# A Combinatorial Approach To RNA Sequence-Structure Alignments

Markus Johann Bauer

Dissertation zur Erlangung des Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

*Freie Universität Berlin*
*April 2008*

Datum des Kolloquiums: ***10. Juli 2008***

Gutachter:
***Prof. Dr. Knut Reinert***, *Freie Universität Berlin, Deutschland*
***Dr. Gunnar Klau***, *Freie Universität Berlin, Deutschland*
***Prof. Dr. Peter F. Stadler***, *Universität Leipzig, Deutschland*

## Short Abstract

Until a couple of years ago the scientific mainstream held that genetic information, stored as DNA strands, is transcribed to RNA, and RNA sequences are in turn translated to proteins, the actual functional units in the cell. RNA was generally believed to be a helper molecule in the cell until the beginning of the new millennium. This view changed. We see the potential of RNA as one of the key cellular players.

In this thesis we present a novel framework for computing sequence-structure alignments of RNA sequences. Our contribution is twofold: first, we give a graph-theoretic model for the computation of multiple sequence-structure alignments. We phrase the model as an integer linear program (ILP) and show how we can relax the ILP such that we are able to compute optimal or near-optimal solutions for the original problem. In a subsequent step, we augment the initial model with stacking energies. Stacking base pairs greatly contribute to the energetic stability of the overall structure and should therefore be additionally rewarded. We extend the original ILP such that stacking energies are incorporated.

Second, we give extensive computational results on real data from the RFAM database. We compare the performance of truly multiple sum-of-pairs sequence-structure alignments to heuristic sequence-structure alignments. We show that the objective function value of the sum-of-pairs model is generally higher compared to the heuristically inferred alignments. At the same time, we sketch the computational limits for the sum-of-pairs multiple sequence-structure model.

The computational costs for computing exact multiple sequence-structure alignments are generally very high. To validate our approach on a larger test set, we run two implementations that take two sequences as their input. LARA and sLARA—based on the initial and the stack model—compute all pairwise sequence-structure alignments and use the external program T-COFFEE to infer a consistency-based multiple sequence-structure alignment. Additionally, we run the progressive versions pLARA and psLARA on the same input data set. Our experiments on the BRALIBASE benchmark set show that our tools are top-ranked for all input classes. Furthermore, our implementations need less running time compared to similar approaches.

Subsequently, we compare two different algorithms for computing the optimal value of the Lagrangian dual and show that in our test setting the conceptually easier subgradient method is superior to the bundle method. Finally, we incorporate our Lagrangian relaxation approach into a branch-and-bound framework. We show for which instances we are able to compute provably optimal solutions and compare our results with previously published results of a branch-and-bound approach for the related quadratic knapsack problem.

## Acknowledgments

Doing a PhD is not the effort of a single person, but many people contributed and helped in various ways. First and foremost, I am grateful that my two advisors Gunnar W. Klau and Knut Reinert acted as my scientific fathers, for teaching me how to do research, for providing a stimulating and pleasant working environment, and for sharing a lot of laughs along the way. I consider it a privilege having had the two of you as my advisors. Thanks for everything.

I greatly appreciate Peter Stadler's willingness to appraise this thesis, and I want to express my gratitude for the generous financial support of the International Max Planck Research School for Computational Biology and Scientific Computing and the Deutsche Forschungsgemeinschaft.

I extend my gratitude to Prof. Vineet Bafna for hosting me in San Diego in 2006, and to Nuno, Banu, Shaojie, and Vikas for making the stay an enjoyable one.

A big 'thank you' goes out to Holger, Dave, Ole, Eva, Sally, Christopher, Tobi, Lars, Diana, Patrick, and Henning for many memorable evenings and a lot of laughs. I want to thank the members of my Viennese home base for all the fun and always providing me a place to crash: thanks Bernhard, Verena, Steff, Geddie, Tanja, Markus, Anton, Vreni, and Evi! I am indebted to Ole, Dave, and Patrick who read earlier drafts of this thesis and greatly improved it by their comments.

Last but not least I want to thank my sisters and my parents for the unconditional support in all these years, for keeping spirits high when the going got tough, and for constantly asking me what my research is actually good for and if I can make a living out of it. Ihnen ist diese Arbeit in Dankbarkeit gewidmet.

# CONTENTS

# Chapter 1 Introduction

Kumm her und huach zua,
i erzähl Dir a Gschicht.
Wolfgang Ambros
(Das Leben, die Liebe und der Tod)

## 1.1 The Past and Future of Genetics

$3,080,419,480$[1] characters encode who we are: the color of our hair, the shape of our body, and the risk for certain diseases. The joint publication of the sequence of the human genome by the Human Genome Project [26] and the private company Celera [140] marked a milestone in human history. Since then, it is possible to read the book of life: but the more we read and the more we begin to understand, the more complicated it gets.

The history of genetics started by rediscovering the work of Gregor Johann Mendel in the early 20th century. In 1865, Mendel performed studies about inheritance patterns in plants and realized that inheritance was not a random process. It took several decades to finally illuminate the molecular basis and main processes of genetics. Hugo de Vries introduced the terms *pangenesis* and *pangen* for the smallest particle responsible for inheritance, a term that was later abbreviated to *gene* by Wilhelm Johannsen. In 1910, Thomas Hunt Morgan showed that genes are located on *chromosomes* and proposed a linear arrangement of genes on the chromosomes. His student Alfred Sturtevant determined the linear order of genes on the chromosomes, however, it was still unclear what the actual molecular basis of inheritance is. The two possibilities were either DNA or proteins. DNA is a biopolymer, a chain of four different nucleotides: either adenine (A), cytosine (C), guanine (G), or thymine (T); proteins are compositions of 20 different amino acids that were first described by Berzelius in 1838.

In 1944, Oswald Avery, Colin McLeod, and Maclyn McCarty discovered that it was DNA and not proteins that harbor the genes. Building upon work by Frederick Griffith they worked with two different strains of the same bacterium, and then removed either proteins or DNA from the bacteria and showed that by removing DNA the first strain could not transform into the second strain. They did not observe this effect by removing proteins. Hence, they had conclusive evidence that DNA is the carrier of genes. In subsequent work, Hershey and Chase

[1] We calculated this number by simply summing up the length of all 24 chromosomes of the human genome from GenBank. We are aware of the fact that giving an exact number is not possible.

discovered DNA as the genetic foundation of viruses. Finally, in 1953 Francis Crick and James Watson revealed the *double helix* as the structure of DNA and they constructed a physical model for the duplication and reconstruction of partner strands. In subsequent years, massive research efforts, aimed at unravelling the mechanism that governs the processes of transforming DNA into proteins, were conducted until finally *transcription* of DNA into RNA, and the *translation* of RNA into proteins became apparent. RNA is similar to DNA, *i.e.*, it is a chain of nucleotides, but there are some important differences: RNA is single-stranded, in contrast to the double-helical structure of DNA, and the nucleotide uracil (U) replaces thymine. By folding back onto itself RNA builds hydrogen bonds and forms the *secondary structure* (see Fig. 1.2 (b) for an example).

In 1958, Francis Crick postulated the *central dogma of molecular biology* [28; 29], which essentially described the processing of genetic information as a linear flow: *DNA* is copied into *messenger RNA* (mRNA in short), and mRNA in turn serves as the template to synthesize the functional units in the cell, the *proteins*. In the original formulation, RNA solely acts as the working copy of DNA, and proteins alone are able to trigger or inhibit functions in the cell. In the years to come, new technologies were developed that revolutionized molecular genetics. Sanger [123] developed the first sequencing method that allows the determination of a DNA sequence. Weber and Myers [145] introduced *whole-genome shotgun sequencing*: this technique was used by the private company Celera to determine the genomic sequence of the human, the fly, and the mouse. The standard sequencing method nowadays is *pyrosequencing* [119] which sharply reduces the costs for (re-)sequencing of genomes.

Together with the growing amount of available sequence data, techniques like *microarrays* allow the measurement of genes that are expressed. *Gene regulation* deals with the mechanisms that control the expression of proteins, and several key players were identified: *promoters* and *transcription factors* that control the transcription of DNA to RNA, or *enhancers*, for instance, that regulate the transcription of certain DNA sequences. Still, the main workflow was still assumed to be valid to a large extend.

There were, however, already divergent opinions from the beginning. Carl Woese [152] postulated the possibility that RNA was not a helper medium from the start, but that RNA sequences—having catalytic properties—built the basis of the origin of life. Altman [54] and Cech [156] finally provided evidence that RNA sequences are indeed able to perform catalytic actions. Walter Gilbert expressed the possibility of an *RNA world* [50] as the origin of life in his comment on the discovery of catalytic RNAs.

In the 1990s, scientific findings finally changed the prevalent understanding of the molecular mechanisms behind genetics. Lee *et al.* [93] describe small RNAs that regulate proteins. Their paper marks the discovery of a new class of catalytic RNAs, the so called *microRNAs*. Their full importance was not realized until 2001 when a series of papers [92; 90; 88] describes them as an abundant class of RNAs. In 2002, *Science Magazine* announced RNA as the breakthrough of the year [27]. Since then, the interest in *noncoding RNAs*, *i.e.*, functional RNA sequences not coding for proteins, has risen tremendously: examples comprise microRNAs, snoRNAs, siRNAs, or piRNAs.

First scans of several genomes [143; 144] point—despite a high false positive rate—to a large number of possible functional elements. At the time this thesis is being written, recent studies [25; 128; 76; 24; 23; 146] even challenge our perception of three layers that are separated, since they provide evidence that the entire transcriptome is in fact a puzzle of overlapping transcripts from both strands of the helix and that almost the entire genome is transcribed at some point. What all these studies have in common is that they substantiate the role of RNA as one of the major players in driving cellular processes. For most noncoding-RNA families, however, the actual function is mostly unknown. One major exception is the above mentioned class of microRNAs: these 22 nucleotide long RNAs are known to be involved in a wide range of mechanisms, ranging from cancer genesis and classification [100; 98; 111], silencing of genes [120], the diversity of anti-bodies [85], or to the division of stem cells [57].

In biology, sequences of high similarity usually share the same structure or function. Therefore, one of the main tasks in bioinformatics is the comparison of different sequences to search for *conserved* patterns, *i.e.*, subsequences that occur in all sequences. *Alignments* are a way to compare different sequences. We write the sequences on top of each other such that characters that are evolutionary related are in the same column. We model genetic variability by inserting *gap characters* into the sequences. Figure 1.1 gives a small example of a multiple sequence alignment.

```
DQ891020.1|     cgggaaaggaacccattgcaaccaagtcgaagtgatagccacactg----
BX640422.1|     actgaagggcaag---------aatcaggagttctgcctgaccgccttca
CP000077.1|     aggcaaa---------------------atggtgtcctttacctatgacg
DQ778054.1|     gatggcaggcaaa---------agagaaatggttatcattacattt----

DQ891020.1|     --------aaggatgggaggaaaatctgcctg------------------
BX640422.1|     tgtcgggcagaagcctggtccgggcgtgcctgtccgacgcg---------
CP000077.1|     acaatggtaagacaggtagaggagctgtaagc------------------
DQ778054.1|     --------aagagcggcgaaacatttcaggtcgaagtcccgggcagtcaa

DQ891020.1|     ---gacccagatgctccc---agaatcaagaaaattgtacagaaaaaatt
BX640422.1|     ---ggacacgagcacgacacgtggttcgacaccatgcttggctttgccat
CP000077.1|     ---gagaaagatgctccaaaagaattattagacatgttagcaagagcaga
DQ778054.1|     catatagactcccagaaaaaagccattgaaaggatgaaggacacattaag
```

Figure 1.1: An example of a multiple sequence alignment of four input sequences. Characters that are evolutionary related are written in the same column of the alignment. Insertion and deletions are modelled by the insertion of gap characters.

Alignments provide the basis for various subsequent tasks: phylogenetic analysis, the study of evolutionary processes, or searching for homologous sequences. The alignment of DNA sequences based on sequence information works well, because the sequence remains evolutionary conserved, *i.e.*, by considering only the characters of the sequence it is possible to build reliable alignments. In the case of RNA the situation is different. Although two RNA sequences can be divergent on the sequence level, they might still share a common structure. This is due to *compensatory mutations*, a central feature in RNA evolution: compensatory mutations of bases that form hydrogen bonds do change the sequence, but they do not alter the secondary structure. As an example, recent studies [48; 138; 139; 148] have shown that the structural similarity is a dominant factor and has to be taken into account in the alignment step of RNA sequences. Instead of computing pure sequence-based alignments we then compute *sequence-structure alignment*, *i.e.*, alignments that consider both the sequence and structure information. Due to the recent findings about the importance of noncoding-RNAs, the development of new approaches for the alignment of RNA sequences that take the secondary structure into account is a worthwile endavour.

In the following, we will present constraints that a valid secondary structure must satisfy. Additionally, we show different representations of RNA structures and sketch the algorithms that compute secondary structures given only the RNA sequence.

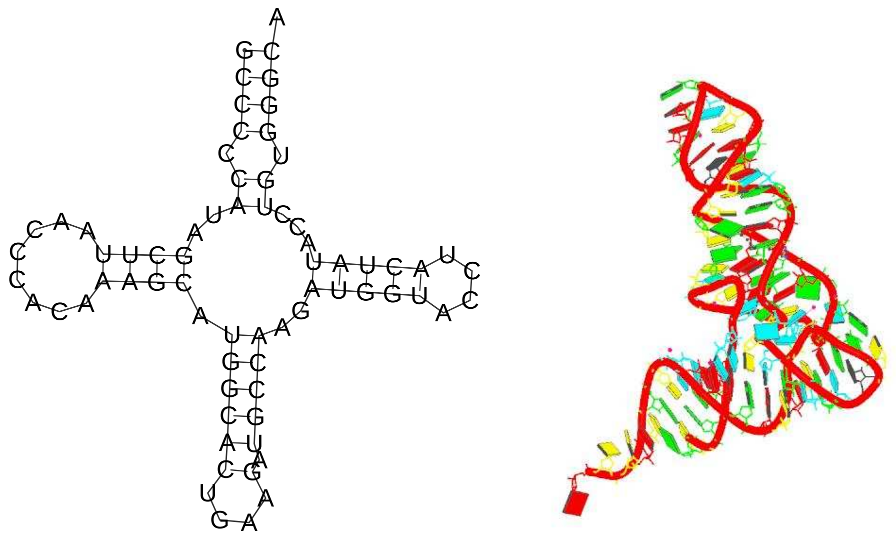
## 1.2 RNA Structures and Structure Prediction

RNA sequences can be represented as strings over the four letter alphabet $\Sigma_{\mathrm{RNA}} = \{A, G, C, U\}$, and—in contrast to DNA sequences—an RNA sequence folds back onto itself and builds hydrogen bonds between complementary nucleotides. We distinguish between the set of *canonical* base pairs G-C and A-U, and the *wobble* base pair G-U. A set $P$ of pairings forms the *secondary structure* of a sequence $s$. The elements of the secondary structure form noncovalent bindings that give rise to the 3D structure of an RNA molecule. Figure 1.2 gives the primary, secondary, and tertiary structure of a tRNA sequence. We call the determination of the secondary and tertiary structure of a sequence $s$ the *structure prediction problem*.

The Holy Grail of RNA structure prediction research is the determination of the tertiary structure of a given sequence, and not only of the secondary structure elements. Unfortunately, the knowledge about the tertiary folding process is far from being complete, similar to the problem of determining the 3D structure of a protein given only its amino acid sequence. Functional RNA molecules usually have a distinctive tertiary structure that is important for their function, and additionally their secondary structure remains evolutionary conserved. Therefore, most of the structure prediction research focuses on the easier problem of predicting the secondary structure of an RNA molecule, since a characteristic secondary structure forms a scaffold for the tertiary structure. This led to efficient algorithms—based on dynamic programming—for a variety of structure prediction problems. We want to stress, however, that the ultimate goal in structure prediction is still the determination of the 3D structure, and not only predicting the secondary structure.

GCCCCCAUAGCUUAACCCACAAAGCAUGGCACUGAAGAUGCCAAGAUGGUACCUACUAUACCUGUGGGCA

(a) Primary sequence



(b) Secondary structure (c) Tertiary structure

Figure 1.2: Primary, secondary, and tertiary structure of a tRNA sequence. The secondary structure (b) was created using RNAFOLD from the Vienna RNA package [66]. We downloaded the tertiary structure (c) from the PDB database.

In the following, Sect. 1.2.1 gives a formal description of RNA secondary structures, along with typical representations for these structures. Finally, Section 1.2.2 describes the energy model that builds the basis for most of the structure prediction algorithms.

### 1.2.1 RNA Structures

Formally, the secondary structure $P$ of a sequence $s \in \Sigma_{\text{RNA}}^*$ is a list of base pairs $(i, j)$ such that the following constraints are satisfied:

(a) for each position $i \in 1, \ldots, |s|$ there is at most one element $(k, l) \in P$ such that $i = k$ or $i = l$, *i.e.*, every nucleotide takes part in at most one base pairing.

(b) for every base pair we have $|i - j| > 3$, *i.e.*, the minimal distance between two paired nucleotides has to be greater than 3 due to physical reasons.

(c) paired bases have to be nested, *i.e.*, $\forall (i, j), (k, l) \in P$ we have $k \in [i, j] \leftrightarrow l \in [i, j]$.

Constraint (a) ensures that a valid secondary structure does not include base triplets or quartets. Such motifs do occur, but only in tertiary structures and they are excluded for secondary structures. The backbone of an RNA sequence cannot bend too sharply; hence, constraint (b) sets the minimal number of residues between any paired bases to three. Finally, constraint (c) marks the major difference between secondary and tertiary structures: the nested character of a valid secondary structure allows the decomposition of the overall structure into smaller independent subproblems. Most RNA related research makes use of this decomposability property and devises algorithms based on *dynamic programming*. Constraints (a)-(c) give rise to a hierarchy of possible structures, namely:

(a) Plain: there are no base pairs at all, *i.e.*, only the sequence information is available.

(b) Chain: every nucleotide is incident to at most one base pair, and there are no nested base pairs, *i.e.*, $\forall (i,j),(k,l) \in P$ we have either $j < k$ or $l < i$.

(c) Nested: every nucleotide is incident to at most one base pair, and we only have nested base pairs.

(d) Crossing: every nucleotide is incident to at most one base pair, and we have crossing base pairs.

(e) Unlimited: there are no restrictions at all.

A base pair that violates constraint (c) is said to form a *pseudoknot*. Pseudoknots are a first step from secondary structures towards tertiary structures, and they exert important biological functions [131]. Like for the complete tertiary structure prediction problem, however, we have an incomplete understanding of folding kinetics and properties of pseudoknots.

There are various representations for secondary structures. Beside the 2D-plot from Fig. 1.2, Fig. 1.3 shows five major representation forms for secondary structures. Due to the nested structure of the pairings we are able to draw a valid secondary structure as an *outer planar graph* with the residues being aligned on a circle and base pairs forming chords of the graph. Another representation frames secondary structures as *trees*: the parent/child relationship of the nodes is given by the nesting of the paired bases. The sequential order of the sequence defines the order of sibling nodes. There are different resolutions for the labeling of the nodes: internal nodes correspond to paired bases, whereas the leaves of the tree represent unpaired bases, or nodes might correspond to stacked regions of the secondary structure. See Fig. 1.3 (c) for an illustration where the nodes correspond to paired and unpaired bases. The *mountain plot* encodes for each residue $i$ the number of pairings that enclose $i$. Each mountain corresponds to a stacked region in the secondary structure. The *dotplot* contains more information than just a single structure: the matrix is divided into two triangles, with the lower triangle containing one single structure of the sequence indicated by black squares. The upper triangle of the dotplot contains the probability for each pair of nucleotides to pair. The bigger the square is, the higher is the probability to form a base pair. Finally, a more technical description of RNA secondary structure is the *Vienna notation*: brackets and dots denote paired and unpaired bases, respectively. Since we do not allow pseudoknots, there is a unique correspondence between paired nucleotides and pairs of opening and closing brackets.

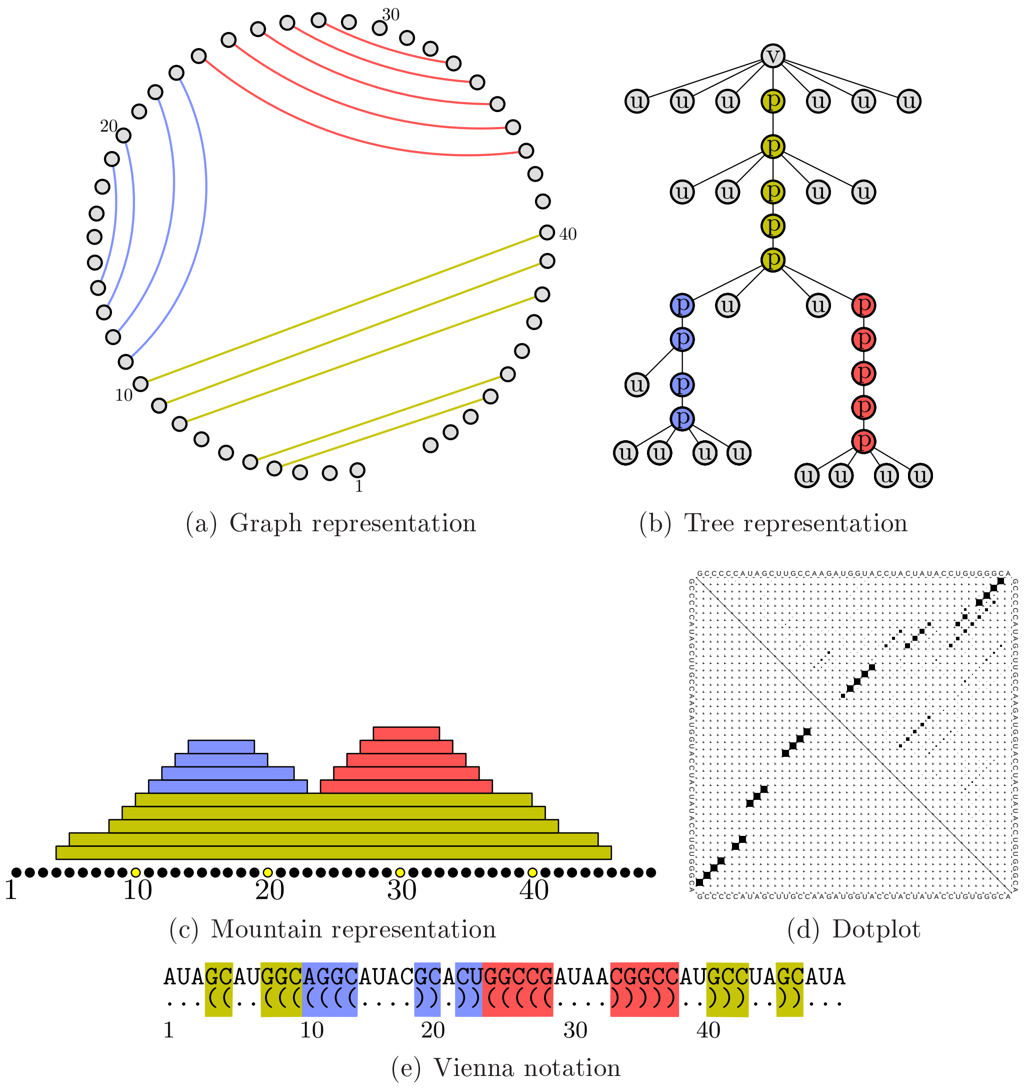


Figure 1.3: Various representations for RNA structures. We have the graph representation (a), RNA structures as a tree (b), the mountain plot (c), the dotplot (d), and the Vienna string notation (e).

So far, we have only discussed the properties that a valid secondary structure must satisfy, and we presented various representations of secondary structures. We did not, however, sketch the algorithms to compute the secondary structure given only the nucleotide sequence. We will make extensive use of these algorithms in our computational experiments, because our default scoring system relies on them. Therefore, this will be the topic of the following section which is mainly based on the exposition of Hofacker and Stadler [69].

## 1.2.2 RNA Structure Prediction

The first attempts to compute the secondary structure of an RNA sequence $s$ aim at maximizing the number of paired base pairs, *i.e.*, we want to find a set $\mathcal{P}$ over all possible structures $P$ such that we have

$$|\mathcal{P}| = \max_{\bar{P} \in P} |\bar{P}| \ .$$

Nussinov *et al.* [110] give recursions for computing $\mathcal{P}$. The recursion handles the two basic cases of a nucleotide $i$, *i.e.*, whether it is paired or unpaired. Let $E[i, \ldots, j]$ be the maximal number of base pairs for substring $s[i, \ldots, j]$. Then, the recursion reads

$$E[i, \ldots, j] = \max \begin{cases} E[i+1, \ldots, j] \\ \max_{k,(i,k)\text{pair}} \left( E[i+1, \ldots, k-1], E[k+1, \ldots, j] \right) + 1 \end{cases} \ .$$

Figure 1.4 gives an illustration.

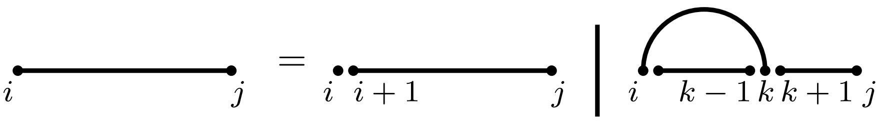


Figure 1.4: The Nussinov algorithm computes the maximal number of paired base pairs of a sequence $s$. The recursions distinguish two basic cases: either a nucleotide $i$ is paired or unpaired.

Due to its simple objective function the experimental performance of the Nussinov algorithm is not satisfactory. Hence, more sophisticated algorithms have been developed that incorporate more knowledge about secondary structures. Nested secondary structure allow the decomposition of the total structure into different *loops*: given an element $(i, j) \in P$, we call nucleotide $h$ *accessible from* $(i, j)$ if there is no other element $(k, l) \in P$ such that $i < k < h < l < j$. A base pair $(k, l) \in P$ is *accessible from* $(i, j)$ if both $k$ and $l$ are. We call the $k - 1$ elements of $P$ and $k'$ unpaired bases that are accessible from the paired nucleotides $(i, j)$ the *k-loop* closed by $(i, j)$. We now distinguish different types of loops according to the number of base pairs accessible from $(i, j)$:

1. A 1-loop is called a *hairpin* loop.
2. If only one single base pair $(i', j')$ is accessible from $(i, j)$, then we call this 2-*loop* a
   (a) *stacked pair* if we have $i' - i = 1$ and $j - j' = 1$.
   (b) *bulge loop* if either $i' - i > 1$ or $j - j' > 1$.
   (c) *interior loop* if both $i' - i > 1$ and $j - j' > 1$.

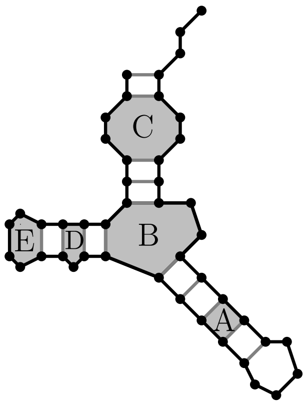


Figure 1.5: The are five main elements in RNA structures: stacked base pairs (A), multiloops (B), interior loops (C), bulges (D), and hairpin loops (E). The black circles and grey lines denote residues and hydrogen bonds between complementary residues.

3. We call a *k-loop* with $k \geq 3$ a *multiloop*.

Figure 1.5 gives an illustration for the different loop types.

The $k$-loop decomposition builds the foundation of the standard *energy model* to predict the secondary structure. Each loop $l$ has an energy contribution $e(l)$, and the total *free energy* of a structure $P$ is given by $\sum_{l \in P} e(l)$. Hence, we switch our objective function for structure prediction from the number of paired bases to the free energy of the ensemble. In particular, we are interested in the structure that has the *minimum free energy* among all possible structures.

Note that the dominant terms for the energy calculation are stacked base pairs, hydrogen bonds, and loop energies. The energy contributions depend on the type and the size of the loop. Furthermore, the overall structure is stabilized by consecutive *stacking* of paired bases: we call consecutive stacked base pairs a *stacked region* or a *stem*. Zuker and Stiegler [161] first proposed recursions for the computation of the minimum free energy, and the main concepts remain valid since then. The time and space complexity of the algorithms is in $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$, respectively. There are two main implementations of the folding recursions, MFOLD [160] and RNAFOLD [66]. Recently, Wexler *et al.* [147] gave a reduction of the running time to $\mathcal{O}(n^2)$.

The recursions use four DP tables—$F$, $C$, $M$, and $M_1$—for storing intermediate results:

(a) $F[i,\ldots,j]$ gives the optimal energy value for subsequence $s[i,\ldots,j]$.

(b) $C[i,\ldots,j]$ gives the optimal energy value for subsequence $s[i,\ldots,j]$ given that $(i,j)$ forms a base pair. This case covers hairpin and interior loops.

(c) $M[i,\ldots,j]$ gives the optimal energy value for subsequence $s[i,\ldots,j]$ such that $s[i,\ldots,j]$ is part of a multiloop with at least one outgoing stem.

(d) $M_1[i, \ldots, j]$ gives the optimal energy value for subsequence $s[i, \ldots, j]$ such that $s[i, \ldots, j]$ is part of a multiloop with exactly one outgoing stem and we have a closing base pair $(i, h)$ with some $h$ satisfying $i < h \leq j$.

Figure 1.6 shows how the computation of the entire optimal secondary structure decomposes into the different cases. For the actual recursions the reader is referred to [69].

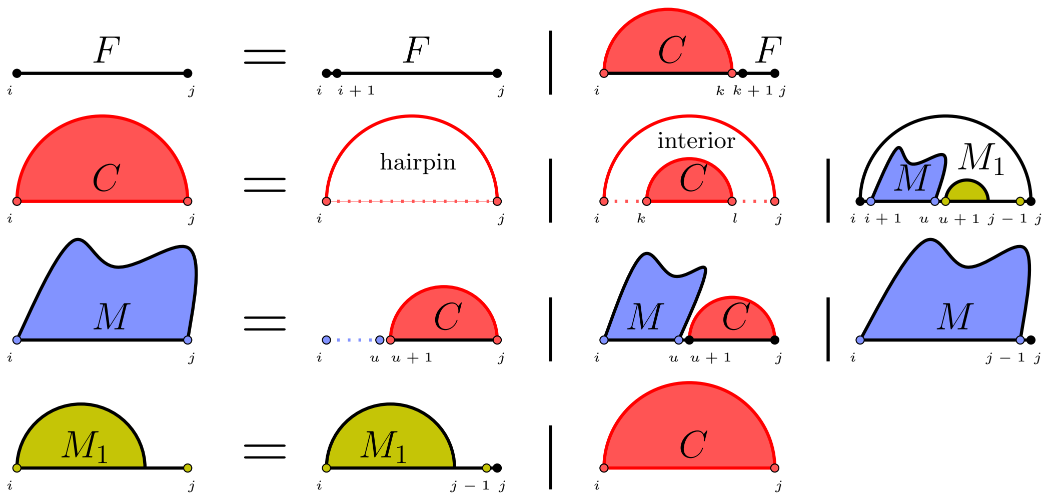


Figure 1.6: A visualization of the recursions energy model for the *ab initio* prediction of RNA secondary structures using the loop-base energy model: all possible RNA structures can be decomposed into these subcases. The illustration is taken from [69].

The energetic contribution of a multiloop is given by $E_{ML} = a + b \cdot \text{degree} + c \cdot \text{size}$. Therefore, table $M_1$ is necessary to keep track of the size and the degree of multiloops. The recursions model a multiloop as the concatenation of a substructure that contains at least one stem, and another substructure that is enclosed by a base pair and contains exactly one outgoing stem. Observe that $F[1, \ldots, |s|]$ only gives the minimum free energy value of the optimal structure. One has to backtrack starting from $F[1, \ldots, |s|]$ to compute the structure.

We now have a model that is much more developed compared to Nussinov's algorithm. The problem is, however, that the minimum free energy structure needs not match the secondary structure that an RNA molecule exhibits. Another way to describe the structural features of an RNA sequence $s$ are *base pair probabilities*. Instead of giving one single structure that we compute using the energy model, McCaskill [105] proposed a way to compute the *partition function* of $s$, and subsequently derive *base pair probabilities* for every pair $(i, j)$ of $s$.

The partition function is an important term from statistical mechanics and links macroscopic phenomena, like the free energy or the entropy of a system, to the microscopic world of molecules or particles. Assume we are given a system of different states and energy levels $E_i$, then the probability $p_i$ to find the system in the $i$th state follows the *Boltzmann distribution*, *i.e.*,

$$p_i = e^{-E_i\beta}$$

with $\beta = \frac{1}{kT}$: $T$ and $k$ are the temperature in Kelvin and the Boltzmann constant, respectively. All $p_i$ have to sum up to 1, because the system has to be in some state. Hence, we have to compute a scaling constant $c$ for the energy values such that $\sum_i p_i = 1$. Then, we have the following terms:

$$\begin{aligned} 1 &= \sum_i c e^{-E_i\beta} \\ c &= \frac{1}{Z} \text{ with } Z = \sum_i e^{-E_i\beta} \quad . \end{aligned}$$

$Z$ is called partition function (observe that $Z$ originates from the German word “Zustandssumme” which captures the meaning of the partition function). Now, we can give the probability for state $i$ as

$$p_i = \frac{e^{-E_i\beta}}{Z} \quad .$$

We can use $Z$ to compute the probability $p_{ij}$ that $(i, j)$ forms a base pair. The main idea is to compute the partition function $\hat{Z}_{ij}$ for structures outside of subsequence $s[i, \ldots, j]$. Then, we need to compute the partition function for structures that include the base pair $(i, j)$. The probability reads

$$p_{ij} = \hat{Z}_{ij} Z_{i+1,j-1} e^{\frac{-E_{ij}\beta}{Z}}$$

with $E_{ij}$ being the energy contribution of base pair $(i, j)$. We will make extensive use of base pair probabilities as structure scores in Chap. 5.

As stated in Sect. 1.2.1 the ultimate goal of RNA structure prediction is the prediction of the entire tertiary structure of an RNA sequence $s$. Both the Nussinov *et al.* and the Zuker/Stiegler algorithm, however, consider only nested structures and discard all possible pseudoknots. There are some approaches [118; 114; 115; 34; 35; 151] that aim at predicting secondary structures including pseudoknots, but all these approaches suffer from two main problems: first, the algorithms are restricted to special classes of pseudoknots, because the general problem of predicting arbitrary pseudoknotted structures was shown to be NP-complete [99]. Even on these restricted cases, the algorithms remain computationally expensive which limits their applicability to short sequences. Second, we lack a set of sound energy parameters for pseudoknotted structures. The energy parameters [102; 155; 104] for nested secondary structures are empirically derived from optical melting experiments. These experiments, however, do not work anymore in the case of pseudoknotted structures, leading to other approaches like learning the parameters from a positive and negative set [34]. Due to the importance of pseudoknots [131], this is an area of active research.

## 1.3 Overview

This thesis introduces a novel model for the computation of multiple sequence-structure alignments which is based on mathematical optimization. Chapter 2 provides all mathematical tools that we will use throughout the rest of the thesis. This includes basic definitions from graph theory in Sect. 2.1, and from linear programs and integer linear programs in Sect. 2.2. Subsequently, Sect. 2.3 gives a description of Lagrangian relaxation and how the associated *dual problem* can be solved. Finally, Sect. 2.4 briefly covers concepts from statistics of which we will make use during the evaluation of our computational results. Chapter 3 describes the main algorithms and concepts for sequence-structure alignments that were presented in the past.

Chapter 4 describes our formulation for the computation of exact multiple sequence-structure alignments. We start by formally defining sequence-structure alignments and show how we can phrase the problem definition in graph-theoretical terms. We prove that the formulation matches the problem that we gave before. Section 4.2 presents the transformation of the graph-based model into an *integer linear program*. We identify a suitable class of constraints that we are able to relax in a Lagrangian fashion. We solve the relaxed problem to provable optimality. We give the computation of a feasible solution to the original problem afterwards: we describe a reduction to the computation of *maximum weight matchings*.

We present an important extension to our initial model in Sect. 4.3: the scoring of consecutive stacked base pairs. Again, we start by formally defining the problem. We then give an integer linear program that matches the problem definition, and this time we drop two classes of constraints and move them to the objective function afterwards.

Chapter 5 starts by describing the input and the parameters that significantly influence the solution process of the models described in Chapt. 4. Thereafter, we give results on exact multiple sequence-structure alignments in Sect. 5.3. In the following section, we present the computational results on the BRAliBase benchmark set. Based on the pairwise case of our multiple model we heuristically compute multiple alignments by either using the external software package T-Coffee or by progressively aligning all input sequences. We compare our results to several state-of-the-art programs both in terms of the quality of the solutions and the running time. Thereafter, we compare the performance of the *subgradient* to the *bundle method*. Finally, we implemented our approach within a *branch-and-bound* framework to obtain provably optimal solutions even if the bounds do not coincide. We report on the applicability and the limits of this method. We conclude the thesis by discussing the major findings and sketching possible lines of future research.

# Chapter 2 Mathematical Preliminaries

Well, your faith was strong,
but you needed proof.
Leonard Cohen
(Hallelujah)

This chapter introduces concepts that we will use throughout the thesis. First, Sect. 2.1 outlines elementary graph theory, whereas Sect. 2.2 and 2.3 describe the basics of (integer) linear programs and how to derive solutions using Lagrangian relaxation. Finally, Sect. 2.4 presents some statistical methods of which we will make use in Sect. 5.

The following exposition is based on various textbooks, for details the interested reader is referred to [33; 10; 107; 116]. Several sources [55; 46; 45; 94] provide additional information especially on Lagrangian relaxation.

First, we introduce some notation from linear algebra: $\mathbb{R}$,$\mathbb{R}_+$, $\mathbb{Z}$, and $\mathbb{Z}_+$ denote the sets of real, nonnegative real, integer, and nonnegative integer numbers, respectively. Given an ordered finite set $E = \{e_1, e_2, \ldots, e_n\}$ and a field $X$, we denote by $X^E$ the set of vectors in which we index the components of each vector by the elements in $E$. In the case of $E = \{1, \ldots, n\}$ we write $X^n$. We consider vectors as column vectors and denote row vectors as transposed column vectors $y^T$. Given a set of vectors $X^E = \{x_1, \ldots, x_k\}$, we call $x$ a *convex combination* of $x_1, \ldots, x_k$ if $x = \sum_{i=1}^{k} \lambda_i x_i$, with $\lambda_i \geq 0$ and $\sum_{i=1}^{k} \lambda_i = 1$.

## 2.1 Graph Theory

A *graph* is a pair $G = (V, E)$ where the sets $V$ and $E$ denote the *vertices* and *edges* of the graph. An edge $e = (u, v) \in E$ denotes a pair of nodes $u$ and $v$. Both $u$ and $v$ are said to be *incident* to edge $e$. Two nodes $u$ and $v$ are *adjacent* if there exists an edge $e = (u, v) \in E$. We denote the number of nodes and edges by $|V|$ and $|E|$.

We call $G' = (V', E')$ a *subgraph* of $G$ if $V' \subseteq V$ and $E' \subseteq E$. If $G'$ is a subgraph of $G$, then we call $G$ the *supergraph* of $G'$.

We call an alternating sequence of vertices and edges $(v_0, e_0, v_1, \ldots, e_{n-1}, v_{n-1})$ with $e_i = (v_i, v_{i+1}), 0 < i < n-1$ where all nodes and edges are distinct a *path of length n*. If the path is closed, *i.e.*, $v_0 = v_{n-1}$, we call the path a *cycle*.

If we are able to partition the vertex set $V$ of $G$ into $k$ disjoint sets $V'_0, \ldots, V'_{k-1}$ such that no two vertices within the same set are adjacent, we call $G$ a *k-partite* graph. A *clique* in a graph $G = (V, E)$ is a subgraph of $G'$ with $V' \subseteq V$ and $E' \subseteq E$ such that every pair of nodes in $V'$ is adjacent.

A *directed graph*, or *digraph* in short, is a pair $D = (V, A)$ with $V$ and $A$ being the sets of vertices and directed *arcs*. An arc $a = (u, v) \subseteq V \times V$ is an ordered pair of elements of $V$, and we call *a incident from* $u$ and *incident to* $v$. Two nodes $u$ and $v$ are adjacent if there exists an arc $a = (u, v) \in A$. For $a = (u, v)$, we call $u$ and $v$ the source and target node of $a$. The two functions $s(a)$ and $t(a)$ return the source and target node for an arc $a$.

A *mixed graph* $G = (V, E, A)$ consists of a vertex set $V$, the edge set $E$, and a set of directed arcs $A$. A path $p = (v_0, e_0, v_1, \ldots, e_{n-1}, v_{n-1})$ in a mixed graph is an alternating series of vertices and edges or arcs such that $e_i = (v_i, v_{i+1}) \in E$ or $e_i = (v_i, v_{i+1}) \in A$ with $0 < i < n - 1$. All vertices and edges of the path are distinct. If at least one edge $e_i \in E$ and one arc $e_i \in A$ are part of path $p$, then we call $p$ a *mixed path*. We call $p$ a *mixed cycle* if $v_0 = v_{n-1}$.

The *transitive closure* of a graph $G = (V, E)$ is defined as the graph $G' = (V, E')$ such that $E'$ contains an edge $e = (u, v)$ if there exists a path from $u$ to $v$ in $G$. The transitive closure of a digraph $G$ is identically defined as for undirected graphs.

An *independent set* of a graph $G = (V, E)$ is defined as a set $I' \subseteq V$ such that there are no two vertices $i, j \in I'$ that are adjacent. The *maximal independent set* is the subset $I'$ with the maximal number of vertices. The computation of a maximal independent set is NP-complete [49].

A *matching* in a graph $G = (V, E)$ is an edge set $M \subseteq E$ such that no two edges $e_0, e_1 \in M$ share the same vertex. If $|M| = \frac{|V|}{2}$ holds true, we call $M$ a *perfect matching*. For a graph where each edge $e_i \in E$ is associated with an edge weight $w_i$, the *matching of maximum weight* is the edge set $M$ such that $M$ is a matching and $\sum_{i=0}^{|M|-1} w_i$ is maximal.

## 2.2 Linear Programming

Intuitively speaking, a *linear programming problem* calls for the computation of an optimal solution with respect to a linear objective function, satisfying a set of linear constraints. The following exposition assumes that we want to maximize the value of the objective function, but we can easily transform the definitions to the case of minimization problems.

Let $A \in \mathbb{R}^{m \times n}$ be a matrix and let $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ two vectors. A linear programming problem is a system of $Ax \leq b$ of linear inequalities and a linear objective function $c^T x$. We call a vector $\bar{x} \in \mathbb{R}^n$ such that $A\bar{x} \leq b$ a *feasible solution* of the problem. If there does not exist such a vector $\bar{x}$, we call the linear program *infeasible*. The *linear programming problem*, or LP in short, addresses the computation of an *optimal feasible solution* $x^*$ with respect to the objective function $c^T x$, *i.e.*,

$$c^T x^* = \max\{c^T x \mid Ax \leq b\}$$

A different notation is

$$\begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax \le b \ . \end{aligned} \tag{P}$$

An important concept in linear programming is *duality theory*. Every linear program (P) has its *dual problem* which is defined as

$$\begin{aligned} \min \quad & y^T b \\ \text{subject to} \quad & A^T y = c \\ & y \ge 0 \end{aligned} \tag{D}$$

We call (P) the *primal problem* and (D) its associated dual. Observe that the dual of (D) is again (P). One deep result of duality theory describes the relationship between the primal and the dual problem.

**Theorem 2.1** (Strong duality theorem of linear programming)**.** *Let* (P) *and* (D) *be linear programs which are dual to each other.*

*(a) If* (P) *and* (D) *have feasible solutions, then they have optimal solutions and the optimal objective function values are the same.*

*(b) If* (P) *is infeasible,* (D) *is either infeasible or unbounded.*

*(c) If* (P) *is unbounded,* (D) *is infeasible.*

The *simplex method*, developed by George Dantzig in 1947, computes an optimal solution of a linear program. Although the algorithm has exponential worst-case complexity, it has proven to work well in practice. Recently, Spielman and Teng [130] gave an explanation for the excellent average-case performance of the simplex algorithm. Karmarkar [77] introduced the *interior point method* to solve linear problems in polynomial time in the worst case.

Linear programs formulating real-world problems often add integrality constraints on the set of feasible solutions. We call such a linear program a *general integer linear program*, or ILP in short, which has the general form

$$\max \quad c^T x \tag{2.1}$$

$$\text{subject to} \quad Ax \le b \tag{2.1.1}$$

$$x \in \mathbb{Z} \tag{2.1.2}$$

If we substitute constraints (2.1.2) by $x \in \{0, 1\}$, we get a *(0/1)-integer linear program*. Computing optimal solutions for an ILP is NP-complete [49]. Popular methods that aim at solving ILPs, for example *branch-and-cut* algorithms, use the *LP-relaxation* of an integer linear program: by dropping the integrality constraints of an ILP, we get the corresponding LP-relaxation. Another popular method to tackle ILP is *Lagrange relaxation* which we will present in the following sections.

## 2.3 Lagrange Relaxation

Consider the following ILP with $A \in \mathbb{R}^{k \times n}$ and $C \in \mathbb{R}^{l \times n}$:

$$\begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & Cx \leq d \\ & x \in \mathbb{Z}_+^n \end{aligned} \tag{IP}$$

Suppose that the constraints $Ax \leq b$ are *difficult* constraints, whereas optimizing over constraints $Cx \leq d$ alone is easy. The main idea is to drop the complicating constraints which yields an ILP that is easier to solve than the original one. Then, consider the following ILP with $\lambda \in \mathbb{R}_+^k$:

$$\begin{aligned} \max \quad & c^T x + \lambda^T (b - Ax) \\ \text{subject to} \quad & Cx \leq d \\ & x \in \mathbb{Z}_+^n \end{aligned} \tag{LR}$$

We call LR($\lambda$) the *Lagrangian relaxation* of the original problem (IP), and the vector $\lambda$ the *Lagrangian multipliers*. Lagrangian multipliers act as penalty terms that become active as soon as constraints $Ax \leq b$ are violated. The following lemma states that $LR(\lambda)$ provides a bound on the optimal value $z_{\mathsf{ip}}$ of (IP).

**Lemma 2.1.** *$LR(\lambda)$ is a relaxation of* (IP) *for all $\lambda \geq 0$.*

A consequence of Lemma 2.1 is that $z_{\mathsf{ip}} \leq LR(\lambda)$ for all $\lambda \geq 0$. Defining a set $Q$ as $Q = \{x \in \mathbb{Z}_+^n \mid Cx \leq d\}$, we can see $LR(\lambda)$ from a different viewpoint, *i.e.*,

$$\max\nolimits_{x_i \in Q} \quad c^T x_i + \lambda^T (b - Ax_i) \tag{2.2}$$

Now, $LR(\lambda)$ is the maximum of a finite set of linear functions in $\lambda$, and therefore it is convex and piecewise linear. We are interested in the tightest bound, *i.e.*, we want to find the value of $\lambda$ that minimizes $LR(\lambda)$:

$$\begin{aligned} \min\nolimits_{\lambda \geq 0} \quad \max \quad & c^T x + \lambda^T (b - Ax) \\ \text{subject to} \quad & Cx \leq d \\ & x \in \mathbb{Z}_+^n \end{aligned} \tag{LD}$$

We call problem (LD) the *Lagrangian dual* of (IP) with respect to $Ax \leq b$. $z_{\mathsf{ld}}$ denotes the optimal value of the Lagrangian dual. It is important to state that the strong duality theorem from Sect. 2.2 does not hold true anymore for the Lagrangian dual. Instead, we have *weak duality* in the case of the Lagrangian dual.

**Lemma 2.2.** *We have $z_{\mathsf{ip}} \leq z_{\mathsf{ld}}$.*

Formulation 2.2 provides another description about the relationship between the optimal value $z_{\mathsf{ip}}$ and $z_{\mathsf{ld}}$: computing the optimal value of the Lagrangian dual is equivalent to the search for a convex combination $x^*$ of elements in $Q$ that satisfies the dropped constraints $Ax \leq b$ as well. Then, we have $c^T x^* = z_{\mathsf{ld}}$. The main observation now is that computing $x^* \in \text{conv}(Q)$ with $Ax^* \leq b$ is dual to the Lagrangian dual. In this case strong duality of linear programming applies. It is important to state that this also implies that computing the optimal value of the Lagrangian dual does not necessarily yield a solution that is also valid for (IP).

An obvious question is the relationship between $z_{\mathsf{ip}}$, $z_{\mathsf{ld}}$, and $z_{\mathsf{lp}}$.

**Lemma 2.3.** *For $z_{\mathsf{ip}}$, $z_{\mathsf{ld}}$, and $z_{\mathsf{lp}}$ we have $z_{\mathsf{ip}} \leq z_{\mathsf{ld}} \leq z_{\mathsf{lp}}$. Additionally, the following holds true:*

*(a) We have $z_{\mathsf{ip}} = z_{\mathsf{ld}}$ for all cost vectors $c$ if and only if*

$$\mathit{conv}(Q \cap \{x \mid Ax \leq b\}) = \mathit{conv}(Q) \cap \{x \mid Ax \leq b\} \ .$$

*(b) We have $z_{\mathsf{lp}} = z_{\mathsf{ld}}$ for all cost vectors $c$ if and only if*

$$\mathit{conv}(Q) = \{x \mid Cx \leq d\} \ .$$

This means that $z_{\mathsf{ip}}$ and $z_{\mathsf{ld}}$ coincide if the polyhedron that is spanned by the complicating constraints $Ax \leq b$, *i.e.*, the set $P = \{x \in \mathbb{R}^n_+ \mid Ax \leq b\}$, has integer extreme points. Furthermore, if the LP-relaxation on $Cx \leq d$ has integral extreme points, then $z_{\mathsf{ld}}$ equals $z_{\mathsf{lp}}$, *i.e.*, the value of the LP-relaxation of the original ILP. Figure 2.1 shows a small polyhedron together with $z_{\mathsf{ip}}$, $z_{\mathsf{ld}}$, and $z_{\mathsf{lp}}$.

A special case of relaxing an ILP in a Lagrangian fashion is *Lagrangian decomposition* [56; 129] which is also known as *variable splitting* [106] or *variable layering* [51]. The main idea is to copy or rename variables in some of the constraints and treat them as independent variables afterwards. We must, however, enforce that the decoupled variables have the same values, *i.e.*, we have to add equality constraints to the ILP. In a subsequent step, we drop the equality constraints and move them to the objective function associated with Lagrangian multipliers. Consider the following ILP:

$$\begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & Cx \leq d \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

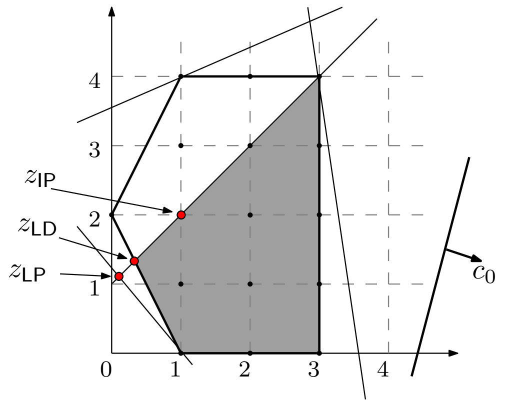


Figure 2.1: The thick lines span the polyhedron given by $\operatorname{conv}(Q)$, *i.e.*, the integer points of the polyhedron induced by the easy constraints $Cx \le d$. The shaded area gives the intersection of $\operatorname{conv}(Q)$ with the area satisfying the complicated constraints $Ax \le b$. For the cost vector $c_0$ we have $z_{\mathsf{ip}} < z_{\mathsf{ld}} < z_{\mathsf{lp}}$. Observe that we can construct cost vectors $c$ such that $z_{\mathsf{ip}} = z_{\mathsf{ld}} = z_{\mathsf{lp}}$ holds true.

The ILP is clearly equivalent to

$$\begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax \le b \\ & Cy \le d \\ & x = y \\ & x, y \in \mathbb{Z}_+^n \end{aligned}$$

Dualizing the equality constraint $x = y$ yields

$$\begin{aligned} \max \quad & c^T x + \lambda^T (y - x) \\ \text{subject to} \quad & Ax \le b \\ & Cy \le d \\ & x, y \in \mathbb{Z}_+^n \end{aligned}$$

which can be decomposed into

$$\max_x\{(c^T - \lambda^T)x \mid Ax \le b, x \in \mathbb{Z}_+^n\} + \max_y\{\lambda^T y \mid Cy \le d, y \in \mathbb{Z}_+^n\} \ .$$

Again, we are interested in the sharpest possible bound, *i.e.*,

$$z_{\hat{\mathsf{ld}}} = \min_\lambda[\max_x\{(c^T - \lambda^T)x \mid Ax \le b, x \in \mathbb{Z}_+^n\} + \max_y\{\lambda^T y \mid Cy \le d, y \in \mathbb{Z}_+^n\}] \ .$$

Guignard and Kim [56] proved that $z_{\hat{\mathsf{ld}}}$, *i.e.*, the bound obtained by dualizing the equality constraints, can dominate the bounds that we get by dualizing either set of constraints.

So far, we only considered how to relax an ILP in a Lagrangian fashion, but we did not explain how we can compute the optimal value of the Lagrangian dual. In the following two sections, we will address this issue by describing two algorithms that aim at computing optimal values of the dual problem.

### 2.3.1 Subgradient Method

The *subgradient method* is a general method to minimize any nondifferentiable convex function. As we described in Sect. 2.3, the Lagrangian dual can be seen as a finite collection of linear functions, and therefore it is convex and nondifferentiable. The subgradient method is similar to *gradient methods* that are used in differentiable optimization problems. There are, however, some important differences: subgradient methods, for instance, are not descent methods, meaning that the function values are not strictly decreasing during the optimization process.

Let $\theta : \mathbb{R}^n \to \mathbb{R}$ be a convex function. We call a vector $g$ a *subgradient* of $\theta$ at position $x$ if $\theta(y) \geq \theta(x) + g^T(y - x)$ holds true for all values of $y$. We call the set of all subgradients of $\theta$ at position $x$ the *subdifferential* of $\theta$ at $x$ and denote it by $\partial\theta(x)$. If $\theta$ is differentiable at $x$, then we have $\partial\theta(x) = \{\nabla\theta(x)\}$, *i.e.*, the subdifferential consists only of the gradient of $\theta$ at $x$. The following lemma gives a necessary and sufficient condition for the minimum of a convex function.

**Lemma 2.4.** *Let $\theta : \mathbb{R}^n \to \mathbb{R}$ be a convex function. A vector $x^*$ minimizes $\theta$ over $\mathbb{R}^n$ if and only if $0 \in \partial\theta(x^*)$.*

Algorithm 2.1 describes the main stages of the iterative subgradient method. The basic principle is to start at some initial point $x_0$ and move along a subgradient $s_t$ for a certain stepsize $\gamma_t$ through the search space. As long as we haven't found the provably optimal solution, *i.e.*, $0 \in \partial\theta(x_t)$, or we have not reached a predefined number of iterations, we move on.

**Algorithm 2.1**: Main steps of the subgradient method.

```
1 Start from an initial point $x_0$, set $t = 0$;
2 while stopping criterion not met do
3     Given $x_t$, choose a subgradient $s_t \in \partial\theta(x_t)$;
4     if $s_t == 0$ then
5         stop;
6     Compute stepsize $\gamma_t$;
7     Set $x_{t+1} = x_t + \gamma_t s_t$;;                // update the current point
```

Since the subgradient method is not a descent method, we have to keep track of the best solution value found so far, *i.e.*, $\theta_{\text{best}} = \min\{\theta(x_0), \ldots, \theta(x_t)\}$. There are various approaches for the computation of stepsize $\gamma_t$, a fundamental result due to Poljak [113] states that for stepsize adaption schemes that satisfy

$$\lim_{k\to\infty} \gamma_k = 0 \quad \text{and} \quad \sum_{k=0}^{\infty} \gamma_k = \infty$$

the series of $\theta(x_t)$ converges to the optimal value $x^*$. Setting $\gamma_k = \frac{1}{k}$ satisfies both conditions, but the practical convergence rate is poor. Held and Karp [60] propose a different way to adapt the stepsize:

$$\gamma_{t+1} = \mu \frac{\theta(x_t) - \theta(\hat{x})}{|s_t|}$$

with $\mu$ being a user-specific parameter and $\theta(\hat{x})$ being an estimate of the optimal value $\theta(x^*)$. Estimates for $\theta(x^*)$ could, for instance, be computed by heuristically inferring solutions that are feasible in the primal problem given the current point $(x_t)$.

In practice, the stopping criterion in Algorithm 2.1 is rarely met. Typically, the optimization process is stopped after a fixed number of iterations. In our computational experiments from Chap. 5, for example, we set the number of iterations to 500, since we usually did not observe any dramatic improvement of the value of the Lagrangian dual after that.

It is possible to use the single $x_t$ to infer solutions $p(x_t)$ that are feasible in the primal problem. Given $p(x_t)$ after $n$ iterations, then we can stop if

$$\theta_{\text{best}} = \max \ \{p(x_t)\} \quad 0 \le t < n \ .$$

Figure 2.2 shows two typical runs of the subgradient method: either the convergence process gets stuck, leaving a gap between the Lagrangian dual and a heuristically inferred primal solution, or we obtain a provably optimal solution for the Lagrangian dual if the upper and lower bound coincide.

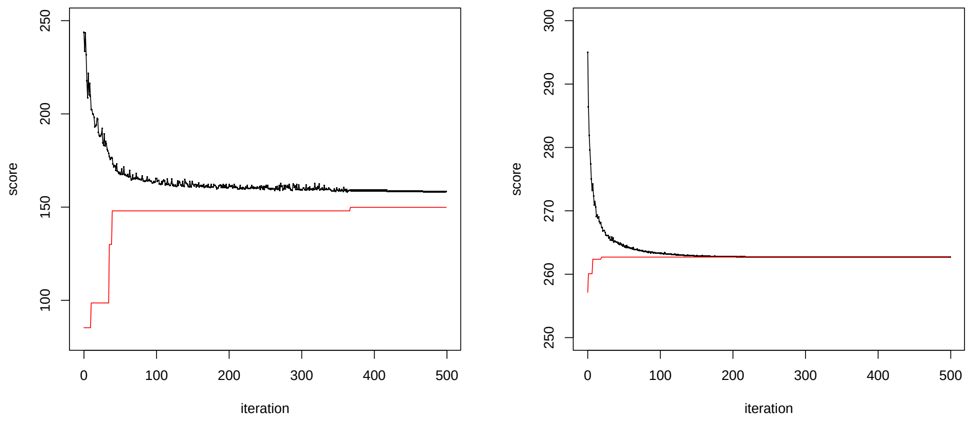


Figure 2.2: Two typical convergence scenarios using the subgradient method. The black line gives the values of the Lagrangian dual, whereas the red line denotes the scores of heuristically inferred primal solutions. Left side: after some hundred iterations the convergence process gets stuck, leaving a gap between the upper bound and primal solution. Right side: there is no gap left between the upper and the lower bound, yielding a provably optimal solution for the Lagrangian dual.

### 2.3.2 Bundle Method

The basic idea of the subgradient method is to evaluate the convex function $\theta$ at some point $x$ and use a subgradient $s$ to obtain the direction towards the next point. We do not, however, keep track of where we came from, *i.e.*, we discard all information about previous points and subgradients that we evaluated.

The *bundle method* removes this limitation. The main idea is to keep a *bundle* $B$ of $k$ subgradients and use the set $B$ to fit a quadratic function to the subgradients. We are keeping a *stability center* $\hat{x}$ which acts as the point from which we build our models. The minimum of the quadratic model yields the next potential point $x_{k+1}$. We do, however, check whether the decrease of the objective function value, $\theta(x_{k+1}) - \theta(\hat{x})$, is big enough. If this is the case, we perform a *descent step* and move the stability center to $x_{k+1}$. Otherwise, we perform a *null step* and simply add $x_{k+1}$—together with an associated subgradient $g_{k+1}$—to the bundle and construct an updated quadratic model. Algorithm 2.2 lists the main stages of the generic bundle method. Observe that we assume to have access to a function oracle that evaluates the function value $\theta(x_{k+1})$ for a point $x_{k+1}$ and that returns a subgradient $g_{k+1}$. The function $\hat{\theta}$ defines the set of piecewise linear functions as given by the bundle and returns the maximal value at position $x_k$; see the left side of Fig. 2.3 for an illustration.

**Algorithm 2.2**: Main steps of the bundle method.

```
   Input   : choose a bundle size $k$ and an initial stability center $\hat{x}$
1  while stopping criterion not met do
2      Solve the quadratic program
           $\min_{(x,r)\in\mathbb{R}^{n+1}} \quad r + \frac{1}{2t}||x-\hat{x}||^2$
           $r \geq \theta(x_i) + g_i^T(x - x_i) \qquad \forall 0 \leq i < k$
3      Set $x_{k+1} = x$;
4      Get $\theta(x_{k+1})$ and $g_{k+1}$ using the function oracle;
5      Determine the regularized gradient $\hat{g} = (\hat{x} - x_{k+1})/t$;
6      Compute $\delta = \theta(\hat{x}) - \hat{\theta}(x_{k+1})$;
7      if $\theta(x_{k+1}) \leq \theta(x_k) - \kappa \cdot \delta$ then
8          $\hat{x} = x_{k+1}$; ;                     // perform a descent step
9      if $|\delta| < \epsilon_\delta$ and $||\hat{g}|| < \epsilon_g$ then
10         stop;
11     Add $(x_{k+1}, g_{k+1})$ to the bundle;
```

Line 2 contains the parameter $t$ that is crucial for the performance of the bundle method, because $t$ specifies the impact of the quadratic term on the objective function value. If $t$ is large, then the impact is small leading to new points $x_{k+1}$

that are far away from the current stability center $\hat{x}$. Small values of $t$, on the other side, constrain the next point $x_{k+1}$ to be in the vicinity of $\hat{x}$. The right side of Fig. 2.3 gives an illustration of the models with different two different settings for $t$.

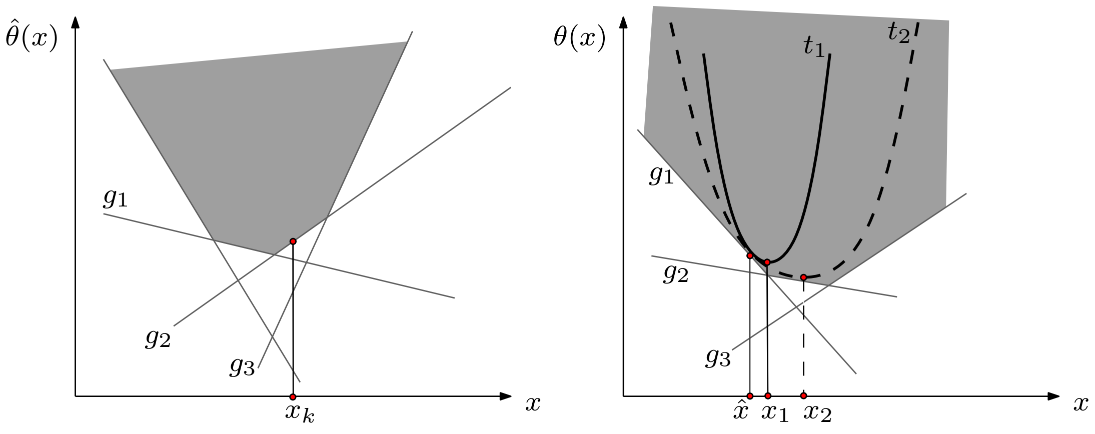


Figure 2.3: Left side: the function $\hat{\theta}$ evaluates the bundle and returns the maximum value at $x_k$. Right side: we get different models for different values of the *spring strength* $t$. If the value $t$ is small, we get models that look like $t_1$ and that are closer to the stability center $\hat{x}$. If we have a large value for $t$, we get models that look like $t_2$.

Line 7 gives the criterion for updating the stability center: $\kappa \in [0, 1]$ denotes a user-specific parameter, and we are testing the decrease of the objective function compared to $\delta$. If the difference is big enough, we perform a *descent step*.

Line 9 contains the stopping criteria for the bundle method. If the expected decrease is smaller than a user-specific threshold $\epsilon_\delta$ and the regularized gradient, *i.e.*, the difference between the current and the new stability center, is small enough, then we stop the method, because we found the minimum of the convex function $\theta$. In Chap. 5.4.4 we compare the performance of the subgradient to the bundle method within the sequence-structure alignment scenario. For a detailed description of the bundle method, the reader is referred to [94; 108].

## 2.4 Statistics

This section briefly covers the statistical algorithms that we will use in Chap. 5. We use *Lowess curves* and rank tests to visualize the results and compare the performance of «««< .mine different programs. A more detailed description of Sect. 2.4.2 can be found in any introductory textbook for statistics. ======= different programs. The two tests described in Sect. 2.4.2 can be found in any statistical textbook. »»»> .r2179

### 2.4.1 Lowess Curves

*Lowess* is also known as *locally weighted polynomial regression*, and it was described by Cleveland [22] for the first time. Lowess regression is used to fit a smoothed curve to a data set to illustrate a trend within the data. The main idea is to fit a low-degree polynomial to a subset of the input data at each data point $x$. We compute the coefficients of the polynomial by the weighted least-squares method, *i.e.*, we assign a higher weight to points that are close to the current point $x$. Typical implementations of the Lowess algorithm use polynomials of first or second order to avoid local overfitting of the data.

More formally, the set $X = \{x_0, \ldots, x_{n-1}\}$ denotes $n$ data points with their associated function values $f(x_i)$. We have a weight function $w_k(x_i)$ that assigns weights to each point $x_k$ while evaluating the current point $x_i$. Then, for each data point $x_i, 0 \leq i < n$, we compute estimates $\hat{\beta}_j(x_i), 0 \leq j \leq d$, of the coefficients in the polynomial regression of degree $d$, *i.e.*, we want to compute $\hat{\beta}_j(x_i)$ that minimize

$$\sum_{k=0}^{n-1} w_k(x_i)(f(x_k) - \beta_0 - \beta_1 x_k - \cdots - \beta_d x_k^d)^2$$

The smoothed function value $\hat{f}(x_i)$ is given by the value of the fitted regression. The subset of points $x_i$ that are used for fitting the polynomial, *i.e.*, points satisfying $w_k(x_i) > 0$ greatly influences the smoothness of the curve. One usually has to set a parameter $s$ that specifies the fraction of all data points used for the computation of the polynomial. With $s = 0.0$ we do not consider any points in the vicinity of $x_i$, yielding a zig-zagged line. The other extreme is setting $s = 1.0$ which yields the smoothest curve possible, because at each point $x_i$ we take all other points into account. Figure 2.4 shows a scatterplot with Lowess curves that have different values for $s$.

### 2.4.2 Friedman Rank Sum Test and Wilcoxon Signed-Rank Test

The nonparametric *Friedman Rank Sum Test* detects differences between test results across $c$ test attempts (or samples). To be more specific, it checks whether the $c$ different sample groups are having the same median.

Given $n$ observations, each consisting of $c$ test values, we then replace the data by their ranks within each of the observation. The smallest one gets rank 1, and we assign rank $c$ to the largest value in the row. If two values are equal, their rank is the average of the ranks that they would have been assigned otherwise. Then, we build a matrix $R \in \mathbb{R}^{n \times c}$ with the matrix entry $R(i, j)$ being the rank of the $j$th test attempt in the $i$th observation.

The null hypothesis $H_0$ assumes that there are no significant differences among the $c$ test attempts, *i.e.*, each ranking within an observation is equally likely. Accepting $H_0$ means that there will be no difference among the average ranks

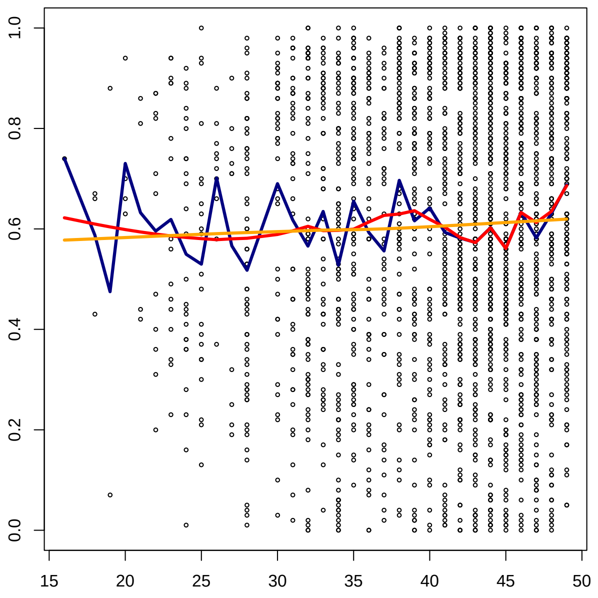


Figure 2.4: The crucial parameter for the shape of the Lowess curve is the parameter $s$ that specifies how many of the data points in the neighborhood should be taken into account. The blue line shows the curve for $s = 0.0$, *i.e.*, no neighboring points are considered. The red and orange lines represent the Lowess curve for $s = 0.2$ and $s = 1.0$. One can clearly see that the curve becomes smoother with an increasing value of $s$.

for each test attempt. Otherwise, we know that there are significant differences among the $c$ test attempts and we have to perform pairwise comparisons to detect significant differences between two test attempts.

The *Wilcoxon Signed-Rank Test* performs such a pairwise comparison. It is a nonparametric test to check whether the median of $n$ paired data differs significantly. The main idea is to rank the differences between the paired data by their absolute value, and assign 1 to the smallest and $n$ to the largest difference. Then we sum up the ranks of the positive and the negative differences. The test statistic is the smaller one of the two values. If the null hypothesis $H_0$ is true, *i.e.*, the median of the two observed samples is the same, then we expect the rank sum of the positive and the negative ranks to be the same. In this case, we accept $H_0$, and we reject it otherwise.

If we perform multiple Wilcoxon tests, then we have to correct the p-value for multiple testing. In our experiments we used the conservative *Bonferroni correction* to adapt the p-values. Using a significance value $p$ for $k$ tests, we have to set the significance level for each test to $\frac{p}{k}$.

Chapter 3

# Previous Work

Früher war alles besser,
früher war alles gut.
Die Toten Hosen
(Wort zum Sonntag)

This chapter summarizes the main concepts of previous approaches for the problem of computing sequence-structure alignments. Section 3.1 reviews various sequence-structure alignment scenarios, together with a brief description of the four main sequence-structure alignment models. Thereafter, we present each model in detail.

Section 3.2 gives two general paradigms for the computation of multiple alignments. Both *progressive* and *consistency-based* alignment algorithms originate from pure sequence-based alignment algorithms, but can be extended to incorporate structural information.

## 3.1 Sequence-Structure Alignments

### 3.1.1 Introduction

Depending on the available knowledge about the (putative) structures that we want to align, there are three different alignment scenarios for two RNA structures, which readily extend to the multiple case.

1. *Structure-to-structure* alignment algorithms align two known secondary structures, typically the *minimum free energy* structures. This scenario applies if one searches for common structural motifs that are shared by both structures and there is reason to believe that the secondary structures are correct.

2. *Structure-to-unknown* alignment algorithms align a given structure to a sequence with unknown structure. Applications are finding homologous sequences by inferring a consensus structure to a sequence. This has been done, for example, in case of the ITS2 database [153].

   RNA filtering software, like FastR[159] or PFastR [158], employ a two-stage strategy to find homologous structures for a given RNA structure. First, they search for regions in the database that show similar sequence or structural properties using fast searching strategies like indices, allowing for a higher number of false positives. Thereafter, a *verification* phase follows that separates the true from the false positives. The verification phase in the FASTR and PFASTR packages are performed as sequence-to-unknown alignments. Searching homologous structures of noncoding-RNAs in large genomic sequences has recently sparked considerable interest in the research community, see [47] for a survey.

3. In the *unknown-to-unknown* alignment problem, no previous structural information is given. It applies when two RNA sequences are suspected to share a common, but still unknown, structure. We constrain the space of possible structures by the entire set of possible Watson-Crick and wobble pairs. A reduction of the size of this space is possible, for instance, by computing the partition function to obtain the base pair probabilities [105]. Then, one only considers those interactions whose probabilities are above a certain threshold.

Figure 3.1 gives cartoon illustrations of the three scenarios.

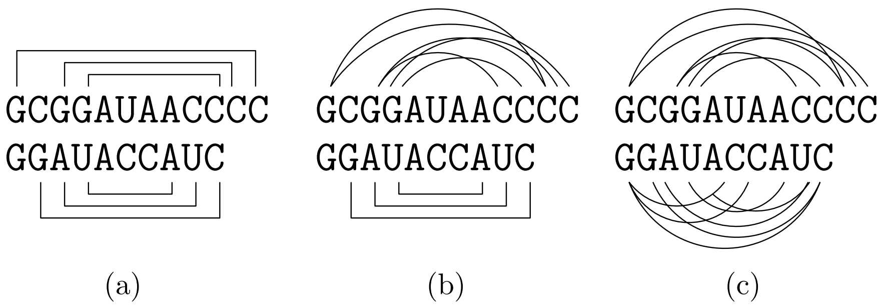


Figure 3.1: Different input alignment scenarios of RNA sequences (pairwise case): (a) the alignment of two known structures, (b) of one known and one unknown structure, and (c) of two unknown structures. The angled and round edges represent fixed and unknown structures, respectively.

There are four major alignment models for RNA structures that tackle the previous described alignment scenarios: *annotated sequences*, *tree models*, *probabilistic models*, and *graph-based models*. We give small examples for each model in Fig. 3.2. Table 3.1 classifies previous work in the area of structural RNA alignment according to the different models and scenarios. In the following sections we will describe previous approaches for each model.

## 3.1.2 Annotated Sequences

We call a sequence augmented by structural information an *annotated sequence*. In the unknown-to-unknown scenario we want to perform a simultaneous computation of the alignment and consensus structure. The computational problem of simultaneously considering sequence and structure of an RNA molecule was initially addressed by Sankoff in [124], where the author proposed a DP algorithm to align and fold a set of RNA sequences at the same time. The CPU and memory requirements of the original algorithm are $\mathcal{O}(n^{3k})$ and $\mathcal{O}(n^{2k})$, with $k$ and

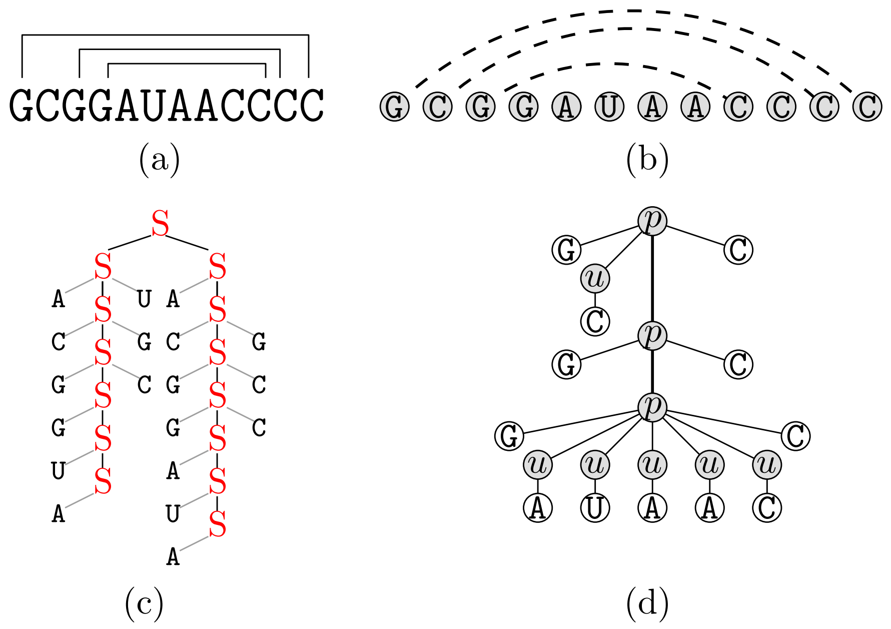


Figure 3.2: Different models representing RNA structures: (a) annotated sequences, (b) graph-based, (c) probabilistic, and (d) tree-based models.

| | **tree-based** | **annotated sequences** |
|---|---|---|
| structure-to-structure | [134; 157; 75] | [3; 44; 74] |
| structure-to-unknown | — | [3; 43; 12] |
| unknown-to-unknown | — | [124; 103; 101; 59; 58; 67; 137; 148; 81; 16; 133; 132] |

| | **probabilistic** | **graph-based** |
|---|---|---|
| structure-to-structure | [41; 122] | [95; 4; 6; 89; 20; 18; 31] |
| structure-to-unknown | [41; 125; 121] | [95; 4; 6; 89; 20; 18; 31] |
| unknown-to-unknown | [72; 70; 71; 38] | [95; 4; 6; 89; 20; 18; 31] |

Table 3.1: Classification of previous work.

$n$ being the number of sequences and their maximal length, respectively. The $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ terms for time and space consumption follow from the recursions for RNA folding: the improvements from [147], where the authors present an algorithm that runs in $\mathcal{O}(n^2)$, brings the running time down to $\mathcal{O}(n^{2k})$. Employing the original Sankoff algorithm restricts the length of the input sequences to $100 - 200$ nucleotides. Therefore, various approaches have been proposed to heuristically prune parts of the solution space. Current implementations modify Sankoff's algorithm by imposing limits on the size or shape of substructures, *e.g.*, DYNALIGN [103; 101], or FOLDALIGN [58] that combine a sliding window and banded alignment approach. These approaches, however, still apply a loop-based

$$S_{i,j;k,l} = \max \begin{cases} S_{i+1;j;k,l} + \gamma \\ S_{i;j;k+1,l} + \gamma \\ S_{i+1,j;k+1,l} + \sigma(A_i, B_k) \\ \max_{h \le j, q \le l} \left( S^M_{i,h;k,q} + S_{h+1,j;q+1,l} \right) \end{cases}$$
$$S^M_{i,j;k,l} = S_{i+1,j-1;k-1,l-1} + p^A_{ij} + p^B_{kl} + \tau(A_i, A_j; B_k, B_l)$$

Figure 3.3: Nussinov-style recursions for computing a sequence-structure alignment of two RNA sequences. The fourth case leads to a running time of $\mathcal{O}(n^6)$ in the unconstrained case. The matrix $S_{i,j;k,l}$ holds the optimal value of the sequence-structure alignment between subsequences $A[i, \ldots, j]$ and $B[k, \ldots, l]$. The values $S^M_{i,j;k,l}$ give the optimal value for the alignment between subsequences $A[i, \ldots, j]$ and $B[k, \ldots, l]$ given that $(i, j)$ and $(k, l)$ form base pairs.

energy model making the computational requirements very expensive. The latest version of FOLDALIGN [59] additionally applies a dynamic pruning algorithm that discards parts of the DP matrix that does not score above a length-dependent threshold.

Hofacker, Bernhart, and Stadler [67] follow a different track: instead of incorporating the complete loop-based folding model they mimic an energy model by computing the *base pair probability matrices*, as given by the partition function [105]. Afterwards, they align the matrices using recursions that are essentially the same as the ones described in [124; 3]. Intuitively, their approach relates to the loop-based Sankoff algorithm like the original Nussinov folding algorithm to the Zuker energy model.

Figure 3.3 gives the recursions to compute a sequence-structure alignment with linear gap costs of two RNA sequences. One recognizes the similarity to the Nussinov recursions presented in Sect. 1.2.2: $\gamma$ represents the gap penalty, $\sigma(A_i, B_k)$ assigns a sequence score to the sequence alignment of $i$th character of sequence $A$ to the $k$th character of sequence $B$. The variable $p^{A|B}_{ij}$ gives the pairing probability for pair $(i, j)$ in sequence $A$ or $B$. Finally, $\tau(A_i, A_j; B_k, B_l)$ denotes the sequence score for matching base pair $(i, j)$ in sequence $A$ with $(k, l)$ in sequence $B$. In the unconstrained case, the recursions have a time and space complexity of $\mathcal{O}(n^6)$ and $\mathcal{O}(n^4)$. By banding the range of possible alignment positions, *i.e.*, by restricting the range of variables $h$ and $q$ for the fourth recursion case in Fig. 3.3, the time and space complexity drops to $\mathcal{O}(n^4)$ and $\mathcal{O}(n^3)$. For the multiple case, they align consensus base pair probability matrices in a progressive fashion. Their original program package PMCOMP is written in PERL which influences the running time and memory consumption. Therefore, there are two reimplementations of the PMCOMP ansatz, FOLDALIGNM [137] and LOCARNA [148] written in JAVA and C++, respectively.

FoldalignM restricts the maximal length difference of the alignment of two subsequences to a parameter $\delta$, which yields a reduced running time of $\mathcal{O}(n^2\delta^2)$. Secondly, a two-stage procedure fills the DP matrix: the authors identify possible branching points in the first place, dividing the sequences into unbranched subsequences. These unbranched parts are then used to align the entire sequences. On the other hand, LocARNA makes use of the sparse nature of base pair probability matrices, *i.e.*, there is only a constant number of significant entries per row. By considering only the significant positions in the DP filling stage, the authors reduce the overall time and space consumption to $\mathcal{O}(n^2(n^2 + m^2))$ and $\mathcal{O}(n^2 + m^2)$, respectively.

Kiryu *et al.* [81] describe a recent reimplementation of the Sankoff recursions, where they employ two strategies—the *strip* and the *skip approximation constraints*—for keeping the running time low. The strip approximation limits the set of possible alignment positions to a band of width $\delta$ around an initial *pair hidden Markov model* alignment, *i.e.*, this first alignment is based on sequence information alone. Secondly, the skip approximation constrains the set of possible bifurcation points to positions that are within the band computed during the strip approximation stage. The set size of putative branching points is additionally controlled via a user-specific parameter.

Bonhoeffer [16] suggested the following idea to align sequences of unknown structures using the base pair probability matrices: one takes the highest up- and downstream probability and uses these values as the scores for a traditional sequence alignment. Given a sequence $A$, let $p^{\mathrm{u}}_{A,i}$ and $p^{\mathrm{d}}_{A,i}$ be the highest up- and downstream base pair probabilities of sequence $A$ at position $i$. Then, the score $s(i,j)$ for a match between positions $i$ and $j$ reads

$$s(i,j) = \sqrt{p^{u}_{A,i} \cdot p^{u}_{B,j}} + \sqrt{p^{d}_{A,i} \cdot p^{d}_{B,j}}$$

for two sequences $A$ and $B$. Given a matrix $\Delta$ with $\Delta(i,j) = s(i,j)$ we compute a traditional sequence alignment using $\Delta$ as the scoring matrix. A recent implementation of this idea is the Stral tool [30].

Tabei *et al.* [133; 132] describe a different approach based on base pair probability matrices. They use the matrices to extract ungapped stem fragments of length $l$. Given a base pair probability matrix $p^A$, a putative stem-fragment is a set $F$ of continuous nested base pairs such that for each $(i,j) \in F$ we have $p^A_{ij} > \alpha$: $\alpha$ defines the threshold for a probability to be considered. The authors align these stem fragments in a consistent fashion, *i.e.*, if we have overlapping stem fragments in sequence $A$, then the aligned stem fragments in $B$ have to be overlapping as well. Note that the aligned stem fragments define the alignment of the helical parts of the sequence, *i.e.*, loops are not aligned at this point. Therefore, in a second step the loop regions are aligned by using the aligned stem fragments as anchor points in a sequence alignment.

In the restricted structure-to-structure scenario, one can resort to more sophisticated edit-models (Edit in short) like the one proposed by Jiang *et al.* in [74]

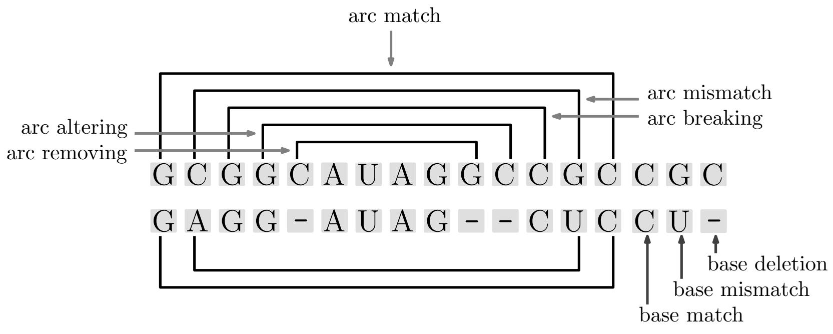


Figure 3.4: The operations in the edit model by Jiang *et al.*: we have both operations on the structure and the sequence level, each one associated with a certain cost. Given two annotated sequences $A$ and $B$, we aim at finding the series of edit operations of minimum cost such that we transform $A$ into $B$.

where the authors specify operations—each associated with a specific cost—both on the sequence and the structure level: the operations are *arc match*, *arc mismatch*, *arc altering*, *arc breaking*, and *arc removing* modifying the structures, and *base match*, *base mismatch*, and *base deletion* on the sequence level. Figure 3.4 gives a cartoon illustration for the single operations.

Similar to the notion of the edit-distance on the level of nucleotide sequences, the authors devise algorithms to compute the edit distance between two annotated sequences. As stated above, each edit operation $e_i$ is associated with a certain cost $\delta(e_i)$. Given a series of edit operations $\Gamma = \{e_0, e_1, \cdots, e_n\}$ the overall cost is given by $\sum_{i=0}^{n} \delta(e_i)$. We now want to find the series $\bar{\Gamma} = \{\bar{e}_0, \bar{e}_1, \cdots, \bar{e}_n\}$ such that $\sum_{i=0}^{n} \delta(\bar{e}_i)$ is minimal and $\bar{\Gamma}$ transforms the first into the second annotated sequence. Jiang *et al.* give a dynamic programming algorithm that runs in $\mathcal{O}(n^4)$ to infer a known structure onto a second sequence, making the computation rather tedious for longer sequences.

Evans [44; 43] started a new line of research by introducing the *longest arc-preserving common subsequence* problem (or Lapcs in short). The Lapcs is defined as follows: we are given two annotated input sequences $(S_1, P_1)$ and $(S_2, P_2)$, with $S_1$ and $S_2$ being sequences from some alphabet $\Sigma$. $P_1$ and $P_2$ are annotations, possibly containing crossing interactions, and we have a target length $l$. The output is true if there exists a mapping $M \subseteq \{1, \ldots, |S_1|\} \times \{1, \ldots, |S_2|\}$ such that $|M| = l$, and false otherwise. Furthermore, the following constraints have to be satisfied:

1. $M$ has to be a proper alignment, *i.e.*, the order of the subsequences has to be preserved.

2. arcs induced by the mapping have to be preserved, *i.e.*, $\forall (i_1, j_1), (i_2, j_2) \in M$ if $(i_1, i_2) \in P_1$ iff $(j_1, j_2) \in P_2$.

3. the subsequence induced by $M$ is a common subsequence, *i.e.*, $\forall (i_1, j_1) \in M$ we have $S_1[i_1] = S_2[j_1]$.

Especially the second constraint distinguishes this model from the other alignment models for annotated sequences, because we have to enforce that arcs from $P_1$ are conserved in $P_2$ via $M$. As we will see in Sect. 3.1.6, computing the LAPCS is already NP-complete if both input annotations are nested structures. Blin *et al.* [12] extend the original model by Evans and introduce the *maximum arc-preserving common subsequence* (or MAPCS in short). The MAPCS introduces two scoring functions $f_a : \Sigma^4 \to \mathbb{N}^*$ and $f_b : \Sigma^2 \to \mathbb{N}^*$ that assign scores to the mapping of nucleotides ($f_b$) and the conservation of arcs ($f_a$). Then, we aim at finding a mapping $(M, Q)$, with $M$ being the common subsequence of $S_1$ and $S_2$, and $Q$ being the conserved arcs of $P_1$ and $P_2$, such that

$$\sum_{(i,j)\in M} f_b(S_1[i], S_2[j]) + \sum_{(i,j,k,l)\in Q} f_a(S_1[i], S_1[j], S_2[k], S_2[l])$$

is maximized. The original constraints for the LAPCS problem remain valid for the computation of the MAPCS.

### 3.1.3 Tree-Based Models

As we have seen in Sect. 1.2, nested RNA secondary structures may be viewed as trees. Hence, algorithms that compare trees can be applied to RNA structures. A first model was introduced by Tai [134] and generalizes the *edit problem on strings* [142] to tree structures which is known as the *tree-edit problem*. Informally speaking, we have a set of operations $\Omega = \{e_0, \dots, e_n\}$, each associated with a certain cost $\delta(e_i), \forall e_i \in \Omega$. We are given trees $T_1$ and $T_2$ whose nodes have labels from some alphabet $\Sigma$, and $n = |T_1|$ and $m = |T_2|$ with $n \geq m$. Let $\bar{\Sigma} = \Sigma \cup \lambda$, with $\lambda$ being the null symbol. We are searching for the series $S$ of edit operations of minimum cost such that $S$ transforms $T_1$ into $T_2$. For sake of simplicity we assume in the following that the nodes and their labels are identical. The tree-edit model provides three distinct operations:

(a) **node relabeling ($X \to Y$):** the label of node $X$ in $T_1$ is changed to $Y$.

(b) **node deletion ($X \to \lambda$):** we delete node $X$ from $T_1$, all children of $X$ become children of the parent node of $X$, preserving the sibling relation of the parent node. If $X$ is the root node of $T_1$, the deletion of $X$ yields the forest of the children nodes of $X$.

(c) **node insertion ($\lambda \to X$):** we insert a new node $X$ into $T_1$.

Figure 3.5 shows a small example of transforming tree $T_1$ into another tree $T_2$. Given a series of edit operations $\Gamma = \{e_0, e_1, \cdots, e_n\}$ the overall cost is given by $\sum_{i=0}^{n} \delta(e_i)$. We now aim at finding a series $\bar{\Gamma} = \{\bar{e}_0, \bar{e}_1, \dots, \bar{e}_n\}$ such that $\sum_{i=0}^{n} \delta(\bar{e}_i)$ is minimal and $\bar{\Gamma}$ transforms $T_1$ into $T_2$. Tai's original algorithm runs in $\mathcal{O}(n \cdot m \cdot \text{leaves}(T_1)^2 \cdot \text{leaves}(T_2)^2)$, which Zhang and Shasha [157] improve to $\mathcal{O}(n \cdot m \cdot \min(\text{leaves}(T_1), \text{depth}(T_1)) \cdot \min(\text{leaves}(T_2), \text{depth}(T_2)))$. There are several recent papers that report on variations on the original Zhang-Shasha algorithm,

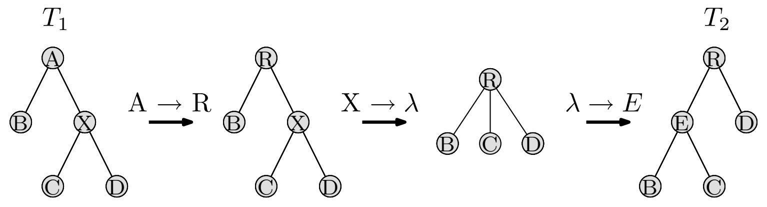


Figure 3.5: We transform $T_1$ into $T_2$ by relabeling node $A$ to $R$ ($A \rightarrow R$), deleting node $X$ ($X \rightarrow \lambda$), and by finally inserting node $E$ ($\lambda \rightarrow E$).

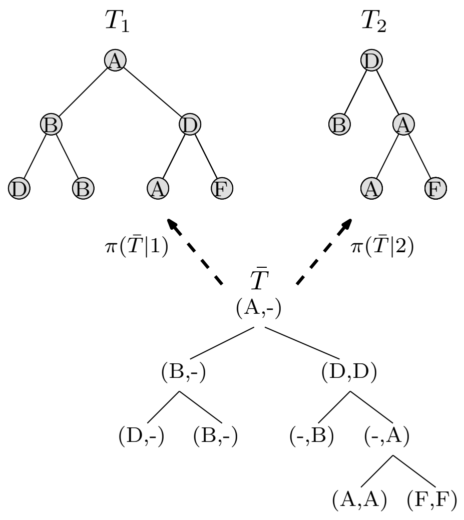


Figure 3.6: Given two trees $T_1$ and $T_2$ we aim at finding a common supertree $\bar{T}$ whose pairwise projections $\pi(\bar{T}|1)$ and $\pi(\bar{T}|2)$ yield the two original input trees.

the interested reader is, for example, referred to [126; 39; 82]. Finally, Demaine *et al.* [32] show that the worst case time complexity for the tree edit problem is in $\mathcal{O}(n^3)$.

An alternative way to compare trees is *tree alignment* which was introduced Jiang *et al.* in [75]. Instead of transforming one tree into another one by a series of edit operations, we are now searching for a common supertree $T$ whose nodes hold labels from $\bar{\Sigma} \times \bar{\Sigma}$, and the pairwise projections $\pi(T|1)$ and $\pi(T|2)$ yield the two input trees $T_1$ and $T_2$. A pairwise projection $\pi(T|1)$ or $\pi(T|2)$ is defined as the tree that we get by taking the first (or second) symbols of the nodes of the common supertree $T$, and by deleting all nodes that have the null symbol $\lambda$ afterwards. Given a cost function $\delta : \bar{\Sigma} \times \bar{\Sigma} \rightarrow \mathbb{R}$ that scores the nodes of the supertree, we want to find the supertree $\bar{T}$ such that $\sum_{(a,b)\in V(T)} \delta(a,b)$ is minimal. Figure 3.6 shows a small example of a tree alignment of two input trees. Tree alignment algorithms have complexities that are on average only slightly worse than conventional sequence alignment. More precisely, their running time is in $\mathcal{O}(n^2 \cdot \Delta^2)$, where $\Delta$ denotes the maximum number of branches of a multiloop in the input structures.

A tool that builds upon the tree alignment paradigm is RNAForester [64; 65]. It computes multiple structure-to-structure alignments of RNA sequences by performing tree alignment in a progressive fashion.

### 3.1.4 Probabilistic Models

The use of *hidden Markov models* (HMMs) and *profile hidden Markov models* has proven to be a very useful concept in the context of genomic sequence analysis. Applying these algorithms directly to RNA related problems is not straightforward, because HMMs are not able to account for the structural information of RNA sequences.

Therefore, Eddy and Durbin [41] (and simultaneously Sakakibara and coworkers [122]) describe *stochastic context free grammars* (SCFGs) for measuring the secondary structure and primary sequence consensus of RNA sequence families. A grammar contains a set of *rules* to generate strings, starting from some *start symbol*. The main components of a grammar are a set $N$ of *nonterminal symbols*, a set $T$ of *terminal symbols*, and a set $P$ of *production rules*. In the case of stochastic grammars each production rule is associated with a probability. The *language* of a grammar are all strings that, starting from the start symbol, can be generated by successively applying the production rules.

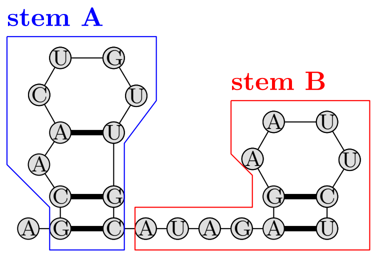
SCFGs are grammars that model the tree-like structure of RNAs. A simple SCFG that captures ungapped RNA structures is the following grammar, with $N = \{W, P, L, R, B, S, E\}$ and $T = \{\epsilon, A, C, G, U\}$. The nonterminal $W$ denotes any of the six other nonterminals ($P$,$L$,$R$,$B$,$S$,$E$), and $a, b \in T$ :

$$\begin{array}{lcll} \text{P} & \rightarrow & \text{aWb} & \text{(pairwise emission)} \\ \text{L} & \rightarrow & \text{aW} & \text{(leftwise emission)} \\ \text{R} & \rightarrow & \text{Wa} & \text{(rightwise emission)} \\ \text{B} & \rightarrow & \text{SS} & \text{(bifurcation)} \\ \text{S} & \rightarrow & \text{W} & \text{(start production rule)} \\ \text{E} & \rightarrow & \epsilon & \text{(end)} \end{array}$$

Then, the RNA secondary structure from Fig. 3.7 yields the corresponding parse tree.

The simple SCFG from above does not incorporate the presence of gaps, and hence has to be extended by insertion and deletion states. The resulting grammars—called *covariance models*—are quite complex, and there are three main algorithms used in the context of covariance models: the *inside*, the *inside-outside*, and the *Cocke-Younger-Kasami* (CYK) algorithm. These algorithms compute the likelihood of an observed sequence $x$ of length $n$, the expected number that each production rule is used, and the maximum likelihood parse of sequence $x$, respectively. The runtime of these algorithms scales in $\mathcal{O}(n^3)$. For a detailed description the reader is referred to [40].

The SCFGs described so far are not suited to compute a sequence-structure alignment of two sequences, because they are emitting at most one single symbol



| | | | stem A | | | stem B | | |
|---|---|---|---|---|---|---|---|---|
| $S_1$ | $\longrightarrow$ | $L_2$ | $S_4$ | $\longrightarrow$ | $P_5$ | $S_{14}$ | $\longrightarrow$ | $L_{15}$ |
| $L_2$ | $\longrightarrow$ | $aB_3$ | $P_5$ | $\longrightarrow$ | $gP_6c$ | $L_{16}$ | $\longrightarrow$ | $aL_{17}$ |
| $B_3$ | $\longrightarrow$ | $S_4S_{14}$ | $P_6$ | $\longrightarrow$ | $cL_7g$ | $L_{17}$ | $\longrightarrow$ | $uL_{18}$ |
| | | | $L_7$ | $\longrightarrow$ | $aP_8$ | $L_{18}$ | $\longrightarrow$ | $aL_{19}$ |
| | | | $P_8$ | $\longrightarrow$ | $aL_9u$ | $L_{19}$ | $\longrightarrow$ | $gP_{20}$ |
| | | | $L_9$ | $\longrightarrow$ | $cL_{10}$ | $P_{20}$ | $\longrightarrow$ | $aP_{21}u$ |
| | | | $L_{10}$ | $\longrightarrow$ | $uL_{11}$ | $P_{21}$ | $\longrightarrow$ | $gP_{22}c$ |
| | | | $L_{11}$ | $\longrightarrow$ | $gL_{12}$ | $P_{22}$ | $\longrightarrow$ | $gL_{23}$ |
| | | | $L_{12}$ | $\longrightarrow$ | $uE_{13}$ | $L_{23}$ | $\longrightarrow$ | $aL_{24}$ |
| | | | $E_{13}$ | $\longrightarrow$ | $\epsilon$ | $L_{24}$ | $\longrightarrow$ | $aL_{25}$ |
| | | | | | | $L_{25}$ | $\longrightarrow$ | $uL_{26}$ |
| | | | | | | $L_{26}$ | $\longrightarrow$ | $uE_{27}$ |
| | | | | | | $E_{27}$ | $\longrightarrow$ | $\epsilon$ |

Figure 3.7: A toy example of an RNA secondary structure with the corresponding parse tree of the SCFG.

on either side. The idea of *pair HMMs* that works for nucleotide sequences can be extended to SCFGs: a *pair SCFG* captures the structural interaction of the input sequences and emits two symbols on either side. The computational complexity to compute structural alignments using a pair SCFG matches the one of the unconstrained Sankoff algorithm, *i.e.*, the space and time requirements scale in $\mathcal{O}(n^2m^2)$ and $\mathcal{O}(n^3m^3)$, respectively. This makes the unconstrained usage of pair SCFGs practical only for short sequences. Hence, there are several papers [72; 70; 71; 38] that propose heuristical constraints to improve the runtime.

In [72] Holmes and Rubin introduced the notion of a *fold envelope*. Instead of iterating over all possible substrings like in the unconstrained case, the authors only consider substrings of the input sequences that are consistent with precomputed secondary structures. Along these lines, Holmes [70; 71] generalizes the concept of fold envelopes to *alignment envelopes*. Alignment envelopes specify a set of positions between the two sequences that have to be aligned. By employing alignment and fold envelopes the author is able to significantly reduce the overall running time. Dowell and Eddy [38] also resort to the concept of alignment envelopes. They call an alignment envelope a *pin* and use pins as anchors in their alignment: a pin is a fixed position in the alignment and they compute a set of pins via the posterior probability for each possible pair of aligned residues using pair HMMs. Subsequently, these pins serve as constraints in their pair SCFG formulation.

Sakakibara [121] combines pair HMMs with the tree alignment algorithm by Jiang *et al.* and performs sequence-structure alignments of a known to an unknown structure using *pair HMMs on tree structures*. In subsequent work [125], Sato and Sakakibara build upon the recursions from [121], but they employ *conditional random fields* (CRFs) [87] to learn the parameters for their model. CRFs represent an undirected graphical model that generalizes standard HMMs in the sense that CRFs are able to model overlapping and non-independent features of the output. Furthermore, arbitrary functions replace the constant transition probabilities of HMMs, and the *feature functions*—which map current observations at a certain node in the graphical model—may depend on the entire observed sequence.

### 3.1.5 Graph-Based Models

Kececioglu [79] has introduced a graph-theoretical model for the classical primary sequence alignment problem. Lenhof, Reinert and Vingron [95] incorporate structural information and frame the sequence-structure alignment problem as an integer linear program. Their objective function maximizes the sum of aligned sequence scores plus the scores of interactions that are conserved by the alignment. They propose a branch-and-cut algorithm and perform structure-to-unknown alignments on data from the *European Ribosomal Database* [154] and compare the performance of their algorithm to sequence and manually curated alignments. With an increasing number of variables, however, the computational requirements become prohibitive.

Based on the formulation from [95], Lancia and coworkers [89] give an ILP formulation for the related problem of computing the *maximal contact map overlap* of two proteins. The contact map of a protein $A$ is a graph $G = (V, E)$ with $V$ and $E$ being the sets of vertices and edges. For each amino acid of the protein we have a vertex $v_i \in V$, and we insert an edge $e_i = (v_i, v_j) \in E$ iff the two corresponding amino acids $i$ and $j$ are spatially close enough, *i.e.*, $i$ and $j$ are in contact with each other. The maximal contact map overlap problem of two contact maps aims at computing a non-crossing mapping of residues from the first onto the second protein such that the number of conserved contacts is maximal. A contact from the first protein is conserved if its mapped endpoints in the second protein are also in contact with each other.

The algorithm in [89] is based on the branch-and-cut principle, and the authors are able to compute the optimal contact map overlap of small- and medium-sized proteins. Following earlier work [21] on the *quadratic knapsack problem*, the authors switch from branch-and-cut to *Lagrangian relaxation* in their subsequent paper [20]. Using Lagrangian relaxation they are able to solve instances to provable optimality that are an order of magnitude bigger compared to the

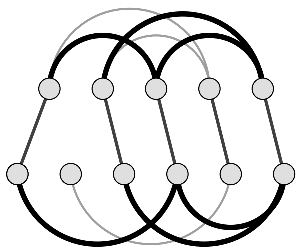


Figure 3.8: Given two proteins, the *maximal contact map overlap* problem calls for the maximal number of contacts that are conserved through a mapping of amino acids from the first onto the second protein. The mapping shown here preserves three contacts, *i.e.*, the contact map overlap is three. Since there is no mapping with a higher number of conserved contacts, the maximal contact map overlap is 3.

branch-and-cut algorithm. We adapted the formulation of Lancia and Caprara [20] for the computation of RNA sequence-structure alignments in previous work [4]. Compared to the first formulation of Lenhof *et al.* [95] we are able to solve instances with a much higher number of variables in less computation time.

Davydov and Batzoglou [31] present a graph-theoretical model for the alignment of multiple RNA structures based on the notion of a *nested linear graph* (we call this model MLG in short). A graph is a *linear graph* if we can place its vertices on some line. Nucleotide sequences naturally give rise to such linear graphs if we take the single nucleotides as the vertices of the graph. We add edges between complementary base pairs. Then, given $m$ linear graphs $G_1, \ldots, G_m$ the authors aim at finding the *largest common nested linear subgraph* (MAX-NLS) among all $m$ graphs. The MAX-NLS is defined as the largest nested graph $G_C$ such that $G_C$ is a subgraph of $G_i$ with $1 \leq i \leq m$. The authors show that finding the MAX-NLS is NP-complete, but they give polynomial time approximation algorithms with an approximation ratio of $\mathcal{O}(\log^2 S)$ with $S$ being the size of the optimal solution.

Note that the graph-based model naturally deals with all three alignment scenarios. In addition, unlike other algorithmic approaches, the graph-based algorithms do not restrict the input in any way and hence can handle arbitrary pseudoknots. They have been shown to play important roles in a variety of biological processes, see [131] for a recent review. Most DP-based algorithms assume nested secondary structures to compute subproblems efficiently. Few exceptions exist, for example [37], but these algorithms are always restricted to certain classes of pseudoknots (like H-type pseudoknots) and do not handle the general case. Brinkmeier [18] presents an algorithm to align various classes of pseudoknots, but the recursion scale in $\mathcal{O}(n^{14})$ and $\mathcal{O}(n^8)$ for time and space, making the algorithm inapplicable even for short sequences.

### 3.1.6 Computational Complexity

The complexity of pairwise sequence-structure alignments of RNA sequences is an intricate topic. The complexity does not only depend on the complexity of the input structure, *i.e.*, we do allow pseudoknots or not, but also on the model that we are using and in some cases also on the scoring system.

The complexity of tree-based and probabilistic sequence-structure alignment models is settled, as we have polynomial time algorithms that perform sequence-structure alignments. Furthermore, the Sankoff algorithm and all its variants run in polynomial time as well. The computation of a LAPCS, a sequence-structure alignment using the edit model and graph-based models is more involved.

In Sect. 1.2 we described the four classes of possible input structures for RNA structures, namely CHAIN, NESTED, CROSSING, and UNLIMITED. Table 3.2 which is taken from [12] gives an overview of the computational complexity in the LAPCS, EDIT, and MLG model (remember that LAPCS, EDIT, and MLG denote the *longest arc-preserving common subsequence*, the edit-model by Jiang *et al.*, and the *maximum linear subgraph*, respectively).

| $A \times B$ | CHAIN | NESTED | | CROSSING | | | UNLIMITED | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | CHAIN | CHAIN | NEST | CHAIN | NEST | CROS | CHAIN | NEST | CROS | UNLIM |
| EDIT | $\mathcal{O}(nm)$ [43] | $\mathcal{O}(nm^3)$ [73] | NPC [13] | MAX-SNP hard [74] | | | | | | |
| LAPCS | $\mathcal{O}(nm)$ [43] | $\mathcal{O}(nm^3)$ [73] | NPC [43; 73] | | | | | | | |
| MLG | $\mathcal{O}(nm)$ [63] | $\mathcal{O}(n^2m)$ [97] | $\mathcal{O}(n^2m^2)$ [97] | $\mathcal{O}(n^4 \log^3 n)$ [86] | | NPC [17; 141] | $\mathcal{O}(n^4 \log^3 n)$ [86] | | NPC [17; 141] | |

Table 3.2: The computational complexity of computing sequence-structure alignments in different models under different input structures.

Blin and Touzet [14] further refine the computational complexity considerations by restricting the allowed operations in the EDIT model. They introduce three submodels of Jiang's general model, such that we allow all substitution operations, base-deletions and arc-removings (model I), additionally arc-alterings (model II), or arc-alterings and arc-breakings (model II). Furthermore, one of the main results of the paper is the proof that the LAPCS model can be reduced to a special case of the EDIT model. In Sect. 3.1.2 we describe the MAPCS as a variant of the original LAPCS problem. Blin *et al.* [12] prove that computing the MAPCS is NP-complete already in the case of two nested input structures.

Of particular interest for this thesis is the table entry for computing the MLG if both input structures are crossing. This problem corresponds exactly to the computation of RNA sequence-structure alignments in our graph-based model. Goldman *et al.* show in [52] that computing the *maximal contact map overlap* is NP-hard in the pairwise case. They also state that the computation of the maximal contact map overlap, if 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. Unfortunately, they omit the proof and there is no full version of the paper available [112].

Vialette [141] introduces the *2-interval pattern* problem that corresponds exactly to the sequence-structure alignment of RNA sequence with crossing input structures. We score each conserved interaction with 1 and discard sequence-specific information. Then, computing the maximal set of conserved 2-interval patterns corresponds to the sequence-structure alignment problem in our model. The authors give an explicit reduction from 3SAT to the 2-interval pattern problem and therefore prove that the problem is NP-complete.

## 3.2 Multiple Alignments

This section covers two general paradigms for computing multiple alignments that were originally developed for the computation of pure sequence-based alignments. They can, however, be extended to incorporate structural information.

### 3.2.1 Building Progressive Alignments

The main idea behind progressive alignment is to build a multiple alignment from a series of pairwise alignments. In the beginning, we align two sequences and take the resulting alignment as fixed. Successively, we choose a third sequence and align it to the fixed alignment. This is repeated until no more sequences are available.

Typically, the order in which the sequences are aligned is given by a *guide tree*. We construct the guide tree using standard phylogenetic algorithms, *e.g.*, *weighted average linkage* (WPGMA) or *average linkage* (UPGMA). Starting from the leaves of the tree we align the sequences in a bottom-up fashion. The main stages of a progressive alignment of $k$ input sequences are the following:

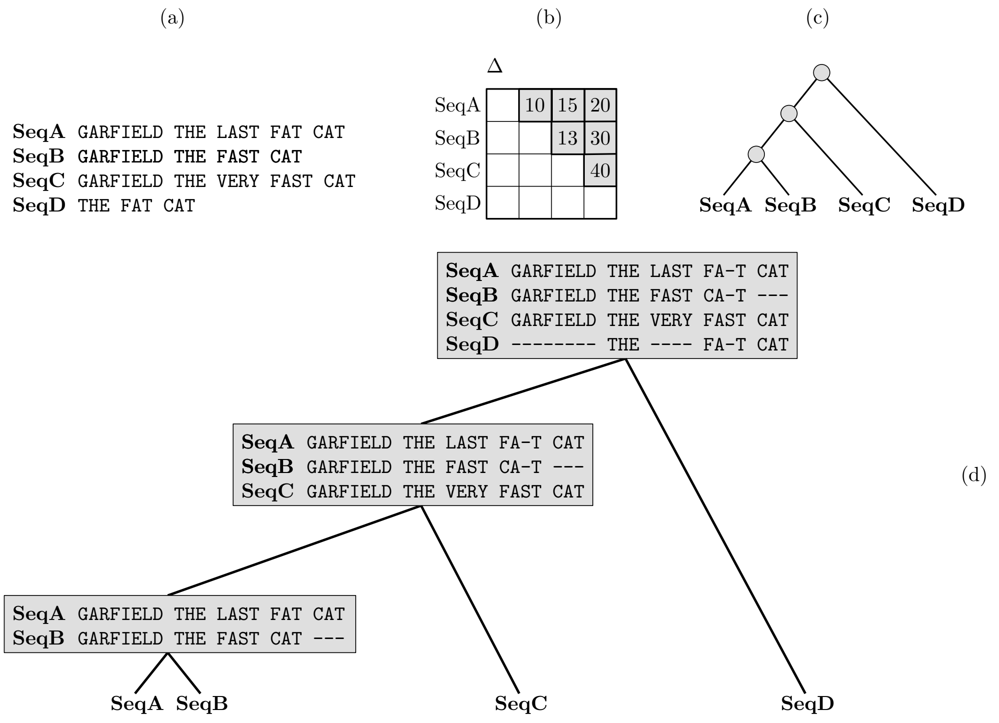
1. compute the *distance matrix* $\Delta$ for the $k$ sequences, *i.e.*, entry $\Delta(i, j)$ denotes the distance between sequences $i$ and $j$.

2. compute the guide tree using a phylogenetic tree construction algorithm like *UPGMA*.

3. perform the progressive alignment along the guide tree.

Figure 3.9 shows a toy example by aligning four input sequences in progressive fashion. The figure also exhibits the main weakness of progressive alignments; mistakes that are made in the lower part of the tree cannot be corrected later on, which is summarized by the *once a gap, always a gap* paradigm. Figure 3.9 exemplifies the weakness of progressive alignments: seqA and seqB are the first pair to be aligned, and `fast cat` of seqB is aligned to `last fat` of seqA. Taking a look at the entire alignment, one realizes that aligning `fast cat` to `fat cat` would improve the overall multiple alignment.

The progressive approach can be extended to incorporate structural information. Previous work, like [65; 67], perform pairwise sequence-structure alignments



(a)
(b)
(c)
(d)
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT
SeqC GARFIELD THE VERY FAST CAT
SeqD THE FAT CAT
Δ
SeqA
SeqB
SeqC
SeqD
10
15
20
13
30
40
SeqA SeqB SeqC SeqD
SeqA GARFIELD THE LAST FA-T CAT
SeqB GARFIELD THE FAST CA-T ---
SeqC GARFIELD THE VERY FAST CAT
SeqD -------- THE ---- FA-T CAT
SeqA GARFIELD THE LAST FA-T CAT
SeqB GARFIELD THE FAST CA-T ---
SeqC GARFIELD THE VERY FAST CAT
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT ---
SeqA SeqB
SeqC
SeqD

Figure 3.9: The main stages of progressive alignments: (a) The input sequences. (b) Compute the matrix Δ containing all pairwise distances between the input sequences, and construct the guide tree using Δ (c). The actual alignment is computed by pairwise alignments along the guide tree (d). The illustration is taken from [109].

along a precomputed guide tree and compute *consensus structures* at the inner nodes of the tree. A straightforward way to compute a consensus score between positions $(i, j)$ is to take the average values of the structure scores between positions $i$ and $j$ and compute the arithmetic mean of them. Figure 3.10 shows an example of a sequence-structure alignment of five tRNA sequences using the PMComp software package [67]. As one can observe, the consensus structure thins out along the guide tree.

### 3.2.2 Building Consistency-Based Alignments

As a remedy for the pure progressive alignment method, the authors of [109] propose *consistency-based* alignments. Although their algorithm is also progressive in nature, they introduce a preprocessing stage that reduces the probability of making a mistake early in the alignment phase.

The main idea behind consistency-based alignments is to perform all pairwise alignments, and then check for each aligned pair of residues how consistent this pair is with the remaining pairwise alignments. Figure 3.11 shows the main concepts by aligning the four input sequences from Fig. 3.9. Given the input

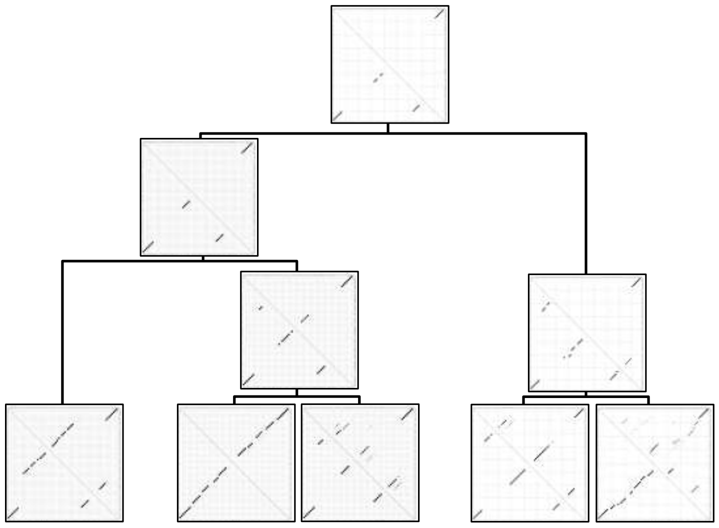
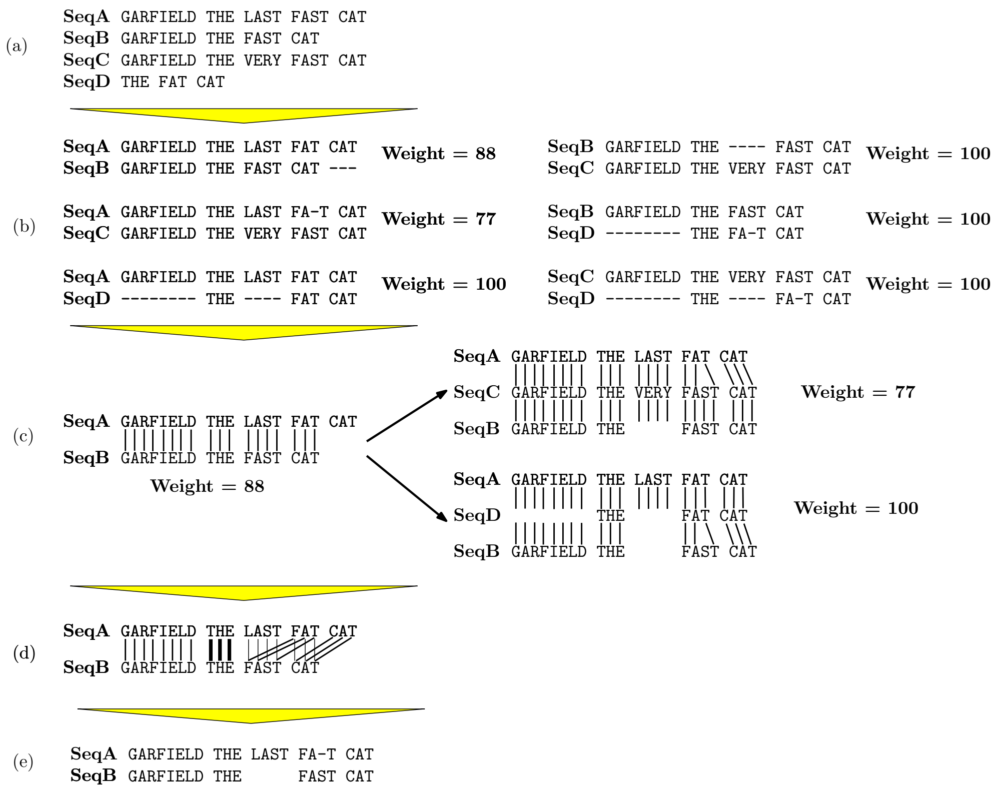


Figure 3.10: Progressive sequence-structure alignment of five tRNA sequences. The figure was generated using the PMCOMP software package [67].

sequences (a), we compute all pairwise alignments and assign their pairwise sequence identity as their weights (b). Then, we check for each pair of aligned positions how well the aligned pair is represented by the remaining pairwise alignments. As an example, we take the `G` in seqA and seqB denoted by $\texttt{G}_A$ and $\texttt{G}_B$. We then examine the alignment of seqA and seqB through seqC. We observe that $\texttt{G}_A$ is also aligned to $\texttt{G}_B$ via $\texttt{G}_C$. Hence, we add the smaller of the two weight values $W_1 = W(\texttt{G}_A, \texttt{G}_C)$ and $W_2 = W(\texttt{G}_C, \texttt{G}_B)$. In our case this sums up to $88 + \min(77, 100) = 165$, with 88 being the weight of the alignment between sequences seqA and seqB. We call the process of checking aligned positions via the alignment of other sequences *library extension*. The weights computed during the library extension computation are used as scores in the progressive alignment phase.

The first implementation of consistency-based alignments is the T-COFFEE software package [109]. Subsequently, several other programs resort to the same idea, like MAFFT [78] or PROBCONS [36]. T-COFFEE is, however, the only program that offers the possibility to incorporate alignment information from external sources. MARNA was the first program that uses this feature to compute multiple sequence-structure alignment heuristically. In [5] we presented a first version of our multiple alignment tool based on the pairwise information from the model presented in Chapt. 4. This eventually led to the first version of the LARA software package.



(a)
SeqA GARFIELD THE LAST FAST CAT
SeqB GARFIELD THE FAST CAT
SeqC GARFIELD THE VERY FAST CAT
SeqD THE FAT CAT
(b)
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT ---
Weight = 88
SeqB GARFIELD THE ---- FAST CAT
SeqC GARFIELD THE VERY FAST CAT
Weight = 100
SeqA GARFIELD THE LAST FA-T CAT
SeqC GARFIELD THE VERY FAST CAT
Weight = 77
SeqB GARFIELD THE FAST CAT
SeqD -------- THE FA-T CAT
Weight = 100
SeqA GARFIELD THE LAST FAT CAT
SeqD -------- THE ---- FAT CAT
Weight = 100
SeqC GARFIELD THE VERY FAST CAT
SeqD -------- THE ---- FA-T CAT
Weight = 100
(c)
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT
Weight = 88
SeqA GARFIELD THE LAST FAT CAT
SeqC GARFIELD THE VERY FAST CAT
SeqB GARFIELD THE FAST CAT
Weight = 77
SeqA GARFIELD THE LAST FAT CAT
SeqD THE FAT CAT
SeqB GARFIELD THE FAST CAT
Weight = 100
(d)
SeqA GARFIELD THE LAST FAT CAT
SeqB GARFIELD THE FAST CAT
(e)
SeqA GARFIELD THE LAST FA-T CAT
SeqB GARFIELD THE FAST CAT

Figure 3.11: Given the input sequences (a), we perform all pairwise sequence alignments and associate the pairwise sequence identity as the weight to the alignments (b). For every aligned position we perform the library extension (c), and get new weights (d) for a standard progressive alignment strategy (e). The illustration is taken from [109].

Chapter 4

# A Model for the Multiple Sequence Case

Sie ist ein Modell,
und sie sieht gut aus.

Kraftwerk
(Das Modell)

In this chapter we present a model for the problem of computing multiple sum-of-pairs sequence-structure alignments. The formulation unifies the models from [4] and [1]. Section 4.3 describes an extension to the initial model that takes the effects of stacking of adjacent base pairs into account. Main parts of this chapter are published as [8]. Mind that for the moment we restrict ourselves to the description of the formulation, we give extensive computational results in Chap. 5.3.

## 4.1 An Exact Framework for the Multiple Sequence-Structure Alignment Problem

Section 4.1.1 starts with mathematical definitions of alignments, gaps, and appropriate scoring functions. We then give a graph-based view of these definitions in Sect. 4.1.2. Subsequently, Sect. 4.2 shows how we can transform the graph-based model into an *integer linear program* (ILP), relax it and solve the relaxed ILP efficiently.

### 4.1.1 Basic Definitions

**Definition 4.1.** Let $\Sigma$ 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 $\Sigma$, 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:

1. the sequences $\hat{s}^i$, $1 \leq i \leq k$, are over the alphabet $\hat{\Sigma}$.

2. all sequences $\hat{s}^i$ have the same length $|A|$.

3. sequence $\hat{s}^i$ without "-" corresponds to $s^i$, for $1 \leq i \leq k$.

4. there is no index $j$ such that $\hat{s}^i_j =$ "-", $1 \leq i \leq k$. By $s^i_j$ we refer to the $j$th 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

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

Figure 4.1: 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.

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 the 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 4.2.** 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 $\sigma$, the *sum-of-pairs* (SPS) score of $A$ is defined by

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

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

**Definition 4.3.** Given a scoring function $\sigma$ and a set $S$ of sequences, we aim at computing an alignment $A^*$ with

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

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 $\sigma$.

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—aligning three 'A's to three gaps—achieves the same score as three separate individual gaps, see Fig. 4.1 (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. 4.1 (c), over the three individual gaps, see Fig. 4.1 (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 $\ell$ in sequence $i$ at position $j$ by a triple $(i, j, \ell)$ and assign it a penalty score $\gamma(i, j, \ell) \in \mathbb{R}_{\leq 0}$.

**Definition 4.4.** 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 $\sigma$, and a gap penalty function $\gamma$. 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

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

Note that $\gamma$ assigns negative scores to gaps in the alignments.

As described in Sect. 1, sequence alignments are in general not sufficient 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 4.5.** 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 4.6.** 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 $(s^i_e, s^i_f) \in p^i$ and $(s^j_g, s^j_h) \in p^j$ a *structural match* if $s^i_e$ is aligned with $s^j_g$ and $s^i_f$ is aligned with $s^j_h$. Two structural matches $(\hat{s}^i_e, \hat{s}^i_f)$, $(\hat{s}^j_e, \hat{s}^j_f)$ and $(\hat{s}^i_g, \hat{s}^i_h)$, $(\hat{s}^j_g, \hat{s}^j_h)$ 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 the case of a structural match of two interactions, their “left” and “right” endpoints are aligned by $A$. Two structural matches are *inconsistent*

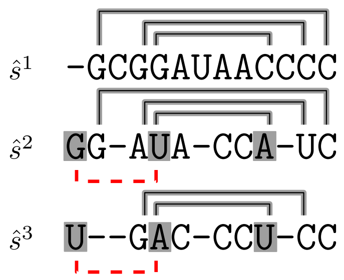

$\hat{s}^1$ -GCGGAUAACCCC
$\hat{s}^2$ GG-AUA-CCA-UC
$\hat{s}^3$ U--GAC-CCU-CC

Figure 4.2: 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 $y$.

if they share an aligned column. In the 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 for RNA sequences. This leads to the definition of *sequence-structure alignments* of RNA structures.

**Definition 4.7.** 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 $\sigma$, $\tau$, $\gamma$ be functions for scoring sequence, structural matches, and gaps. Then, the *gapped structural sum-of-pairs* score of $A$ is defined by $\mathrm{GSSPS}(A, \sigma, \tau, \gamma) =$

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

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

Figure 4.2 gives an illustration for the definitions from above. In analogy to the optimal sequence alignment problem, we consider the *optimal sequence-structure alignment* of RNA structures:

**Definition 4.8.** Given scoring functions $\sigma$, $\tau$, and $\gamma$ 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

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

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

### 4.1.2 Graph-Based 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 mixed graph $(V, L \cup F \cup D \cup G)$. The set $V$ denotes the

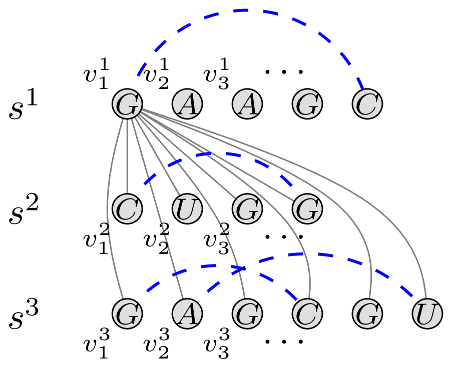

$s^1$
$s^2$
$s^3$
$v_1^1$ $v_2^1$ $v_3^1$ ...
G A A G C
C U G G
$v_1^2$ $v_2^2$ $v_3^2$ ...
G A G C G U
$v_1^3$ $v_2^3$ $v_3^3$ ...

Figure 4.3: Basic graph model of three annotated sequences containing lines (grey solid lines) and interaction edges (blue dotted edges). For sake of clarity we do not show all alignment edges, only the ones incident to $v_1^1$.

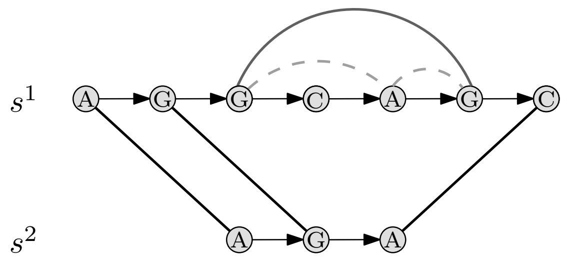
vertices of the graph, in this case the bases of the sequences, and we write $v_j^i$ for the $j$th base of the $i$th sequence. The set $L$ contains undirected *alignment edges* between vertices of two different input sequences—for sake of better distinction called *lines*. A line $l \in L$ with $l = (v_k^i, v_l^j), i \neq j$ 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 $t(l)$. For $l = (v_k^i, v_l^j)$ we have $s(l) = v_k^i$ and $t(l) = v_l^j$. The set $L^{ij}_{v_k^i}$ is the subset of $L^{ij}$ containing only alignment edges whose source node is $v_k^i$. 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 $(v_k^i, v_l^i)$ representing the interaction between vertices $v_k^i$ and $v_l^i$. Figure 4.3 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.*, $D = \{(v_j^i, v_{j+1}^i) \mid 1 \leq i \leq k, 1 \leq j < |s^i|\}$.

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 $a_{ef}^{ij}$ from $v_e^i$ to $v_f^i$ representing the fact that no character of the substring $s_e^i \ldots s_f^i$ is aligned to any character of the sequence $j$, whereas $s_{e-1}^i$ (if $e > 1$) and $s_{f+1}^i$ (if $f + 1 \leq |s^j|$) are aligned with some characters in sequence $j$. We say that $v_e^i, \ldots, v_f^i$ are *spanned* by the gap arc $a_{ef}^{ij}$. The entire set $G$ is partitioned into distinct subsets $G^{ij}$ with $i, j = 1, \ldots, k$, $i \neq j$, and $G^{ij} = \{a_{lm}^{ij} \in G \mid 1 \leq l \leq m \leq |s^i|\}$. Intuitively, for each sequence $i$ we have $k - 1$ arcs between each pair of nodes $(v_e^i, v_f^i)$ in order to represent gaps between the sequence and the remaining $k - 1$ sequences.

Two gap arcs $a_{ef}^{ij}$, $a_{mn}^{ij} \in G^{ij}$, w.l.o.g. $e < m$, are *in conflict* with each other if $\{e, \ldots, f + 1\} \cap \{m, \ldots, n\} \neq \varnothing$, that is, we do not allow overlapping or even



```
AGGCAGC
AG----A
```

Figure 4.4: 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.

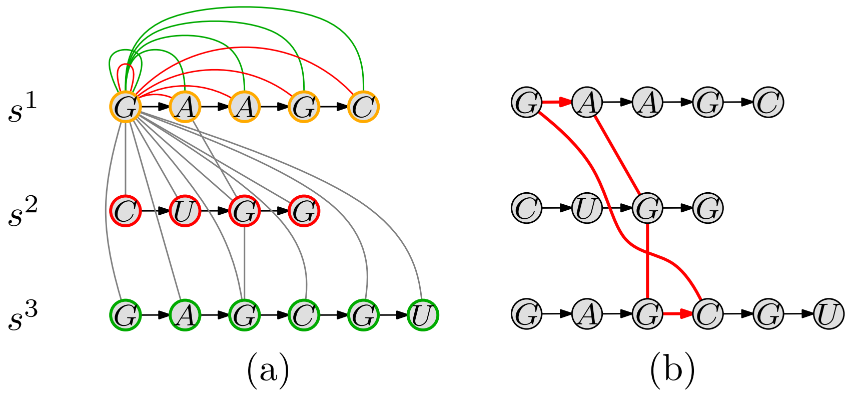


Figure 4.5: (a) Basic graph model augmented by gap edges (interaction edges are not displayed). The colour of the gap edges indicates to what other input sequence the gap edges refer to. The right side (b) shows an instance of a mixed cycle.

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 $v^i_e \ldots v^i_f$. See Fig. 4.4 for an illustration.

**Mixed Cycles** A *path* in $(V, L \cup D)$ is an alternating sequence $v_1, e_1, v_2, e_2, \ldots$ 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 $s^1, \ldots s^k$ if $\mathcal{L} \cup D$ does not contain a mixed cycle [79; 117]. In this case, we use the term alignment for $\mathcal{L}$.

**Interaction Match** Two interaction edges $r = (v_k^i, v_l^i) \in p^i$ and $s = (v_m^j, v_n^j) \in p^j$ form an *interaction match* if two lines $e = (v_k^i, v_m^j)$ and $f = (v_l^i, v_n^j)$ exist such that $e$ and $f$ do not cross each other. A subset $\mathcal{L} \subset L$ *realizes* the interaction match $(e, f)$ if $e, f \in \mathcal{L}$. Observe that the definition of an interaction match is a graph-based reformulation of a structural match as defined in Sect. 4.1.1. The set $I$ contains all possible 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 $i, j = 1, \ldots, k, i \neq j$ we define $\mathcal{L}^{ij} = L^{ij} \cap \mathcal{L}$. Then, for $l = 1, \ldots, |s^i|$ the vertex $v_l^i$ 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 $a_{kl}^{ij}, a_{mn}^{ij} \in \mathcal{G}$ such that $a_{kl}^{ij}$ is in conflict with $a_{mn}^{ij}$.
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. 4.6(a) for an illustration.

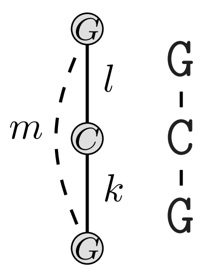
Fig. 4.6(b) shows a valid gapped structural trace and the corresponding alignment.

**Observation 4.1.** *There is a one-to-one mapping between alignments realizing structural matches and gapped structural traces.*

*Proof.* The correspondence follows the observation in [1]. 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 can set each weight independently, line weights are usually set by empirically derived mutation score matrices where $\sigma(s_k^i, s_l^j)$ gives a high value for identical (or similar) characters. In Sect. 5.4.2 we will further elaborate on commonly used scoring schemes.

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 $s_k^i \ldots s_l^i$ with gap characters in sequence $j$.



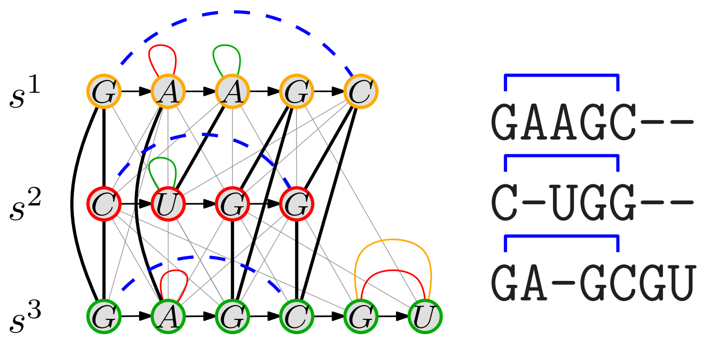


Figure 4.6: (a) Transitive edges must be realized: if $k$ and $l$ are part of the alignment, then $m$ has to be realized as well. (b) Example of a valid gapped structural trace of three annotated sequences. Three interaction matches are conserved by the alignment.

## 4.2 Integer Linear Program and Lagrangian Relaxation

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

### 4.2.1 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 having the following interpretation: 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 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 $\mathcal{L}$ 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 4.7 gives an illustration of the variable splitting. This does not change the underlying model, it just makes the ILP formulation more convenient for further processing as we shall see in the sections to come.

Splitting interaction matches has first been proposed by Caprara and Lancia in the context of contact map overlap [20]. The general concept of *variable*

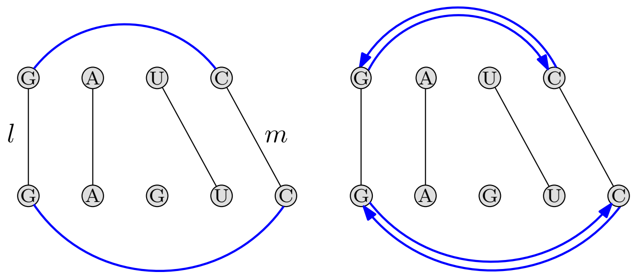


Figure 4.7: One interaction match is split into two *directed* interaction matches.

*splitting*, or *Lagrangian decomposition*, is, however, a well-known technique in mathematical programming [56].

$$
\begin{array}{llr}
\max & \sum_{l\in L} w_l x_l + \sum_{g\in G} w_g z_g + \sum_{l\in L}\sum_{m\in L} w_{lm} y_{lm} & (4.1)\\
\text{s. t.} & \sum_{l\in L\cap M} x_l \le |L\cap M| - 1 \qquad \forall\, M\in\mathcal{M} & (4.2)\\
& x_l + x_k - x_m \le 1 \quad \forall\,(l,k,m)\in L,\ (x_l,x_k,x_m) \text{ forming a cycle} & (4.3)\\
& \sum_{a\in C} z_a \le 1 \qquad \forall\, C\in\mathcal{C} & (4.4)\\
& \sum_{l\in L^{ij}_{s(m)}} x_l + \sum_{a\in G^{ij}_{s(l)\leftrightarrow s(l)}} z_a = 1 \qquad 1\le i,j\le k, i\ne j, \forall\, m\in L^{ij} & (4.5)\\
& \sum_{\substack{m\in L,(l,m)\\ \text{not crossing}}} y_{lm} \le x_l \qquad \forall\, l\in L & (4.6)\\
& y_{lm} = y_{ml} \qquad \forall\, l,m\in L & (4.7)\\
& x\in\{0,1\}^L \quad y\in\{0,1\}^{L\times L} & (4.8)\\
& z\in\{0,1\}^G & (4.9)
\end{array}
$$

Figure 4.8: Master ILP

**Definition 4.9.** We call the ILP (4.1)–(4.9) of Fig. 4.8 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. 4.1.2, and therefore we have $w_g < 0$ for $g\in G$.

**Lemma 4.1.** *A feasible solution to the ILP* (4.1)–(4.9) *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 (4.2) 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 (4.2) 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 (4.3).

Constraints (4.4) guarantee that there are no mutually crossing gap edges: assume there exist 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 (4.4) is constrained by 1.

Equality (4.5) 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 (4.6).

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 4.10.** We call the relaxed ILP consisting of (4.1)–(4.9) without (4.7) 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 (4.7), variables $y_{lm}$ appear only in constraints (4.6); 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 (4.6), all $y_{lm}$ must be zero as well. If, however, a line $l = (v^i_k, v^j_l)$ is part of an alignment, its maximal contribution to the score is given by solving the ILP shown in Fig. 4.9. Inequality (4.11) 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 (4.10) 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 (4.12) 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.*, $p_l = w_l + w_{l\hat{m}}$.

$$p_l := \max \quad w_l + \sum_{m \in L} w_{lm} y_{lm} + \sum_{a \in \{G^{ij}_{s(l)\leftrightarrow s(l)} \cup G^{ji}_{t(l)\leftrightarrow t(l)}\}} w_a z_a \tag{4.10}$$

$$\text{s. t.} \quad \sum_{\substack{m \in L,(l,m) \\ \text{not crossing}}} y_{lm} \le 1 \tag{4.11}$$

$$\sum_{a \in \{G^{ij}_{s(l)\leftrightarrow s(l)} \cup G^{ji}_{t(l)\leftrightarrow t(l)}\}} z_a = 0 \tag{4.12}$$

$$x \in \{0,1\}^L \quad y \in \{0,1\}^{L \times L} \tag{4.13}$$

$$z \in \{0,1\}^G \tag{4.14}$$

Figure 4.9: Constraints that have to satisfied if an alignment edge $l$ is part of the alignment, *i.e.*, if $x_l = 1$.

In the second step, we compute the optimal score by solving the ILP consisting of the remaining constraints, which is listed in Fig. 4.10.

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

Figure 4.10: Computing the solution for the relaxed problem. Observe that the ILP only contains $x$ and $z$ variables, because the values of the $y$ variables depend on the $x$ variables.

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 [1].

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 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 (4.6). To see that $(x^*, y^*, z^*)$ is optimal, observe that its value is determined by

$$\begin{aligned}\sum_{l\in L} p_l x^*_l + \sum_{g\in G} w_g z^*_g &= \sum_{l\in L}(w_l + w_{l\hat{m}})x^*_l + \sum_{g\in G} w_g z^*_g \\ &= \underbrace{\sum_{l\in L} w_l x^*_l + \sum_{g\in G} w_g z^*_g}_{\text{optimal sol. for MSA}} + \underbrace{\sum_{l\in L}\sum_{m\in L} w_{lm} y^*_{lm}}_{\text{optimal sol. for } y_{l\hat{m}} \text{ due to (4.10)–(4.14)}}\end{aligned}$$

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.

■

### 4.2.2 Lagrangian Relaxation

Obviously we have not yet solved the master ILP, since we dropped equalities (4.7). 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 the case the dropped constraint is not satisfied. Moving constraints (4.7) into the objective function yields the *Lagrangian dual*, which is the slave ILP with the objective function

$$\max \sum_{l\in L} w_l x_l + \sum_{g\in G} w_g z_g + \sum_{l\in L}\sum_{m\in L} w_{lm} y_{lm} + \sum_{l\in L}\sum_{m\in L} \lambda_{lm}(y_{lm} - y_{ml}) \ . \tag{4.15}$$

Exploiting the fact that $\lambda_{lm} = -\lambda_{ml}$, which we ensure below, (4.15) can be reformulated to

$$\max \sum_{l\in L} w_l x_l + \sum_{g\in G} w_g z_g + \sum_{l\in L}\sum_{m\in L} (w_{lm} + \lambda_{lm}) y_{lm} \ . \tag{4.16}$$

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.

We want to determine the Lagrangian multipliers that provide the best bound to the original problem. In practice, iterative *subgradient optimization*, that we described in Sect. 2.3.1, is widely used. This method determines the multipliers of the current by adapting the values from the previous iteration.

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

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

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

Here, $\mu$ 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. The computation of a lower bound is more involved and we show in Sect. 4.2.3 how to use the solution of the relaxed problem 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* [94]. However, currently the described subgradient optimization exhibits better convergence properties than bundle methods as the results from Sect. 5.4.4 show.

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 the case of instances that share medium or high structural similarity, the lower and upper bound often coincide yielding provably 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. We report the results of the branch-and-bound implementation in Sect. 5.5.4.

**Solving the relaxed problem in the pairwise case.** The solution of the relaxed problem in the multiple case amounts to the computation of an exact multiple sequence alignment. If we consider the special case of two input sequences $s^1$ and $s^2$, with $n = |s^1|$ and $m = |s^2|$ and $n > m$, then we can use standard dynamic programming algorithms to solve the relaxed problem in $\mathcal{O}(n^2)$.

For the total running time of $k$ iterations we have to additionally consider other factors. We have $\mathcal{O}(n^2)$ possible alignment edges, and if we allow interactions between every pair of nucleotides, then every alignment edge has $\mathcal{O}(n)$ possible partner edges. We store the partner edges in a priority queue leading to a complexity of $\mathcal{O}(n \lg n)$ for building and updating each one of the $\mathcal{O}(n^2)$ priority queues. This yields a complexity of $\mathcal{O}(n^3 \lg n)$ for a fixed number of iterations. Priority queues are necessary, because we adapt the Lagrangian multipliers in each iterations, and we want to access the highest scoring interaction match in constant time.

For an RNA sequence the number of potential interactions is, however, typically constant, leading to a constant number of possible partner edges in the case of sequences. Therefore, the $\mathcal{O}(n \lg n)$ term is in fact constant, yielding a total running time of $\mathcal{O}(n^2)$.

### 4.2.3 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. 4.11 (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 4.11 (a) illustrates the problem: the alignment $\mathcal{L} = (l, k, m, n, o)$ provides 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 completion* of an alignment $\mathcal{L}$.

**Definition 4.11.** 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*: we have $M_S = (V, E)$ 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 $i$th 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. 4.11 (b)). The weight of edge $e_i$ is given by the weight of the two interaction edges $(v^i_k, v^i_l)$ and $(v^j_m, v^j_n)$.

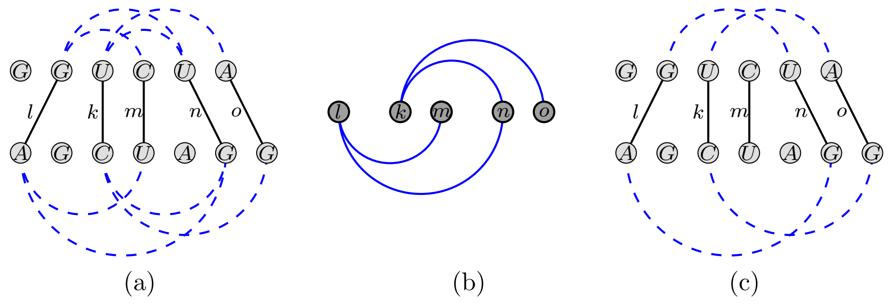


Figure 4.11: Given the alignment $\mathcal{L} = (l, k, m, n, o)$ , we have different possibilities to augment the alignment with structural matches. Creating an interaction matching graph (b) and calculating a general matching of maximum weight yields the best structural completion of $\mathcal{L}$ (c).

**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. ■

## 4.3 Incorporating Stacking Energies Into the Model

Section 1.2 describes the loop-energy model that builds the basis for the computational prediction of RNA structures. The *stacking energies* of paired bases build the prevalent contribution to the overall stability of an RNA structure. The model that we described in Sect. 4.1.2 does not account for stacking energies, because it treats every interaction separately. There is no additional benefit for realizing adjacent paired bases.

We call two interaction matches $(l, k)$ and $(m, n)$ with $s(l) = s(m) - 1$, $s(k) = s(n) + 1$, $t(l) = t(m) - 1$, and $t(k) = t(n) + 1$ the *stacking interaction match* $[(l, k), (m, n)]$. Figure 4.12 shows the stacking interaction match $[(l, k), (m, n)]$.

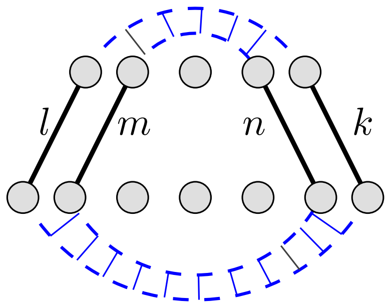


Figure 4.12: The two interaction matches $(l, k)$ and $(m, n)$ form the stacking interaction match $[(l, k), (m, n)]$.

In the following, we will extend the model from Sect. 4.1.2 by incorporating stacking interaction matches. For sake of simplicity, we shall start from a stripped-down version of the full model. We will consider the pairwise case, and we will include only the alignment and structure edges. The extension to the multiple case including gap edges is straightforward, but it makes the description more tedious.

First, we define the problem formally. In Section 4.1.1 we gave the formal definition for sequence-structure alignments which we will extend by stacking scores.

**Definition 4.12.** Given two strings $s^1$ and $s^2$ and an alignment $A$ consisting of the two strings $\hat{s}^1$ and $\hat{s}^2$. We define a scoring function $\upsilon : \Sigma^4 \rightarrow \mathbb{R}$ that assigns a score to quadruples of characters representing the benefit of stacking interactions, *i.e.*, $\upsilon(s^1_l, s^2_l, s^1_m, s^2_m)$ scores the stacking between interaction matches $(s^1_l, s^2_l, s^1_m, s^2_m)$ and $(s^1_{l-1}, s^2_{l-1}, s^1_{m+1}, s^2_{m+1})$.

Let $\sigma$, $\tau$, $\upsilon$ be functions for scoring sequence, structural matches, and stacking interactions. Then, the *stacking sum-of-pairs* score SSPS$(A, \sigma, \tau, \upsilon)$ of $A$ is defined as

$$\sum_{l=1}^{|A|} \sigma(\hat{s}^1_l, \hat{s}^2_l) + \sum_{l=1}^{|A|-1} \sum_{m=l+1}^{|A|} \tau(\hat{s}^1_l, \hat{s}^2_l, \hat{s}^1_m, \hat{s}^2_m) + \sum_{l=2}^{|A|-2} \sum_{m=l+1}^{|A|-1} \upsilon(\hat{s}^1_l, \hat{s}^2_l, \hat{s}^1_m, \hat{s}^2_m) \quad .$$

We do not score inconsistent structural matches, that is, every base is part of at most one structural match, and we only score stacking contributions between realized adjacent structural matches.

Similar to the optimal sequence alignment problem, we consider the *optimal sequence-structure-stacking alignment* of RNA structures:

**Definition 4.13.** Given scoring functions $\sigma$, $\tau$, and $\upsilon$ for scoring sequence, structural matches, and stacking contributions, we aim at computing an alignment $A^*$ of two sequences $s^1$ and $s^2$ with

$$\text{SSSPS}(A^*, \sigma, \tau, \upsilon) = \max\nolimits_{A \in \mathcal{A}} \text{SSSPS}(A, \sigma, \tau, \upsilon) \quad ,$$

where $\mathcal{A}$ is the set of all possible pairwise alignments for $s^1$ and $s^2$. We call $A^*$ an *optimal pairwise sequence-structure-stacking* alignment of $S$.

We do not have to add new edges to the graph-based model described in Sect. 4.1.2 to model the stacking contributions, because they are implicitly represented by the structure edges. In the following, we will adapt the ILP formulation to take stacking scores into account.

## 4.3.1 Integer Linear Program Including Stacking Scores

Figure 4.13 shows the ILP describing the pairwise sequence-structure alignment model without considering gap edges. Remember that we associate an $x$ and $y$

$$
\begin{aligned}
\max \quad & \sum_{l \in L} w_l x_l + \sum_{l \in L} \sum_{m \in L} w_{lm} y_{lm} & \\
\text{s.t.} \quad & \sum_{l \in C_L} x_l \leq 1 & \forall\, C_L \in \mathcal{C}_L \\
& \sum_{\substack{m \in L, (l,m) \\ \text{not crossing}}} y_{lm} \leq x_l & \forall\, l \in L \\
& y_{lm} = y_{ml} & \forall\, l, m \in L \\
& x \in \{0,1\}^L \quad y \in \{0,1\}^{L \times L} &
\end{aligned}
$$

Figure 4.13: The ILP that describes pairwise sequence-structure alignment without gap costs.

variable each every alignment edge and directed interaction match, respectively. We now add variables $z$ that model potential stacking between pairs of adjacent interaction matches. We have $z_{lm|nk} = 1$ if and only if the stacking match $[(l,k),(m,n)]$ between the adjacent interaction matches $(l,k)$ and $(m,n)$ is realized, and $z_{lm|nk} = 0$ otherwise. If we have $z_{lm|nk} = 1$, then $(l,k)$ and $(m,n)$ realize a *stacking interaction match*.

Similar to the splitting of an interaction match $(l,k)$ into two directed interaction matches $(l,k)$ and $(k,l)$, we also split a stacking interaction match $[(l,k),(m,n)]$ into two directed stacking interaction matches, associated with separate $z$ variables $z_{lm|nk}$ and $z_{nk|lm}$.

Figure 4.14 gives the ILP that describes the model extended by the stacking variables. Observe that the ILP only enforces $z_{lm|nk} \leq y_{mn}$, but we do not have to explicitly enforce $z_{lm|nk} \leq y_{lk}$ since this is automatically satisfied in the case of feasible solutions. If we have $z_{lm|nk} = 1$, then $z_{nk|lm} = 1$ is true as well due to constraint (4.22). With $z_{nk|lm}$ being set to 1 we have $y_{kl} = 1$ because of constraint (4.20), and then in turn $y_{lk} = 1$ due to equality constraints (4.21).

**Lemma 4.4.** *A feasible solution to the ILP* (4.17)–(4.23) *matches the definition of a sequence-structure-stacking alignment from Def. 4.12.*

*Proof.* We first prove that a feasible solution $(\hat{x}, \hat{y}, \hat{z})$ of the ILP describes a valid sequence-structure-stacking alignment.

Observe that constraints (4.18) and (4.19) guarantee that the subset of alignment and structure edges (represented by the $\hat{x}$ and $\hat{y}$ variables) form a valid sequence-structure alignment. There are no crossing edges and every alignment edge realizes at most one interaction edge.

Furthermore, constraint (4.20) ensures that the alignment only incorporates stacking scores, if the two stacking interaction matches are realized. The score obviously equals the score of the alignment.

$$
\begin{aligned}
\max \quad & \sum_{l \in L} w_l x_l + \sum_{l \in L} \sum_{m \in L} w_{lm} y_{lm} + \sum_{l,m,n,k \in L} w_{lm|nk} z_{lm|nk} && && (4.17)\\
\text{s.t.} \quad & \sum_{l \in C_L} x_l \leq 1 && \forall\, C_L \in \mathcal{C}_L && (4.18)\\
& \sum_{\substack{m \in L, (l,m) \\ \text{not crossing}}} y_{lm} \leq x_l && \forall\, l \in L && (4.19)\\
& z_{lm|nk} \leq y_{mn} && \forall\, l,m,n,k \in L && (4.20)\\
& y_{lm} = y_{ml} && \forall\, l,m \in L && (4.21)\\
& z_{lm|nk} = z_{nk|lm} && \begin{array}{ll}(l,m) & \text{stacked,} \\ (n,k) & \text{stacked}\end{array} && (4.22)\\
& x \in \{0,1\}^L \quad y \in \{0,1\}^{L \times L} \quad z \in \{0,1\}^{L \times L \times L \times L} && && (4.23)
\end{aligned}
$$

Figure 4.14: The ILP that incorporates stacking energies.

To complete the proof, we have to show that a valid sequence-structure-stacking alignment represents a feasible solution to the ILP. Given $(\mathcal{L}, \mathcal{I})$ with $\mathcal{L} \subseteq L$ and $\mathcal{I} \subseteq I$, we set the values of the $\hat{x}$ and $\hat{y}$ variables in correspondence if the respective edges are part of $\mathcal{L}$ and $\mathcal{I}$. Observe that the values of the $z$ variables are implicitly given by the $y$ variables. ■

Figure 4.15 shows an illustration of the three different sets of variables.

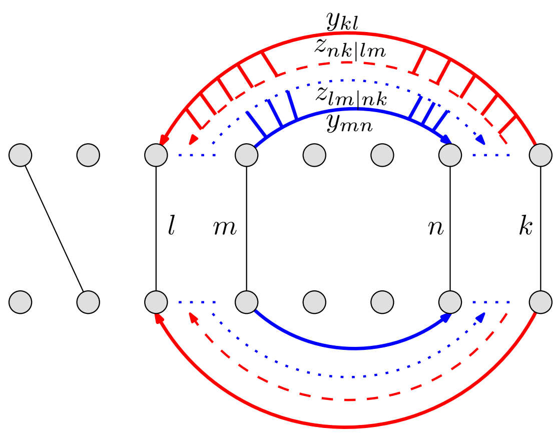


Figure 4.15: Illustration of the ILP incorporating stacking energies. The blue and red arrows represent stacking variables together with their respective structure edges.

Thus, after relaxing constraints (4.21) and (4.22) and moving them to the objective function, we get the ILP shown in Fig. 4.16.

$$
\begin{aligned}
\max\ & \sum_{l\in L} w_l x_l + \sum_{l\in L}\sum_{m\in L}(w_{lm}+\lambda_{lm})y_{lm} + \sum_{l,m,n,k\in L}(w_{lm|nk}+\lambda_{lm|nk})z_{lm|nk} && (4.24)\\
\text{s. t.}\ & \sum_{l\in C_L} x_l \le 1 && \forall\, C_L \in \mathcal{C}_L \quad (4.25)\\
& \sum_{\substack{m\in L,(l,m)\\ \text{not crossing}}} y_{lm} \le x_l && \forall\, l \in L \quad (4.26)\\
& z_{lm|nk} \le y_{mn} && \forall\, l,m,k \in L \quad (4.27)\\
& x \in \{0,1\}^L \quad y \in \{0,1\}^{L\times L} \quad z \in \{0,1\}^{L\times L\times L\times L} && (4.28)
\end{aligned}
$$

Figure 4.16: The ILP incorporating stacking energies relaxed by two classes of constraints.

We solve the relaxed problem in a similar way as the ILP without stacking bonuses. Like in the case of the initial model from Sect. 4.1, we again have to distinguish between $x_m = 0$ and $x_m = 1$: if $x_m = 0$, then all $y_{ml}$ will be 0 due to (4.26). With $y_{ml} = 0$ for all possible interaction matches, constraint (4.27) will set all $z_{lm|nk}$ variables to 0.

In the case of $x_m = 1$, however, at most one $y_{ml}$ can be set to 1. Then, for the variable $y_{ml} = 1$ the corresponding stacking interaction match variable $z_{lm|nk}$ can be set to 1. We have a cascading of the $x$, $y$, and $z$ variables. The $x$ variables possibly switch some $y$ variables from 0 to 1, and the $y$ variables in turn set $z$ variables to 1.

The computation of the maximal profit for each alignment edge has to be adapted compared to the description in Sect. 4.2.2. Instead of maximizing the sum of the alignment edge score and the highest scoring directed interaction match, we additionally evaluate the maximum profit that a stacking variable plus the associated structure edge can possibly achieve. For an alignment edge $m$, associated with an alignment score $w_m$, we have $\hat{p}_0 = w_{m\hat{n}}$ as the maximum score of only an interaction match associated with alignment edge $\hat{n}$. The value $\hat{p}_1 = w_{m\bar{n}} + w_{lm|\bar{n}k}$ is the maximum score that an interaction match $(m, \bar{n})$ plus the corresponding stacking interaction match $[(l,k),(m,\bar{n})]$ can realize. The profit of alignment edge $m$ is then given by

$$p_m = w_m + \max\{\hat{p}_0, \hat{p}_1\}\ .$$

Following the description in Sect. 4.2.2, we get a solution for the relaxed ILP by computing a standard sequence alignment problem with the profit values $p_m$ as the matching scores for each alignment edge $m$.

We compute optimal or near-optimal solutions for the dual problem—the ILP consisting of (4.24)-(4.28)—by again resorting to subgradient optimization. We adapt the Lagrangian multipliers the same way as in the model for multiple sequence-structure alignments with arbitrary gap costs.

### 4.3.2 Computing a Feasible Solution

Solving the relaxed problem (4.24)-(4.28) does not usually yield a solution that is also valid for the original problem. If this is the case, then we have found an optimal solution for the original problem, because the number of subgradients is zero. In Sect. 4.2.3 we described how we generate a feasible solution for the original problem, given the solution of the relaxed problem. We build the interaction matching graph and perform a maximum-weight matching computation in it. The matching corresponds to a feasible solution in our original problem, see Fig. 4.11 for an illustration.

In principle, the same algorithm also works in the extended model. We compute the matching of maximum weight and add the scores for stacking interaction matches in a postprocessing step. The computational experiments in Sect. 5.4.3.1 show that the resulting pairwise alignments are competitive or better than the alignments without stacking energies. There is, however, one problem. The value of the maximum-weight matching plus the scores of realized stacking scores does not necessarily have to be the optimal value.

Figure 4.17 gives a toy example where the matching routine does not compute the structural completion of maximum weight. The matching selects the edges $(l, r)$ and $(m, o)$ as the structural completion of the alignment. Due to the stacking score of 100 for the stacking match $[(k, r), (m, o)]$, the edges $(k, r)$ and $(m, o)$ yield a higher score.

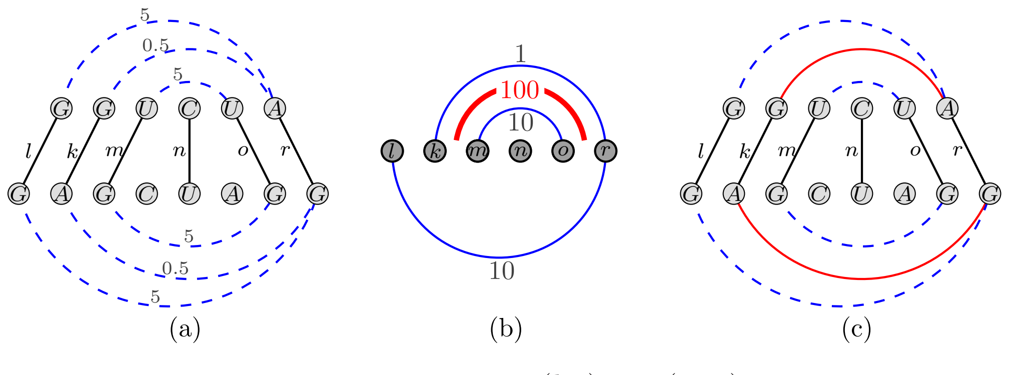


Figure 4.17: The matching selects the edges $(l, r)$ and $(m, o)$ as the structural completion of the alignment. Due to the stacking score of 100 for the stacking interaction matches $(k, r)$ and $(m, o)$, the edges $(k, r)$ and $(m, o)$ form the optimal solution.

Figure 4.18 shows an alternative way for the computation of a feasible solution that includes the stacking contributions. We reduce the problem to the computation of an *independent set of maximum weight*. We first give a definition

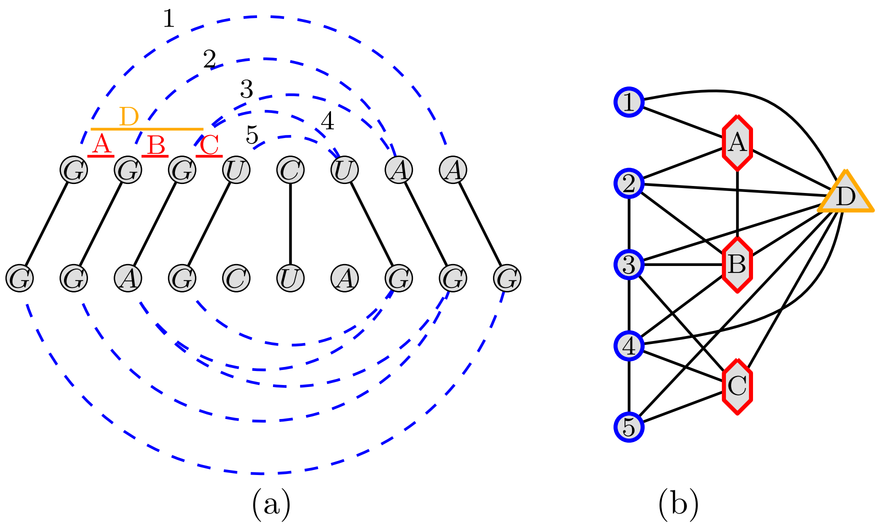


Figure 4.18: We determine the stacked structural completion of an alignment $A$ by computing an *independent set* of maximum weight. We have an alignment $A$ (a) and construct an auxiliary graph (b) in which we determine the independent set of maximum weight.

of what we want to maximize, and afterwards we describe the contruction of an auxiliary graph.

**Definition 4.14.** We are 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 *stacked structural completion*—and that maximizes the interaction match scores of $\hat{\mathcal{I}}$ plus the stacking scores that are induced by $\hat{\mathcal{I}}$.

We formulate this problem as an *independent set of maximum weight problem* in an auxiliary graph $M_{\mathrm{IS}}$, the *independent set graph*. The graph $M_{\mathrm{IS}} = (V_I \cup V_S, E)$ contains the sets $V_I$, $V_S$ and $E$ that constitute vertices and edges, respectively. For an alignment $\mathcal{L} \subset L$ we create a node $v_i \in V_I$ for every possible interaction match that this alignment realizes. Furthermore, for every possible combination of stacking interaction matches we add another vertex $v_s \in V_S$ to the graph. Observe that this includes not only all pairwise stacking interaction matches, but also vertices for several consecutive stacking interaction matches. We insert an edge $e \in E$ between every two nodes that are in conflict with each other, *i.e.*,

1. if two interaction matches $m$ and $n$—represented by vertices $v_m$ and $v_n$—share an endpoint.

2. if interaction match $m$ is part of stacking interaction match $n$, we insert an edge between $v_m$ and $v_n$.

For $v_i \in V_I$ the weight $w(v_i)$ of vertex $v_i$ is the weight of the corresponding interaction match. For $v_s \in V_S$, with $v_s$ representing interaction matches $y_0, \ldots, y_m$ and stacking interaction matches $z_0, \ldots, z_n$, the node weight is the sum of the weights of all the (stacking) interaction matches, *i.e.*, we have

$$w(v_s) = \sum_{i=0}^{m} w(y_i) + \sum_{j=0}^{n} w(z_j) \quad .$$

**Lemma 4.5.** *An independent set of maximum weight in the independent set graph $M_{\text{IS}}$ corresponds to the best stacked structural completion of $\mathcal{L}$.*

*Proof.* The equivalence follows directly from the construction of $M_{\text{IS}}$ and the definition of an independent set. ∎

Constructing a feasible solution in our augmented model by solving an independent set problem is the last resort that we have, because this computation is NP-complete [49]. Determining a feasible solution in the initial, *i.e.*, stackless model could also be reduced to MIS, but in this case we can reduce it to max-weight matching computations instead. The question is whether this holds also true in the case of stacking scores, *i.e.*, whether there exists an algorithm running in polynomial time that computes the maximal stacking completion for an alignment $\mathcal{L}$. The other option is to prove that the problem is indeed NP-hard.

# Chapter 5 Computational Results

It's been a hard day's night,
and I've been working like a dog.
The Beatles
(A Hard Day's Night)

This chapter describes the computational experiments that we performed with our prototypical implementations of the models from Chap. 4. We first present how we generate the input graph for our model, and subsequently show how we score the edges of the input graph. Sections 5.3 and 5.4 contain the computational experiments using the exact and heuristic approach to multiple sequence-structure alignments. Section 5.5 lists the results for computing pairwise alignments using the bundle method. Furthermore, we give results on running the Lagrange approach within a branch-and-bound framework to verify the optimality of the solutions. Sections 5.3 and 5.4 are published as parts of [8] and [7].

## 5.1 Constructing the Input Graph

### 5.1.1 Generation of Alignment Edges

For sake of simplicity, we will restrict ourselves to the description of the pairwise case. The same ideas apply to the multiple case as well.

We use different strategies for the generation of alignment edges. The first natural choice is to insert all possible alignment edges between the two sequences, yielding the complete bipartite graph as shown on the left side of Fig. 5.1. Every nucleotide of the first sequence can be mapped onto every nucleotide of the second sequence.

Most of these edges, however, will not be part of any optimal or near-optimal sequence-structure alignment. We therefore follow the strategy that we already employed in previous work [4; 6; 5; 95]: 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 real multiple structural alignment, *e.g.*, a hand-curated alignment like an RFAM seed alignment, our experiments on the RFAM database show that RFAM reference alignments consist of alignment edges of small suboptimality. To this end, we randomly extracted 10 sequences from the seed alignment of eight random RFAM families (RF00001, RF00005, RF00020, RF00023, RF00029, RF00031, RF00059, RF00515) and computed the alignment edges that score at most 40 below the optimal score for all pairwise projections. For all but one family we can generate the alignment edges that form the original alignment at a suboptimality level of 40. For TPP riboswitches (RF00059) this level of suboptimality does not suffice to cover all alignment edges. There are six pairwise projections that miss alignment edges. At a suboptimality level of 90, however, all alignment edges are created.

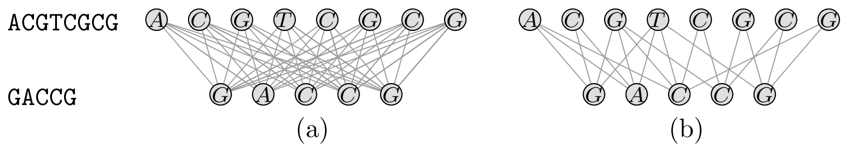


Figure 5.1: Given the two sequences on the left, we either create the complete bipartite graph (a), or thin out the graph using an available bound or suboptimal matches (b).

Another way to generate alignment edges is to start from the complete bipartite graph and subsequently delete alignment edges that cannot be part of the optimal—with respect to our objective function—alignment. Given two sequences $s^1$ and $s^2$, we start from a scoring matrix $\Delta$ with $\Delta(i,j)$ being the score of alignment edge $a = (s^1_i, s^2_j)$. Then, we compute the maximum score $\mathcal{S}(\mathcal{A})$ of an alignment $\mathcal{A}$ realizing alignment edge $a$ by summing up the score of $a$ and the sum of the prefix and suffix alignment induced by $a$. Given a valid sequence-structure alignment $\mathcal{A}_{ss}$ of score $\mathcal{S}(\mathcal{A}_{ss})$, we can safely drop alignment edge $a$ if $\mathcal{S}(\mathcal{A}) < \mathcal{S}(\mathcal{A}_{ss})$, because alignment $a$ cannot be part of an optimal sequence-structure alignment. We compute a valid solution $\mathcal{A}_{ss}$ of the sequence-structure alignment problem by running our Lagrange approach with a limited number of iterations or by simply computing a traditional sequence alignment and adding the scores of conserved interaction matches afterwards.

In our experiments we resort to the generation of alignment edges using suboptimal sequence matches, because it needs less computation time while the performance is comparable to the second procedure described above.

## 5.1.2 Generation of Interaction Edges

The generation of interaction edges reflects the knowledge that we have about the structural properties of the sequences. If we do not want to constrain the structure in any way, then we simply insert an interaction edge between any two nucleotides that can form hydrogen bonds. If we have, however, structural information for one of the sequences available, for example from the *Comparative RNA web (CRW) site* [19], then we insert only those interaction edges that form the secondary structure.

The LaRA default setting lies in between: we compute the partition function for the sequence (see Sect. 1.2) and derive *base pairing probabilities* for each pair of nucleotides using McCaskill's algorithm [105]. We then insert an interaction edge between every pair of nucleotides whose base pairing probability is higher than a minimum value $p_{\min}$. A typical value for $p_{\min}$ is 0.003.

## 5.2 Lagrange-Specific Parameters

In this section we describe the scores that we use throughout our experiments. Additionally, we briefly specify Lagrange-specific parameters that have significant impact on the convergence of the Lagrange solution process. We resort to subgradient optimization for adapting the Lagrangian multipliers.

### 5.2.1 Scores

**Sequence Scores.** We used different schemes for scoring the alignment edges. First, in our theoretical contributions [6; 8] we employed ad-hoc chosen matrices, like scoring matches and mismatches by 4 and 1 or by 2 and 1. Subsequently, we resort to more elaborate scoring systems like the RIBOSUM matrices [83] that were derived along the lines of ribosomal gold-standard alignments. The authors count the number of occurrences of the respective matches and derive log-odds scores by comparing them to a uniformly distributed background model.

We provide a parameter $\tau$ by which the user is able to scale the original scoring matrix values. If we do not want to put too much emphasis on the sequence, then $\tau$ will be very small. If sequence is important, like in the case of riboswitches, then one is able to scale the scores accordingly. By default, the value of $\tau$ is 1. In our experiments we use the RIBOSUM65 matrix as our default scoring scheme.

**Structure Scores.** The scoring system for the interaction edges is based on base pair probability matrices (BPP scoring in short). It transforms the probabilities into the additive log-scores in spirit of PMCOMP [67]. More precisely, given the probability $p_{ij}$ that nucleotides $i$ and $j$ pair, the score $s_{ij}$ reads

$$s_{ij} = \lg\left(\frac{p_{ij}}{p_{\min}}\right)$$

where lg is the natural logarithm and $p_{\min}$ is the smallest probability that we consider.

**Stacking Scores.** The score for the stacking weights $w_{[i,i+1|j-1,j]}$ are derived along the lines of the conditional stacking probabilities [15]. The value $p_{i,i+1|j-1,j} = P[(i+1,j-1)|(i,j)]$ is the probability that nucleotides $(i+1,j-1)$ form hydrogen bonds given that $(i,j)$ already pair. Then, the weight reads

$$w_{[i,i+1|j-1,j]} = \lg\left(\frac{p_{i,i+1|j-1,j}}{p_{\min}}\right) \quad .$$

We set the value of $p_{\min}$ to the same value as for the structure scores.

### 5.2.2 Other Program-Specific Parameters

There are several other parameters that influence the performance of the Lagrange solution process. First, the number of Lagrange iterations specifies how often the Lagrange multipliers can be adapted: the higher the number of iterations, the better the bounds are in general. Second, we need to specify the parameter $\mu$ which acts as a regulating factor in the computation of the step size $\gamma$ (see Sect. 4.1 for details). Finally, a common feature to all implementations of the subgradient solver is a number $n$ of non-decreasing rounds: if the value of the best upper bound does not decrease within $n$ iterations, we halve the value of $\mu$. This leads to smaller step sizes $\gamma$. In practice, we observe that the smaller step sizes support the convergence of the solution process if the algorithm got stuck at a certain point, *i.e.*, if the upper bound does not decrease within several iterations.

## 5.3 Results for the Exact Multiple Case

mLARA (*m*ultiple *La*grangian *r*elaxed *a*lignments) is our prototypical implementation of the formulation for multiple structural alignments presented in Chap. 4. The algorithm is easy to implement and comprises only a couple of hundred lines of code. For the computation of the lower bound, however, we use the matching routines from the LEDA library [91].

In the following, we shall give a proof-of-concept of our approach by running experiments on real data of moderate size, setting all gap costs to zero, and we assign scores of 4 and 1 to matches and mismatches, respectively. We set the scaling parameter $\tau$ to 1.0. The mLARA software package directly uses the values from the dotplot files—created by the RNAfold program—as the input to the log-odds transformation described in Sect. 5.2.1.

From the Rfam database [53] we downloaded sequences that belong to the families of ribosomal L19 leader proteins, tRNAs, and ribosomal 5S RNAs (Rfam IDs: RF00556, RF00005, and RF00001).

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 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. 5.2, the number of alignment edges greatly influences the running time for computing an exact multiple structural alignment. We therefore follow the strategy described

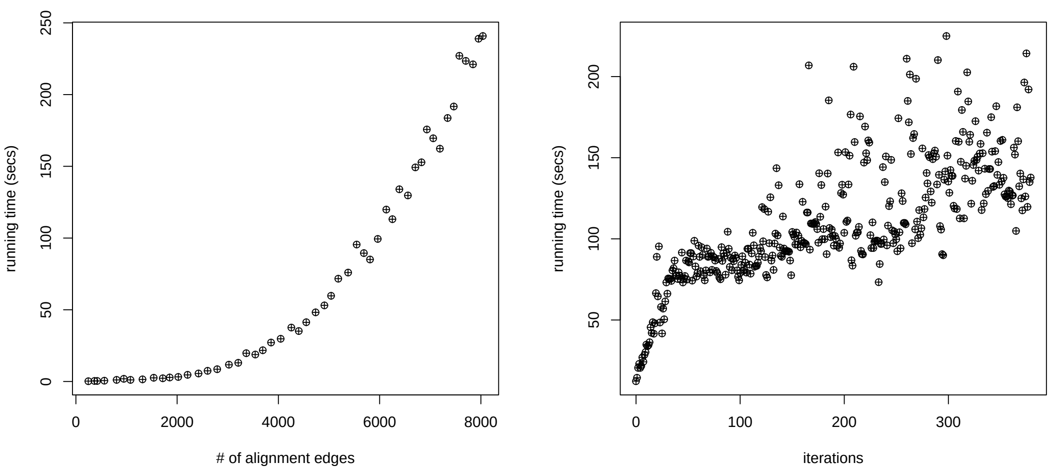


Figure 5.2: Typical behavior for the multiple case. Left: the time to compute an exact multiple sequence alignment increases non-linearly with the number of alignment edges. Right: the time to compute one single iteration for an instance containing 4106 alignment edges increases rapidly with the number of iterations. This is due to the adaption of the Lagrangian multipliers.

in Sect. 5.1.1 to thin out the graph.

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 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 unpredictable increase in the running time per iteration over the course of all iterations. The right side of Fig. 5.2 shows the development for an instance of three L19 leader protein sequences.

As a second experiment, we assess the improvement of the objective function value between heuristically inferred multiple structural alignments [7] and provably optimal or near-optimal solutions of the exact multiple sequence-structure model. Note that at this stage we are especially interested to what extent heuristical multiple sequence-structure alignments approximate the objective function values of the exact sequence-structure framework.

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 [7], we compute all pairwise alignments of a given instance and feed them to the T-Coffee software [109] 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 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 5.1 shows the objective function values of the alignments generated by LaRA and mLARA for these 40 instances. Note that we provide two different evaluations for LaRA alignments: the first column LaRA $sub_5$ gives the objective function value at a suboptimality of 5, *i.e.*, exactly the set of alignment edges that we used for the computation of the sequence-structure alignments. T-Coffee, however, additionally inserts potential alignment edges when it heuristically infers the multiple alignment. To take the augmented set of alignment edges into account we again evaluate the LaRA alignment with a suboptimality value of 20, such that all alignment edges are considered. As one can see in Tab. 5.1 the difference between the two objective function values is significant in many cases.

Generally, mLARA reaches higher objective function values than those computed by LaRA. There are, however, 12 instances where the heuristically inferred alignments yield better objective function values than mLARA. A closer inspection of those instances reveals three main reasons:

1. The computation time limit is too tight. Hence, mLARA 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. 5.3 shows the computation time per iteration of tRNA instances #1 and #15 (1174 and 1178 alignment edges, represented by the circles andred squares, respectively) from Tab. 5.1. Although the number of alignment edges differs only by four, the computation time per iterations varies dramatically. Consequently, mLARA performs 259 and only 59 iterations for instances #1 and #15.

2. The right side of Fig. 5.3 shows the solution process for 5S instance #13 from Tab. 5.1. 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.

3. The T-Coffee software potentially augments the set of alignment edges when it heuristically builds a multiple structural alignment based on all

| | Instance | LaRA sub$_5$ | LaRA sub$_{20}$ | mLARA |
|---|---|---|---|---|
| **tRNA** | #0 | 1050.88 | 1051.88 | **1193.34 (0.94)** |
| | #1 | 1091.6 | 1137.9 | **1194.33 (0.94)** |
| | #2 | 1402.11 | **1453.81** | 1453.06 (0.99) |
| | #3 | 1468.2 | 1468.2 | **1469.63 (0.98)** |
| | #4 | 797.29 | 907.628 | **1014.61 (0.83)** |
| | #5 | 1153.69 | 1172.08 | **1184.89 (0.88)** |
| | #6 | 1174.83 | 1285.38 | **1299.14 (0.97)** |
| | #7 | 1229.24 | 1267.6 | **1304.31 (0.98)** |
| | #8 | 1710.11 | 1711.11 | **1772.04 (1.00)** |
| | #9 | 1184.9 | **1213.68** | 1193.55 (0.92) |
| | #10 | 1084.26 | **1148.6** | 1134.20 0.90) |
| | #11 | 1103.91 | **1125.58** | 1043.95 (0.80) |
| | #12 | 1099.66 | **1119.71** | 1113.45 (0.91) |
| | #13 | 1329.08 | **1329.08** | 1323.94 (0.97) |
| | #14 | 1108.17 | 1177.21 | **1254.51 (0.96)** |
| | #15 | 1089.84 | **1293.95** | 1077.07 (0.88) |
| | #16 | 878.656 | 955.553 | **1019.92 (0.88)** |
| | #17 | 971.056 | 1086.05 | **1133.84 (0.85)** |
| | #18 | 1238.3 | 1238.3 | **1320.11 (0.99)** |
| | #19 | 1254.7 | 1280.46 | **1366.26 (0.99)** |
| **5S** | | | | |
| | #0 | 1845.66 | 1888.1 | **1922.20 (0.96)** |
| | #1 | 1809.14 | 1810.34 | **2097.22 (0.99)** |
| | #2 | 2199.47 | 2221.18 | **2259.01 (1.00)** |
| | #3 | 2015.68 | 2034.04 | **2049.05 (0.98)** |
| | #4 | 1641.18 | 1669 | **1735.34 (0.92)** |
| | #5 | 1718.62 | **1721.6** | 1696.58 (0.88) |
| | #6 | 1589.02 | 1616.35 | **1682.68 (0.93)** |
| | #7 | 1609.94 | 1695.44 | **1740.73 (0.90)** |
| | #8 | 2052.95 | **2194.9** | 1956.62 (0.89) |
| | #9 | 1957.43 | 2028.3 | **2107.10 (1.00)** |
| | #10 | 1949.51 | **2048.08** | 1946.53 (0.93) |
| | #11 | 1547.51 | **1873.47** | 1715.54 (0.92) |
| | #12 | 1932.32 | 1933.32 | **2023.18 (0.99)** |
| | #13 | 2113.55 | **2197.52** | 1996.10 (0.86) |
| | #14 | 2218.78 | 2229.18 | **2267.25 (0.99)** |
| | #15 | 1956.95 | 1987.45 | **2064.64 (0.97)** |
| | #16 | 2084.55 | 2086.55 | **2116.64 (0.99)** |
| | #17 | 1716.94 | 1818.26 | **1884.92 (0.94)** |
| | #18 | 2090.59 | 2091.59 | **2171.81 (0.99)** |
| | #19 | 2134.05 | 2183.33 | **2407.99 (0.99)** |

Table 5.1: The comparison between the objective function values of LaRA and mLARA on 40 randomly generated tRNA and 5S RNA instances. Column LaRA sub$_5$ gives the mLARA objective function values at a suboptimality level of 5, whereas LaRA sub$_{20}$ gives the evaluation at a suboptimality value of 20, *i.e.*, we make sure that all alignment edges that are induced by T-Coffee are considered. The numbers in brackets in column mLARA give the level of optimality of the solution. 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.

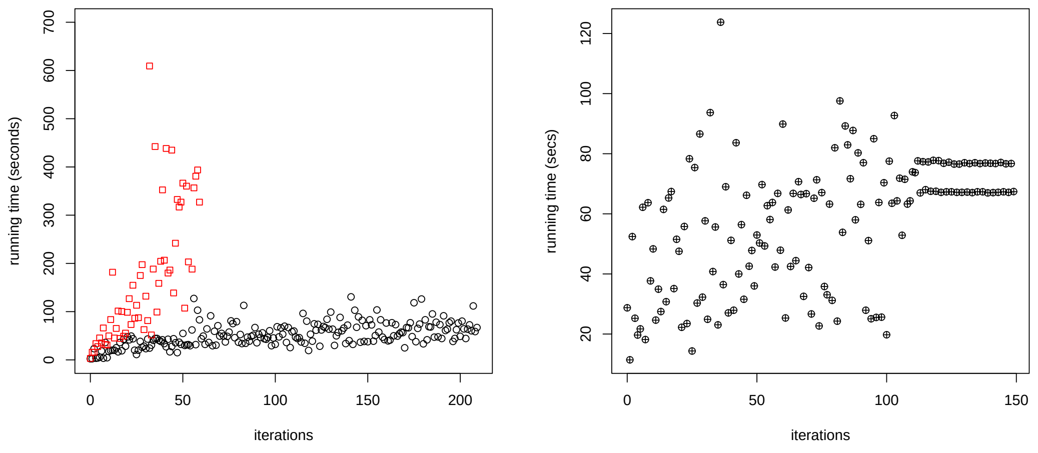


Figure 5.3: Left: the computation time per iterations oscillates dramatically even between instances having almost the same number of alignment edges. The red squares and circles represent the iterations of tRNA instance #15 (1178 edges) and #1 (1174 edges) from Tab. 5.1, respectively. Right: the solution 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. 5.1.

pairwise alignments. This happens, for example, in the case of 5S instance #11: mLARA yields a value of 1715.54 with an upper bound of 1868.31. The LARA alignment, on the other hand, has a value of 1873.47 which is only possible, because the set of alignment edges is augmented while heuristically inferring the multiple alignment.

## 5.4 Results for the Heuristic Multiple Case

The experiments in Sect. 5.3 show that exact multiple sequence-structure alignments are computationally very expensive. Our aim is, however, to evaluate the approach on a large data set. The application of mLARA is too expensive for a large-scale comparison of various sequence-structure alignment programs. Hence, we resort to the implementation of the pairwise model which is called LARA. Building upon the consistency-based approach that we described in Sect. 3.2.2 we infer multiple sequence-structure alignments based on the pairwise alignment information. We also give the results for the progressive alignment version of our approach which is called pLARA (short for *progressive* LARA). Additionally, we report on the performance of the extended model that we describe in Sect. 4.3. The consistency-based and progressive variant of the model are called sLARA (*stacked* LARA) and psLARA (*progressive stacked* LARA).

### 5.4.1 BRAliBase 2.1

The BRAliBase data set[1] [149; 150], was created with the objective to provide reference alignments for the fair comparison of different sequence and sequence-structure alignment programs. We compare our implementations to other current programs on this data set.

BRAliBase 2.1 reference alignments are based on the manually curated seed alignments of the *Rfam 7.0* database [53]. Out of the pool of all ncRNA families that have more than 50 sequences in their seed alignment, test instances of the BRAliBase were constructed the following way: all pairwise projections that are within a certain *average pairwise sequence identity* (APSI) range form the pool of pairwise candidate alignments. Then, single sequences are randomly deleted from the sequence pool and added to a candidate alignment, until the candidate alignment holds the desired number of sequences. If the alignment satisfies the sequence and structure conservation constraints, *i.e.*, the APSI of the generated instances has to be within a predefined range and the structural conservation has to be higher than a given threshold, the instance is accepted. Otherwise, the algorithm restores the sequence pool and starts over again. If we look at the problem through a graph-theoretic lens, we represent each sequence from the seed alignment by a vertex, and we connect two vertices by an edge if the APSI value of these two sequences is within a certain range. Creating input instances of size $k$ corresponds to finding cliques of size $k$ in that graph.

The BRAliBase data set is divided into alignment instances containing either 2, 3, 5, 7, 10 or 15 sequences. In the following, we stick to the BRAliBase naming convention and refer to the sets of instances by $k2$, $k3$, $k5$, $k7$, $k10$, and $k15$, depending on the number of sequences per instance. BRAliBase 2.1 contains 36 different RNA families, ranging from approximately 26 nucleotides long Histone 3'UTR stem-loop motifs to about 300 nucleotides long eukaryotic SRP RNAs. The interested reader is referred to [150] for a detailed listing of all instances.

Unfortunately, the way the input instances are created leads to an over representation of certain RNA families within BRAliBase. The data set contains a higher number of instances from families that have more sequences in their seed alignments. Consequently, a few ncRNA families represent the major instances of all BRAliBase instances: tRNA instances, for example, constitute 56% of all pairwise instances. This percentage rises to 66%, 73%, 75%, 74%, and 80% for $k3$, $k5$, $k7$, $k10$, and $k15$ instances, respectively.

The primary reason to perform our experiments on the BRAliBase data set is to evaluate the performance on an independent benchmark set that we did not compile ourselves. There are various recent papers, for example [81; 132], where the authors claim to describe the best available sequence-structure alignment program. They compile their own data sets, which are all based on data from the Rfam database, and on these data sets their programs perform best. We doubt

[1] Available at `http://www.biophys.uni-duesseldorf.de/bralibase/`.

that this is the right way to go. There should be a standard benchmark database for RNA structural alignments in the spirit of BAliBase or Prefab for amino acids, where various alignment programs can be benchmarked against each other in a sound manner. In our opinion the BRAliBase is a first step in the right direction.

## 5.4.2 Assessment Scores

The quality assessment of a structural alignment is a non-trivial task. If reliable "gold-standard" alignments are available, the comparison on the sequence and structure level is sufficient for a sound comparison. In the following, we describe assessment scores that we use in our comparison.

**Sequence assessment.** In [135] the authors introduced the *sum-of-pairs score*, or SPS in short, to define the similarity between a test and reference alignment on the sequence level. The main idea is to count the number of aligned residues of the test alignment that are identically aligned as the reference. More formally, given an alignment $\mathcal{A} = \hat{s}_0, \ldots, \hat{s}_{n-1}$ of $n$ sequences with $|\mathcal{A}| = m$, then we have an indicator variable $p_{krs}$ with $p_{krs} = 1$ if the residue $\hat{s}_r^k$ and $\hat{s}_s^k$ are aligned as in the reference alignment, and 0 otherwise. Then, we define $s_{\mathrm{SPS}}$ as

$$s_{\mathrm{SPS}} = \frac{\sum_{k=0}^{m-1} \sum_{r=0}^{n-2} \sum_{s=r+1}^{n-1} p_{krs}}{\sum_{i=0}^{m-1} \binom{n}{2}}$$

A value of 0 indicates that not a single column is correctly aligned with respect to the reference alignment, whereas a value of 1 indicates perfect agreement with the reference alignment.

The program compalign developed by Sean Eddy, that is part of the SQUID library [42], represents an advancement by considering not only the aligned residues, but also what residues are aligned to a gap character. compalign builds the foundation for the program compalignp that is being used in the BRAliBase benchmark set. We use compalign in the following to benchmark the sequence accuracy.

**Structure assessment.** For some ncRNA families manually curated multiple alignments exist that are annotated with published structures. Prominent examples are tRNA, ribosomal 5S RNA, or the TPP riboswitches. If reference structures are available, then one compares predicted paired nucleotides to the annotation of the reference alignment. The *Matthew's correlation coefficient* (MCC) assesses the structural similarity. We define the MCC score $s_{\mathrm{MCC}}$ as

$$s_{\mathrm{MCC}} = \frac{\mathrm{TP} \cdot \mathrm{FN} - \mathrm{FN} \cdot \mathrm{FP}}{\sqrt{(\mathrm{TN} + \mathrm{FN})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TP} + \mathrm{FP})}}$$

with TP, FN, FP, and TN being the number of true positives, false negatives, false positives, and true negatives, respectively.

The value of $s_{\mathrm{MCC}}$ is bounded by $-1$ and 1, with a $s_{\mathrm{MCC}} = 1$ being the best result: every pairing that is predicted is a correct interaction, and no false pairing is predicted. On the other hand, a value of $-1$ indicates that not a single interaction was correctly predicted, and the number of wrongly predicted interactions is maximal.

In reality, however, the number of reliably annotated gold-standard alignments is limited. This holds also true for the RFAM database, where for parts of the ncRNA families the structural annotation was created using consensus folding algorithms like RNAALIFOLD [68] or PFOLD [84]. Therefore, the creators of the BRALIBASE benchmark set chose the score *structural conservation index* [143], or SCI in short, to assess the structural quality of an alignment.

The SCI gives the degree of conservation of a consensus structure induced by a multiple alignment of $n$ sequences in relation to the *minimum free energy* structures of the $n$ single sequences. Let $E_c$ be the energy value of the consensus structure induced by the alignment, and let $E_0, \ldots, E_{n-1}$ be the minimum free energy values of the $n$ aligned sequences with $\bar{E}$ being the arithmetic mean of the $n$ values. Then, we define the SCI as

$$\mathrm{SCI} = \frac{E_c}{\bar{E}} \quad .$$

A SCI value of $\approx 1$ indicates high structural conservation, whereas a value around 0 indicates no structural conservation at all. Note that the SCI score can be greater than 1, since covariance information is additionally rewarded during the computation of the consensus structure. Furthermore, the computation of the consensus structure is done via the RNAALIFOLD program which is susceptible to changes in the alignment. This especially means that a higher Compalign value does not necessarily imply a higher SCI score, *e.g.*, running LARA with default parameter settings for two SECIS instances from BRALIBASE (SECIS.apsi-45.sci-68.no-1.raw.fa and SECIS.apsi-45.sci-79.no-1.raw.fa) we get Compalign scores of 0.45 and 0.44. The corresponding SCI scores are 0.26 and 0.58, respectively. Setting the LARA parameters to optimized values, the Compalign scores increase to 0.60 and 0.48. The corresponding SCI values, however, drop to 0.14 and 0.00.

In the following experiments we use the program SCIF from the BRALIBASE website to assess the SCI of the computed alignments.

### 5.4.3 Results

**Parameter Training.** There are three important LARA parameters: gap open and extension penalties $\gamma_o$ and $\gamma_e$, and the parameter $\tau$ that represents the scaling factor for the sequence scores. In [5; 7] we used rather ad-hoc values that tried to mimic the parameter settings within the PMCOMP software package. This leads to initial values of $-6$, $-2$, and 0.05 for $\gamma_o$, $\gamma_e$, and $\tau$.

For the final evaluation in this thesis, however, we took the data set that the authors of [96] assembled and experimentally evaluated different parameter sets. The parameter setting that yielded the highest MCC scores were chosen as the final ones. Accordingly, we set the gap open and extension penalty, and $\tau$ to $-12$, $-5$, and 1.0 for the following evaluations.

sLaRA and psLaRA, the implementations that incorporate stacking energies, have two additional parameters: the structure and stacking scaling factors $\sigma_s$ and $\sigma_{ss}$. During the initial test phase it turned out that—similar to the sequence scaling factor $\tau$—we have to scale the structure and stacking contribution to balance these two scores. The final parameter set consists of $-10$,$-5$,1.0,0.6, and 0.9 for $\gamma_o$,$\gamma_e$,$\tau$,$\sigma_s$, and $\sigma_{ss}$, making the stacking contributions more important than the structure scores.

It has to be remarked, however, that several different parameter sets yield almost the same performance on the data set from [96], and the values of the parameters differ significantly. The values for the gap penalties, for instance, vary between $-10$ and $-20$. Therefore, it is likely that the performance of sLaRA and psLaRA can be further improved by examining the various parameter sets on different data sets in an automated manner.

Finally, we set the number of overall iterations to 500 for all implementations. If the upper bound does not improve within 50 iterations, we halve the value of parameter $\mu$. Table 5.2 gives an overview of the parameters that we use for the computation of the alignments throughout the rest of the thesis.

| **Paramter** | **LaRa** | **sLaRA** |
|---|---:|---:|
| suboptimality for alignment edge generation | 40 | 40 |
| gap open penalty $\gamma_o$ | $-12.0$ | $-10.0$ |
| gap extend penalty $\gamma_e$ | $-5.0$ | $-4.0$ |
| sequence contribution $\tau$ | 1.0 | 1.0 |
| sequence scoring matrix | RIBOSUM65 | RIBOSUM65 |
| structure contribution $\sigma_s$ | 1.0 | 0.6 |
| stacking contribution $\sigma_{ss}$ | — | 0.9 |
| structure scoring system | bpp | bpp |
| minimal probability considered | 0.003 | 0.003 |
| Lagrange iterations | 500 | 500 |
| subgradient parameter $\mu$ | 1.0 | 1.0 |
| halve $\mu$ after $n$ non-decreasing iterations | 50 | 50 |
| T-Coffee version | 4.70 | 4.70 |

Table 5.2: A summary of the parameters that we use for our program runs. We applied the same parameters from LaRA for pLaRA, and the parameter set of sLaRA for psLaRA. The structure scoring system *bpp* refers to structure scoring based on base pair probabilities.

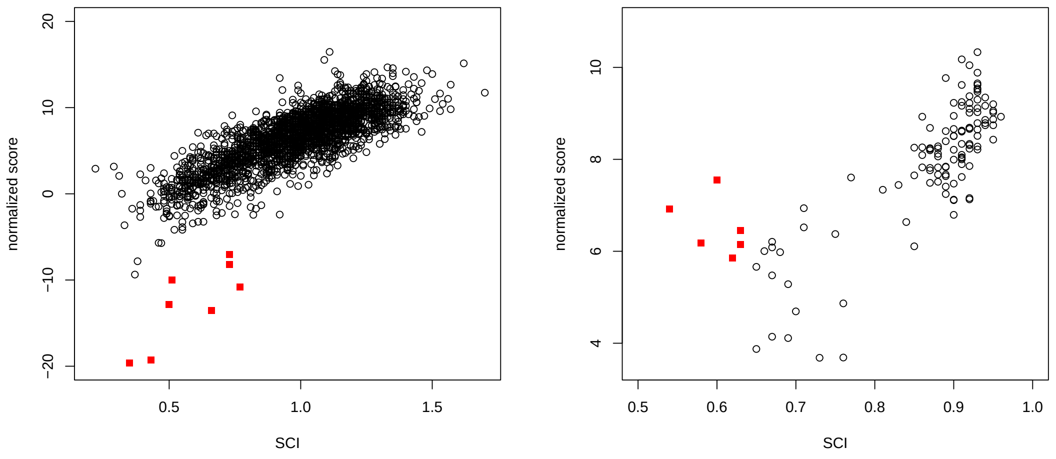


Figure 5.4: All 2251 BRAliBase $k2$ (left side) or 123 BRAliBase $k15$ (right side) instances of low pairwise sequence identity where each black circle or red square corresponds to one instance. The x-axis gives the SCI score, whereas the y-axis codes the structure-normalized score. The red squares mark the outlier instances.

**Score vs. Structural Conservation.** We were interested in to what extent the accuracy of our alignments correlates with the objective function value of our model, *i.e.*, the sum of sequence contributions plus the structure scores based on the base pair probabilities. Since the score depends on the length of the input sequences, we normalized the score with respect to the number of paired bases in the *minimum free energy* structure. Note that we did not use the structure, but only the number of base pairs in the structure to get a rough estimate of how many pairings we expect in the structure. Then, let $\hat{p}$ and $n$ be the LaRA score and the number of base pairs in the MFE structure, then the *normalized score* is given by $\frac{\hat{p}}{n}$.

The scores of conserved structural interactions build the lion's share of the final LaRA score, *i.e.*, the higher the normalized score is, the better is the structural conservation. Consequently, there should be a correlation between the normalized score and the SCI score, because the more structural similarities the alignment captures, the better should the consensus folding perform during the computation of the SCI score. Figure 5.4 shows the results for all $k2$ (left side) and $k15$ (right side) instances.

Most of the $k2$ instances behave as expected: the higher the normalized score is, the better is the SCI value. There is, however, a group of eight outliers that have a high SCI score, but a very low normalized score. A closer inspection revealed that the input sequences differ tremendously in length, for example one Intron_gpII instance contains sequences of length 78 and 142. Although the SCI score is relatively good, the normalized score is decreased by the high number of (necessary) gaps.

In the case of the $k15$ instances, we are facing a different situation. Again, most of the input instances behave as expected, but now we have a group of instances that show a relatively low SCI score together with a high normalized score. It turns out that all these instances are either SRP RNAs or SECIS elements. In [7] we already showed that scoring both the sequence and structure using RIBOSUM matrices yields better results for SRP RNAs and SECIS elements.

#### 5.4.3.1 Comparison to Other Programs

Table 5.3 lists the programs that we compare in this section, together with their respective program calls and the program version we used in our experiments.

| Program | Model | Complexity | Cite |
|---|---:|---:|---:|
| LaRA | graph-based | $\mathcal{O}(n^2)$ | [7; 8] |
| sLaRA | graph-based | $\mathcal{O}(n^2)$ | Sect. 4.3 |
| pLaRA | graph-based | $\mathcal{O}(n^2)$ | Sect. 3.2.1 |
| psLaRA | graph-based | $\mathcal{O}(n^2)$ | Sect. 3.2.1 |
| FoldalignM | Sankoff | $\mathcal{O}(n^4)$ | [137] |
| MURLET | Sankoff | $\mathcal{O}(n^4)$ | [81] |
| MARNA | edit-distance | $\mathcal{O}(n^4)$ | [127] |
| MXSCARNA | annotated sequence | $\mathcal{O}(n^2)$ | [133] |
| Stral | annotated sequence | $\mathcal{O}(n^2)$ | [30] |
| Mafft | sequence-based | $\mathcal{O}(n^2)$ | [78] |

| Program | Program Call | Version |
|---|---|---|
| LaRA | `lara -i <input_file>` | 1.3.2 |
| sLaRA | `slara -i <input_file>` | 1.0 |
| pLaRA | `plara -i <input_file>` | 1.0 |
| psLaRA | `pslara -i <input_file>` | 1.0 |
| FoldalignM | `java FoldalignM_McCaskill <input_file>` | 1.0.1 |
| MURLET | `murlet <input_file>` | 1.0 |
| MARNA | `marna.pl -g 2 -n 3 <input_file>` | 1.0 |
| MXSCARNA | `scarna -clustalw <input_file>` | 1.3 |
| Stral | `stral -i <input_file>` | 0.5.4 |
| Mafft | `mafft <input_file>` | 5.861 |

Table 5.3: The upper table lists the programs that we used in our computational experiments. We give the actual program calls in the lower table.

In the following, we give a short description of each program, Chap. 3 provides a more detailed overview.

There are two recent implementations of the original PMComp software: LocARNA [148] and FoldalignM [137]. Both take base pair probability matrices as their input, and using the recursions from [124; 3] they compute the nested substructure that maximizes the sum of the probabilities plus a sequence score. Since LocARNA filters the base pair probability matrices to increase its efficiency, we only considered FoldalignM, because it considers all probabilities and relies on the same recursions like LocARNA.

FoldalignM performs an alignment and clustering of the input sequences at the same time. In some instances, FoldalignM splits the input sequences into two clusters. Since the scores that we use depend on the number of input sequences, we dropped those FoldalignM alignments that did not contain all sequences in the final alignment. This leads to 29, 30, 11, 15, 9, and 6 instances that we did not consider in the case of $k2$, $k3$, $k5$, $k7$, $k10$, and $k15$ instances.

MURLET is another tool that builds upon the Sankoff recursions. It additionally applies heuristics to reduce the DP search, namely the *strip* and the *skip* approximations. The strip approximation limits possible alignment positions to a band of length $\delta$ around an initial alignment. The initial alignment is computed using pairwise HMMs which is similar in spirit to previous pairwise approaches [38]. The skip approximation limits the number of possible branching points within the Sankoff recursions.

MARNA is an implementation of the general edit model for RNA structures proposed by Jiang [74]. There are operations either on the sequence level (base match, base mismatch, and base deletion) or on the structure level (arc match, arc mismatch, arc breaking, arc altering, and arc removing), each associated with a certain weight. MARNA aims for the alignment that transforms one structure into the other, minimizing the overall costs for the edit operations. The interested reader is referred to [74] or to Chap. 3 for details.

MXSCARNA uses the base pairing matrices to compute *stem fragments*, *i.e.*, ungapped parts of helices, of sequences $A$ and $B$. It then discards the entire sequence information and aligns the stem fragments in a consistent manner. The alignment is consistent if we align two stacking stem fragments from sequence $A$ to stacking fragments in sequence $B$. Subsequently, the aligned stem fragments serve as anchors in a traditional sequence alignment.

Stral builds upon an idea by Bonhoeffer *et al.* [16] to incorporate the highest up- and downstream probabilities for each pair of aligned residues and incorporate these scores into the computation of a traditional sequence alignment.

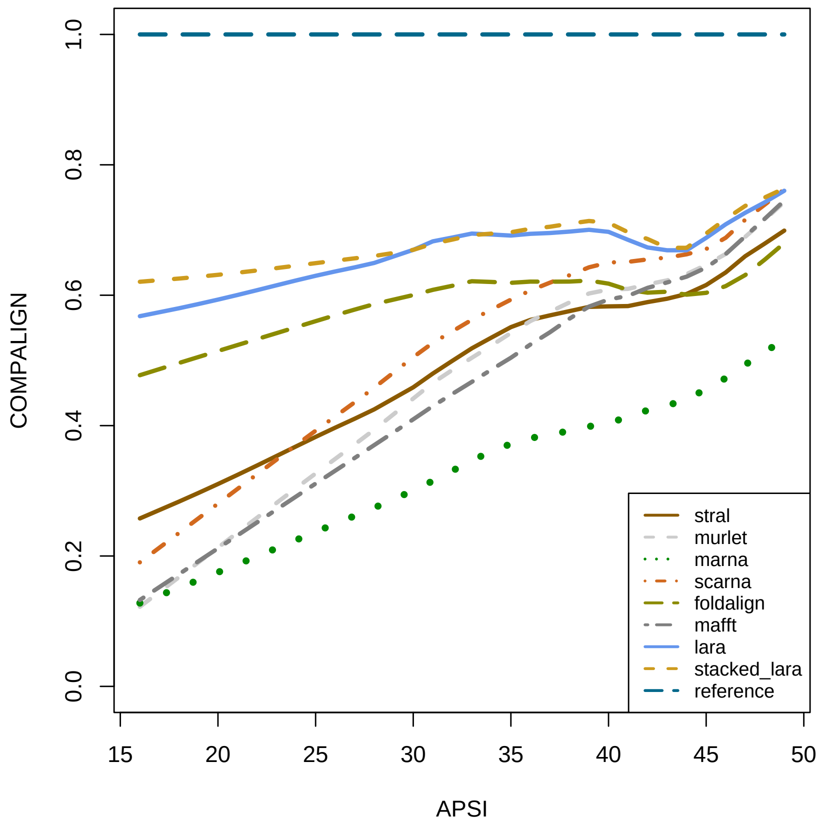
Finally, we want to compare the performance of the sequence-structure alignment programs to a pure sequence-based program. Therefore, we chose Mafft, because it performs very well on the established *BAliBase* [136] and *PREFAB* [11] benchmark sets. In Figs. 5.5 to 5.10 we show the results of our experiments broken down to the different input classes (either $k2$, $k3$, $k5$, $k7$, $k10$, or $k15$) using the Compalign and SCI scores as the quality measure. These graphics have the average pairwise sequence identity as their $x$-axis. The upper part of each figure shows the Compalign performance, whereas the lower part gives the results with respect to the SCI score. We use Lowess regression that we described in Sect. 2.4 for the computation of the lines.

#### 5.4.3.2 Comments on the Results

In the pairwise case sLaRA is ranked first both in terms of the Compalign and the SCI score. The difference between sLaRA and LaRA and FoldalignM, that are ranked second with respect to the Compalign and SCI score, are, however, not significant. Taking a look at Fig. 5.5 we recognize that the curves are almost the same. The SCI performance of sLaRA is better than the performance of LaRA: obviously, the incorporation of stacking probabilities enhances the structural quality of the alignment. On the data sets with an increasing number of input sequences, the better pairwise alignment quality does not pay off. The Compalign performance remains almost the same compared to LaRA. In the case of the $k10$ and $k15$ instances LaRA performs slightly better than sLaRA, but again the difference is not significant in this case.

One has to observe the composition of the $k15$ input data set: 99 of all 123 instances are tRNA instances. Furthermore, on the left side of Fig. 5.11 we show the density plot for all pairwise sequence identities of these 99 tRNA instances. The surprising thing is that about a quarter of all pairwise projections forming the $k15$ instances have a pairwise sequence identity higher than 0.50, the average pairwise sequence identities of the instances, however, are smaller than 0.50. Remember that in the pairwise case sLaRA is superior to LaRA with respect to both the Compalign and the SCI score. We are therefore interested whether this holds true for $k2$ tRNA instances with a sequence identity above 0.50. The right side of Fig. 5.11 shows the Lowess plot for these 780 tRNA instances. In these cases, LaRA performs better than sLaRA and the SCI performance is almost identical (plot not shown). Since a quarter of the pairwise $k15$ alignments are input instances with an identity higher than 0.50, this contributes to the slightly worse performance of sLaRA compared as to LaRA.

In the pairwise case, *i.e.*, the $k2$ instances, up to a sequence identity of $\approx 42\%$ LaRA and sLaRA show a similar Compalign performance, with their respective curves shifted by about 0.1 to the top compared to the Sankoff variant FoldalignM. For the range of $\approx 42 - 50\%$ all programs (even the sequence-based Mafft) have comparable performance (except for MARNA). With an increasing number of input sequences per instance, especially for the $k10$ and $k15$ sequences, the results change tremendously. LaRA outperforms the other programs, yielding average Compalign scores of $\approx 0.9$, whereas the other structure-based alignment programs have average scores around $\approx 0.55 - 0.75$. This is quite remarkable, especially considering that FoldalignM, LaRA, and sLaRA show a similar performance in the pairwise case. FoldalignM, however, computes multiple alignments in a progressive fashion, whereas LaRA and sLaRA compute *all* pairwise alignments and leave it to T-Coffee to compute an alignment that is highly consistent with all pairwise alignments. With an increasing number



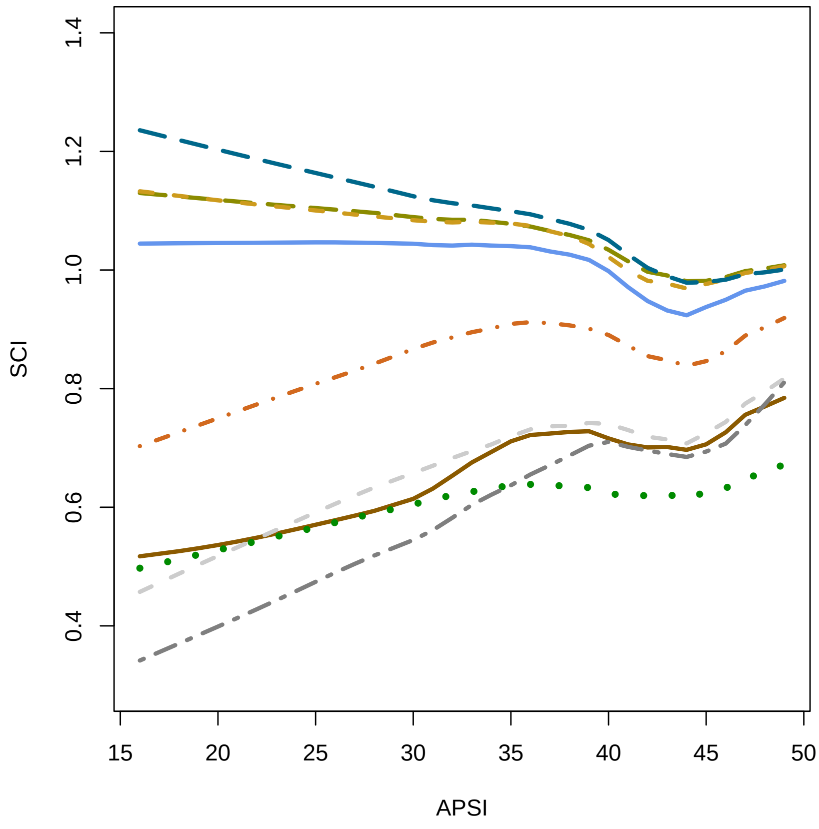
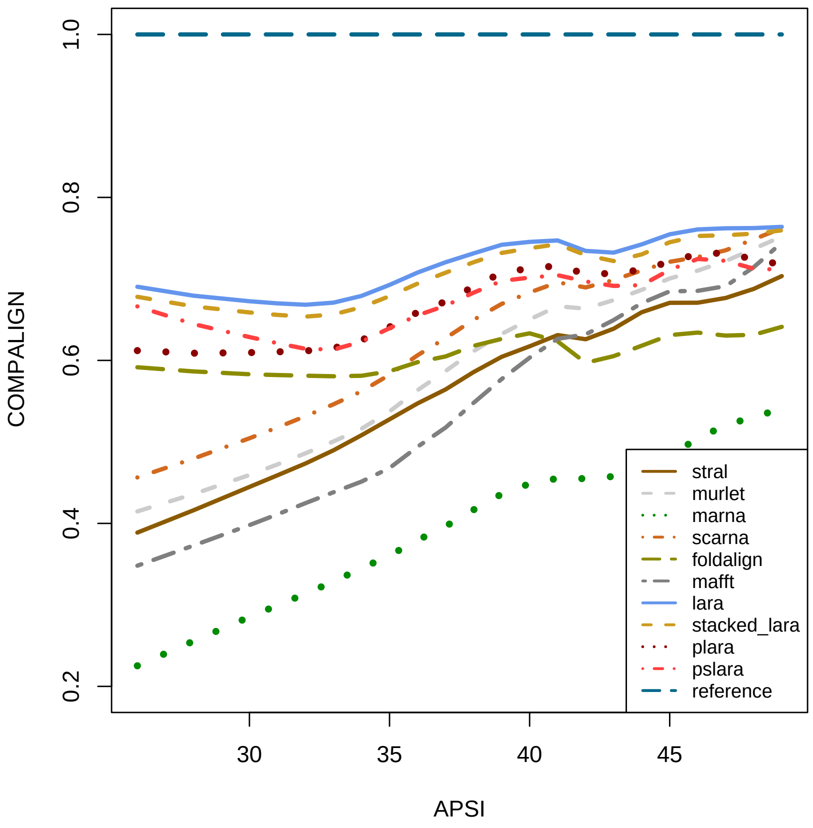


Figure 5.5: Results on all low homology BRALIBASE instances containing 2 input sequences. The $x$- and $y$-axis give the average pairwise sequence identity (APSI) and the Compalign score (upper plot) or the SCI score (lower plot). The legend of the upper plot also applies to the lower one.



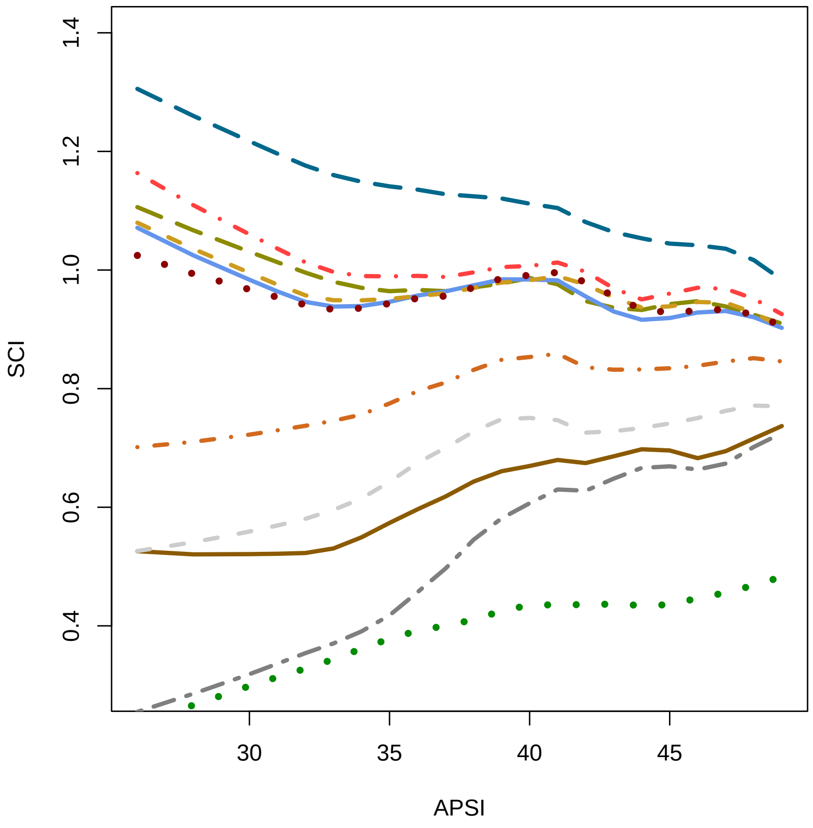
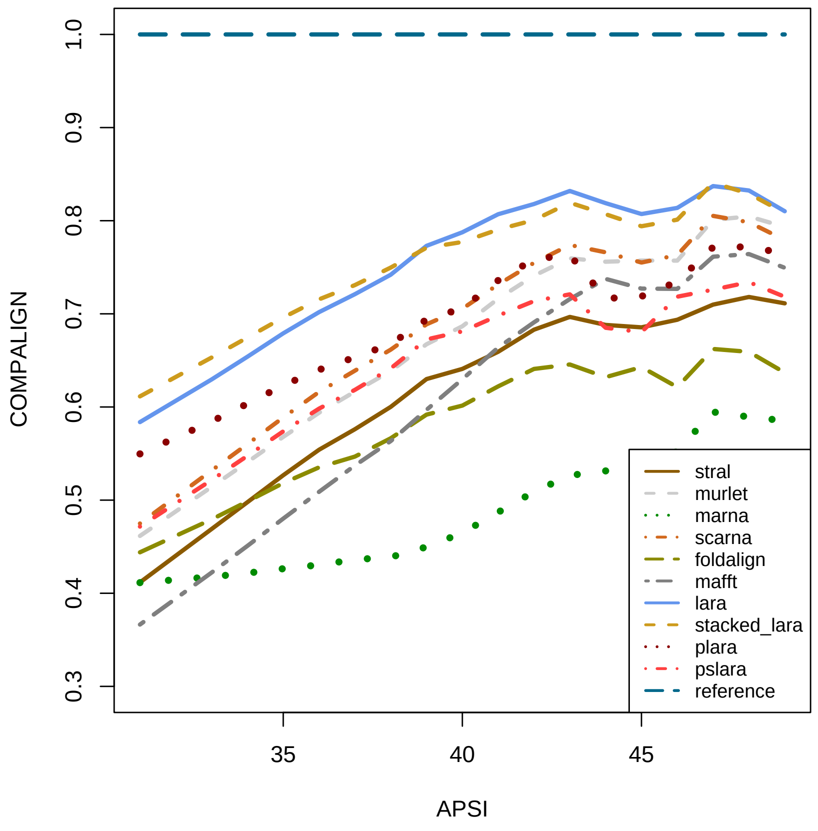


Figure 5.6: Results on all low homology BRAliBase instances containing 3 input sequences. The $x$- and $y$-axis give the average pairwise sequence identity (APSI) and the Compalign score (upper plot) or the SCI score (lower plot). The legend of the upper plot also applies to the lower one.



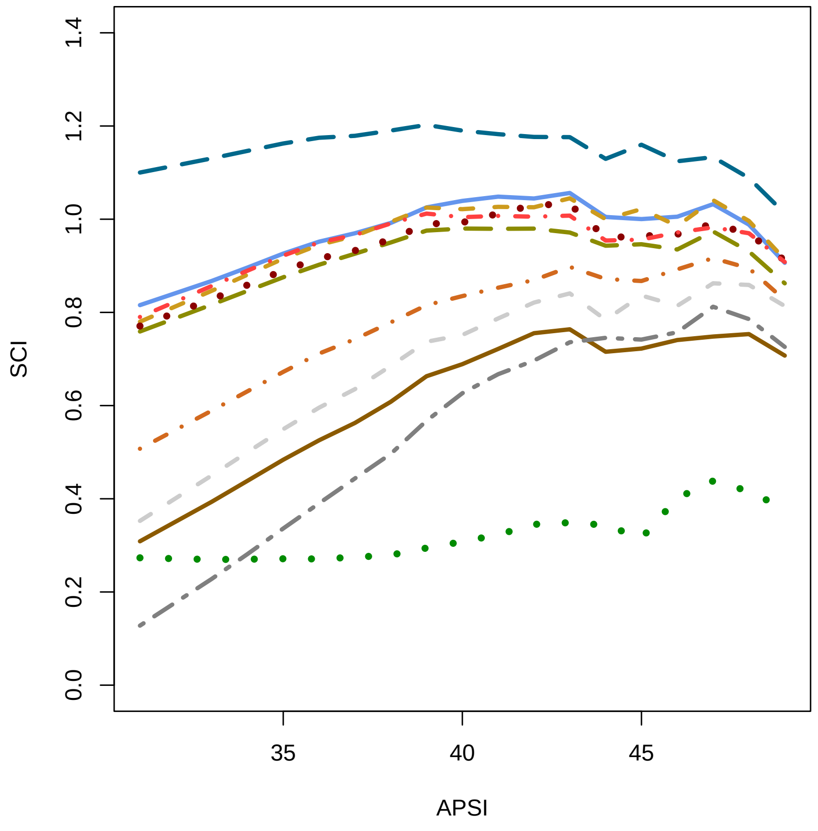
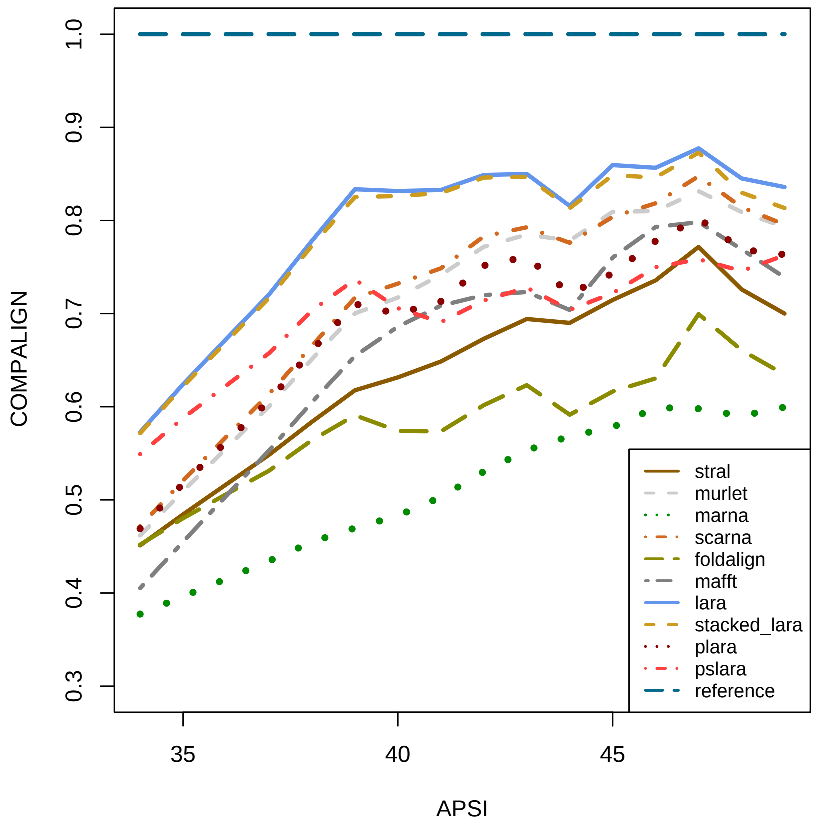


Figure 5.7: Results on all low homology BRAliBase instances containing 5 input sequences. The $x$- and $y$-axis give the average pairwise sequence identity (APSI) and the Compalign score (upper plot) or the SCI score (lower plot). The legend of the upper plot also applies to the lower one.



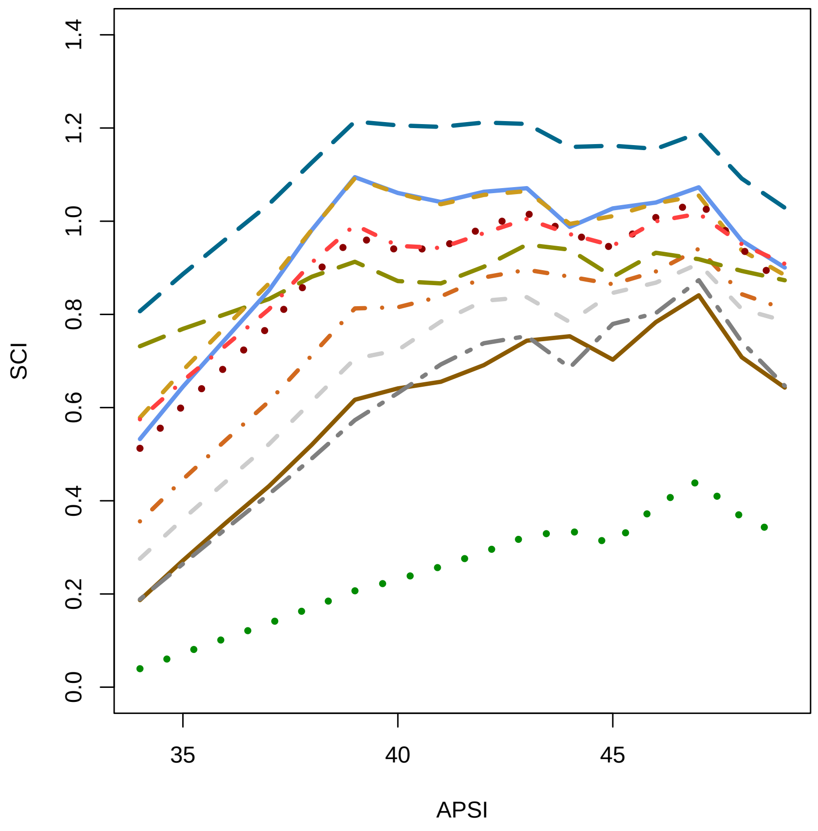
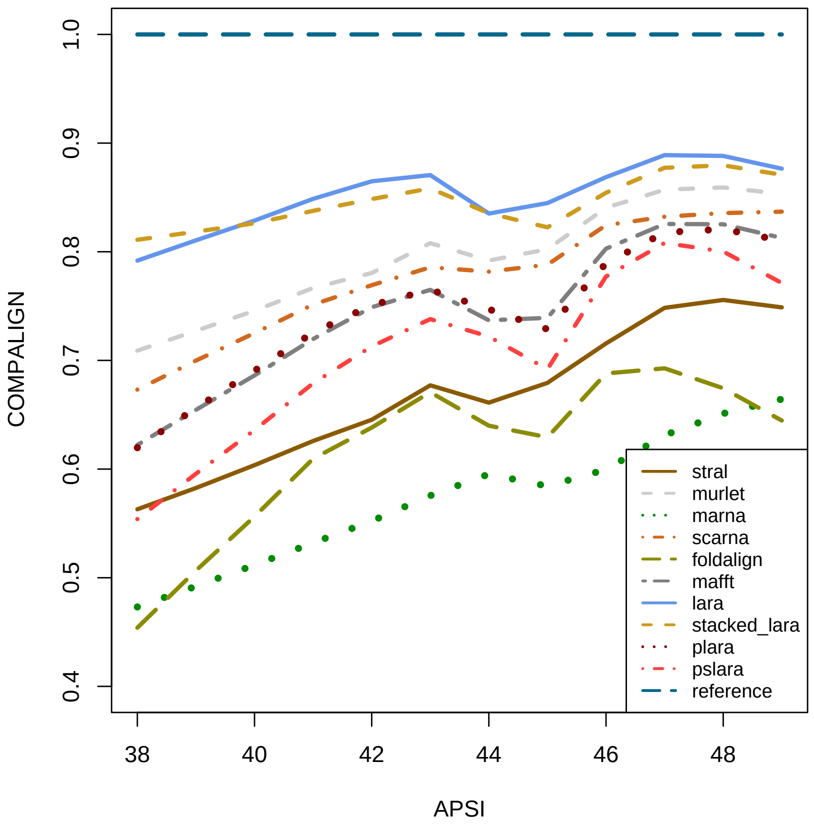


Figure 5.8: Results on all low homology BRAliBase instances containing 7 input sequences. The $x$- and $y$-axis give the average pairwise sequence identity (APSI) and the Compalign score (upper plot) or the SCI score (lower plot). The legend of the upper plot also applies to the lower one.



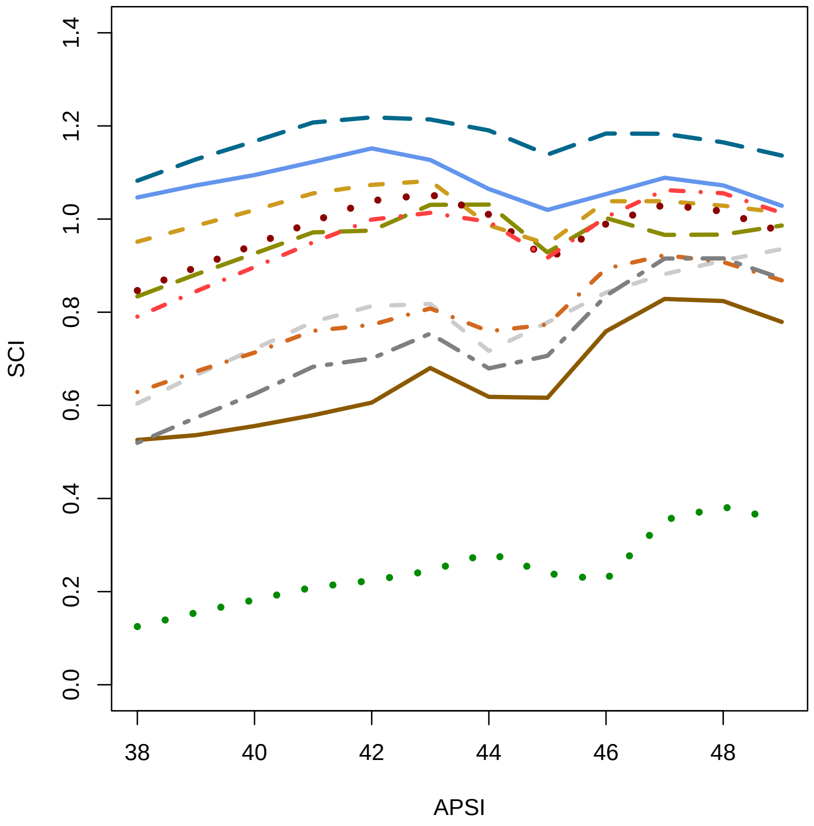


Figure 5.9: Results on all low homology BRALIBASE instances containing 10 input sequences. The $x$- and $y$-axis give the average pairwise sequence identity (APSI) and the Compalign score (upper plot) or the SCI score (lower plot). The legend of the upper plot also applies to the lower one.

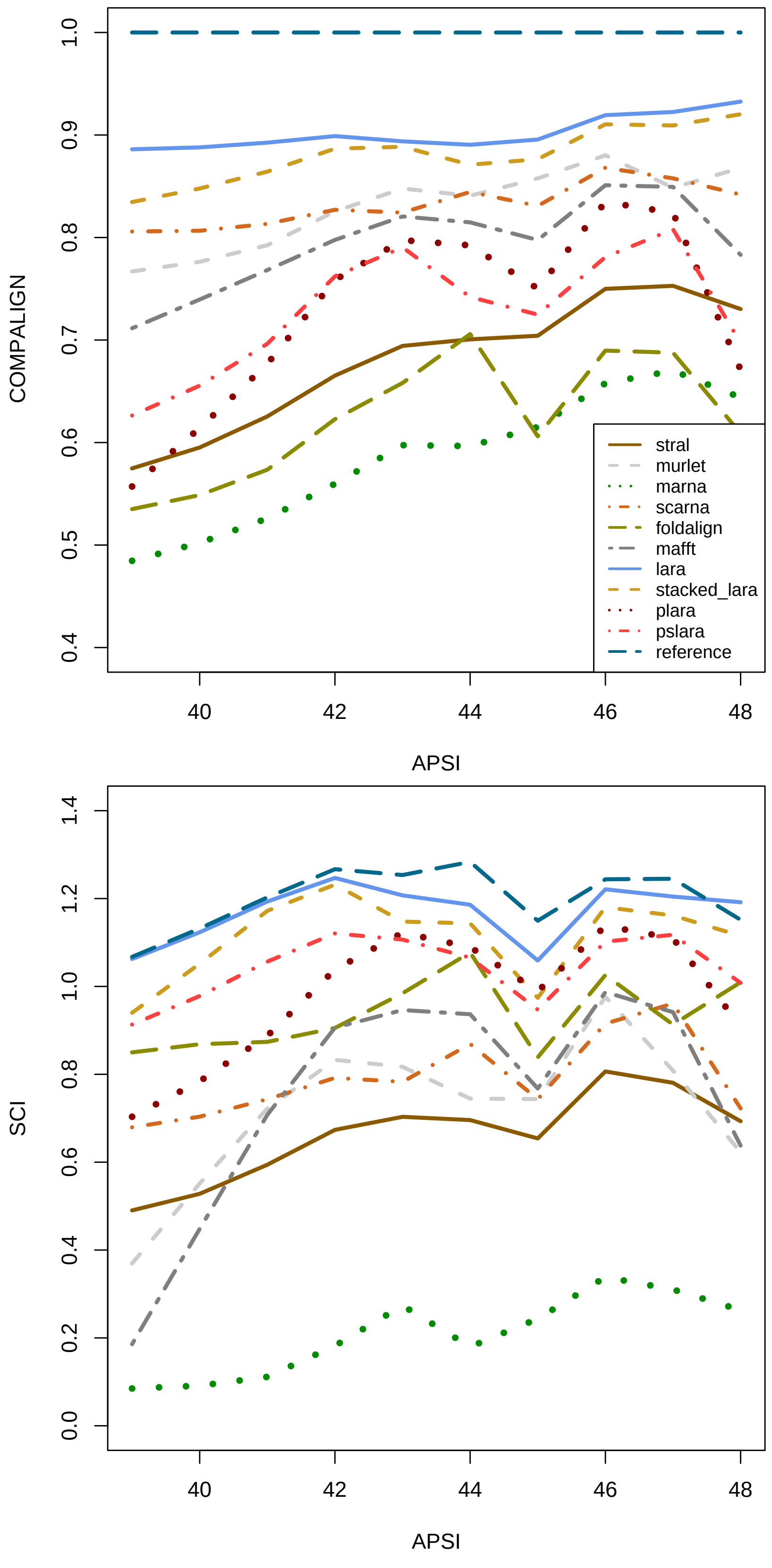


Figure 5.10: Results on all low homology BRALIBASE instances containing 15 input sequences. The $x$- and $y$-axis give the average pairwise sequence identity (APSI) and the Compalign score (upper plot) or the SCI score (lower plot). The legend of the upper plot also applies to the lower one.

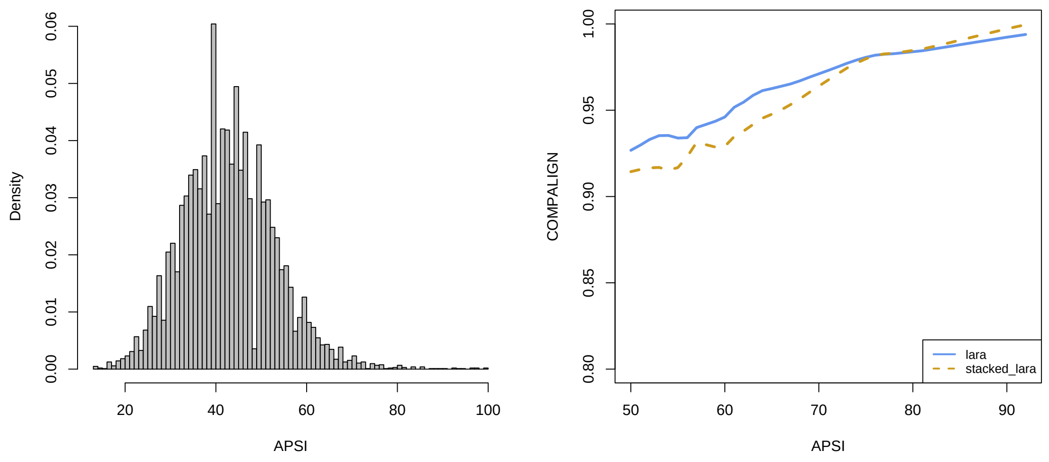


Figure 5.11: Left side: the distribution of pairwise sequence identity values of all $k15$ tRNA instances having an APSI value smaller than 0.50. Right side: comparison between LaRA and sLaRA on all 780 $k2$ tRNA instances that show a pairwise sequence identity greater than 0.50.

of input sequences, the consistency-based approach generates better alignments than the progressive methods (at least in our experimental setup). This holds also true for the progressive variants of our models, pLaRA and psLaRA. Their performance—relative to LaRA and sLaRA—becomes worse with an increasing number of input sequences.

Another astonishing observation is the performance of Mafft, a sequence-based program. The $k2$ and $k3$ instances show a comparable performance to all the other sequence-structure alignment programs for instances above $\approx 42\%$, which is already surprising. With a growing number of input instances, the performance of Mafft becomes even better. In the case of 15 input instances, the program yields—on average—the second best results (behind the various LaRA implementations), outperforming even FoldalignM and Stral which incorporate structural information. The corresponding SCI plots, however, show that the structural features of these instances are not conserved at all, leading to low SCI scores. In the case of FoldalignM, the situation is exactly vice versa: the Compalign scores are low, whereas the SCI scores are relatively high which means that the helical regions—in contrast to the loop regions—are correctly aligned.

The comparison between FoldalignM and pLaRA shows that pLaRA performs consistently better than FoldalignM on the various input data sets. The two programs optimize the same objective function by maximizing the sequence and structure score and compute multiple alignments in a progressive fashion. There are, however, two important differences: pLaRA and FoldalignM use different parameter sets, and as the authors of [9] show, the construction of the guide tree is of great importance for the overall result. In the current implementation of pLaRA and psLaRA we build the guide tree by computing pure sequence-based alignments using the RIBOSUM sequence scores. We performed various tests with alternative approaches, *e.g.*, computing sequence-structure alignments with a low number of iterations, but surprisingly it turns out that the quality of the alignments does not increase if one spends more time on the construction of the guide tree.

Taking a look at the result plots one immediately recognizes the bad performance of the MARNA software. For the final evaluations in this thesis, we double-checked that we did not change any parameters to run the software on the BRAliBase data set with the original settings. In [30] the authors perform a comparison of sequence-structure alignments on the original BRAliBase compilation from [48]. They show that the performance of MARNA is comparable to the one of clustalW even if the APSI is smaller than 0.50. There are two possible reasons: first, MARNA builds upon the general edit-operations for RNA structures and uses fixed structural information (either a fixed structure or the *shape* of the sequence) which ultimately means that the alignment quality greatly depends on single MFE structures. If these structures are wrong, then the algorithm is mislead. Second, the command to execute T-Coffee with the MARNA compiled library reads

```
system("t_coffee -in=Lcoffee.lib,Mclustalw_pair");
```

which means that, in addition to the library file `coffee.lib` that MARNA creates, T-Coffee uses a second library based on clustalW alignments. This is unfortunate, however, since the clustalW information seems to blur the structural information of MARNA. Moreover, a closer look into the MARNA library file shows that almost all weights in the library are in the range between 95 and 100. This does not allow a discrimination between stacked and unstacked regions, introducing additional difficulties for T-Coffee to assemble a proper alignment. Hence, parameter training on a recent Rfam data set, like the data set compiled for the MASTR paper, dropping the clustalW information, and setting proper library weights might enhance the overall quality of the MARNA alignments.

During our computational experiments we evaluated two different modes for the generation of libraries. We either write the entire alignment into the library file, *i.e.*, both the loop and the stacked regions, or the stacked regions alone. The reasoning for supplying only stacked regions is that these should serve as anchor regions, and T-Coffee should perform its consistency-based ansatz on the nucleotide level. If the sequence conservation in the loop regions is high enough, this works well as we could, for example, observe with sequences from the `ITS2` database [153]. Libraries specifying only the stacked regions produced better alignments in the case of `ITS2` sequences. In general, however, due to the low sequence conservation of the input sequences T-Coffee introduces too many gaps into the loop regions which lowers the overall alignment quality.

Second, we observe significant differences in the performance of various T-Coffee releases. For our computational experiments in Chap. 5 we use the version 4.70 that we originally used for our computational study in [7]. Since the release of 4.70 in November 2006, there have been several new program versions, but these releases show inferior performance compared to release 4.70.

### 5.4.3.3 Friedman Tests

In Chap. 2.4 we described the Friedman testing procedure which compares multiple samples without assuming anything about the distribution of the input data. In our case we have the results of various programs and want to compare their performance on the BRAliBase input sets. The null hypothesis of the Friedman test is that there is no significant difference between the various programs. In the case the null hypothesis is rejected, *i.e.*, there are significant differences between various groups, one has to perform pairwise Wilcoxon signed-rank tests to detect significant differences between the programs. To limit the hassle of multiple testing, we perform the Wilcoxon test only between the program that is ranked first and the remaining programs. We perform all the tests with a significance level of 0.05, and we correct for the multiple Wilcoxon tests using the Bonferroni correction, *i.e.*, we set the p-value to $\frac{0.05}{k}$ with $k$ being the number of the tested programs.

Table 5.4 lists the results of the testing procedure for both the Compalign and the SCI scores on all six data sets of the BRAliBase.

### 5.4.3.4 Comparison of the Running Times

We compared the programs tested on the same computing server with an Intel Xeon CPU running at 3.2 GHz, 3.5 GB RAM, and Linux kernel version 2.6.16. It turned out that memory requirement was not an issue, but the computation time instead. Especially MARNA scales in $\mathcal{O}(n^4)$, which makes the alignment of longer sequences (for example the SRP instances of BRAliBase) rather time-consuming. This, however, is not the case with LaRA and Foldalign, since these two programs have running times in $\mathcal{O}(n^2)$. To evaluate the time consumption within reasonable time, we therefore set a time limit of 20 minutes per instance. If the computation was not finished within 20 minutes, the process was killed and we took 20 minutes as the running time. In Table 5.5 we list the number of instances that the programs were not able to align within 20 minutes.

We were especially interested in how the running times of the programs that use structure information scaled with respect to the number of the input sequences. Foldalign, as a progressive approach, computes $(n-1)$ pairwise alignments given $n$ input sequences. MARNA and LaRA, however, compute all $\frac{n(n-1)}{2}$ pairwise alignments. Table 5.6 shows the execution time of all programs on all $k2$, $k3$, $k5$, $k7$, $k10$, and $k15$ instances. As one can see, with an increasing number of input sequences, a progressive alignment strategy pays off compared to the computation of all pairwise alignments.

| | k2 | | k3 | | k5 | |
|---|---|---|---|---|---|---|
| **Program** | SPS | SCI | SPS | SCI | SPS | SCI |
| LaRA | 2 | 3 | $\mathbf{1}^{(\mathbf{3-10})}$ | 4 | $\mathbf{1}^{(\mathbf{3-10})}$ | 2 |
| sLaRA | $\mathbf{1}^{(\mathbf{3-8})}$ | $\mathbf{1}^{(\mathbf{3-8})}$ | 2 | 2 | 2 | $\mathbf{1}^{(\mathbf{4-10})}$ |
| pLaRA | – | – | 3 | 3 | 5 | 3 |
| psLaRA | – | – | 6 | $\mathbf{1}^{(\mathbf{6-10})}$ | 7 | 4 |
| FoldalignM | 4 | 2 | 9 | 5 | 9 | 5 |
| MURLET | 5 | 6 | 5 | 7 | 4 | 7 |
| MARNA | 8 | 8 | 10 | 10 | 10 | 10 |
| MXSCARNA | 3 | 4 | 4 | 6 | 3 | 6 |
| Stral | 7 | 5 | 8 | 8 | 8 | 9 |
| Mafft | 6 | 7 | 7 | 9 | 6 | 8 |

| | k7 | | k10 | | k15 | |
|---|---|---|---|---|---|---|
| **Program** | SPS | SCI | SPS | SCI | SPS | SCI |
| LaRA | $\mathbf{1}^{(\mathbf{3-10})}$ | $\mathbf{1}^{(\mathbf{3-10})}$ | $\mathbf{1}^{(\mathbf{3-10})}$ | $\mathbf{1}^{(\mathbf{3-10})}$ | $\mathbf{1}^{(\mathbf{3-10})}$ | $\mathbf{1}^{(\mathbf{3-10})}$ |
| sLaRA | 2 | 2 | 2 | 2 | 2 | 2 |
| pLaRA | 5 | 3 | 6 | 3 | 6 | 3 |
| psLaRA | 7 | 4 | 7 | 5 | 7 | 4 |
| FoldalignM | 9 | 5 | 9 | 4 | 9 | 5 |
| MURLET | 3 | 7 | 3 | 6 | 3 | 8 |
| MARNA | 10 | 10 | 10 | 10 | 10 | 10 |
| MXSCARNA | 4 | 6 | 4 | 7 | 4 | 7 |
| Stral | 8 | 9 | 8 | 9 | 8 | 9 |
| Mafft | 6 | 8 | 5 | 8 | 5 | 6 |

Table 5.4: Results of the Friedman test for the Compalign and SCI scores. The p-value for the test is 0.05. For the programs with the highest rank-sum, *i.e.*, the programs that are ranked first, we perform pairwise Wilcoxon signed-rank tests: the superscript numbers denote the ranks of the programs to which significant differences exist.

Table 5.6 shows that the running time of MURLET is very high. This is quite in contrast to what the authors of the corresponding paper [81] claim, namely the development of a fast and practical variant of the Sankoff algorithm. Taking a closer look at the paper, one recognizes that they performed their test on a self-assembled data set from the recent release of the Rfam data base comprising alignment instances above an average pairwise sequence identity of 0.45. Figure 5 of their paper shows the reduction of memory and time consumption over the APSI value for the Hammerhead_3 ribozyme family. The striking aspect is that the reduction sharply drops for the APSI range between 0.55 and 0.60, and there are no data given for instances below an APSI of 0.45. Given the shape of the curve, we speculate that the curve steeply goes up for instances below APSI values of 0.50 which would explain the high computation time of MURLET.

| Program | k2 | k3 | k5 | k7 | k10 | k15 |
|---|---|---|---|---|---|---|
| LaRA | 0 | 0 | 0 | 0 | 0 | 0 |
| sLaRA | 0 | 0 | 0 | 0 | 0 | 0 |
| pLaRA | 0 | 0 | 0 | 0 | 0 | 0 |
| psLaRA | 0 | 0 | 0 | 0 | 0 | 0 |
| FoldalignM | 0 | 0 | 0 | 0 | 0 | 0 |
| MURLET | 1 | 25 | 32 | 55 | 28 | 16 |
| MARNA | 0 | 49 | 23 | 17 | 12 | 6 |
| MXSCARNA | 0 | 0 | 0 | 0 | 0 | 0 |
| Stral | 0 | 0 | 0 | 0 | 0 | 0 |
| Mafft | 0 | 0 | 0 | 0 | 0 | 0 |

Table 5.5: Number of unsolved instances for all input instances within a time limit of 20 minutes.

| Program | k2 | k3 | k5 | k7 | k10 | k15 |
|---|---|---|---|---|---|---|
| | (2251) | (1048) | (512) | (323) | (189) | (123) |
| LaRA | 3157.74 | 4400.22 | 6397.29 | 17632.74 | 11399.62 | 16261.14 |
| sLaRA | 5234.15 | 7405.09 | 11014.26 | 30818.55 | 20099.32 | 28513.16 |
| pLaRA | – | 2844.08 | 2628.53 | 5262.44 | 2410.37 | 2318.04 |
| psLaRA | – | 5202.97 | 4934.24 | 9834.59 | 4594.13 | 4265.50 |
| FoldalignM | 10360.44 | 14208.05 | 10995.36 | 10095.93 | 9977.03 | 7871.85 |
| MURLET | 9575.54 | 88355.02 | 76051.10 | 126883.04 | 51836.57 | 43345.91 |
| MARNA | 56434.11 | 25230.23 | 30463.49 | 38143.15 | 42146.56 | 55457.50 |
| MXSCARNA | 478.74 | 380.42 | 307.61 | 616.23 | 313.21 | 271.00 |
| Stral | 18.72 | 25.21 | 19.24 | 42.57 | 24.13 | 28.96 |
| Mafft | 53.14 | 30.83 | 17.18 | 25.12 | 7.72 | 7.20 |

Table 5.6: The overall runtime (in seconds) of the programs. If a program did not compute the alignment within 20 minutes, we killed the process and took 20 minutes as the running time. The number in brackets give the number of instances per input class.

### 5.4.4 Computing the Upper Bound via the Bundle Method

Section 2.3.2 describes the bundle method which is an alternative approach for computing solutions for the Lagrangian dual. Instead of adapting the Lagrangian multipliers according to one single subgradient, the bundle method accumulates a bundle of subgradients and fits a quadratic function to them. Then, the quadratic function is used to compute the new values of the Lagrangian multipliers. Section 2.3.2 gives details about the bundle method. The ConicBundle library

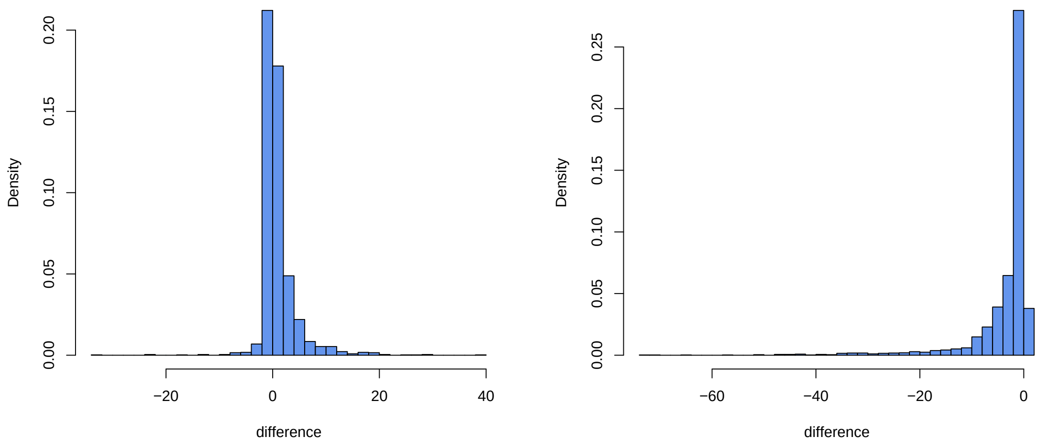


Figure 5.12: The distribution of the differences (bundle bound − subgradient bound) between the values of the upper bounds computed via the subgradient and the bundle method. A positive value means that the bundle bound is higher than the corresponding subgradient value. Left side: the distribution of the differences after maximal 500 iterations. Right side: the distribution of the difference allowing a maximal computation time of five minutes.

[62] contains generic code for using the bundle method, and we implemented an interface to the library within the LiSA library. We set the parameters according to [61], *i.e.*, we restricted the bundle size to 2 and added at most one new subgradient to the bundle.

Our test set are all $k2$ instances below an APSI value of 50%, yielding 2251 test instances. We are interested in comparing the quality of the upper bounds using the subgradient and the bundle method. We performed two different test settings: first, we stopped the subgradient and the bundle code after a maximum of 500 iterations which corresponds to the default setting in the LaRA software. Second, we allowed a maximal running time of five minutes and stopped the computation afterwards. This should clarify whether the bounds would improve significantly over a longer computation time.

The left side of Fig. 5.12 shows that in 69% of all instances (1563 out of 2251) the upper bound produced by the subgradient procedure is as good or better than the bound computed via the bundle procedure if we allow a maximal number of 500 iterations for both algorithms. If the bundle procedure does better, then the improvement is typically small. The mean and standard deviation of these instances are $-0.82$ and 2.23.

On the other hand, the difference between the bounds are a bit higher if the bundle method performs worse: the mean and standard deviation for these values are 2.44 and 3.7. The essence of these experiments is the following: if

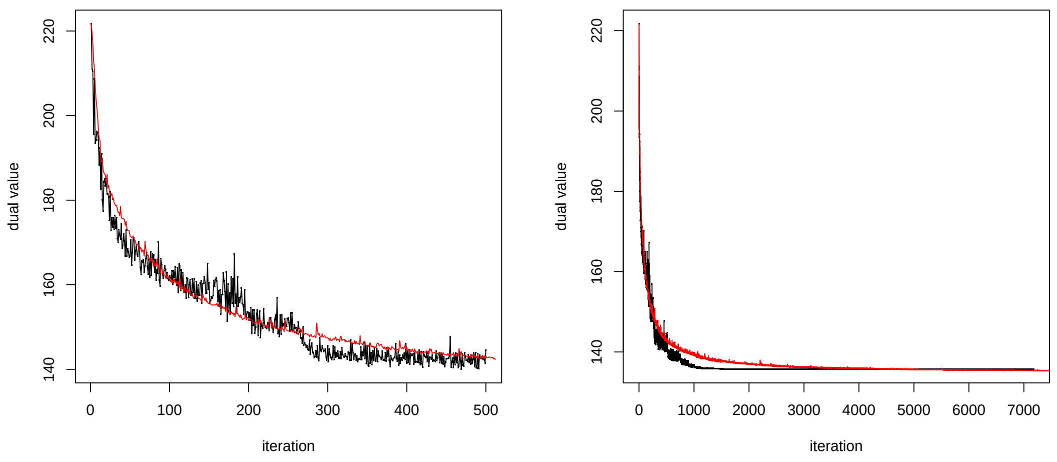


Figure 5.13: Typical behavior of the subgradient (black) and the bundle procedure (red). Left side: within a couple of hundred instances, the subgradient procedure typically produces competitive or better bounds compared to the bundle method. Right side: allowing more computation time (in this case 300 seconds), the bundle method yields slightly better bounds.

bundle performs well, then it does better, but only by a small amount. The performance of subgradient optimization is generally as good or better than the bundle method.

If we allow a maximal computation time of five hours, the picture changes. The bundle method produces better bounds in general, only in 23% of all instances (531 out of 2251) the subgradient method performs better. The mean and standard deviation of the difference between bundle and subgradient bound is small: $-0.85$ and $0.81$. Figure 5.13 shows the typical development of the upper bound both for the subgradient and the bundle method. As one can see, the bundle method produces a curve that is more smoothly, whereas the subgradient method shows more a staircase-shaped curve.

In [94] Lemaréchal states that the subgradient procedure is basically “only used by amateurs”. While it is true that subgradient optimization is conceptually much easier than bundle methods, we have to state that his opinion does not hold true in general. In our problem setting subgradient optimization is appropriate, because our primary goal is to compute good bounds as quickly as possible. The comparison of the two procedures shows that the subgradient method works better.

We are aware of the fact that by resorting to subgradient optimization we sacrifice the advantages of the bundle method, *e.g.*, explicitly estimating the difference of the current solution and the optimal value of the Lagrangian dual, or retrieving primary information. This information,however, is not important

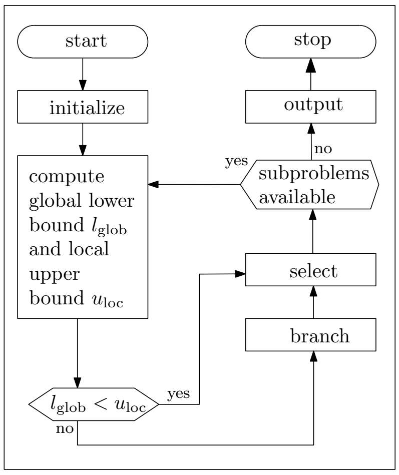

start
stop
initialize
output
no
yes
compute global lower bound $l_{\text{glob}}$ and local upper bound $u_{\text{loc}}$
subproblems available
select
branch
$l_{\text{glob}} < u_{\text{loc}}$
yes
no

Figure 5.14: Generic concept of branch-and-bound.

to the user of sequence-structure alignment algorithms. The user typically wants to have good solutions for the alignment problem as fast as possible, and this is what the subgradient procedure provides in our setting.

## 5.5 Computing Provably Optimal Solutions

Every iteration in the Lagrange solution process yields an upper and lower bound on the optimal solution value. Unless the upper and lower bound do coincide, one cannot be sure whether the best solution found is the optimal one. We therefore implemented a *branch-and-bound* algorithm that exhaustively searches the solution space given the best lower bound found during the subgradient phase.

Branch-and-bound uses a *divide-and-conquer* strategy to divide the original problem into smaller ones. This yields an enumeration tree of subproblems, where the root node corresponds to the original problem. The nodes of the tree represent constrained subproblems where variables are either set to 1 or 0. Each inner node has two subnodes where a new variable is set to 1 and 0. To avoid the exhaustive search of the entire tree, at each node we compute a local upper bound on the subproblem, and compare this upper bound to a global lower bound. If the local upper bound is smaller than the global lower bound, we can safely backtrack, because we cannot do better if we step down further that subtree. Otherwise we might achieve a better solution value in the subtree, we choose a variable $x_v$ and create two new subproblems by adding the constraints $x_v = 1$ and $x_v = 0$. Figure 5.14 summarizes the steps of the generic branch-and-bound algorithm.

There are only two main components in every branch-and-bound algorithm: the computation of the local upper bound and strategy of choosing the next branching variable. In Sect. 5.5.2 and Sect. 5.5.3 we shall elaborate on these issues. Additionally, in our case we employ a preprocessing phase to lower the number of branching variables which we shall describe in the following section.

### 5.5.1 Preprocessing the Input

Similar to [21] we can preprocess the input and eliminate alignment edges variables that cannot be part of the optimal solution. To this end, for each alignment edge $e$ we compute the score $s_e$ of the *best* alignment that includes $e$. If $s_e$ is smaller than the value of a feasible solution to the sequence-structure alignment problem, we can safely drop $x_e$ from the pool of branching variables, because $e$ will never be part of the optimal alignment.

To be more specific: given two sequences $s_0$ and $s_1$, we have a scoring matrix $\Delta$ where $\Delta(i,j)$ holds the maximum profit that alignment edge $e = (i,j)$ can possibly achieve. Then, the value $s_e$, *i.e.*, the score of the best alignment $\mathcal{A}$ with $s_i$ being alignment to $s_j$, is

$$\begin{aligned} s_e \quad = \quad & \text{align}(s_0[0,\ldots,i-1], s_1[0,\ldots j-1]) + \\ & \Delta(i,j) + \\ & \text{align}(s_0[i+1,\ldots,(|s_0|-1)], s_1[j+1,\ldots,(|s_1|-1)]) \end{aligned}$$

The matrix $\text{align}(s_0[i_0,\ldots,i_1], s_1[j_0,\ldots,j_1])$ holds the value of the optimal sequence alignment between subsequences $s_0[i_0,\ldots,i_1]$ and $s_1[j_0,\ldots,j_1]$. If $s_e$ is smaller than some global lower bound $l$, then we drop $e$ from the list of branching variables. In our experiments we start from the complete bipartite graph and we observe that the reduction of branching variables typically ranges from $75-95\%$. If the reductions amounts to 95% of all variables, then the upper and lower bound obtained after the subgradient optimization procedure are already almost the same.

### 5.5.2 Computation of the Upper Bound

The computation of a local upper bound on a subproblem requires the solution of a constraint alignment problem. The set $C$ contains the positions that have to be aligned, whereas $N$ comprises the set of positions that must not be aligned.

We solve this constrained alignment problem in a straightforward manner: for alignment edges $c = (i,j) \in C$ we set their score to some high value $M$, *i.e.*, $\Delta(i,j) = M$. This forces the optimal alignment path to run through these positions. For alignment edges $n = (k,l) \in N$ we have $\Delta(k,l) = -M$ which means that these matches will never be realized. Given the resulting score matrix $\Delta$ we solve the sequence alignment problem which yields the optimal sequence alignment value of $\Omega$. We get the alignment score $\hat{\Omega}$ by substracting the bonuses $M$, *i.e.*, $\hat{\Omega} = \Omega - |C| \cdot M$. The value of $\hat{\Omega}$ then gives the local upper bound on the problem.

### 5.5.3 Choosing the Branching Variables

Solving the Lagrange relaxed problem using subgradient optimization—in contrast to bundle methods— does not directly yield information on the value of the primal variables. We therefore compute reasonable values of primal variables the following way: let $k$ be the number of overall iterations during the subgradient phase, then for every alignment edge $e$ the value $c(e)$ denotes how often alignment edge $e$ was part of the solution set from the relaxed problem. Then, we take $p(e) = \frac{c(e)}{k}$ as an approximation for the primal value of each alignment edge.

The literature provides various schemes for the selection of the next branching variable based on the primal information. The most commonly used strategies select values either close to 0, 1, or to 0.5. In our experiments we could not observe big performance differences between various branching schemes, so in the following we report on our results for choosing the variables that are closest to 0.

### 5.5.4 Results

As we describe in Sect. 5.3, the algorithm to compute an exact multiple sequence alignment shows an unpredictable runtime behavior in practice. We therefore constrain ourselves to the pairwise case, since we compute optimal solutions of the relaxed problem in $\mathcal{O}(n^2)$.

Table 5.7 shows the results for the branch-and-bound algorithm on the pairwise BRAliBase instances. We allowed a maximal runtime of two hours. If the branch-and-bound algorithm did not stop within the time limit, we killed the process. Note that there are two Cobalamin instances where the implementation quits due to a memory overflow. In these two instances there are still 27828 and 27795 variables left after the variable reduction phase. A large gap between the upper and lower bound deprives us from the average reduction of $75 - 95\%$ of all variables, but only 58% and 61% for these two instances. This, in turn, is due to the extreme length differences of the input sequences. The sequences are 178 and 268, and 177 and 256 nucleotides long, respectively.

The authors of [21] develop a branch-and-bound algorithm for the solution of *quadratic knapsack problems* which is similar to the computation of RNA sequence-structure alignments. During our experiments we could confirm their observations. In [21] they state that

> One can observe that the upper and lower bounds are generally very tight, making it possible to reduce a majority of the variables, on average more than 75%. (. . . ) Despite this effective preprocessing, the final branch-and-bound phase demands some hours and a huge number of nodes for the largest instances, as many variables have to be fixed by branching before closing the gap, despite the latter is typically very small already at the root node.

We observe exactly the same behavior in our experiments. The values for the upper and lower bound in Tab. 5.7 (columns *ub* and *lb*) show that we are only able to solve instances to provable optimality where the gap between the upper and lower bound is already fairly small after the subgradient procedure. A small gap between upper and lower bounds leads to a small number of variables after

| | APSI | # | ⌀ len | # vars | lb | ub | opt | ratio | time |
|---|---|---|---|---|---|---|---|---|---|
| **tRNA** | 22 | 1 | 71 | 838 | 162.58 | 163.13 | 162.58 | (1.00) | 165.99 |
| | 23 | 2 | 72 | 844 | 166.19 | 170.96 | 168.13 | (0.99) | 3856.02 |
| | 24 | 3 | 71 | 658 | 127.04 | 127.93 | 127.04 | (1.00) | 274.45 |
| | 25 | 1 | 70 | 819 | 106.17 | 106.37 | 106.17 | (1.00) | 163.91 |
| | 26 | 2 | 73 | 855 | 128.94 | 129.47 | 128.94 | (1.00) | 390.90 |
| | 27 | 2 | 71 | 704 | 159.04 | 162.32 | 159.04 | (1.00) | 1081.04 |
| | 28 | 6 | 71 | 613 | 166.79 | 168.41 | 167.09 | (1.00) | 317.67 |
| | 30 | 4 | 71 | 578 | 145.26 | 146.60 | 145.86 | (1.00) | 119.09 |
| | 31 | 5 | 72 | 821 | 159.30 | 160.62 | 159.30 | (1.00) | 306.68 |
| | 32 | 8 | 72 | 689 | 160.93 | 161.70 | 160.93 | (1.00) | 652.90 |
| | 33 | 9 | 74 | 773 | 154.87 | 156.85 | 155.09 | (1.00) | 1872.71 |
| | 34 | 16 | 73 | 695 | 168.53 | 170.92 | 168.75 | (1.00) | 1198.23 |
| | 35 | 3 | 73 | 679 | 181.93 | 185.18 | 182.24 | (1.00) | 1907.45 |
| | 36 | 4 | 72 | 421 | 191.49 | 193.87 | 191.71 | (1.00) | 198.56 |
| | 37 | 3 | 69 | 535 | 149.54 | 150.30 | 149.80 | (1.00) | 360.81 |
| | 38 | 11 | 73 | 666 | 164.44 | 165.83 | 164.45 | (1.00) | 871.19 |
| | 39 | 7 | 75 | 869 | 157.62 | 161.39 | 158.61 | (0.99) | 2022.37 |
| | 40 | 14 | 74 | 707 | 175.98 | 177.94 | 175.98 | (1.00) | 1071.43 |
| | 41 | 5 | 73 | 647 | 190.95 | 194.16 | 192.20 | (0.99) | 1713.44 |
| | 42 | 10 | 73 | 721 | 170.10 | 171.74 | 170.13 | (1.00) | 712.24 |
| | 43 | 12 | 74 | 873 | 163.88 | 165.40 | 163.93 | (1.00) | 1782.08 |
| | 44 | 16 | 73 | 668 | 191.51 | 193.83 | 191.95 | (1.00) | 907.14 |
| | 45 | 8 | 73 | 821 | 162.56 | 165.44 | 163.01 | (1.00) | 1681.94 |
| | 46 | 18 | 73 | 579 | 189.49 | 190.76 | 189.52 | (1.00) | 405.01 |
| | 47 | 14 | 74 | 781 | 182.72 | 184.92 | 183.05 | (1.00) | 1661.17 |
| | 48 | 17 | 71 | 688 | 179.61 | 181.43 | 179.96 | (1.00) | 1427.49 |
| | 49 | 23 | 74 | 798 | 179.54 | 181.50 | 179.94 | (1.00) | 959.37 |
| **5S** | 41 | 2 | 119 | 1701 | 307.52 | 307.62 | 307.52 | (1.00) | 2910.48 |
| | 42 | 2 | 118 | 1757 | 233.00 | 233.88 | 233.00 | (1.00) | 3895.03 |
| | 44 | 2 | 116 | 863 | 262.86 | 263.72 | 262.86 | (1.00) | 1915.21 |
| | 45 | 4 | 119 | 1487 | 239.88 | 240.07 | 239.90 | (1.00) | 2375.55 |
| | 46 | 1 | 120 | 871 | 326.24 | 326.71 | 326.24 | (1.00) | 1339.70 |
| | 47 | 5 | 117 | 1584 | 240.86 | 241.61 | 240.86 | (1.00) | 4568.20 |
| | 48 | 1 | 120 | 559 | 369.06 | 369.42 | 369.06 | (1.00) | 439.11 |
| | 49 | 1 | 119 | 2446 | 250.03 | 250.03 | 250.03 | (1.00) | 1983.03 |
| **Coba-lamin** | 47 | 1 | 192 | 1754 | 404.55 | 404.55 | 404.55 | (1.00) | 3758.07 |

Table 5.7: All instances solved by the branch-and-bound algorithm. We report the average values grouped according to the pairwise sequence identity of the input sequences. The second and third column give the number of solved instances and the average length of the input sequence, respectively. Column # vars gives the number of variables after the preprocessing phase. Columns lb and ub represent the lower and upper bound at the root node. Furthermore, columns opt and ratio give the value of the optimal solution and the degree of optimality of the lower bound found at the root node. Finally, the last column reports the runtime in seconds.

the preprocessing step and provides sharp bounds during the bounding phase of the branch-and-bound algorithm. We therefore did not even try to solve instances to optimality whose ratio between lower and upper bound is smaller than 0.95.

Second, we are also facing the fact that we have to set a high number of variables to 1 or 0, before we close the gap between the upper and lower bound. In many cases, we have to constrain the entire alignment, until we are able to prune the subtree. For a typical tRNA instance this means that we are setting $50 - 60$ variables to 1, before the upper bound finally becomes smaller than the global lower bound.

Furthermore, the lower bounds computed during the subgradient phase are optimal in almost all cases. Table 5.7 lists only three bins in which the optimal solution deviates from the value of the best lower bound found during the subgradient phase. This observation is, however, not surprising given the fact that the gap between the lower and upper bound is typically very small in the instances that we tackle with our branch-and-bound code. Table 5.8 gives an overview over all 1624 pairwise instances that we tackled using the branch-and-bound algorithm.

| **Group** | **Lagrange solved** | **Gap too big** | **Solved by BB** | **Unsolved by BB** |
|---|---:|---:|---:|---:|
| tRNA | 476 | 466 | 224 | 92 |
| 5S | 67 | 122 | 18 | 42 |
| Cobalamin | 0 | 110 | 1 | 6 |

Table 5.8: Summary over all instances processed by the branch-and-bound framework. *Lagrange solved*, *Gap too big*, *Solved by BB*, and *Unsolved by BB* give the numbers of instances that are solved to optimality after the subgradient phase, whose gap after the subgradient phase is too big, that are solved to optimality by the branch-and-bound algorithm, and that exceeded the branch-and-bound time limit of two hours, respectively.

CHAPTER 6

# Conclusion and Future Work

I glaub i geh jetzt,
weil i was genau,
wenn i noch länger bleib,
geht ma der Schmäh aus,
und des wü i net.

Wolfgang Ambros
(I glaub i geh jetzt)

In this thesis we presented a framework for computing sequence-structure alignments of RNA structures based on techniques from combinatorial optimization. The comparison of our implementations with several other state-of-the-art programs shows that we performed very well on the established BRALIBASE benchmark set. Both the consistency-based LARA and sLARA, and the progressive versions pLARA and psLARA are top-ranked for all input classes.

We refrain, however, from claiming that our tools are the best alignment programs for each input class. Each of the tested programs has its strengths and weaknesses: FOLDALIGNM, for example, generally performs better on SRP instances compared to our programs, whereas LARA and sLARA outperform FOLDALIGNM on tRNA sequences. Therefore, one cannot speak of one single best sequence-structure alignment program for all input classes as other authors [81; 132] did, because tests on self-compiled data sets usually show that their program works best on their data. Consequently, we chose the BRALIBASE benchmark to evaluate our programs, because this way we avoid putting a data set together and afterwards claim that we performed best on it. We believe that the community should work and agree on a benchmark set BRALIBASE *next generation* that allows a fair comparison of different structure alignment programs. This benchmark should eliminate the deficiencies of the current BRALIBASE release:

(a) update the sequences to the latest version of the RFAM database,

(b) remove the bias of input instances towards some RNA families that have a large number of sequences in their seed alignments,

(c) incorporate published structures into the alignment, and subsequently use the MCC instead of the SCI as the structural assessment score,

(d) incorporate 3D information—if available—to enhance the alignment quality.

We are aware that creating a large benchmark set satisfying these constraints is a non-trivial task, especially since for most of the RNA families in the RFAM the sequence data base is sparse.

During the evaluation of our programs it became very clear that the performance of a program does not only depend on the model or formulation it is using, since a great deal of improvement or decline in performance accounts for the setting of the parameters. For most programs, parameter settings are either determined by systematically trying out various parameter settings [96], or values are chosen that seem to work well in practice [67]. LaRA has three main parameters: the sequence contribution, and the gap open and extension penalty. We used the MASTR data set as our training set and examined various parameter possibilities, until the values worked well on the MASTR data. In the case of sLaRA, the situation becomes even more involved, because apart from the sequence scaling we have to balance the structure and stacking contribution thus yielding an expanded parameter search space. We tested seven different values for both the structure and stacking contribution yielding 49 times more possibilities to explore. There are various distinct parameter sets that yield almost the same performance as the sets that we finally chose for our evaluation. It is very likely that a combination of these parameter sets yields comparable or even better results. Hence, a proper parameter training method is of utmost importance. In the case of pure sequence-based alignments Kececioglu and Kim [80] propose an approach based on linear programming: given an alignment, the *inverse alignment problem* calls for determining the (user-specified) parameters such that the optimal sequence alignment—using these parameters—yields the input alignment. In principle, their approach is applicable to sequence-structure alignments as well. For a multiple alignment annotated with the conserved interactions, it is possible to determine the optimal parameter set. Input alignments could be taken, for instance, from the Rfam or the *European rRNA database*. One has to take care not to overfit the parameters to certain input groups; hence, the input alignments should be distributed among various classes of noncoding RNAs.

Besides the gap and scaling parameters, the scoring system greatly influences the performance of our implementations. We use scores based on the base pairing probability matrices. The pairing probabilities in turn are derived from the partition function which takes all possible nested secondary structures into account. Our model allows for all possible pseudoknots, because the only constraint is that a nucleotide might pair with at most one other residue. In our experiments we observe that the structural completion computed via the maximum-weight matching computation often contains pseudoknots that do not violate the definition of a secondary structure, but that will not be observed in Nature. Figure 6.1 gives an example of such a typical case. Therefore, for the computation of a secondary structure we resort to RNAlifold that computes a nested consensus structure given an alignment. sLaRA generally inserts fewer arbitrary pseudoknots, because it favours the consecutive stacking of base pairs.

One of the main advantages of our formulation is its ability to deal with pseudoknots. In contrast to most of the DP based approaches and all the tree-based models, we can align structures that contain pseudoknots. Right now,

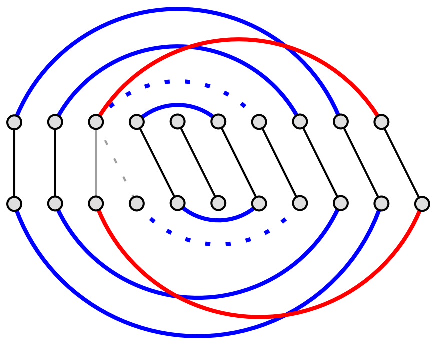


Figure 6.1: The solid lines represent the interactions that are conserved via the alignment. The red interaction match denotes the pseudoknot that is inserted, since we do not reward stacking of base pairs in our initial model. If we take the dotted instead of the solid alignment edge, the alignment conserves the interaction match denoted by blue dotted lines, yielding a more resonable structure.

however, we are not able to take advantage of it, because algorithms predicting secondary structures or base pair probabilities including pseudoknots suffer from two main drawbacks. First, all these approaches are computationally expensive (their time complexity scales at least in $\mathcal{O}(n^4)$), which makes them applicable only to short sequences. Second, there are no sound energy parameters available for pseudoknotted structures. This is even more significant, because it means that even if we have the algorithms and models to incorporate pseudoknots, we are optimizing an objective function that is misleading. Hence, the only scenario where LaRA could make use of its pseudoknot alignment abilities is the alignment of experimentally verified structures that contain pseudoknots.

At the time this thesis is being written, new approaches have been proposed that aim at avoiding the high computational costs of the Sankoff variants. Mafft 6, the latest version of the Mafft [78] alignment program, introduces the concept of *four-way consistency* which extends the consistency-based sequence alignment described in Sect. 3.2 to incorporate structural information. Preliminary tests of the beta version on the BRAliBase show an improved performance compared to the previous Mafft versions, while the running time increased only slightly.

**Future Work.** The work presented in this thesis provides several lines of future research. Our graph-based model can be modified to tackle other alignment problems such as *local* sequence-structure alignments. We already implemented a prototype that searches local sequence-structure motifs in the spirit of RNAForester, *i.e.*, subsequences of the input sequences that share a common structural motif. Running our prototype on the same data as RNAForester, we are not only able to find the local motifs that RNAForester finds, but also other elements that are published in the literature. Backofen and Will [2] describe a different version of local sequence-structure alignment. Instead of only considering entire subsequences, the authors also allow sequence-structure motifs where parts of the sequence are omitted. This is the case, for instance, if we have a helix in the first sequence that does not exist in the second input sequence. The only constraint is, however, that the motif has to be connected either in the sequence or structure. In principle, the graph-based model is also able to capture such motifs, but the handling of omitted subsequences is an unsolved issue at this moment.

The original MXSCARNA approach [133] inspired the model that incorporates stacking scores. Naturally, the halfstems of stem fragments correspond to nodes in the graph, and we have an interaction edge between nodes that form a stem. Remember that MXSCARNA first aligns the stem fragments and uses these as anchors in a subsequent sequence alignment. We implemented a prototype that goes beyond the model of [133] by aligning stem fragments and loop regions at the same time. The recursions for the extended model become intricate and the prototype did not yield satisfactory results. By going back one step, however, and applying the graph-based model only on the stem fragments, we reduce the size of the problems, because a node now corresponds to a stem fragment and not to a single nucleotide anymore. In the end, this would lead to decreased running times of our approach.

A theoretical problem that needs further research is the computation of a feasible solution—given the solution of the relaxed problem—including stacking energies. In Sect. 4.3.2 we show how we can compute an exact solution by solving a max-weight independent set problem. This reduction is, however, of little practical interest, since the computation of an independent set is NP-complete. Hence, the problem consists in either proving the NP-hardness of the problem, or giving a polynomial time algorithm that computes an exact solution.

# Appendix A Deutsche Zusammenfassung

Puh,
das war harter Stoff.

Die Ärzte
(Zusammenfassung)

Wissenschaftliche Entdeckungen der letzten Jahre haben die Molekulargenetik revolutioniert: bis dahin ging man von einem linearen Informationsfluss aus, in dem DNA zu RNA, und RNA in Proteine übersetzt wird. RNA nahm dabei die Rolle eines Hilfsmoleküls ein, das selbst—bis auf wenige Ausnahmen—keinerlei katalytische Eigenschaften hat. In den letzten Jahren zeigte sich jedoch, dass man von einer viel komplexeren Organisation der zellulären Prozesse ausgehen muss: Nichtkodierende RNA-Sequenzen, d.h. RNA-Sequenzen die keine Proteine kodieren, spielen dabei eine wesentliche Rolle. Bei der Analyse von RNA-Sequenzen ist es wichtig, Strukturinformation zu beachten, da die sogenannte Sekundärstruktur, und nicht so sehr die eigentliche Sequenzinformation erhalten bleibt. Alignmentprogramme von divergenten RNA-Sequenzen müssen deshalb Strukturinformation miteinbeziehen, um zuverlässige Alignments zu erstellen.

In dieser Arbeit stellen wir ein neues Modell für das Berechnen von multiplen Sequenz-Struktur-Alignments von RNA-Sequenzen vor. Wir beschreiben Struktur-Alignments als graphentheoretisches Problem und zeigen danach, wie man dieses Modell als ganzzahliges lineares Programm (ILP) formulieren kann. Wir relaxieren das ILP im Folgenden in einer Lagrangeschen Weise, d.h. wir verschieben eine Klasse von Bedingungen—versehen mit einem Strafterm-Vektor—in die Zielfunktion und lösen das resultierende ILP. Zusätzlich beschreiben wir eine Erweiterung des ILPs, bei der sogenannte Stackingenergien in die Berechnung des Sequenz-Struktur-Alignments einfließen.

Im Rahmen einer umfangreichen Auswertung vergleichen wir die Implementierungen unserer Modelle mit zahlreichen anderen aktuellen Programmen. Unsere Programme liefern auf einem kürzlich publizierten Benchmark-Datensatz die besten Ergebnisse für alle Klassen von Eingabedaten. Zusätzlich geben wir einen Vergleich zwischen dem Subgradienten-Verfahren und der Bündel-Methode zum Lösen des dualen Problems. Wir können zeigen, dass für Standard-Eingabeinstanzen das Subgradienten-Verfahren normalerweise bessere Ergebnisse liefert. Den Abschluss der praktischen Auswertung bildet die Beschreibung eines Branch-und-Bound-Verfahrens, das—gegeben die Schranken aus dem Subgradienten-Verfahren—beweisbar optimale Lösungen berechnet. Wir zeigen, dass der Anwendungsrahmen dieses Ansatzes in etwa dem entspricht, was für das verwandte Problem des quadratischen Rucksackproblems publiziert wurde.

# BIBLIOGRAPHY

[1] E. Althaus, A. Caprara, H.-P. Lenhof, and K. Reinert. A branch-and-cut algorithm for multiple sequence alignment. *Mathematical Programming*, 105(2-3):387–425, 2006.

[2] R. Backofen and S. Will. Local sequence-structure motifs in RNA. *J. of Bioinformatics and Computational Biology*, 2(4):681–698, Dec 2004.

[3] V. Bafna, S. Muthukrishnan, and R. Ravi. Computing similarity between RNA strings. In *Proc. CPM'95*, number 937 in LNCS, pages 1–16. Springer, 1995.

[4] M. Bauer and G. W. Klau. Structural alignment of two RNA sequences with Lagrangian relaxation. In *Proc. ISAAC'04*, number 3341 in LNCS, pages 113–123. Springer, 2004.

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

[6] M. Bauer, G. W. Klau, and K. Reinert. Multiple structural RNA alignment with Lagrangian relaxation. In *Proc. WABI'05*, volume 3692 of *LNBI*, pages 303–314, 2005.

[7] M. Bauer, G. W. Klau, and K. Reinert. Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. *BMC Bioinformatics*, 8:271, Jul 2007.

[8] M. Bauer, G.W. Klau, and K. Reinert. An exact mathematical programming approach to multiple RNA sequence-structure alignment. *Algorithmic Operations Research*, 2008. to appear.

[9] A.B. Bellamy-Royds and M. Turcotte. Can Clustal-style progressive pairwise alignment of multiple sequences be used in RNA secondary structure prediction? *BMC Bioinformatics*, 8:190, 2007.

[10] D. Bertsimas and J.N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, 1997.

[11] G. Blackshields, I.M. Wallace, M. Larkin, and D.G. Higgins. Analysis and comparison of benchmarks for multiple sequence alignment. *In Silico Biol*, 6(4):321–339, 2006.

[12] G. Blin, G. Fertin, G. Herry, and S. Vialette. Comparing RNA structures: Towards an intermediate model between the Edit and the LAPCS problems. In *Proc. BSB'07*, pages 101–112, 2007.

[13] G. Blin, G. Fertin, I. Rusu, and C. Sinoquet. Extending the hardness of RNA secondary structure comparison. In *Proc. ESCAPE'07*, pages 140–151, 2007.

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

[15] A. Bompfünewerer, R. Backofen, S. Bernhart, J. Hertel, I.L. Hofacker, P.F. Stadler, and S. Will. Variations on RNA folding and alignment: lessons from benasque. *J Math Biol*, 56(1-2):129–144, Jan 2008.

[16] S. Bonhoeffer, J. S. McCaskill, P. F. Stadler, and P. Schuster. RNA multi-structure landscapes. a study based on temperature dependent partition functions. *Eur Biophys J*, 22(1):13–24, 1993.

[17] P. Bose, J. F. Buss, and A. Lubiw. Pattern matching for permutations. *Inf. Process. Lett.*, 65(5):277–283, 1998.

[18] M. Brinkmeier. Structural alignments of pseudoknotted RNA-molecules in polynomialtime. Technical report, Technical University of Ilmenau, 2003.

[19] J. J Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M D'Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Muller, N. Pande, Z. Shang, N. Yu, and R. R. Gutell. The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3, 2002.

[20] A. Caprara and G. Lancia. Structural alignment of large-size proteins via lagrangian relaxation. In *Proc. RECOMB'02*, pages 100–108, 2002.

[21] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS J. on Computing*, 11(2):125–137, 1999.

[22] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *J. of the American Statistical Association*, 74(368):829–836, 1979.

[23] ENCODE Consortium. Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. *Nature*, 447(7146):799–816, Jun 2007.

[24] ENCODE Project Consortium. The ENCODE (ENCyclopedia Of DNA Elements) project. *Science*, 306(5696):636–640, Oct 2004.

[25] FANTOM Consortium. Analysis of the mouse transcriptome based on functional annotation of 60,770 full-length cDNAs. *Nature*, 420(6915):563–573, Dec 2002.

[26] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, Feb 2001.

[27] J. Couzin. Breakthrough of the year: Small RNAs make big splash. *Science*, 298(5602):2296–2297, 2002.

[28] F. Crick. On protein synthesis. *Symp Soc Exp Biol*, 12:138–163, 1958.

[29] F. Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, Aug 1970.

[30] 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(13):1593–1599, 2006.

[31] E. Davydov and S. Batzoglou. A computational model for RNA multiple structural alignment. *Theoretical Computer Science*, 368(3):205–216, 2006.

[32] E. D. Demaine, S. Mozes, B. Rossman, and O. Weimann. An optimal decomposition algorithm for the tree edit distance. In *Proc. ICALP'07*, pages 146–157, 2007.

[33] R. Diestel. *Graph Theory*. Springer, 1996.

[34] R.M. Dirks and N.A. Pierce. A partition function algorithm for nucleic acid secondary structure including pseudoknots. *J. of Compututational Chemistry*, 24(13):1664–1677, Oct 2003.

[35] R.M. Dirks and N.A. Pierce. An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *J. of Computational Chemistry*, 25(10):1295–1304, Jul 2004.

[36] C. B Do, M.S.P. Mahabhashyam, M. Brudno, and S. Batzoglou. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15(2):330–340, Feb 2005.

[37] B. Dost, B. Han, S. Zhang, and V. Bafna. Structural alignment of pseudoknotted RNA. In *Proc. RECOMB'06*, pages 143–158, 2006.

[38] R. Dowell and S.R. Eddy. Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*, 7:400, 2006.

[39] S. Dulucq and L. Tichit. RNA secondary structure comparison: exact analysis of the zhang-shasha tree edit algorithm. *Theoretical Computer Science*, 306(1-3):471–484, 2003.

[40] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.

[41] S. P. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22(11):2079–2088, 1994.

[42] S.R. Eddy. SQUID, C function library for sequence analysis, December 2007. `http://selab.janelia.org/software.html`.

[43] P. Evans. *Algorithms and complexity for annotated sequence analysis*. PhD thesis, University of Victoria, 1999.

[44] P. Evans. Finding common subsequences with arcs and pseudoknots. In *Proc. of CPM'99*, number 1645 in LNCS, pages 270–280. Springer, 1999.

[45] M. L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 50(12 Supplement):1861–1871, 2004.

[46] A. Frangioni. About lagrangian methods in integer optimization. *Annals of Operations Research*, 139(1):163–193, 2005.

[47] E.K. Freyhult, J.P. Bollback, and P.P Gardner. Exploring genomic dark matter: A critical assessment of the performance of homology search methods on noncoding RNA. *Genome Research*, 17(1):117–125, 2007.

[48] P. Gardner, A. Wilm, and S. Washietl. A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Research*, 33(8):2433–2439, 2005.

[49] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman., 1979.

[50] W. Gilbert. Origin of life: The RNA World. *Nature*, 319:618, 1986.

[51] F. Glover and D. Klingman. Layering strategies for creating exploitable structure in linear and integer programs. *Mathematical Programming*, 40(2):165–181, 1988.

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

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

[54] C. Guerrier-Takada, K. Gardiner, T. Marsh, N. Pace, and S. Altman. The RNA moiety of ribonuclease p is the catalytic subunit of the enzyme. *Cell*, 35(3 Pt 2):849–857, Dec 1983.

[55] M. Guignard. Lagrangean relaxation. *TOP*, 11(2):151–200, 2003.

[56] M. Guignard and S. Kim. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39(2):215–228, 1987.

[57] S. D. Hatfield, H. R. Shcherbata, K. A. Fischer, K. Nakahara, R. W. Carthew, and H. Ruohola-Baker. Stem cell division is regulated by the microRNA pathway. *Nature*, 435(7044):974–978, Jun 2005.

[58] J.H. Havgaard, R. Lyngsø, G. Stormo, and J. Gorodkin. Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics*, 21:1815–1824, 2005.

[59] J.H. Havgaard, E. Torarinsson, and J. Gorodkin. Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. *PLoS Computational Biology*, 3(10), Oct 2007.

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

[61] C. Helmberg. Personal communication.

[62] C. Helmberg. The ConicBundle library for convex optimization, January 2008. `http://http://www-user.tu-chemnitz.de/~helmberg/ConicBundle/`.

[63] D.S. Hirschberg. *The Longest Common Subsequence Problem*. PhD thesis, Princeton University, 1975.

[64] M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz. Local similarity in RNA secondary structures. In *Proc. CSB'03*, pages 159–168, 2003.

[65] M. Höchsmann, B. Voss, and R. Giegerich. Pure multiple RNA secondary structure alignments: a progressive profile approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):53–62, 2004.

[66] I. L. Hofacker. The Vienna RNA secondary structure server. *Nucleic Acids Research*, 31:3429–3431, 2003.

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

[68] I. L. Hofacker, M. Fekete, and P. F. Stadler. Secondary structure prediction for aligned RNA sequences. *J. of Molecular Biology*, 319:1059–1066, 2002.

[69] I.L. Hofacker and P.F. Stadler. *Bioinformatics - From Genomes to Therapies*, chapter RNA Secondary Structure, pages 439–491. Wiley-Vch, 2007.

[70] I. Holmes. A probabilistic model for the evolution of RNA structure. *BMC Bioinformatics*, 5:166, 2004.

[71] I. Holmes. Accelerated probabilistic inference of RNA structure evolution. *BMC Bioinformatics*, 5:73, 2005.

[72] I. Holmes and G. M. Rubin. Pairwise RNA structure comparison with stochastic context-free grammars. In *Proc. PSB'02*, pages 163–174, 2002.

[73] T. Jiang, G. Lin, B. Ma, and K. Zhang. The longest common subsequence problem for arc-annotated sequences. In *Proc. CPM'00*, pages 154–165, 2000.

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

[75] T. Jiang, J. Wang, and K. Zhang. Alignment of trees—an alternative to tree edit. *Theoretical Computer Science*, 143:137–148, 1995.

[76] P. Kapranov, J. Cheng, S. Dike, D.A. Nix, R. Duttagupta, A.T. Willingham, P.F. Stadler, J. Hertel, J. Hackermüller, I.L. Hofacker, I. Bell, E. Cheung, J. Drenkow, E. Dumais, S. Patel, G. Helt, M. Ganesh, S. Ghosh, A. Piccolboni, V. Sementchenko, H. Tammana, and T.R. Gingeras. RNA maps reveal new RNA classes and a possible function for pervasive transcription. *Science*, 316(5830):1484–1488, Jun 2007.

[77] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[78] K. Katoh, K.-I. Kuma, H. Toh, and T. Miyata. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research*, 33(2):511–518, 2005.

[79] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. In *Proc. CPM'93*, volume 684 of *LNCS*, pages 106–119, 1993.

[80] J.D. Kececioglu and E. Kim. Simple and fast inverse alignment. In *RECOMB*, pages 441–455, 2006.

[81] H. Kiryu, Y. Tabei, T. Kin, and K. Asai. Murlet: a practical multiple alignment tool for structural RNA sequences. *Bioinformatics*, 23(13):1588–1598, Jul 2007.

[82] P. N. Klein. Computing the edit-distance between unrooted ordered trees. In *ESA*, pages 91–102, 1998.

[83] R. Klein and S. R. Eddy. RSEARCH: Finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4:44, 2003.

[84] B. Knudsen and J. Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Research*, 31(13):3423–3428, 2003.

[85] S.B. Koralov, S.A. Muljo, G.R. Galler, A. Krek, T. Chakraborty, C. Kanellopoulou, K. Jensen, B.S. Cobb, M. Merkenschlager, N. Rajewsky, and K. Rajewsky. Dicer ablation affects antibody diversity and cell survival in the b lymphocyte lineage. *Cell*, 132(5):860–874, Mar 2008.

[86] M. Kubica, R. Rizzi, S. Vialette, and T. Walen. Approximation of RNA Multiple Structural Alignment. In *CPM*, pages 211–222, 2006.

[87] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML'01*, pages 282–289, 2001.

[88] M. Lagos-Quintana, R. Rauhut, W. Lendeckel, and T. Tuschl. Identification of novel genes coding for small expressed RNAs. *Science*, 294(5543):853–8, 2001.

[89] 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. In *Proc. RECOMB'01*, pages 193–202, 2001.

[90] N. C. Lau, L. P. Lim, E. G. Weinstein, and D. P. Bartel. An abundant class of tiny RNAs with probable regulatory roles in *Caenorhabditis elegans*. *Science*, 294(5543):858–62, 2001.

[91] LEDA. *The LEDA User Manual*. Algorithmic Solutions, March 2007. `http://www.algorithmic-solutions.info/leda_manual/MANUAL.html`.

[92] R. C. Lee and V. Ambros. An extensive class of small RNAs in *Caenorhabditis elegans*. *Science*, 294(5543):862–4, 2001.

[93] R. C. Lee, R. L. Feinbaum, and V. Ambros. The c. elegans heterochronic gene lin-4 encodes small RNAs with antisense complementarity to lin-14. *Cell*, 75(5):843–854, Dec 1993.

[94] C. Lemaréchal. *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions*, chapter Lagrangian Relaxation, pages 112–156. Springer Berlin, 2001.

[95] H.-P. Lenhof, K. Reinert, and M. Vingron. A polyhedral approach to RNA sequence structure alignment. *J. of Computational Biology*, 5(3):517–530, 1998.

[96] S. Lindgreen, P.P. Gardner, and A. Krogh. MASTR: multiple alignment and structure prediction of non-coding RNAs using simulated annealing. *Bioinformatics*, 23(24):3304–3311, Dec 2007.

[97] A. Lozano and G. Valiente. *String Algorithmics*, chapter On the Maximum Common Embedded Subtree Problem for Ordered Trees, pages 155–169. College Publications, 2004.

[98] J. Lu, G. Getz, E.A. Miska, E. Alvarez-Saavedra, J. Lamb, D. Peck, A. Sweet-Cordero, B.L. Ebert, R.H. Mak, A.A. Ferrando, J.R. Downing, T. Jacks, H.R. Horvitz, and T.R. Golub. MicroRNA expression profiles classify human cancers. *Nature*, 435(7043):834–838, Jun 2005.

[99] R. B. Lyngsø and C. N. S. Pedersen. RNA pseudoknot prediction in energy-based models. *J. of Computational Biology*, 7:409–427, 2000.

[100] L. Ma, J. Teruya-Feldstein, and R.A. Weinberg. Tumour invasion and metastasis initiated by microRNA-10b in breast cancer. *Nature*, 449(7163):682–688, Oct 2007.

[101] D. Mathews. Predicting a set of minimal free energy RNA secondary structures common to two sequences. *Bioinformatics*, 21:2246–2253, 2005.

[102] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner. Expanded sequence dependence of thermodynamic parameters provides robust prediction of RNA secondary structure. *J. of Molecular Biology*, 288:911–940, 1999.

[103] D. H. Mathews and D. H. Turner. Dynalign: An algorithm for finding secondary structures common to two RNA sequences. *J. of Molecular Biology*, 317:191–203, 2002.

[104] D.H. Mathews, M.D. Disney, J.L. Childs, S.J. Schroeder, M. Zuker, and D.H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc. of the National Academy of Sciences*, 101(19):7287–7292, May 2004.

[105] J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–1119, 1990.

[106] M. Näsberg, K.O. Jörnsten, and P.A. Smeds. Variable spliting - a new Lagrangean relaxation approach to some mathematical programming problems. Technical report, Linkoping University, 1985.

[107] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Publication, 1988.

[108] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research, 1999.

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

[110] R. Nussinov, G. Piecznik, J. R. Griggs, and D. J. Kleitman. Algorithms for loop matching. *SIAM J. Appl. Math.*, 35(1):68–82, 1978.

[111] K.A. O'Donnell, E.A. Wentzel, K.I. Zeller, C.V. Dang, and J.T. Mendell. c-myc-regulated microRNAs modulate e2f1 expression. *Nature*, 435(7043):839–843, Jun 2005.

[112] C.H. Papadimitriou. Personal communication.

[113] B.T. Poljak. A general method of solving extremum problems. *Soviet Mathematics Doklady*, 8:593–597, 1967.

[114] J. Reeder and R. Giegerich. Design, implementation and evaluation of a practical pseudoknot folding algorithm based on thermodynamics. *BMC Bioinformatics*, 5:104, Aug 2004.

[115] J. Reeder, P. Steffen, and R. Giegerich. pknotsRG: RNA pseudoknot folding including near-optimal structures and sliding windows. *Nucleic Acids Research*, 35(Web Server issue):W320–W324, Jul 2007.

[116] K. Reinert. *A Polyhedral Approach to Sequence Alignment Problems*. PhD thesis, Universität des Saarlandes, 1999.

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

[118] E. Rivas and S. R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. of Molecular Biology*, 285:2053–2068, 1999.

[119] M. Ronaghi, M. Uhlén, and P. Nyrén. A sequencing method based on real-time pyrophosphate. *Science*, 281(5375):363, 365, Jul 1998.

[120] Jeffrey S Ross, J. Andrew Carlson, and Graham Brock. miRNA: the new gene silencer. *Am J Clin Pathol*, 128(5):830–836, Nov 2007.

[121] Y. Sakakibara. Pair hidden Markov models on tree structures. *Bioinformatics*, 19:i232–240, 2003.

[122] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. Recent methods for RNA modeling using stochastic context-free grammars. In *CPM*, pages 289–306, 1994.

[123] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc. of the National Academy of Sciences*, 74(12):5463–5467, Dec 1977.

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

[125] K. Sato and Y. Sakakibara. RNA secondary structural alignment with conditional random fields. *Bioinformatics*, 21(suppl_2):ii237–242, 2005.

[126] D. Shasha and K. Zhang. Fast algorithms for the unit cost editing distance between trees. *J. of Algorithms*, 11(4):581–621, 1990.

[127] S. Siebert and R. Backofen. MARNA: Multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, 21(16):3352–3359, 2005.

[128] L.M. Mendes Soares and J. Valcárcel. The expanding transcriptome: the genome as the 'Book of Sand'. *EMBO J*, 25(5):923–931, Mar 2006.

[129] R. Soenen. *Contribution à l'étude des systèmes de conduite en temps réel en vue de la commande d'unités de fabrication.* PhD thesis, Université de Lille, 1977.

[130] D.A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. of the ACM*, 51(3):385–463, 2004.

[131] D. W. Staple and S. E. Butcher. Pseudoknots: RNA structures with diverse functions. *PLoS Biology*, 3(6):e213, 2005.

[132] Y. Tabei, H. Kiryu, T. Kin, and K. Asai. A fast structural multiple alignment method for long RNA sequences. *BMC Bioinformatics*, 9(1):33, Jan 2008.

[133] Y. Tabei, K. Tsuda, T. Kin, and K. Asai. SCARNA: fast and accurate structural alignment of RNA sequences by matching fixed-length stem fragments. *Bioinformatics*, 22(14):1723–1729, Jul 2006.

[134] K.-C. Tai. The tree-to-tree correction problem. *J. of the ACM*, 26(3):422–433, 1979.

[135] J. D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27(13):2682–2690, Jul 1999.

[136] J.D. Thompson, P. Koehl, R. Ripp, and O. Poch. BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, 61(1):127–136, Oct 2005.

[137] E. Torarinsson, J. H. Havgaard, and J. Gorodkin. Multiple structural alignment and clustering of RNA sequences. *Bioinformatics*, 23(8):926–932, 2007.

[138] E. Torarinsson, M. Sawera, J.H. Havgaard, M. Fredholm, and J. Gorodkin. Thousands of corresponding human and mouse genomic regions unalignable in primary sequence contain common RNA structure. *Genome Research*, 16(7):885–889, Jul 2006.

[139] E. Torarinsson, Z. Yao, E.D. Wiklund, J.B. Bramsen, C. Hansen, J. Kjems, N. Tommerup andW.L. Ruzzo, and J. Gorodkin. Comparative genomics beyond sequence-based alignments: RNA structures in the ENCODE regions. *Genome Research*, 18(2):242–251, Feb 2008.

[140] J.C. Venter et al. The sequence of the human genome. *Science*, 291(5507):1304–1351, 2001.

[141] S. Vialette. On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science*, 312(2-3):223–249, 2004.

[142] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. of the ACM*, 21(1):168–173, 1974.

[143] S. Washietl, I. Hofacker, M. Lukasser, A. Hüttenhofer, and P. Stadler. Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome. *Nature Biotechnology*, 23(11):1383–1390, 2005.

[144] S. Washietl, J. S. Pedersen, J. O Korbel, C. Stocsits, A. R. Gruber, J. Hackermüller, J. Hertel, M. Lindemeyer, K. Reiche, A. Tanzer, C. Ucla, C. Wyss, S. E. Antonarakis, F. Denoeud, J. Lagarde, J. Drenkow, P. Kapranov, T.R. Gingeras, R. Guigó, M. Snyder, M.B. Gerstein, A. Reymond, I.L. Hofacker, and P.F. Stadler. Structured RNAs in the ENCODE selected regions of the human genome. *Genome Research*, 17(6):852–864, Jun 2007.

[145] J. L. Weber and E.W. Myers. Human whole-genome shotgun sequencing. *Genome Research*, 7(5):401–409, May 1997.

[146] George M Weinstock. ENCODE: more genomic empowerment. *Genome Research*, 17(6):667–668, Jun 2007.

[147] Y. Wexler, C. Zilberstein, and M. Ziv-Ukelson. A study of accessible motifs and RNA folding complexity. *J. of Computational Biology*, 14(6):856–872, 2007.

[148] S. Will, K. Reiche, I.L. Hofacker, P.F. Stadler, and R. Backofen. Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Computational Biology*, 3(4):e65, Apr 2007.

[149] A. Wilm. *RNA-Alignments and RNA-Structure in silico*. PhD thesis, Heinrich-Heine University Düsseldorf, 2006.

[150] A. Wilm, I. Mainz, and G. Steger. An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms for Molecular Biology*, 1(1):19, 2006.

[151] C. Witwer, I.L. Hofacker, and P.F. Stadler. Prediction of consensus RNA secondary structures including pseudoknots. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(2):66–77, 2004.

[152] C. R. Woese. *The Genetic Code*. Harper & Row, 1968.

[153] M. Wolf, M. Achtziger, J. Schultz, T. Dandekar, and T. Müller. Homology modeling revealed more than 20,000 rRNA internal transcribed spacer 2 (ITS2) secondary structures. *RNA*, 11(11):1616–1623, 2005.

[154] J. Wuyts, G. Perrière, and Y. Van De Peer. The European ribosomal RNA database. *Nucleic Acids Research*, 32:D101–D103, Jan 2004.

[155] T. Xia, J. SantaLucia, M. E. Burkard, R. Kierzek, S. J. Schroeder, X. Jiao, C. Cox, and D. H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry*, 37(42):14719–14735, Oct 1998.

[156] A. J. Zaug and T. R. Cech. The intervening sequence RNA of Tetrahymena is an enzyme. *Science*, 231(4737):470–475, 1986.

[157] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.

[158] S. Zhang, I. Borovok, Y. Aharonowitz, R. Sharan, and V. Bafna. A sequence-based filtering method for ncRNA identification and its application to searching for riboswitch elements. *Bioinformatics*, 22(14):e557–e565, Jul 2006.

[159] S. Zhang, B. Haas, E. Eskin, and V. Bafna. Searching genomes for noncoding RNA using FastR. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(4):366–379, 2005.

[160] M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13):3406–3415, Jul 2003.

[161] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, Jan 1981.