Algorithmic Operations Research

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

Markus Bauer, Gunnar W. Klau and Knut Reinert

Volume 3, Number 2, Fall 2008

URI: https://id.erudit.org/iderudit/aor3_2art03

See table of contents

Publisher(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (digital)

Explore this journal

Cite this article

Bauer, M., Klau, G. W. & Reinert, K. (2008). An Exact Mathematical Programming Approach to Multiple RNA Sequence-Structure Alignment. *Algorithmic Operations Research*, *3*(2), 130–146.

Article 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, which can be represented by graph-like secondary structures. Often, secondary structures are more conserved than the actual sequence. Hence, computing reliable alignments of RNA molecules should take this additional information into account.

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. We run experiments on data from the RFAM database and compare the performance of our model to heuristically inferred multiple structural alignments. Finally, experiments on a recently published benchmark show that in the pairwise case our algorithm achieves results comparable to those obtained by more costly dynamic programming algorithms while needing less computation time.

Algorithmic Operations Research Vol.3 (2008) 130–146

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

Markus Bauer

International Max Planck Research School & Free University Berlin, Department of Mathematics and Computer Science, Arnimallee 3, 14195 Berlin, Germany

Gunnar W. Klau

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Knut Reinert

Free University Berlin, Department of Mathematics and Computer Science, Takustr. 9, 14195 Berlin, Germany

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, which can be represented by graph-like secondary structures*. Often, secondary structures are more conserved than the actual sequence. Hence, computing reliable alignments of RNA molecules should take this additional information into account.*

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. We run experiments on data from the* RFAM *database and compare the performance of our model to heuristically inferred multiple structural alignments. Finally, experiments on a recently published benchmark show that in the pairwise case our algorithm achieves results comparable to those obtained by more costly dynamic programming algorithms while needing less computation time.*

Key words: RNA sequence structure alignment, mathematical programming

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 [1,49] that marked a milestone in modern biology. Besides that, biologists use programs like BLAST [6] 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* [33], and has been extended to various fields in subsequent years [3, 5, 8, 14, 35, 39, 44]. The interested reader is referred to [24] 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 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 char-

Email: Markus Bauer [mbauer@inf.fu-berlin.de], Gunnar W. Klau [gunnar.klau@cwi.nl], Knut Reinert [reinert@inf.fu-berlin.de].

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

Fig. 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)

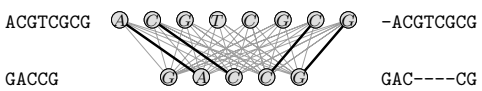

Fig. 2. Alignment graphs. Vertices correspond to characters in a sequence, solid lines to alignments of characters: Given the input sequences on the left, we construct a complete bipartite graph. The subset of edges shown in bold represent the alignment on the right side

acter, 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 [42], which has been subsequently modified to detect locally similar subsequences [47]. 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 [44] 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. Until very recently, the central dogma of molecular biology was that DNA is transcribed into its working copy RNA, and RNA sequences in turn are translated into proteins, the actual functional units in the cell. In the last years, however, it became evident that RNA itself is able to trigger or inhibit important functions in the cell [40], boosting the interest in the study of RNA molecules.

From an algorithmic point of view, the sequence alignment algorithms for DNA still apply in case of RNA sequences, the only difference is that the four-letter alphabet Σ contains a U instead of the T. It has been shown, however, 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 should take structural information into account. Figure 3 shows an example of two possible alignments 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 [23]. 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 based on the principle of *dynamic programming*. In addition it is well known that the multiple alignment problem is NP-hard [50] even without considering secondary structure.

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

More specifically, let A denote an alignment of the sequences. We define by $s_S(A)$ the sequence score

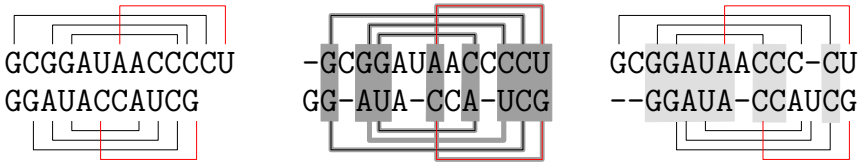

Fig. 3. Given two RNA sequences with their corresponding secondary structure (a), alignment that maximizes sequences and structure score (in grey) (b), alignment maximizing sequence score alone (in light grey) (c)

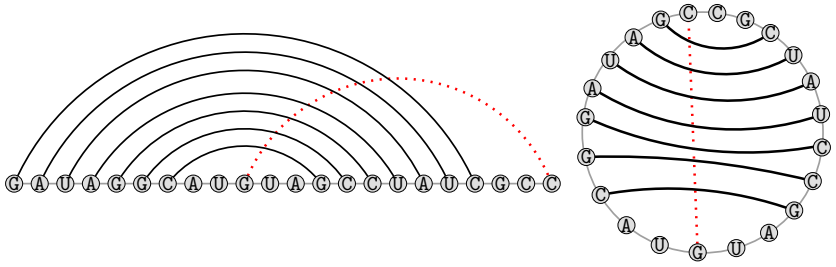

Fig. 5. Graph-based representations of RNA structures. The left side shows the standard graph representation, whereas on the right side a circular graph representation is given. Adding the dotted red edge yields a pseudoknot, i.e., crossing base pairs, in the secondary structure

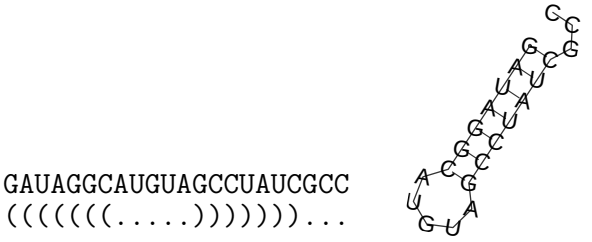

Fig. 4. Two ways to depict an RNA sequence and corresponding secondary structure. Left: the bracket notation in which pairing brackets indicate base pairs. Right: an alternative way to represent the structure using a graph

of alignment A and by $s_P(A)$ the score of structural matches that are realized by the alignment A. We aim at maximizing the combined sequence-structure score, that is, finding an alignment A^* that maximizes $s_S(A^*) + s_P(A^*)$.

2. Previous Work

Sankoff described the first algorithm for the simultaneous alignment of sequence and folded structure in his seminal paper [45]: 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 [30], or by keeping some aligned positions fixed a priori [19, 29]. Bafna et al. [7] give recursions for the simultaneous alignment of sequence and structure that build the basis for subsequent work [28, 48].

In [52] 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 [32] propose an alternative view by introducing the alignment of trees.

Jiang et al. [31] 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 [21] and reduces the computation of sequence-structure alignments to the computation of the *longest arc-preserving common subsequence*. The

authors of [18] present a novel computational model for aligning multiple RNA structures based on the notion of a *linear* graph.

Lenhof et al. [39] gave a different approach for comparing RNA structures: they phrase their graph-based model as an *integer linear program*, which they solve using the *branch-and-cut* principle. They 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 many problem instances of that size their algorithm already needs prohibitive resources. Lancia and coworkers developed a branch-and-cut algorithm [36] that is similar to [39] for the related problem of aligning contact maps. In subsequent work [14] Lancia and Caprara 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 [15] or the Quadratic Assignment Problem [16].

In [8] 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 [39] for solving the same instances with respect to the same objective function. Along these lines, [10] describes an initial integer linear programming formulation for solving multiple RNA structures simultaneously. Althaus et al. [4] 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 [10] and [4] 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 heuristic application and comparison of our new method to state-of-the art tools [11].

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. contains computational experiments on data from the RFAM database: we compare the objective function values of truly multiple structural alignments and heuristically inferred ones. Furthermore, we briefly report on computational results on the recently published benchmark set BRALIBASE [51]. We show that our approach yields comparable results in the pairwise case to other state-of-the-art programs while needing less computation time.

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 [8] and [4].

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 refer by $M_i^{-1}(j)$ to 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 $\sigma : \hat{\Sigma} \times \hat{\Sigma} \to \mathbb{R}$ 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. We 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) score of A is defined by

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

Intuitively speaking, the sum-of-pairs score adds up all scores of pairs of aligned characters in the alignment

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

Fig. 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

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

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

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.

This score model does not explicitly model gaps; they are 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—e.g., aligning three 'A's to three gaps—achieves the same score as three separate individual gaps, see Fig. 6 (b) and (c).

Biological findings motivate a more complicated gap model: the beginning 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 clearly favors the single gap, see Fig. 6 (c), over the three individual gaps, see Fig. 6 (b).

We therefore 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. We denote a gap of length ℓ in sequence i at position j by a triple (i, j, ℓ) and assign it a penalty score $\gamma(i, j, \ell) \in \mathbb{R}_{\leq 0}$.

Definition 4. Given a set S of k strings $s^1, \ldots, s^k$, an alignment A consisting of strings $\hat{s}^1, \ldots, \hat{s}^k$, a sequence scoring function σ, and a gap penalty function γ. We denote the gaps in A with

$$G(A) := \{(i, j, \ell) \mid \text{sequence } i \text{ has a gap of length } \ell \text{ at position } j \text{ in } A\}$$

The *gapped sum-of-pairs* (GSPS) score of A is defined by

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

Note that γ assigns negative scores to gaps in the alignments.

As described in Sect. 1., sequence alignments are in general not sufficient enough to build reliable RNA alignments. Therefore, in addition to the gaps, we propose 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), (C, G), (A, U), (U, A), (G, U), or (U, G). The set p of interactions is called the *annotation* of sequence s. Two interactions (s_e, s_f) and (s_g, s_h) are said to be *inconsistent*, if they share one base; they form a *pseudoknot* if they "cross" each other that is if $e < g < f < h$ or $g < e < h < f$. 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 $(s_e^i, s_f^i) \in p^i$ and $(s_g^j, s_h^j) \in p^j$ a *structural match* if s_e^i is aligned with s_g^j and s_f^i is aligned with s_h^j. Two structural matches $(\hat{s}_e^i, \hat{s}_f^i)$, $(\hat{s}_e^j, \hat{s}_f^j)$ and $(\hat{s}_g^i, \hat{s}_h^i)$, $(\hat{s}_g^j, \hat{s}_h^j)$ are *inconsistent* if either $e = g$, $f = g$, $e = h$, or $f = h$. 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

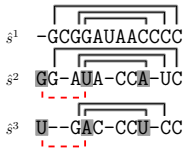

Fig. 7. Realized structural matches are highlighted with grey edges: the structural match $x = [(\hat{s}^2_1, \hat{s}^2_5), (\hat{s}^3_1, \hat{s}^3_5)]$ (the red dotted edges) is inconsistent with the structural match $y = [(\hat{s}^2_5, \hat{s}^2_{10}), (\hat{s}^3_5, \hat{s}^3_{10})]$, that is we either score x or y

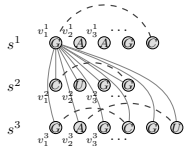

Fig. 8. Basic graph model of three annotated sequences containing lines (grey solid lines) and interaction edges (bold dotted edges). For sake of clarity we do not show all alignment edges, only the ones incident to v^1_1

$G(A)$ be the set of all gaps of A, and let σ, τ, γ be functions for scoring sequence, structural matches, and gaps. Then the *gapped structural sum-of-pairs* score of A is defined by

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

which does not score inconsistent structural matches, that is, every base is part of at most one structural match.

Figure 7 gives an illustration for the definitions given above. In analogy 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 S be a set of k sequences $s^1, \ldots, s^k$. We aim at computing an alignment A^* with

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

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

3.2. *Graph-Theoretical Model for Structural RNA Alignment*

3.2.0.1 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 mixed graph $(V, L \cup F \cup D \cup G)$. 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 set $L^{ij}_{v^i_k}$ is the subset of L^{ij} containing only alignment edges whose source node is v^i_k. Observe that the graph (V, L) is k-partite.

The edge set F models the annotation of the input sequences in our graph. Consequently, we have *interaction edges* between vertices of the same sequence, i.e., edges (v^i_k, v^i_l) representing the interaction between vertices v^i_k and v^i_l. Figure 8 illustrates these definitions.

3.2.0.2 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}_{ef} from v^i_e to v^i_f representing the fact that no character of the substring $s^i_e \ldots s^i_f$ is aligned to any character of the sequence j, whereas s^i_{e-1} (if $e > 1$) and s^i_{f+1} (if $f + 1 \leq |s^j|$) are aligned with some characters in sequence j. We say that $v^i_e, \ldots, v^i_f$ are *spanned* by the gap arc a^{ij}_{ef}. 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_e, v^i_f) in order to represent gaps between the actual sequence and the remaining $k-1$ sequences.

Two gap arcs $a^{ij}_{ef}, a^{ij}_{mn} \in G^{ij}$, w.l.o.g. $e < m$, are *in*

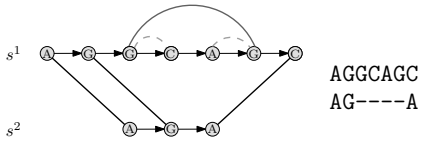

Fig. 9. A longer gap cannot be split into two shorter gaps: the two dashed gap edges are in conflict with each other and are replaced by the solid gap edge spanning the two shorter gap edges

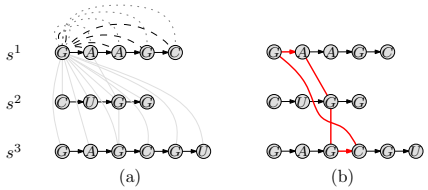

Fig. 10. (a) Basic graph model augmented by gap edges (interaction edges are not displayed) , (b) showing an instance of a mixed cycle

conflict with each other if $\{e,\dots,f+1\}\cap\{m,\dots,n\}\neq\emptyset$, 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. We define a set $\mathcal{C}$ containing all maximal sets of pairwise conflicting gap arcs. Finally, we define $G^{ij}_{v^i_e\leftrightarrow v^i_f}$ as the set of gap arcs that span the nodes $v^i_e\dots v^i_f$. See Fig. 9 for an illustration.

3.2.0.3 Mixed Cycles A *path* in $(V, L\cup D)$ is an alternating sequence $v_1, e_1, v_2, e_2, \dots$ of vertices $v_i\in V$ and lines or arcs $e_i\in L\cup D$. 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 corresponds to lines that cross each other. The 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 $\mathcal{L}\cup D$ does not contain a mixed cycle [33, 44]. In this case, we use the term alignment for $\mathcal{L}$.

3.2.0.4 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 two lines $e=(v^i_k,v^j_m)$ and $f=(v^i_l,v^j_n)$ exist such that e and f do not cross each other. A subset $\mathcal{L}\subset L$ *realizes* the interaction match (e,f) if $e,f\in\mathcal{L}$.

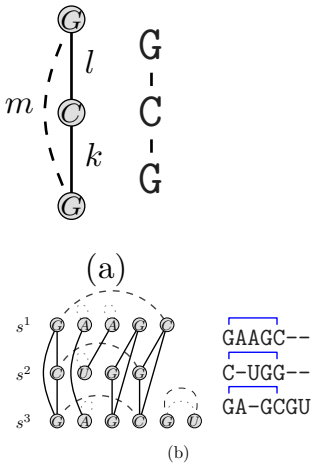

Fig. 11. (a) Transitive edges must be realized: If k and l are part of the alignment, then m has to be realized as well. (b) Example of a valid gapped structural trace of three annotated sequences. Three interaction matches are conserved by the alignment

Observe that the definition of an interaction match is a graph-theoretical reformulation of a structural match as defined in Sect. 3.1.. The set I contains all possible interaction matches of L.

3.2.0.5 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 $i,j=1,\dots,k, i\neq j$ we define $\mathcal{L}^{ij}=L^{ij}\cap\mathcal{L}$: Then, for $l=1,\dots,|s^i|$ 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 $M\cap L=M$.
(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 $\mathcal{L}$, we denote by $H(\mathcal{L})$ the transitive closure 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.

Observation 1. *There is a one-to-one mapping between alignments realizing structural matches and*

gapped structural traces.

Proof. The correspondence follows the observation in [4]. In our case, however, we have to additionally map structural matches to realized interaction matches in the gapped structural trace. Due to the one-to-one mapping between structural matches and interaction matches, this is straightforward. □

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 $\sigma(s_k^i, s_l^j)$ gives a high value in case of identical (or similar) characters. We assign scores to interaction edges by calculating *base pair probabilities* [41]. The base pair probability $\text{bpp}(v_k^i, v_l^i)$ gives the probability that nucleotides s_k^i and s_l^i fold onto each other, i.e., $\text{bpp}(v_k^i, v_l^i)$ will be close to 1 if v_k^i is very likely to form a hydrogen bond with v_l^i. To use the probabilities in an additive scoring scheme, we perform a logarithmic transformation, i.e., the actual score p_{kl}^i for an interaction between s_k^i and s_l^i is given by

$$p_{kl}^i = \lg\left(\frac{\text{bpp}(v_k^i, v_l^i)}{p_{\min}}\right)$$

where $p_{\min}$ is the minimal probability that we consider. The weight $w_{\hat{\imath}\hat{\jmath}}$ for an interaction match of lines $\hat{\imath} = (v_k^i, v_m^j)$ and $\hat{\jmath} = (v_l^i, v_n^j)$ is then given by $w_{\hat{\imath}\hat{\jmath}} = p_{kl}^i + p_{mn}^j$, 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_{kl}^{ij} representing the gap penalty for aligning substring $s_k^i \dots s_l^i$ with gap characters in sequence j.

3.3. *Complexity*

Jiang and Wang showed that computing an optimal multiple sequence alignment is NP-hard [50]. Along these lines, Elias proves that the problem remains NP-hard for different scoring functions [20].

The complexity of sequence-structure alignments depend on the input of the problem and on the actual model one is using: optimal pairwise sequence-structure alignments of RNA structures—as defined in Sect. 3.1.—where pseudoknots are not allowed can be computed in polynomial time [7]. Goldman et al. show in [23] that computing the *maximal contact map overlap*—a problem similar to RNA alignment—is NP-hard in the pairwise case. They also state that the computation of the maximal contact map overlap, where every node has a maximum degree of 1, is already NP-hard. 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 [21] Evans gave an NP-hardness proof for the computation of the longest arc-preserving common subsequence. Along these lines, Blin et al. give several NP-completeness proofs [12, 13] for variants of the arc-annotated sequence model.

Computing sequence-structure alignments in the general edit-model of [31] 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

This section begins with our *integer linear programming formulation* for the multiple sequence-structure alignment problem, which is based on the graph-theoretical model from the previous section. We then show how to compute solutions to this integer linear program (ILP) using the Lagrangian relaxation method.

4.1. *Integer Linear Program*

We associate binary variables with each line, interaction match, and gap edge, and model the constraints of a valid gapped structural trace by suitable inequalities in the ILP.

The handling of lines and gap edges is straightforward: We associate an x and a z variable to each line and gap edge, respectively, with the following interpretation: $x_l = 1$ 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 match, 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 $\mathcal{L}$

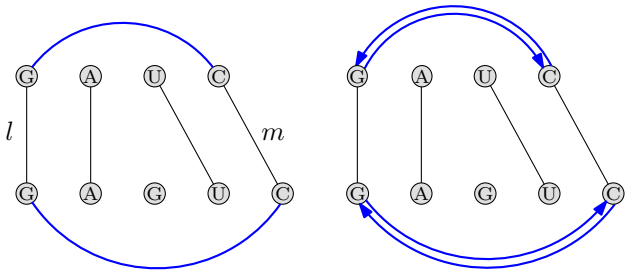

Fig. 12. One interaction match is split into two *directed* interaction matches

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.

Splitting interaction matches has first been proposed by Caprara and Lancia in the context of contact map overlap [14], whereas *variable splitting*, or *Lagrangian decomposition*, is a well-known technique in mathematical programming [26].

$$\max \quad \sum_{l \in L} w_l x_l + \sum_{g \in G} w_g z_g + \sum_{l \in L} \sum_{m \in L} w_{lm} y_{lm} \tag{1}$$

$$\text{s.t.} \quad \sum_{l \in L \cap M} x_l \leq |L \cap M| - 1 \forall M \in \mathcal{M} \tag{2}$$

$$x_l + x_k - x_m \leq 1 \quad \forall (l,k,m) \in L,\ (x_l, x_k, x_m) \text{ forming a cycle} \tag{3}$$

$$\sum_{a \in C} z_a \leq 1 \quad \forall C \in \mathcal{C} \tag{4}$$

$$\sum_{l \in L^{ij}_{s(m)}} x_l + \sum_{a \in G^{ij}_{s(l) \leftrightarrow s(l)}} z_a = 1 \quad 1 \leq i, j \leq k, i \neq j, \forall m \in L^{ij} \tag{5}$$

$$\sum_{m \in L} y_{lm} \leq x_l \quad \forall l \in L \tag{6}$$

$$y_{lm} = y_{ml} \quad \forall l, m \in L \tag{7}$$

$$x_l \in \{0,1\} \quad \forall l \in L \tag{8}$$

$$y_{lm} \in \{0,1\} \quad \forall l, m \in L \tag{9}$$

$$z_g \in \{0,1\} \quad \forall g \in G \tag{10}$$

Fig. 13. Master ILP

Definition 9. We call the ILP (1)–(10) of Fig. 13 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 $w_g < 0$ for $g \in G$.

Lemma 4..1. *A feasible solution to the ILP* (1)–(10) *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 $m \notin \hat{\mathcal{L}}$ 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^{ij}_{kl} and a^{ij}_{mn} 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)–(10) 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); thus, each variable x_l is 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

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

$$p_l := \max \quad w_l + \sum_{m \in L} w_{lm} y_{lm} + \sum_{a \in G^{ij}_{s(l)\leftrightarrow s(l)}, G^{ji}_{t(l)\leftrightarrow t(l)}} w_a z_a \tag{11}$$

$$\text{s.t.} \quad \sum_{m \in L} y_{lm} \leq 1 \tag{12}$$

$$\sum_{a \in G^{ij}_{s(l)\leftrightarrow s(l)}, G^{ji}_{t(l)\leftrightarrow t(l)}} z_a = 0 \tag{13}$$

$$x_l \in \{0,1\} \ \ \forall l \in L \tag{14}$$

$$y_{lm} \in \{0,1\} \ \ \forall l, m \in L \tag{15}$$

$$z_g \in \{0,1\} \ \ \forall g \in G \tag{16}$$

Inequality (12) states that we can choose only one single interaction match from the set of outgoing interaction matches that alignment edge l can possibly realize. According to the objective function (11) 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_k^i or v_l^j, since otherwise constraints (13) 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 $\hat{m}$ are not in conflict, which can be done in linear time. Therefore, the profit a line can possibly achieve is solely computed by considering the weights of line l and of the best directed interaction match $(l, \hat{m})$ that line l can realize, 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, which is given in Fig. 14.

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 correspond to the *multiple sequence alignment* formulation given in [4].

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 realize), 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

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

Fig. 14. Computing the profit

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

$$\begin{aligned}
\sum_{l \in L} p_l x^*_l + \sum_{g \in G} w_g z^*_g &= \sum_{l \in L} (w_l + w_{l\hat{m}}) x^*_l + \sum_{g \in G} w_g z^*_g \\
&= \underbrace{\sum_{l \in L} w_l x^*_l + \sum_{g \in G} w_g z^*_g}_{\text{optimal sol. for MSA}} + \\
&\qquad \underbrace{\sum_{l \in L} \sum_{m \in L} w_{lm} y^*_{lm}}_{\text{optimal sol. for } y_{l\hat{m}} \text{ due to (11)–(16)}}
\end{aligned}$$

We now proof that (x^*, y^*, z^*) is indeed the optimal solution. Assume that there exists a valid solution $(\bar{x}^*, \bar{y}^*, \bar{z}^*)$ that has a higher objective function value than (x^*, y^*, z^*). Clearly, (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 the objective function in case the dropped constraint is not satisfied.

Moving constraints (7) into the objective function yields the *Lagrangian dual*, which is the slave ILP with objective function

$$\max \sum_{l\in L} w_l x_l + \sum_{g\in G} w_g z_g + \sum_{l\in L}\sum_{m\in L} w_{lm} y_{lm} + \sum_{l\in L}\sum_{m\in L} \lambda_{lm}(y_{lm} - y_{ml}) \ . \tag{17}$$

Exploiting the fact that $\lambda_{lm} = -\lambda_{ml}$, which we ensure below, (17) can be reformulated to

$$\max \sum_{l\in L} w_l x_l + \sum_{g\in G} w_g z_g + \sum_{l\in L}\sum_{m\in L} (w_{lm} + \lambda_{lm}) y_{lm} \ . \tag{18}$$

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. In practice, iterative *subgradient optimization* as proposed by Held and Karp [27] is widely used. 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

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

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

Here, μ is a common adaption parameter and v_U and v_L denote the best upper and lower bounds, respectively.

In each iteration of the subgradient optimization procedure we get a value for the Lagrangian dual. Given this series $(v^1, v^2, \ldots, v^n)$ 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* [38]. However, currently the described subgradient optimization exhibits better convergence properties than bundle methods.

Note that unless the lower and the upper bound v_L and v_U coincide, we cannot guarantee optimality. 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 solutions 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 Fig. 15 (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 an 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 to find a subset $\hat{I}$ of interaction edges of maximum weight such that the structural information for each sequence is valid, that is, each base is paired with at most one other base. Figure 15 (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 would result in an invalid secondary structure. We therefore define the problem of finding the best *structural comple-*

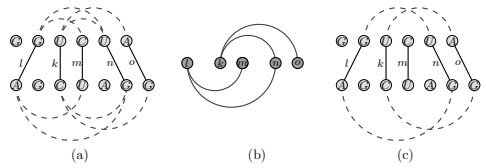

Fig. 15. Given the alignment $\mathcal{L} = (l, k, m, n, o)$, we have different possibilities to augment the alignment with structural matches. Creating an interaction matching graph (b) and calculating a general matching of maximum weight yields the best structural completion of $\mathcal{L}$ (c)

tion 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—the *structural completion*—of maximal weight on $\mathcal{L}$.

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_k^i, v_l^i) and (v_m^j, v_n^j) whose endpoints are adjacent to a pair $(o, p) \in \mathcal{L} \times \mathcal{L}$ (see Fig. 15 (b)). The weight of edge e_i is given by the weight of the two interaction edges (v_k^i, v_l^i) and (v_m^j, v_n^j).

Lemma 4..3. *A matching of maximum weight in the interaction matching graph M_S corresponds to the best 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

MLARA is our prototypical implementation of the formulation for multiple structural alignments presented in the previous section and is publicly available as part of the open source library LISA [34]. The actual algorithm is easy to implement and does not depend on commercial LP-solvers. For the computation of the lower bound, however, we use the matching routines from the LEDA library [37]. According to Lemma 4..2, the solution to an instance of the Lagrangian problem amounts to the computation of an exact multiple sequence alignment problem with arbitrary gap costs. Although this problem is NP-hard, the branch-and-cut algorithm of Althaus et al. [4] is able to solve medium-sized instances within reasonable computation time, and we use their code as a subroutine.

Table 1

	Instance	lara	mLARA
tRNA	#0	1072.37	**1193.34** (**0.94**)
	#1	1188.72	**1194.33** (**0.94**)
	#2	1431.85	**1453.06** (**0.99**)
	#3	1439.87	**1469.63** (**0.98**)
	#4	958.19	**1014.61** (**0.83**)
	#5	1167.46	**1184.89** (**0.88**)
	#6	1285.38	**1299.14** (**0.97**)
	#7	1262.62	**1304.31** (**0.98**)
	#8	**1808.26**	1772.04 (1.00)
	#9	1141.16	**1193.55** (**0.92**)
	#10	1130.97	**1134.20** (**0.90**)
	#11	**1134.41**	1043.95 (0.80)
	#12	1109.39	**1113.45** (**0.91**)
	#13	1274.95	**1323.94** (**0.97**)
	#14	**1299.85**	1254.51 (0.96)
	#15	**1275.82**	1077.07 (0.88)
	#16	857.06	**1019.92** (**0.88**)
	#17	1110.28	**1133.84** (**0.85**)
	#18	1280.29	**1320.11** (**0.99**)
	#19	1314.67	**1366.26** (**0.99**)
5S			
	#0	1853.12	**1922.20** (**0.96**)
	#1	1810.34	**2097.22** (**0.99**)
	#2	2233.86	**2259.01** (**1.00**)
	#3	2008.30	**2049.05** (**0.98**)
	#4	1687.17	**1735.34** (**0.92**)
	#5	**1702.27**	1696.58 (0.88)
	#6	1605.52	**1682.68** (**0.93**)
	#7	**1911.39**	1740.73 (0.90)
	#8	**2198.24**	1956.62 (0.89)
	#9	2014.57	**2107.10** (**1.00**)
	#10	**2048.08**	1946.53 (0.93)
	#11	1693.10	**1715.54** (**0.92**)
	#12	1927.36	**2023.18** (**0.99**)
	#13	**2208.30**	1996.10 (0.86)
	#14	2224.99	**2267.25** (**0.99**)
	#15	1954.85	**2064.64** (**0.97**)
	#16	2076.54	**2116.64** (**0.99**)
	#17	**1915.61**	1884.92 (0.94)
	#18	2096.72	**2171.81** (**0.99**)
	#19	2131.44	**2407.99** (**0.99**)

The comparison between the objective function values of LARA *and* MLARA *on* 40 *randomly generated tRNA and 5S RNA instances. The numbers in brackets give the level of optimality. Note that in some cases the heuristic algorithm produces better results, which is possible due to the time limit and the fact that* T-COFFEE *adds more alignment edges to the graph*

In the following we will give a proof-of-concept of our approach by running experiments on real data of moderate size, setting all gap costs to zero. From the RFAM database [25] we downloaded sequences that belong to the families of ribosomal L19 leader proteins,

tRNAs, and ribosomal 5S RNAs (the RFAM IDs are RF00556, RF00005, and RF00001, respectively).

As a first example we take L19 leader protein sequences (accession numbers: AL935256.1, AE014216.1, and AP006627.1) and compute the optimal multiple alignment given the complete k-partite graph containing 4106 alignment edges. We find a provably optimal solution after 19 hours of computation. There are two interesting observations: First, the optimal solution to the input instance is found within the first 10 iterations of the computation, that is, only 70 seconds after starting the program. MLARA spends the remaining time on proving the optimality of this solution. Second, although we need the complete k-partite graph to ensure optimality, many alignment edges are not very likely to be part of the optimal structural alignment, e.g., edges running from the first vertex in the first sequence to the last vertex in the second sequence. As one can see on the left side of Fig. 16, the number of alignment edges greatly influences the running time for computing an exact multiple structural alignment. We therefore follow the strategy that we already employed in previous work [8–10, 39]: We generate a set of reasonable alignment edges by computing a conventional sequence alignment with affine gap costs and subsequently insert all alignment edges realized by any suboptimal alignment scoring better than a fixed threshold s below the optimal score. Although we cannot guarantee that the set of alignment edges always contains the edges forming the optimal multiple structural alignment, our experiments on the RFAM database show that RFAM reference alignments consist of alignment edges of small suboptimality.

We again take the sequences from our first example and compute the multiple structural alignment based on a reduced set of alignment edges: Already a suboptimality level of 5 suffices to generate all alignment edges that are part of the provably optimal solution. The reduced number of alignment edges—465 instead of 4106—brings the overall running time down from 19 hours to 43.35 seconds.

In our experiments we realized that not only the number of alignment edges influences the overall computation time. As described in Sect. 4.2. we resort to subgradient optimization to solve the Lagrangian dual. By iteratively adapting the Lagrangian multipliers and computing the multiple sequence alignment afterwards, we observe an increase in the running time per iteration over the course of all iterations. The right side of Fig. 16 shows the development for an instance of three

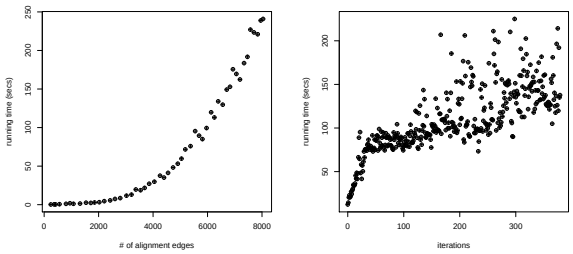

Fig. 16. Typical behavior for the multiple case. Left: The time to compute an exact multiple sequence alignment increases non-linearly with the number of alignment edges. Right: The time to compute one single iteration for an instance containing 4106 alignment edges increases rapidly with the number of iterations. This is due to the adaption of the Lagrangian multiplier

L19 leader protein sequences.

As a second experiment, we assess the improvement of the objective function value between heuristically inferred multiple structural alignments [11] and provably optimal or near-optimal solutions of the exact multiple sequence-structure model described in this article: To this end, we randomly drew 20 instances containing three input sequences of either tRNA or ribosomal 5S RNA sequences (RFAM IDs: RF00001 and RF00005), resulting in 40 instances in total. Using our tool LARA, which yields the best results on the BRALIBASE benchmark set [11], we compute all pairwise alignments of a given instance and feed them to the T-COFFEE software [43] to heuristically infer a consistency-based multiple structural alignment. Given this alignment, we again evaluate it under the sum-of-pair objective function of MLARA.

Then, we take MLARA and compute the multiple structural alignments: We allow a maximal computation time of three hours per instance. If MLARA does not terminate within three hours, we stop the computation and report the best solution found so far. We then compare the objective function values. We want to stress the fact that we use exactly the same settings for both programs, i.e., we use the same scoring scheme and generate the same alignment edges such that the results are comparable.

Table 1 shows the objective function values of the alignments generated by LARA and MLARA on the 40 instances. Generally, MLARA reaches higher objective function values than those computed by LARA. There are, however, some instances where the heuristically in-

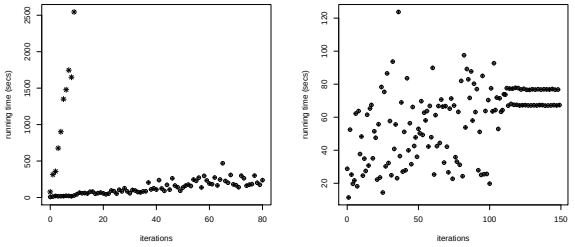

Fig. 17. Left: The computation time per iterations oscillates dramatically even between instances having almost the same number of alignment edges: the circles and stars represent the iterations of tRNA instances #2 (1755 edges) and #4 (1756 edges) from Tab. 1, respectively. Right: The solution process may get stuck between two solutions and jumps back and forth between these two, and therefore does not find the global optimal solution. The plot shows the solution process of the 5S instance #13 from Tab. 1

ferred alignments yield better objective function values than MLARA. A closer inspection of those instances reveals the following reasons:

- The computation time limit is too tight. Hence, our program performs only a small number of iterations, and is therefore not able to adapt the Lagrangian multipliers accordingly.

  In many instances, the time spent on one single iteration is not predictable: The left side of Fig. 17 shows the computation time per iteration of the tRNA instances #2 (1755 alignment edges, represented by the circles) and #4 (1756 alignment edges, represented by the stars) from Tab. 1. Although the number of alignment edges differs only by one, the computation time per iterations varies dramatically: Consequently, MLARA performs 80 and only 9 iterations for instances #2 and #4, respectively.
- The right side of Fig. 17 shows the solution process for 5S instance #13 from Tab. 1. After 110 iterations MLARA gets stuck between two solutions and oscillates between these two (represented by the two parallel lines from iterations 110-165). From this point on, the algorithm is not able to further converge to the global optimal solution.
- The T-COFFEE software potentially augments the set of alignment edges when it heuristically builds a multiple structural alignment based on all pairwise alignments. This happens, for example, for instance #8.

As described above and shown in Fig. 16, the overall running time of MLARA depends on the way we solve the multiple sequence alignment problem, which appears as the Lagrangian dual. The computation of optimal multiple structural alignments requires at least a couple of hours of computation time making the solution process rather tedious: We nevertheless want to analyze the performance of our model on a large data set such as the recently published BRALIBASE benchmark set of RNA sequence-structure alignments [51]. We therefore restrict ourselves to the special case of computing pairwise sequence-structure alignments: in the pairwise case standard dynamic programming algorithms always solve the Lagrangian relaxed problem in time $\mathcal{O}(nm)$, with n and m being the lengths of the two input sequences, independent from the values of the Lagrangian multipliers. This means that the running time per iteration remains constant over the course of all iterations.

We want to show that our model yields on average comparable or better alignments than other state-of-the-art programs that build upon more costly dynamic programming algorithms. We do believe that this comparison is useful, because it shows that the actual model works: The limiting factor for computing larger instances of multiple structural alignments faster is the code that solves the multiple sequence alignment problem. In recent months there has been progress with respect to this matter, as new algorithms for the exact multiple-sequence-alignment problem are being published, e.g., [2]. As soon as these programs become available, we can simply plug them into our model and will then able to solve larger instances of multiple sequence-structure alignments.

Note that the following description of the pairwise instances is just a short summary of the extensive analysis that we give in our companion paper: The interested reader is referred to [11] for the full description of our results on the entire BRALIBASE data set.

Gardner et al. observe in [22] that structure-based alignment programs produce significantly better alignments compared to sequence-based programs if the sequence similarity drops below approximately 50-60 percent. For our tests, we therefore excluded all instances that had a pairwise sequence similarity greater than 50 percent, resulting in 2251 instances containing two input sequences.

We compared our algorithm to three other sequence-structure alignment programs: FOLDALIGNM [48] which is based on a variant of the Sankoff algorithm, MARNA [46], and STRAL [17]. These programs have

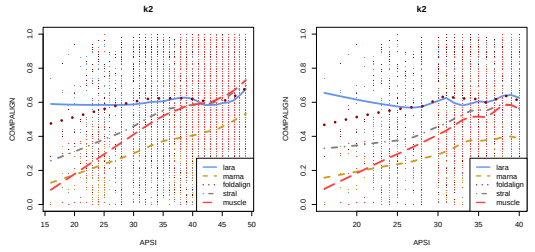

Fig. 18. Results of our implementation on instances containing two input sequences. One dot corresponds to one alignment, the lines represent the Lowess function, i.e., they give the trend of the computed alignments. A line at 1.0 means that every alignment is identical to the reference alignment: Hence, the closer the line is to 1.0, the better the alignments are on average. The left side shows all 2251 pairwise instances, whereas the right side shows only instances of an average pairwise sequence identity below 40%

running times 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. All these programs do not consider pseudoknots, i.e., crossing interactions. Additionally, we took MUSCLE to compare our alignments with a program that is purely sequence-based.

Figure 18 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 reference alignment 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 a value of 0 denotes that no column was correctly aligned with respect to the reference alignment. Hence, the higher the value, the bigger is the similarity of an alignment to the reference alignment.

In the pairwise case LARA and FOLDALIGNM show almost the same COMPALIGN performance: over all 2251 input instances LARA and FOLDALIGNM reach an average value of 0.60 and 0.61, respectively. LARA, however, only needs 86 minutes to compute all 2251 pairwise sequence-structure alignment. On the same input set FOLDALIGNM needs 172 minutes. The other structural alignment programs, STRAL and MARNA, yield worse average COMPALIGN scores of 0.58 and 0.41 in 2.5 and 941 minutes of computation time, respectively. The purely sequence based program MUSCLE reaches an average value of 0.58 in only 12 seconds of computation time. The left side of Fig. 18 shows the trend of the results for the tested programs using the Lowess function, i.e., locally weighted regression.

Looking at these values, the rather small difference of 0.02 in the average COMPALIGN values between LARA and STRAL does not seem to justify the higher running time of LARA. Concentrating, however, on the hard-to-align input instances that show very little sequence conservation, i.e., sequences showing an average pairwise sequence identity below 40%, the picture changes:

For these 884 instances LARA, FOLDALIGNM, STRAL, MARNA, and MUSCLE reach average COMPALIGN values of 0.60, 0.60, 0.52, 0.35, and 0.49, respectively, see also the plot on the right side of Fig. 18. This clearly shows that it indeed does pay off using slower but exact methods when aligning input instances of low average pairwise sequence identity, where sequence information does not suffice to build good alignments.

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. The authors also thank the anonymous referees for their constructive remarks, which helped improve our initial submission.

References

[1] M. D. Adams et al., *The Genome Sequence of Drosophila melanogaster*, Science **287** (2000), no. 5461, 2185–2195.

[2] E. Althaus and S. Canzar, *A Lagrangian Relaxation Approach for the Multiple Sequence Alignment Problem*, Proc. COCOA'07, LNCS, vol. 4616, 2007, pp. 267–278.

[3] E. Althaus, A. Caprara, H.-P. Lenhof, and K. Reinert, *Multiple sequence alignment with arbitrary gap costs: Computing an optimal solution using polyhedral combinatorics*, Bioinformatics **18** (2002), no. 90002, S4–S16.

[4] E. Althaus, A. Caprara, H.-P. Lenhof, and K. Reinert, *A branch-and-cut algorithm for multiple sequence alignment*, Mathematical Programming (2006), no. 105, 387–425.

[5] E. Althaus, O. Kohlbacher, H.-P. Lenhof, and P. Müller, *A combinatorial approach to protein docking with flexible side-chains.*, Proc. RECOMB'00, 2000, pp. 15–24.

[6] S. F. Altschul, W. Gish, E. W. Myers, and D. J. Lipman, *Basic local alignment search tool*, Journal of Molecular Biology **215** (1990), no. 3, 403–410.

[7] V. Bafna, S. Muthukrishnan, and R. Ravi, *Computing similarity between RNA strings*, Proc. CPM'95, LNCS, no. 937, Springer, 1995, pp. 1–16.

[8] M. Bauer and G. W. Klau, *Structural Alignment of Two RNA Sequences with Lagrangian Relaxation*, Proc. ISAAC'04, LNCS, no. 3341, Springer, 2004, pp. 113–123.

[9] M. Bauer, G. W. Klau, and K. Reinert, *Fast and accurate structural RNA alignment by progressive Lagrangian relaxation*, Proc. CompLife'05, LNBI, vol. 3695, 2005, pp. 217–228.

[10] M. Bauer, G. W. Klau, and K. Reinert, *Multiple structural RNA alignment with Lagrangian relaxation*, Proc. WABI'05, LNBI, vol. 3692, 2005, pp. 303–314.

[11] M. Bauer, G. W. Klau, and K. Reinert, *Accurate Multiple Sequence-Structure Alignment of RNA Sequences Using Combinatorial Optimization*, BMC Bioinformatics **8** (2007), no. 1, 271.

[12] G. Blin, G. Fertin, R. Rizzi, and S. Vialette, *What Makes the Arc-Preserving Subsequence Problem Hard?*, T. Comp. Sys. Biology **2** (2005), 1–36.

[13] G. Blin and H. Touzet, *How to compare arc-annotated sequences: The alignment hierarchy.*, Proc. SPIRE'06, 2006, pp. 291–303.

[14] A. Caprara and G. Lancia, *Structural Alignment of Large-Size Proteins via Lagrangian Relaxation*, Proc. RECOMB'02, 2002, pp. 100–108.

[15] A. Caprara, D. Pisinger, and P. Toth, *Exact solution of the quadratic knapsack problem*, INFORMS Journal on Computing **11** (1999), no. 2, 125–137.

[16] 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, 1994, pp. 147–160.

[17] D. Dalli, A. Wilm, I. Mainz, and G. Steger, *STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time*, Bioinformatics **22** (2006), no. 13, 1593–1599.

[18] E. Davydov and S. Batzoglou, *A computational model for RNA multiple structural alignment*, Theoretical Computer Science **368** (2006), no. 3, 205–216.

[19] R. Dowell and S.R. Eddy, *Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints*, BMC Bioinformatics **7** (2006), no. 1, 400.

[20] I. Elias, *Settling the intractability of multiple alignment.*, Journal of Computational Biology **13** (2006), no. 7, 1323–1339.

[21] P. Evans, *Finding common subsequences with arcs and pseudoknots*, Proc. of CPM'99, LNCS, no. 1645, Springer, 1999, pp. 270–280.

[22] 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.

[23] D. Goldman, S. Istrail, and C. H. Papadimitriou, *Algorithmic aspects of protein structure similarity.*, Proc. FOCS'99, 1999, pp. 512–522.

[24] H.J. Greenberg, W. E. Hart, and G. Lancia, *Opportunities for combinatorial optimization in computational biology*, INFORMS Journal on Computing **16** (2004), no. 3, 211–231.

[25] S. Griffiths-Jones, S. Moxon, M. Marshall, A. Khanna, S. R. Eddy, and A. Bateman, *Rfam: annotating non-coding RNAs in complete genomes*, Nucl. Acids Res. **33** (2005), D121–124.

[26] M. Guignard and S. Kim, *Lagrangean decomposition: A model yielding stronger Lagrangean bounds*, Mathematical Programming **39** (1987), no. 2, 215–228.

[27] M. Held and R.M. Karp, *The traveling-salesman problem and minimum spanning trees: Part II*, Mathematical Programming **1** (1971), 6–25.

[28] I. L. Hofacker, S. H. F. Bernhart, and P. F. Stadler, *Alignment of RNA base pairing probability matrices*, Bioinformatics **20** (2004), 2222–2227.

[29] I. Holmes, *Accelerated probabilistic inference of RNA structure evolution*, BMC Bioinformatics **5** (2004), 73.

[30] 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.

[31] T. Jiang, G.-H. Lin, B. Ma, and K. Zhang, *A general edit distance between RNA structures*, Journal of Computational Biology **9** (2002), 371–388.

[32] T. Jiang, J. Wang, and K. Zhang, *Alignment of trees — an alternative to tree edit*, Theor. Comput. Sci. **143** (1995), 137–148.

[33] J. Kececioglu, *The maximum weight trace problem in multiple sequence alignment*, Proc. CPM'93, LNCS, vol. 684, 1993, pp. 106–119.

[34] G. W. Klau, S. Andreotti, M. Bauer, P. May, and L. Petzold, *http://www.planet-lisa.net*, [web page], [Accessed 2 Nov. 2007].

[35] G.W̃. Klau, S. Rahmann, A. Schliep, M. Vingron, and K. Reinert, *Optimal robust non-unique probe selection using Integer Linear Programming*, Bioinformatics **20 (Suppl. 1)** (2004), i186–i193.

[36] 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. RECOMB'01, 2001, pp. 193–202.

[37] LEDA, *The LEDA user manual*, Algorithmic Solutions, March 2007, http://www.algorithmic-solutions.info/leda_manual/MANUAL.html.

[38] C. Lemaréchal, *Computational combinatorial optimization, optimal or provably near-optimal solutions*, ch. Lagrangian Relaxation, pp. 112–156, Springer Berlin, 2001.

[39] H.-P. Lenhof, K. Reinert, and M. Vingron, *A polyhedral approach to RNA sequence structure alignment*, Journal of Computational Biology **5** (1998), no. 3, 517–530.

[40] J. S. Mattick, *The functional genomics of noncoding RNA*, Science **309** (2005), no. 5740, 1527–1528.

[41] J. S. McCaskill, *The Equilibrium Partition Function and Base Pair Binding Probabilities for RNA Secondary Structure*, Biopolymers **29** (1990), 1105–1119.

[42] 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.

[43] C. Notredame, D. G. Higgins, and J. Heringa, *T-Coffee: A novel method for fast and accurate multiple sequence alignment*, Journal of Molecular Biology **302** (2000), 205–217.

[44] K. Reinert, H.-P. Lenhof, P. Mutzel, K. Mehlhorn, and J. D. Kececioglu, *A branch-and-cut algorithm for multiple sequence alignment.*, Proc. RECOMB'97, 1997, pp. 241–250.

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

Received 23 March 2007; revised 24 October 2007; accepted 15 November 2007

[46] S. Siebert and R. Backofen, *MARNA: Multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons*, Bioinformatics **21** (2005), no. 16, 3352–3359.

[47] T. F. Smith and M. S. Waterman, *Identification of Common Molecular Subsequences*, Journal of Molecular Biology **147** (1981), 195–197.

[48] E. Torarinsson, J. H. Havgaard, and J. Gorodkin, *Multiple structural alignment and clustering of RNA sequences*, Bioinformatics **23** (2007), no. 8, 926–932.

[49] J. Craig Venter et al., *The Sequence of the Human Genome*, Science **291** (2001), no. 5507, 1304–1351.

[50] L. Wang and T. Jiang, *On the complexity of multiple sequence alignment.*, Journal of Computational Biology **1** (1994), no. 4, 337–348.

[51] A. Wilm, I. Mainz, and G. Steger, *An enhanced RNA alignment benchmark for sequence alignment programs*, Algorithms for Molecular Biology **1** (2006), no. 1, 19.

[52] 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.