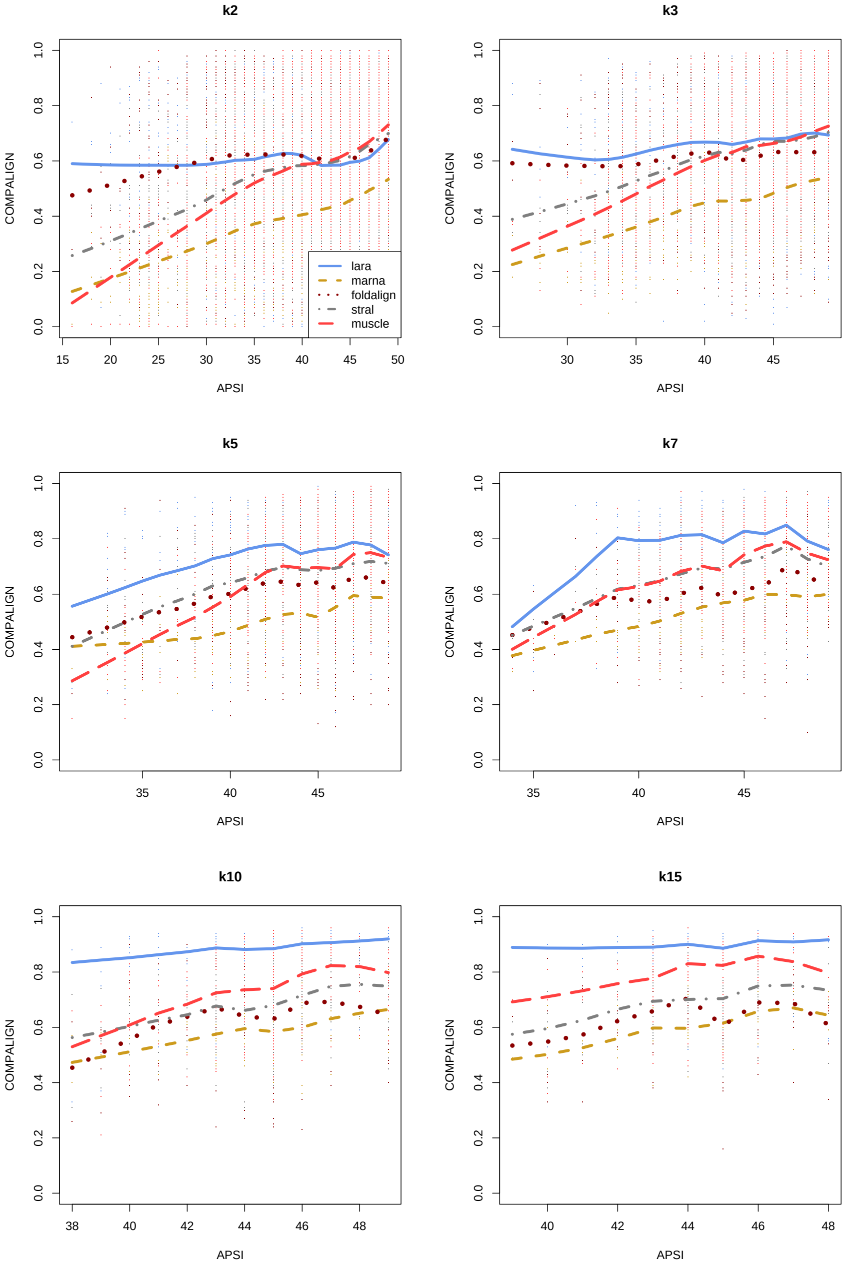


Figure 14. Results of our implementation on instances containing 2 (top left), 3 (top right), 5 (middle left), 7 (middle right), 10 (bottom left), and 15 (bottom right) input sequences. One dot corresponds to one alignment, the lines represent the Lowess function, i.e. they give the trend of the computed alignments. A line at 1.0 means that every alignment is identical to the reference alignment: Hence, the closer the line is to 1.0, the better the alignments are on average. The legend from the top left applies to all other plots as well.

As one can see, in the pairwise case Lara and FoldalignM show the same Compalign performance: Lara, however, only needs 86 minutes to compute all 2251 pairwise sequence-structure alignment. On the same input set FoldalignM needs 172 minutes.

With an increasing number of input sequences, however, Lara outperforms all other programs: in case of the 123 instances containing 15 input sequences per instance, Lara yields an average Compalign score of 0.82. The second best sequence-structure alignment tool, Stral, has an average value of 0.69. Surprisingly, sequence-based Muscle achieves an average Compalign score of 0.76. For a detailed analysis the interested reader is referred to [8] for an in-depth analysis of the entire dataset. Furthermore, the paper also discusses the biological soundness of the model presented in Sect. 4.

Acknowledgements

This work has been partly supported by DFG grant KL 1390/2-1. Support from the International Max Planck Research School for Computational Biology and Scientific Computing is gratefully acknowledged.

References

1. Mark D. Adams and al., *The Genome Sequence of Drosophila melanogaster*, Science **287** (2000), no. 5461, 2185–2195.
2. E. Althaus, A. Caprara, H.-P. Lenhof, and Reinert K., *Multiple sequence alignment with arbitrary gap costs: Computing an optimal solution using polyhedral combinatorics*, Bioinformatics **18** (2002), no. 90002, S4–S16.
3. Ernst Althaus, Alberto Caprara, Hans-Peter Lenhof, and Knut Reinert, *A branch-and-cut algorithm for multiple sequence alignment*, Mathematical Programming (2006), no. 105, 387–425.
4. Ernst Althaus, Oliver Kohlbacher, Hans-Peter Lenhof, and Peter Müller, *A combinatorial approach to protein docking with flexible side-chains.*, RECOMB, 2000, pp. 15–24.
5. V. Bafna, S. Muthukrishnan, and R. Ravi, *Computing similarity between RNA strings*, Proc. of CPM'95, LNCS, no. 937, Springer, 1995, pp. 1–16.
6. M. Bauer and G. W. Klau, *Structural Alignment of Two RNA Sequences with Lagrangian Relaxation*, Proc. of ISAAC'04, LNCS, no. 3341, Springer, 2004, pp. 113–123.
7. M. Bauer, G. W. Klau, and K. Reinert, *Multiple structural RNA alignment with Lagrangian relaxation*, Proc. WABI'05 (R. Casadio and G. Myers, eds.), LNBI, vol. 3692, 2005, pp. 303–314.
8. Markus Bauer, Gunnar W. Klau, and Knut Reinert, *Accurate Multiple Sequence-Structure Alignment of RNA Sequences Using Combinatorial Optimization*, Tech. Report TR-B-07-06, Dept. of Mathematics and Computer Science, Free University Berlin, 2007, Submitted to BMC Bioinformatics.
9. Guillaume Blin, Guillaume Fertin, Romeo Rizzi, and Stéphane Vialette, *What Makes the Arc-Preserving Subsequence Problem Hard?*, T. Comp. Sys. Biology **2** (2005), 1–36.
10. Guillaume Blin and Hélène Touzet, *How to compare arc-annotated sequences: The alignment hierarchy.*, SPIRE, 2006, pp. 291–303.
11. A. Caprara and G. Lancia, *Structural Alignment of Large-Size Proteins via Lagrangian Relaxation*, Proc. of RECOMB'02, ACM Press, 2002, pp. 100–108.

12. Alberto Caprara, David Pisinger, and Paolo Toth, *Exact solution of the quadratic knapsack problem*, INFORMS J. on Computing **11** (1999), no. 2, 125–137.
13. P. Carraresi and F. Malucelli, *A reformulation scheme and new lower bounds for the quadratic assignment problem*, Quadratic Assignment and Related Topics, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pp. 147–160.
14. Deniz Dalli, Andreas Wilm, Indra Mainz, and Gerhard Steger, *STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time*, Bioinformatics **22** (2006), no. 13, 1593–1599.
15. Eugene Davydov and Serafim Batzoglou, *A computational model for RNA multiple structural alignment*, Theor. Comput. Sci. **368** (2006), no. 3, 205–216.
16. Robin Dowell and Sean Eddy, *Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints*, BMC Bioinformatics **7** (2006), no. 1, 400.
17. Isaac Elias, *Settling the intractability of multiple alignment.*, J Comput Biol **13** (2006), no. 7, 1323–1339.
18. P. Evans, *Finding common subsequences with arcs and pseudoknots*, Proc. of CPM'99, LNCS, no. 1645, Springer, 1999, pp. 270–280.
19. P. Gardner, A. Wilm, and S. Washietl, *A benchmark of multiple sequence alignment programs upon structural RNAs*, Nucl. Acids Res. **33** (2005), no. 8, 2433–2439.
20. Deborah Goldman, Sorin Istrail, and Christos H. Papadimitriou, *Algorithmic aspects of protein structure similarity.*, FOCS, 1999, pp. 512–522.
21. Harvey J. Greenberg, William E. Hart, and Giuseppe Lancia, *Opportunities for combinatorial optimization in computational biology*, INFORMS J. on Computing **16** (2004), no. 3, 211–231.
22. M. Held and R.M. Karp, *The traveling-salesman problem and minimum spanning trees: Part II*, Mathematical Programming **1** (1971), 6–25.
23. I. L. Hofacker, S. H. F. Bernhart, and P. F. Stadler, *Alignment of RNA base pairing probability matrices*, Bioinformatics **20** (2004), 2222–2227.
24. I. Holmes, *Accelerated probabilistic inference of RNA structure evolution*, BMC Bioinformatics **5** (2004), 73.
25. J. Hull Havgaard, R. Lyngsø, G. Stormo, and J. Gorodkin, *Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%*, Bioinformatics **21** (2005), 1815–1824.
26. T. Jiang, G.-H. Lin, B. Ma, and K. Zhang, *A general edit distance between RNA structures*, J. of Computational Biology **9** (2002), 371–388.
27. T. Jiang, J. Wang, and K. Zhang, *Alignment of trees — an alternative to tree edit*, Theor. Comput. Sci. **143** (1995), 137–148.
28. J. Kececioglu, *The maximum weight trace problem in multiple sequence alignment*, Proc. CPM'93, LNCS, vol. 684, 1993, pp. 106–119.
29. Gunnar W. Klau, Sven Rahmann, Alexander Schliep, Martin Vingron, and Knut Reinert, *Optimal robust non-unique probe selection using Integer Linear Programming*, ISMB/ECCB (Supplement of Bioinformatics), 2004, pp. 186–193.
30. G. Lancia, R. Carr, B. Walenz, and S. Istrail, *101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem*, Proc. of the Fifth Annual International Conference on Computational Biology, ACM Press, 2001, pp. 193–202.
31. Claude Lemarechal, *Computational combinatorial optimization, optimal or provably near-optimal solutions*, ch. Lagrangian Relaxation, pp. 112–156, Springer Berlin, 2001.
32. H.-P. Lenhof, K. Reinert, and M. Vingron, *A polyhedral approach to RNA sequence structure alignment*, Journal of Comp. Biology **5** (1998), no. 3, 517–530.
33. J. S. Mattick, *The functional genomics of noncoding RNA*, Science **309** (2005), no. 5740, 1527–1528.
34. John S. McCaskill, *The Equilibrium Partition Function and Base Pair Binding Probabilities for RNA Secondary Structure*, Biopolymers **29** (1990), 1105–1119.

35. S.B. Needleman and C.D. Wunsch, *A general method applicable to the search for similarities in the amino-acid sequence of two proteins*, Journal of Molecular Biology **48** (1970), 443–453.
36. C. Notredame, D. G. Higgins, and J. Heringa, *T-Coffee: A novel method for fast and accurate multiple sequence alignment*, Journal of Molecular Biology (2000).
37. B.T. Poljak, *A general method of solving extremum problems*, Soviet Mathematics Doklady **8** (1967), 593–597.
38. K. Reinert, H.-P. Lenhof, P. Mutzel, K. Mehlhorn, and J. D. Kececioglu, *A branch-and-cut algorithm for multiple sequence alignment.*, RECOMB, 1997, pp. 241–250.
39. D. Sankoff, *Simultaneous solution of the RNA folding, alignment, and protosequence problems*, SIAM J. Appl. Math. **45** (1985), 810–825.
40. Altschul S.F., Gish W., Myers E.W., and Lipman D.J., *Basic local alignment search tool*, Journal of Molecular Biology **215** (1990), no. 3, 403–410.
41. S. Siebert and R. Backofen, *MARNA: Multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons*, Bioinformatics (2005), In press.
42. Temple F. Smith and Michael S. Waterman, *Identification of Common Molecular Subsequences*, Journal of Molecular Biology **147** (1981), 195–197.
43. Elfar Torarinsson, Jakob H. Havgaard, and Jan Gorodkin, *Multiple structural alignment and clustering of RNA sequences*, Bioinformatics (2007), to appear.
44. J. Craig Venter and et al., *The Sequence of the Human Genome*, Science **291** (2001), no. 5507, 1304–1351.
45. L. Wang and T. Jiang, *On the complexity of multiple sequence alignment.*, J Comput Biol **1** (1994), no. 4, 337–348.
46. Andreas Wilm, Indra Mainz, and Gerhard Steger, *An enhanced RNA alignment benchmark for sequence alignment programs*, Algorithms for Molecular Biology **1** (2006), no. 1, 19.
47. K. Zhang and D. Shasha, *Simple fast algorithms for the editing distance between trees and related problems*, SIAM J. Comput. **18** (1989), no. 6, 1245–1262.

Markus Bauer, International Max Planck Research School & Free University Berlin, Dept. of Mathematics and Computer Science, Arnimallee 3, 14195 Berlin, Germany

E-mail address: `mbauer@inf.fu-berlin.de`

Gunnar W. Klau, DFG Research Center Matheon & Free University Berlin, Dept. of Mathematics and Computer Science, Arnimallee 3, 14195 Berlin, Germany

E-mail address: `gunnar@math.fu-berlin.de`

Knut Reinert, Free University Berlin, Dept. of Mathematics and Computer Science, Arnimallee 3, 14195 Berlin, Germany

E-mail address: `reinert@inf.fu-berlin.de`