# Using Local RNA Secondary Structures for Computational Comparison and Clustering of RNA Molecules

**Dissertation**

zur Erlangung des akademischen Grades

Doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat
der Technischen Fakultät
der Albert-Ludwigs-Universität Freiburg

von Diplom-Bioinformatiker
**Steffen Heyne**

**Dekan:**
Prof. Dr. Yiannos Manoli

**Gutachter:**
Prof. Dr. Rolf Backofen
PD Dr. Björn Voss

Die vorliegende Arbeit wurde erfolgreich am 16.06.2014 in Freiburg verteidigt.

# Abstract

The past decade in molecular biology was characterised by a multitude of genome-wide studies which revealed fascinating insights into the complexity of genomic organization in all kingdoms of life. Surprisingly, a large extend of the transcriptional output consists of non-coding RNAs (ncRNAs), transcripts not being translated into proteins. The sheer amount of functional identified ncRNAs is just overwhelming and high-throughput sequencing technologies produce genomic and transcriptomic sequence data with an ever increasing rate. However, a precise functional annotation of the majority of RNA transcripts remains a challenge. Comparative bioinformatic approaches are commonly used for large-scale functional analysis and annotation of this immense amount of sequence data. The close linkage of sequential and structural properties in ncRNAs necessitates comparison approaches with usually high computational complexity, which in turn makes even small-sized studies often infeasible for existing approaches. In my thesis, I address these challenges and describe efficient and accurate computational methods for the comparative analysis of RNAs based on their sequence and structure.

In the first part of this thesis, I present `ExpaRNA`, a pairwise RNA comparison approach that uses a novel similarity measure based on exact matching substructures (EPMs). In contrast to previous methods, `ExpaRNA` detects conserved substructures during the RNA comparison, which is a key feature to detect functional motifs. Instead of generating a full sequence-structure alignment, the developed dynamic programming algorithm efficiently computes an optimal, non-crossing arrangement of matching substructures. I have shown that this longest common subsequence of exact pattern matchings is in good agreement with existing comparison approaches, but can be obtained in a fraction of runtime. In addition, I investigated a generally applicable approach to speedup ressource-expensive sequence-structure alignment methods by using `ExpaRNA`'s EPM set as anchor constraints. The evaluation showed that there is a trade-off between overall quality and speedup that can be controlled by the minimal size of EPMs used.

With `ExpaRNA-P`, I present in the second part of this thesis a method that generalizes exact matching substructures from fixed RNA secondary structures to energy-based RNA structure ensembles. The developed algorithm not only identifies ensemble-based EPMs in quadratic time, but it is moreover the first $O(n^2)$ time RNA comparison algorithm

that is based on full RNA structure ensembles. This major leap is achieved by novel in-loop probabilities in combination with a new structural sparsification scheme. The evaluation showed that these algorithmic improvements lead to a better quality of the identified EPMs according to reference alignments as well as to higher speedups when using ensemble-based EPMs as anchor constraints for full RNA alignments. Moreover, our method provides a generally applicable method that can also be adopted for other RNA comparison problems.

In the last part of this thesis, I present `GraphClust`, an efficient structure-based RNA clustering method for large RNA datasets. The core of the clustering procedure runs in linear time and, thus, eliminates the runtime bottleneck of previous approaches. The very high quality of the found clusters was shown in extensive benchmark studies on already annotated RNA sequences. `GraphClust` achieves a drastic runtime reduction from, for example, approximately 1 year to 36 hours for a dataset of 3900 RNA sequences. This major improvement is realized by an alignment-free clustering step using a fast graph kernel in combination with an efficient approximate nearest neighbour search query. The key for the fast clustering is an explicit representation of the kernel subgraphs that eventually allows to retrieve dense candidate clusters in linear time by using locality sensitive hashing. The feasibility of this approach was demonstrated with a software pipeline that integrates the clustering algorithm together with a structure-based refinement procedure to obtain RNA clusters of high quality. In a pilot study, large-scale datasets with up to 118,000 RNA sequences have been processed and several structural RNA clusters in, for example, human long intergenic non-coding RNAs and `RNAz` screens from fly and fish have been detected.

# Zusammenfassung

In der Molekularbiologie konnte im letzten Jahrzehnt dank stark weiter entwickelter Sequenziertechnologien ein tiefgreifendes Verständnis über Aufbau und Organisation von Genomen verschiedenster Organsimen erlangt werden. Dabei wurde überraschenderweise deutlich, dass zum Beispiel Menschen und andere höhere Organismen viel weniger Gene für Proteine enthalten als früher angenommen, aber dennoch ein großer Teil der Erbinformation in Form von DNA nach RNA transkribiert wird. Insbesondere sogenannte nicht-kodierende RNA (ncRNA), RNA Moleküle, die keine Vorlage für ein Protein bilden, stellen den Großteil der genomischen Transkripte dar. Es konnte gezeigt werden, dass diese ncRNAs, wie zum Beispiel die nur 21 Nukleotide umfassenden micro-RNAs, außerordentlich wichtige Faktoren für die Regulation von zellulären Prozessen darstellen.

Jedoch ist die Menge an identifizierten RNA-Molekülen so immens, dass bisher nur sehr wenigen Transkripten eine genaue Funktion zugewiesen werden konnte. Bioinformatische Methoden zur vergleichenden Analyse sind deshalb zwingend notwendig, um diesen riesigen Berg an Sequenzdaten zu durchforsten und einen Beitrag zu dessen funktioneller Analyse zu leisten. Die strukturellen Eigenschaften von ncRNAs erfordern komplexe Algorithmen, welche selbst die Analyse von kleinen Datensätzen mit bestehenden Ansätzen fast unmöglich macht. In meiner Arbeit stelle ich mich diesen Herausforderungen und entwickle effiziente bioinformatische Methoden, die RNA-Moleküle basierend auf deren Sequenz und Struktur vergleichen und analysieren können.

Im ersten Teil meiner Arbeit stelle ich `ExpaRNA` vor - ein Verfahren zum paarweisen Vergleich von RNAs, das auf dem Prinzip von exakt übereinstimmenden Teilstrukturen, sogenannten EPMs, basiert. Im Gegensatz zu bestehenden Methoden benutzt `ExpaRNA` evolutionär konservierte Teilstrukturen für den Vergleich und ist damit besser für die Erkennung von funktionellen Motiven geeignet. Der entwickelte dynamische Programmieralgorithmus berechnet kein vollständiges Alignment, sondern eine optimale Menge von Teilstrukturen, welche in beiden RNA-Sekundärstrukturen exakt enthalten und ähnlich angeordent sind. Ich habe gezeigt, dass die ausgewählten Teilstrukturen gut mit den Ergebnissen bisheriger Methoden übereinstimmen, jedoch in einem Bruchteil der Laufzeit berechnet werden können. Zusätzlich habe ich einen Ansatz untersucht, der das Ergebnis von `ExpaRNA` benutzt, um aufwendige Sequenz-Struktur-Alignment-Methoden

zu beschleunigen. Dabei werden gemeinsame Teilstrukturen als Ankerpunkte zur Berechnung eines vollständigen Alignments benutzt. Genauigkeit und Geschwindigkeitsvorteil können dabei über die minimale Größe der verwendeten Strukturen balanciert werden.

Mit `ExpaRNA-P` präsentiere ich im zweiten Teil ein Methode, die exakt übereinstimmende Teilstrukturen in RNAs von festen Sekundärstrukturen auf energiebasierte RNA-Strukturensembles erweitert. Der entwickelte Algorithmus findet nicht nur EPMs innerhalb quadratischer Laufzeit, er stellt auch den ersten RNA-Vergleichsalgorithmus überhaupt dar, der auf Grundlage von RNA-Strukturensembles eine Laufzeit von $O(n^2)$ erreicht. Dieser Fortschritt wird durch neue sogenannte „in-loop"-Wahrschein-lichkeiten auf dem Strukturmodell erreicht, so dass weniger Matrixeinträge berechnet werden müssen. Dieses Verfahren lässt sich auch auf allgemeinere RNA-Vergleichsmethoden anwenden. In der Auswertung konnte gezeigt werden, dass EPMs auf Basis von Strukturensembles eine höhere Qualität aufweisen. Wenn diese EPMs als Ankerpunkte für Alignments verwendet werden, kann dadurch eine noch höhere Beschleunigung erreicht werden.

Im letzten Teil dieser Arbeit präsentiere ich `GraphClust` - ein sehr effizientes Verfahren zum strukturbasierten Clustern von großen RNA-Datensätzen. Das entwickelte Verfahren stellt die erste brauchbare Lösung für dieses Problem dar, da der zentrale Algorithmus von `GraphClust` nur lineare Laufzeit benötigt und ohne Alignments auskommt. Dadurch kann hinsichtlich der Laufzeit der Flaschenhals früherer Ansätze überwunden werden und es ist erstmals möglich, große RNA-Datensätze auf Basis von Sequenz und Struktur zu clustern. In breiten Benchmarkstudien auf bereits annotierten RNA-Sequenzen konnte die hohe Qualität der gefundenen Cluster gezeigt werden. Im Vergleich zu bisherigen Verfahren wird der Vorteil von `GraphClust` besonders offensichtlich, da die Laufzeit von, zum Beispiel, einem Jahr auf 36 Stunden für einen Datensatz von 3900 RNA Sequenzen verringert werden konnte. Diese starke Verbesserung ist möglich durch die Verwendung eines schnellen Graphkernels in Kombination mit einem sehr effizienten Verfahren, welches die k-nächsten-Nachbarn einer RNA approximiert. RNA-Sekundärstrukturen werden dafür in Graphen umgewandelt und lokale RNA-Strukturelemente als Features kodiert. Der Schlüssel für das sehr schnelle Clusterverfahren liegt in der expliziten Repräsentation dieser Features, so dass anhand eines inversen Index eine Menge sehr ähnlicher RNAs in linearer Zeit berechnet werden kann. Die Umsetzbarkeit dieser Methode wird mit Hilfe einer kompletten Software-Pipeline zum Prozessieren von großen RNA-Datensätzen demonstriert. In einer Pilotstudie wurden damit mehrere Datensätze von bis zu 118.000 RNA Sequenzen bearbeitet und es konnte eine Reihe ähnlicher RNA Moleküle zum Beispiel in langer, intergenischer, nicht-kodierender RNA vom Menschen sowie in `RNAz`-Screens in Fruchtfliegen und Fischen gefunden werden.

# Danksagung

Diese Arbeit konnte nur entstehen, weil ich Unterstützung
von vielen Einzelnen erfahren habe.

An erster Stelle möchte ich mich ganz besonders bei meinem Doktorvater Rolf Backofen bedanken, der mir die Promotion ermöglicht hat und mit seiner ansteckenden Begeisterung mich immer wieder für dieses spannende Forschungsgebiet motiviert hat, für die Betreuung in den letzten Jahren und die erfolgreiche Zusammenarbeit mit den gemeinsamen Publikationen.

Ein großer Dank geht auch an Björn Voss, für sein Interesse am Inhalt meiner Arbeit und seine Bereitschaft für die Begutachtung.

Ja und natürlich geht ein großer Dank an *ALLE* Kollegen am Lehrstuhl, die den Arbeitsalltag durchweg sehr angenehm gemacht haben und immer hilfreich zur Seite standen. In Erinnerung bleiben viele Kochaktionen, Filmabende, Grillaktionen, Parties und natürlich unsere Kuchen-Dart-Kaffeepausen genauso wie erlebnisreiche Reisen. Danke an Christina Schmiedl und Fabrizio Costa fürs Korrekturlesen und unsere sehr gute Zusammenarbeit (hier insbesondere auch an Mathias Möhl, Dominic Rose und Sebastian Will). Ein besonderes Dankeschön geht an Andreas Richter für seine Korrekturen und als beständiger, immer hilfreicher, Zimmergenosse.

Wissenschaft lebt vom Austausch und Kooperationen, und dafür gab es in Freiburg sehr gute Möglichkeiten. Die Treffen in Bled und Tschechien bleiben unvergessen und dafür ein großes Dankeschön nach Wien und Leipzig. Hier möchte ich mich insbesondere bei Kristin Reiche, Jan Engelhardt und Jörg Hackermüller sowie den RNA-Faltern aus Wien bedanken. Genauso geht großer Dank an Nuno Mendes und Marie-France Sagot aus Lyon sowie Mika Amit von der Universität Haifa.

Herzlichen Dank auch an meine Eltern und meine Familie, die mich immer unterstützt hat und so manche stressige Zeit ertragen musste. Danke Annkatrin, danke Johanna, danke Pauline!

# Own Publications

## This thesis is based on the following publications:

1. **Steffen Heyne***, Fabrizio Costa*, Dominic Rose, and Rolf Backofen. GraphClust: alignment-free structural clustering of local RNA secondary structures. *Bioinformatics*, 28(12):i224-i232, 2012.

2. Christina Schmiedl*, Mathias Möhl*, **Steffen Heyne***, Mika Amit, Gad M. Landau, Sebastian Will, and Rolf Backofen. Exact pattern matching for RNA structure ensembles. In *Proceedings of the 16th International Conference on Research in Computational Molecular Biology (RECOMB 2012)*, volume 7262 of LNCS, pages 245-260. Springer-Verlag, 2012.

3. Cameron Smith *, **Steffen Heyne***, Andreas S. Richter*, Sebastian Will*, and Rolf Backofen. Freiburg RNA Tools: a web server integrating IntaRNA, ExpaRNA and LocARNA. *Nucleic Acids Research*, 38(Web Server Issue):W373-7, 2010.

4. **Steffen Heyne**, Sebastian Will, Michael Beckstette, and Rolf Backofen. Lightweight comparison of RNAs based on exact sequence-structure matches. *Bioinformatics*, 25(16):2095-2102, 2009.

5. **Steffen Heyne**, Sebastian Will, Michael Beckstette, and Rolf Backofen. Lightweight comparisqon of RNAs based on exact sequence-structure matches. In *Proceedings of the German Conference on Bioinformatics (GCB'2008)*, volume P-136 of *Lecture Notes in Informatics (LNI)*, pages 189-198. Gesellschaft für Informatik (GI), 2008.

*Joint first authors

## Further publications:

- Kelly Leon, Thomas Boulo, Astrid Musnier, Julia Morales, Christophe Gauthier, Laurence Dupuy, **Steffen Heyne**, Rolf Backofen, Anne Poupon, Patrick Cormier, Eric Reiter, and Pascale Crepieux. Activation of a GPCR leads to eIF4G phosphorylation at the 5' cap and to IRES-dependent translation. *Journal of Molecular Endocrinology*, 52(3):373-82, 2014.

- Sebastian Will*, Michael F. Siebauer*, **Steffen Heyne**, Jan Engelhardt, Peter F. Stadler, Kristin Reiche, and Rolf Backofen. LocARNAscan: Incorporating thermodynamic stability in sequence and structure-based RNA homology search. *Algorithms for Molecular Biology*, 8(1):14, 2013.

- Nuno D. Mendes, **Steffen Heyne**, Ana T. Freitas, Marie-France Sagot, and Rolf Backofen. Navigating the unexplored seascape of pre-miRNA candidates in single-genome approaches. *Bioinformatics*, 28(23):3034-41, 2012.

- Mika Amit, Rolf Backofen, **Steffen Heyne**, Gad M. Landau, Mathias Möhl, Christina Schmiedl, and Sebastian Will. Local exact pattern matching for non-fixed RNA structures. In *Proceedings of the 23th Annual Symposium on Combinatorial Pattern Matching (CPM 2012)*, volume 7354 of LNCS, pages 306-320. Springer-Verlag, 2012.

(until June 2014)

# Contents

# Chapter 1

# Introduction

When James D. Watson and Francis H. C. Crick put in 1953 all pieces of scientific discoveries in the right place and presented the first correct description of the complex, double-helical structure of DNA, it marked only the beginning of a new era in understanding the foundations of life [175]. It was also around this time when Crick wrote for the first time his central dogma of molecular biology which states that information passes only from DNA to proteins via RNA [35]. Although not being wrong, his statement influenced molecular biology and put the focus of research for many years predominantly on the protein universe. The popular assumption was that biological complexity correlates with the number of protein-coding genes. RNA in contrast was considered as passive molecule being only a piece in the machinery that converts DNA into proteins. In the meantime, many rather surprising lessons have been learned and RNA molecules seem to be the dark matter orchestrating the development of complex organisms [160]. It is even an established opinion that all life on earth evolved from a primordial RNA world to the modern RNA world we recognize today [26].

When the first draft of the human genome sequence was presented in 2001, it became clear that the initial estimates of up to 150,000 humans genes had to be corrected to only 25,000 genes or even less [45, 46]. In the following years, emerging sequencing technologies sparked an explosion of large-scale genome sequencing projects for many higher eukaryotes like for mouse and chimpanzee, but also for yeast and bacteria. Different evidence from tiling arrays and high-throughput sequencing revealed basically in all higher eukaryotes a much more complex transcriptome together with a much larger amount of DNA transcribed into RNA than one could explain by protein-coding genes alone. For example, in the human genome about 85% of the genome is transcribed into RNA, while less than 1.5% is contained in protein-coding exons [46]. This large amount of non-coding RNA is probably the key to finally explain biological complexity, or more philosophical: what makes us human? For example, the number of genes in the roundworm *Caenorhabditis elegans* is about the same as found in human or mice. In contrast, the ratio between

coding and non-coding RNA sequences is much higher in humans [160]. The discovery of small interfering RNAs (siRNAs) and micro RNAs (miRNAs) around the turn of the millennium mark in retrospective the dawn of the ongoing effort to understand this hidden layer of cellular regulation [52, 98]. Numerous studies revealed that non-coding RNAs are an essential part of nearly all regulatory processes in all kingdoms of life. They are involved, for example, in cell development, gene activation or silencing and chromatin remodelling [87]. There is also clear evidence that many ncRNA are linked to complex diseases [48].

### Subject of this thesis

Even though RNAs are a vital part of current research in molecular biology, the key question remains: what are the precise functions of this immense bulk of RNA transcripts? Experimental methods are often costly and time-consuming, and hence, computational methods are necessary. Bioinformatics is already an integral part of RNA research and the sheer amount of data makes it also mandatory to keep step with the ever increasing rate of novel sequence data. For example, recent computational screens suggest more than 30,000 conserved ncRNAs only in humans [128, 169]. Several studies for genome-wide transcriptomics show that a broad RNA transcriptome is a widespread feature not only of humans and other mammals, but also of fish, invertebrates and bacteria [45, 46, 104, 109, 120]. However, a precise functional annotation of the majority of RNA transcripts remain elusive in many cases and the fraction of ncRNA candidates lacking a reliable class assignment is rapidly expanding [115]. Computational methods complementing a detailed functional analysis of ncRNAs are requested, but constitute a challenging task. The reason is that conserved, and thus functional, signals are traceable mostly by structural properties of the RNAs whereas their sequence alone is often not sufficient. On the other hand, structure-aware approaches immensely increase the computational complexity and runtime of comparative algorithms, which makes even small-sized comparison studies infeasible [58, 173]. A further complicating fact is that ncRNAs are overall very inhomogeneous with vastly different structures, functions, and evolutionary patterns, for example ranging from short micro RNAs (miRNA) to long intergenic non-coding RNAs (lincRNAs) [17].

Therefore, the subject of this thesis are efficient computational methods for the comparative analysis of RNAs based on their sequence and structure. More precisely, I present a fast pairwise RNA comparison method based on exact matching substructures that can also speedup complex sequence-structure alignment algorithms. The second method describes an efficient solution to identify exact local motifs in complex RNA structural ensembles. Finally, I present the first feasible solution for large-scale, structure-based

clustering of RNA sequences. This method can also be used for the purpose of *de-novo* identification and prediction of functional RNA elements. A pilot study revealed conserved structural RNA elements in the human genome and elsewhere.

### Thesis outline

The following introduction gives an overview on important biological functions of RNAs and major classes of RNA molecules. Then, I introduce general concepts in bioinformatics necessary for modelling RNA structure and RNA folding. Finally, I summarize existing approaches for RNA comparison, clustering and ncRNA gene finding. In the second chapter, I present `ExpaRNA`, a fast tool for RNA comparison based on exact matching substructures. The third chapter of this thesis presents `ExpaRNA-P`, a fast RNA comparison approach which generalizes the concept of exact matchings to RNA structure ensembles. In the fourth chapter, I present `GraphClust`, a very efficient method for large-scale structure-based clustering of RNA sequences. A summary of this thesis and ideas for future work are presented in the final chapter.

Bioinformatics is an interdisciplinary research area that is nowadays designated by collaborations. The work presented in thesis results from close collaboration with other scientists, who contributed with ideas, discussions or implementations. Therefore, I use first person plural instead of first person singular in my thesis. The presented work is based on publications which already appeared in international journals or as conference proceedings [70–72, 150, 157]. Figures, tables and text passages from these publications are used in the thesis without further notice.

## 1.1 RNA in Biological Systems

In all living organisms, RNAs are predominantly synthesized by the transcription of the DNA sequence stored in specific genomic loci.[1] For many years, it was assumed that the majority of these transcripts are protein-coding messenger RNAs (mRNAs), which are translated into an amino acid sequence, i.e proteins. This protein-centric view led to the discovery of only a small number of non-coding RNAs (ncRNAs), mainly involved in basic cellular processes like translation. Among these infrastructural RNAs are tRNAs that transfer a single amino acid to the ribosome in order to elongate a protein sequence. Many ncRNAs are also often found in complexes where they act together with proteins. The ribosome itself is, for example, a large arrangement of ribosomal RNAs (rRNAs) and proteins. The discovery of catalytic active RNAs (ribozymes) like self-splicing introns or RNaseP recovered RNAs from being passive and not worth to be studied cellular players [26]. Other common RNAs comprise the small nuclear RNAs

[1] Although not considered as living organisms, retroviruses store their genome directly as RNA.

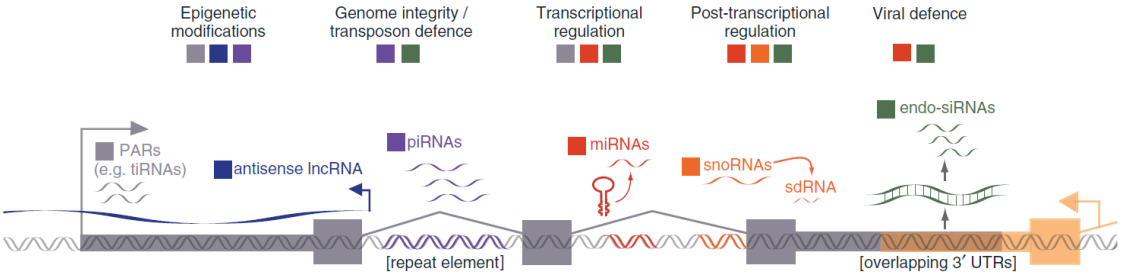


**Figure 1.1. Simplified representation of regulatory ncRNAs and their functions.** A genomic locus can be origin or target of several ncRNA species with different functional roles. Generalized gene models are presented in dark grey and light orange and overlap the double-strand DNA structure (light grey). Classes of regulatory RNAs and their functions are defined by a colour. PARs, promoter-associated RNAs; lncRNAs, long non-coding RNAs; miRNAs, micro RNAs; snoRNAs, small nucleolar RNAs; sdRNAs, sno-derived RNAs; endo-siRNAs, endogenous siRNAs; piRNAs, PIWI-interacting RNAs; tiRNAs, transcription initiation RNAs. The figure is adapted from Taft et al. [160] with kind permission of John Wiley & Sons (License Number: 3610150730390).

(snRNAs) U1-U6, which are part of the eukaryotic spliceosome, and snoRNAs, which are involved in the post-transcriptional modification of rRNAs. The 7SL ncRNA is part of the signal recognition particle (SRP), a complex that is important for correct protein localization [115].

With the discovery of RNA interference (RNAi) in eukaryotes, the fundamental role of RNAs for regulatory processes like gene silencing became evident [34]. RNAi-related pathways result in about 22nt long RNA molecules derived either from a hairpin or double-stranded precursors. These short interfering RNAs (siRNAs) along with micro RNAs (miRNAs) share a similar pathway to form RNA-RNA interactions with mRNAs, which usually lead to gene silencing. Many studies showed that miRNAs play a central role in the regulation of developmental processes, cell proliferation and apoptosis [160]. Among further short RNA classes of the steadily expanding RNA universe, PIWI-interacting RNAs (piRNAs) are, for example, responsible for the silencing of transposons in the germ line of animals [156].

The class of long non-coding RNAs (lncRNAs) includes a broad variety of transcripts with no clear classification. Most of them are localized in the nucleous, show low expression levels, some have a poly-A signal or get spliced. Their location with respect to protein coding genes can be sense, antisense, intronic, intergenic or bidirectional [87]. Long-intergenic RNAs (lincRNAs) form a subgroup with a specific chromatin signature and are, for example, linked to epigenetic regulation by interacting with chromatin remodelling complexes [87]. A comprehensive overview of known ncRNA classes is given in Bompfünewerer Consortium et al. [17] and Taft et al. [160]. Figure 1.1 gives an overview on eukaryotic regulatory RNAs and their functional classification.

Prokaryotes also contain a large number of small regulatory RNAs (sRNAs) involved in the regulation of many critical processes. Most of them act by base pairing to mRNAs at the ribosome binding site and thus lead to a downregulation of the encoded protein. However, activation of translation is also reported [61]. Prokaryotic sRNAs are usually between $50-500nt$ long, whereas in eukaryotes arbitrarily everything $< 200nt$ is regarded as short RNA [88].

Another type of regulatory RNAs are *cis*-acting elements embedded in untranslated regions (UTRs) of mRNAs [115]. These elements often form a local secondary structure that acts as binding platform for *trans*-acting signals like RNA binding proteins or metabolites as, for example, in case of riboswitches. A well-studied example of *cis*-acting elements is the iron-responsive element (IRE) found in ferritin mRNAs. The IRE folds into a small stem-loop structure and is bound by IRE-binding proteins (IRE-BPs) in low iron conditions. If iron level increases, the IRE-BP changes its folding and gets released from the IRE. This triggers the translation of ferritin, an iron storage component [2]. Other *cis*-regulatory RNAs are, for example, SECIS elements, responsible for the insertion of selenocystein into proteins in response to UGA codons, or localization elements [111, 161].

## 1.2 RNA Structure

An RNA molecule is an hetero-polymer composed of the four nucleotide monomers adenine (A), cytosine (C), guanine (G) and uracil (U). Each nucleotide consists of a phosphate, a ribose ring, phosphate groups and the corresponding nucleobase. Adjacent nucleotides are linked together by phosphodiester bonds between the third and fifth carbon atom of the ribose. This asymmetric linkage of nucleotides induces a direction which is always specified from 5' to 3'. The sequence of nucleotides denotes the primary structure; Watson-Crick pairs and other non-crossing pairs are called secondary structure and all other contacts and pairs like pseudoknots are subsumed as tertiary structure.

Most RNA monomers fold into a secondary structure formed by intramolecular hydrogen bonds between complementary bases according to standard Watson-Crick pairs (A-U and G-C) and non-standard pairs like G-U. Although all possible kinds of base pairs occur in nature, these three types contribute most to the free energy of an RNA molecule [100]. Furthermore, RNA folding is a hierarchical kinetic process that is mainly influenced by these base pairs. Once small helices are built, additional contacts and pairs like pseudoknots are made in order to form the three-dimensional structure [20]. Figure 1.2 shows different illustrations of an RNA secondary structure.

### 1.2.1 Formal Abstraction of RNA Structures

We describe an RNA structure as combination of its primary and secondary structure and choose the notion of arc-annotated sequences [49, 50]. We define an arc-annotated sequence as follows:

**Definition 1.2.1** (arc-annotated sequence)**.** *An arc-annotated sequence is a tupel* $(S, P)$*, where* $S$ *is a string over the alphabet* $\{A, C, G, U\}$*. The nucleotide at the* $i$*-th position of* $S$ *is denoted by* $S_i$*, the subsequence from position* $i$ *to* $j$ *by* $S[i\,..\,j]$ *and its length by* $|S|$*. The secondary structure* $P$ *is a set of arcs* $(i, j)$ *such that* $1 \leq i < j \leq |S|$*. Furthermore, we require that each sequence position is involved in at most one base pair, i.e. for all* $(i, j), (i', j') \in P$*:* $i = i' \Leftrightarrow j = j'$ *and* $i \neq j'$*. An arc* $(i, j) \in P$ *is called* crossing *if there exists an arc* $(i', j') \in P$ *with* $i < i' < j < j'$*. An arc-annotated sequence is* crossing *if it contains a crossing arc, otherwise we call it* non-crossing *or* nested *.*

The set of arcs is usually restricted to complementary Watson-Crick base pairs (A-U or G-C) and the non-standard G-U base pair. In order to discriminate two RNAs $A$ and $B$, we simply write $(A, P_A)$ and $(B, P_B)$. A sequence position is indicated with $A_i$ or $B_i$.

Natural RNA structures like those of transfer-messenger RNAs (tmRNAs) or RNase P often form so-called pseudo-knot motifs that contain crossing base pairs. However, crossing arcs are usually ignored because the computational complexity for problems like structure prediction as well as structure comparison becomes NP-hard, and thus, infeasable for RNAs of reasonable length [50, 107]. In order to circumvent the NP-hardness for structure prediction, tools like `KnotSeeker` and `IPknot` use heuristics, while other approaches restrict the type of possible pseudoknots [136, 149, 159]. Most of these approaches are, however, limited to small-scale analysis tasks. In the course of this thesis, we hence consider only non-crossing structures.

### 1.2.2 RNA Motifs

There exists no general definition of an "RNA motif", although there is an intuitive idea of a recurrent part of an RNA behind it. Motifs are often linked to special functions, either directly or indirectly. Known examples comprise short RNA sequences responsible for RNA-protein interactions or for sensing of metabolites or temperature in riboswitches [19, 32]. RNA motifs, also denoted as RNA modules, that are necessary for the compact folding of complex RNAs were in detail analyzed by Eric Westhof and colleagues, who state that "*RNA motifs are directed and ordered stacked arrays of non-Watson-Crick base pairs forming distinctive foldings of the phosphodiester backbones of the interacting RNA strands*" [101]. Recurrent classes of such motifs are tetraloops in hairpins like the GNRA or UNGC tetraloops, loop E motif, bulged G motif, kink-turn motif and the C-motif [177]. As these motifs include very special non-Watson-Crick base pairs, they are usually missed

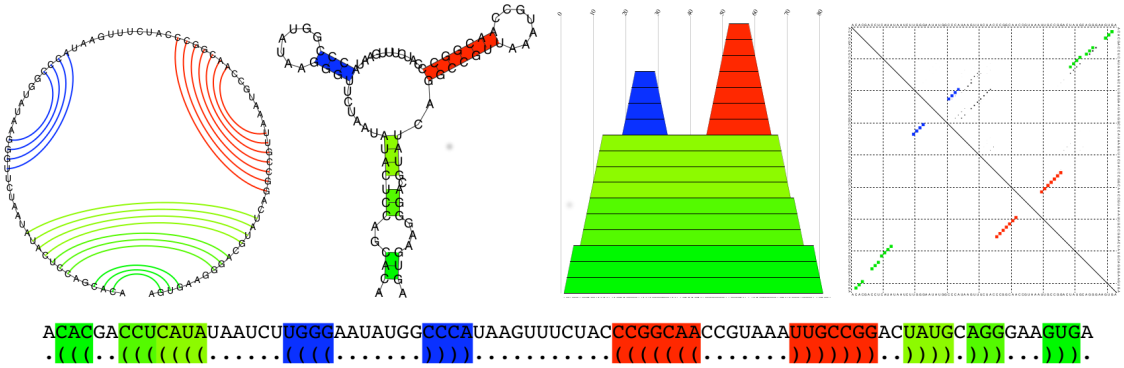


**Figure 1.2. Different representations of RNA secondary structure.** From left to right: Circle plot, conventional secondary structure graph, mountain plot and dot plot. Corresponding colors in each plot indicate the same base pairs. The diagram at the bottom shows the secondary structure in dot-bracket notation, where each base pair corresponds to a pair of matching parentheses. The structure shown is the purine riboswitch (Rfam accession number RF00167). The figure is reused from Hofacker and Stadler [75] with kind permission of John Wiley & Sons (License Number: 3632720643620).

in structure prediction methods. An exception are, for example, certain tetraloops with experimentally measured energies that can be predicted by `RNAfold`. Some of these motifs have special patterns in the secondary structure, which allows to find them indirectly for example in structure based RNA alignments.

In this thesis, we use the term RNA motifs to denote distinct but recurrent secondary structure patterns that can be utilized to identify related RNAs. However, we do not link motifs to specific functions as they can occur anywhere in the structure. Accordingly, an RNA motif could be any substructure like a hairpin, an internal loop, two stacking base pairs or a complete multiloop. The most important characteristic of known RNA motifs seems to be their structural locality, which we incorporated in the developed methods to the best of our knowledge.

## 1.3 RNA Structure Prediction

Similar to proteins, there is a close connection between structure and function for an RNA molecule. A good structural model is therefore crucial to determine the function of an RNA. As it is a very tedious task to determine a complete 3D model experimentally, computational methods are often used to predict structure while focusing on the secondary structure only [174].

RNA secondary structures can be decomposed into loops. The resulting structural elements are distinguished by their composition of base pairs and unpaired bases. We discriminate the following elements: hairpin loop, interior loop, bulge loop, multi-branch

loop, exterior loop and stacked base pairs (see Figure 1.3a). The assumption behind this decomposition is that the free energy contribution is additive and independent of other building blocks. We call this the nearest neighbor or loop-based energy model (see Figure 1.3b). Although this concept lacks some parameters of biological relevant RNAs, it enables the usage of dynamic programming (DP) algorithms for RNA structure prediction. The free energy contributions of single loops can be determined experimentally; a set of widely used parameters was generated by the Turner lab [113]. In general, stacking base pairs have a stabilizing energy contribution while long unpaired loops are destabilizing.

One of the first approaches to solve the RNA folding problem was the Nussinov algorithm [124]. This method finds the RNA secondary structure with the maximum number of base pairs in $O(n^3)$ time and $O(n^2)$ space. The utilized decomposition is simplified in Figure 1.3c. By incorporating the loop-based energy model, one can predict for a given RNA sequence the structure with the minimum free energy (MFE). The Zuker algorithm is the most common variant for this task [193]. Loop energies are either tabulated or modelled linearly to the size of the loop. When the size of internal loops is bound by some constant, the Zuker algorithm runs in $O(n^3)$ time; otherwise it needs $O(n^4)$ time. Tools like `RNAfold` from the Vienna RNA package or `RNAstructure` are well-known implementations of this algorithm [76, 105, 133]. Lyngso et al. [108] proposed an $O(n^3)$ time algorithm for unbound internal loop sizes. There is also a wide range of algorithms that are able to predict structures with pseudoknots. For more details see the recent review by Washietl et al. [173].

The accuracy of energy-based models is compromised by several issues as neglecting all tertiary interactions or too many loop combinations for experimental parameter measurement. It was shown that about a third of the predicted base pairs are wrong and a similar fraction is missed completely [40, 113, 174]. In contrast to energy-based models, only 20% of the base pairs can be correctly predicted by using Nussinov-style algorithms. Prediction accuracy can be enhanced by predicting canonical RNA structures, i.e. by omitting unstacked, "lonely" base pairs [16]. Further improvements can be achieved by integrating folding constraints found by structural probing experiments. Such external knowledge helps to restrict the folding space in tools like `RNAfold`. Structural information obtained by sequencing techniques like SHAPE can be integrated via pseudo energies [37, 172].

Probabilistic models constitute an alternative to energy-based models. Folding parameters can be estimated from a set of verified RNA structures and inferred onto RNA sequences with an unknown structure. Stochastic context free grammars (SCFG) can be used for this task, although actually better suited for RNA homology search [43].

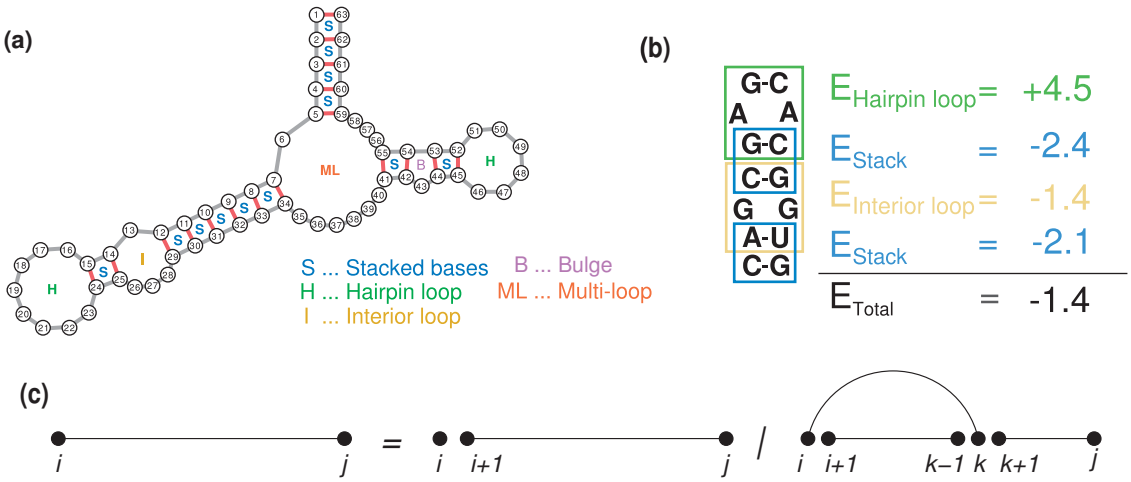


**Figure 1.3. Principles of RNA structure prediction.** (a) A secondary structure can be uniquely decomposed into basic elements (e.g. stacked bases) that are independent from each other. (b) Example of energy evaluation of a small RNA structure. Thermodynamic folding algorithms assign free energies to the structural elements. In the example shown, two stacks and a symmetric interior loop stabilize the structure (negative free energy), while the hairpin loop destabilizes the structure (positive free energy). The total free energy of the structure is the sum of the energy of all its structural elements. (c) The dynamic programming principle allows for efficient folding algorithms. In order to find the minimum free energy between the positions $i \dots j$, we use the solution for $i+1 \dots j$ and either add an unpaired base or a base pair $(i, k)$. Here $k$ divides the problem into smaller subproblems which can be recursively solved. The figure is adapted from Washietl et al. [173] with kind permission of John Wiley & Sons (License Number: 3610141326722).

`CONTRAFOLD` is an interesting approach that augments SCFGs by using conditional random fields [38]. It can score all types of possible base pairs and advanced scoring schemes can be easily integrated. However, it is not suited to predict MFE structures.

### 1.3.1 Structure Ensemble

At physiological relevant temperatures, an RNA sequence is not fixed into one structure but it can form an ensemble of different structures. The fact that many stabilizing effects are ignored in the energy model leads to the problem that a predicted MFE structure is often different to the structure with biological function. The analysis of suboptimal structures is, however, difficult because the number of structures increase exponentially with the length of the sequence [77]. Using the partition function for Boltzmann-weighted ensembles instead allows for a computational tractable method to analyze the RNA structure space [117]. The underlying assumption is that a thermodynamic system in its equilibrium occupies states with a probability depending on the energy of the state. Adopted to the RNA world, a state is a structure $P$ from the set of all possible secondary

structures $\mathcal{P}$ that can be formed by a sequence $S$. The energy at a temperature $T$ of a structure $P$ is denoted as $E(P)$. Now we can use the Boltzmann distribution to define the probability of a structure $P$ of sequence $S$ as

$$\Pr\{P|S\} = \frac{e^{-\beta E(P)}}{Z} \tag{1.1}$$

where $\beta = (\mathsf{k}_B T)^{-1}$ is the inverse temperature, $Z$ the partition function and $\mathsf{k}_B$ is the Boltzmann constant. Basically, the probability of a structure $P$ is proportional to its Boltzmann factor $e^{-\beta E(P)}$ and gets normalized by $Z$. As all probabilities add up to one, we can define the partition function $Z$ as

$$Z = \sum_{P \in \mathcal{P}} e^{-\beta E(P)}. \tag{1.2}$$

In addition to the probabilities of single structures, we can also compute the probabilities of smaller structural features like base pairs. To this end, we sum up all Boltzmann weights of structures that contain a base pair $p = (i, j)$ and divide this term by $Z$, i.e.

$$\Pr\{(i,j)|S\} = \frac{Z_{ij}}{Z} = \frac{\sum_{(i,j)\in\mathcal{P}} e^{-\beta E(P)}}{Z}. \tag{1.3}$$

Fortunately, the partition function can also be computed in $O(n^3)$ time complexity as folding into the MFE structure, although with a different constant factor. McCaskill [117] formulated a DP algorithm to compute $Z$ and its basic recursion scheme is related to the Zuker algorithm. With the McCaskill algorithm, one can for example compute the probability matrix of all possible base pairs $(i, j)$ and obtain a very compact and yet powerful way to abstract the folding space of an RNA sequence. These "RNA dot plots" (see Figure 1.2) are widely used and permit powerful structure-based algorithms. The Vienna RNA package contains a widely used implementation of McCaskill's algorithm, which is used by tools like `RNAfold` [105].

The `RNAShapes` approach for structure prediction is also based on the partition function and can efficiently group similar foldings into different levels of abstraction, called shapes [57]. Each shape class is represented by the structure of minimal free energy within the shape and thus allows to easily identify meaningful suboptimal foldings of a single RNA sequence.

### 1.3.2 Comparative Structure Prediction

Using a single RNA sequence for structure prediction is usually not sufficient to obtain a clear picture of its structure, not to mention its function. Homologous sequences can improve structure prediction by exploiting RNA specific substitution patterns particularly found in structured regions. Similar to proteins, related RNAs preserve structural fea-

tures, while the underlying nucleotide sequences can diverge more easily. In the secondary structure this can be observed by either consistent or compensatory mutations on base pairs, for example a change from A-U to G-U base pair or a full swap like from A-U to G-C. For large RNA sets, the mutual information content can identify highly correlated columns. Another strategy for small datasets is employed by tools like `RNAalifold` [12], which try to fold a given multiple sequence alignment with a variant of the Zuker algorithm. A consensus structure is inferred by averaging the individual energy contributions and adding a special reward for observed covariance. An alternative approach is, for example, provided by `PETfold`, which combines evolutionary and thermodynamic information to derive a consensus structure [151].

The aforementioned approaches rely on a given multiple alignment, usually obtained by sequence alignment tools like `MAFFT` [90]. Especially for RNAs, this is adverse because the named mutation patterns are ignored by sequence alignments algorithms. It has been shown that sequence alignment methods fail for RNA sequences below 60% sequence identity even if they are related. Therefore, structure-aware RNA alignment methods are required and the broad range of existing approaches can be discriminated by how structural information is incorporated. Tools like `MARNA` [154] and `RNAforester` [73] first predict an RNA structure for each sequence and try to align them afterwards. As these tools use mainly the MFE structure as input, their accuracy is also clearly limited.

The assumed "gold standard" is the simultaneous alignment and folding approach as introduced by David Sankoff [147]. Here a common structure is inferred during the computation of the alignment. Tools like `Dynalign` [114] and `FoldAlign` [67] use an energy model and implement the Sankoff method with heuristics as the original algorithm runs in $O(n^6)$ time. More recent advances use a sparse version of the structure ensemble based on base pair probability dot plots, which allows for less expensive $O(n^4)$ time algorithms, for example implemented in `LocARNA` [179] and `FoldAlignM` [163]. The sequence-structure alignment tool `Lara` uses integer linear programming to compute near-optimal solutions by using methods from combinatorial optimization [10]. There are also SCFG-based tools like `Stemloc` [79] that follow the Sankoff framework.

## 1.4 RNA Gene Finding and Annotation

The identification of functional RNAs or RNA genes is a difficult task and remains a key problem especially with the evidence of pervasive transcription for example in mammals [30]. Comparative methods are superior to single sequence methods because specific conservation patterns can be used to distinguish functional regions from non-functional ones. Obviously, *de-novo* RNA gene finding relies on structure prediction methods as outlined above and hence their performance. Furthermore, it has been shown that RNA structure stability alone is not sufficient to identify RNA genes [137]. The problem is

further complicated by the fact that functional RNA elements can be embedded in longer RNA transcripts and can only be detected by employing local search strategies. Such examples comprise *cis*-acting RNAs like SECIS elements or riboswitches. There is also evidence that long non-coding RNAs contain independent functional units that might act in *cis* or *trans* [165].

One of the first tools for *de-novo* ncRNA prediction is `QRNA`, which uses probabilistic models and was successfully used to screen pairwise `BLASTn` alignments of bacteria for novel ncRNAs [138, 139]. More recent approaches often use multi-species alignments of whole genomes (e.g. from `MULTIZ` [14]) as input and employ a sliding-window approach to assess the "gene" potential of a homologous region. `RNAz` for example uses a SVM trained on both sequence and structure conservation to classify ncRNA candidates [170]. The probabilistic approach `EvoFold` combines phylogenetic information with a SCFG for structure prediction [128]. Although many (potential) ncRNAs could be, for example, identified in human [171], the employed sequence alignments limit the capacity of such approaches. By adopting a structure-based whole genome realignment beforehand, `RNAz` is able to predict many new ncRNA candidates [183].

Structure-based RNA clustering can be used to discover new RNA classes out of unaligned sequences. This scheme was introduced with the `LocARNA`-based `RNAclust` pipeline [179]. First, a cluster tree is generated from the pairwise distance matrix and a potential new ncRNA class is then identified by evaluating all subtree alignments with tools like `RNAalifold` or `RNAsoup` [86]. However, the required, but expensive, all-against-all pairwise sequence-structure alignment step limits this approach to datasets of some thousand RNA sequences. Other tools like `EvoFam` employs a similar clustering scheme based on `EvoFold` hits, although entirely based on profile SCFGs [126]. The tool `RNApromo` uses a special EM (expectation maximization) algorithm for SCFGs to train models based on unaligned local secondary structures [131]. `CMfinder` also starts from unaligned sequences and successfully identified novel ncRNAs in bacteria [176, 188, 189]. It was one of the first tools for RNAs that used an EM method on SCFGs to improve initial family models.

RNA gene annotation often refers to the identification of known RNA genes or elements. For this task, the `Infernal` suite uses covariance models to describe known RNA families [42, 43]. It contains constantly improved tools like `CMsearch`, which efficiently finds family members in genomes or databases [94, 121, 122]. In addition, there exist several tools which can identify RNAs of specific RNA families: for example `tRNAscan-SE` for tRNAs [106], `RNAmmer` for ribosomal RNAs [97], and `SnoReport` for snoRNAs [69].

### RNA Databases

There exists a large variety of ncRNA-related databases [51] motivated by the fact that general sequence databases like Ensembl or GenBank often lack appropriate ncRNA specific annotations (e.g. secondary structure). For structured RNAs, `Rfam` is probably the most comprehensive collection and its current release 11.0 lists 2208 families [23]. It comprises many infrastructural RNA families, miRNA, snoRNAs as well as *cis*-regulatory elements. Each family is build upon a manually curated covariance model which can be used to identify new family members. Benchmark data sets like `BRAliBase` are also build upon `Rfam` in order assess different RNA alignment methods [54, 184]. A comprehensive database for micro RNAs and their target sites is miRBase [95]. In addition, there exists specialized databases like for RNase P [22].

# Chapter 2

# ExpaRNA: Fast RNA Comparison on Fixed Structures

In this Chapter we introduce the ExpaRNA approach, a new motif-based method for fast pairwise RNA comparison. The advantage over existing approaches is that ExpaRNA uses exact matching substructures to identify similarities common to two RNA molecules. Furthermore we show how the predicted motifs can be used to speed-up Sankoff-style alignment algorithms. We have evaluated ExpaRNA's performance on a reliable benchmark set. Finally we present a web server that allows to use ExpaRNA via an easy to use web interface. This chapter is based on the publications Heyne et al. [70, 71] and Smith et al. [157].

## 2.1 The ExpaRNA Approach

The multitude of discovered RNAs lacking precise functional annotations is rapidly expanding. One major reason of so many unannotated ncRNAs is that in contrast to protein-coding genes, ncRNAs belong to a diverse array of classes with vastly different structures, functions, and evolutionary patterns [17]. In addition, genes of non-coding RNA often have no discernible homology at the sequence level but still share common structural and functional properties.

Likewise to proteins, specific functions of ncRNAs are often associated with evolutionary conserved motifs that contain specific sequence and structure properties. Examples for such regulatory RNA elements, whose functions are mediated by sequence-structure motifs, are selenocysteine insertion sequence (SECIS) elements [82] (see Figure 2.1 for an example) which occur in the 3' untranslated (UTR) region of mammalian messenger RNAs. They facilitate the integration of selenocysteines, the 21st amino acid, in the peptide chain of proteins. Further examples of regulatory RNA elements are iron-responsive

elements (IRE) [68], different riboswitches [152], or internal ribosomal entry sites (IRES) [112]. IRES elements occur often in viral genomes where they allow the translation of the virus' RNA in a cap-independent manner in the host cell.

Although the detection of similar structural motifs in different RNAs is an important aspect for function determination, it is often negligibly handled in pairwise RNA comparison methods. Functionally important and common substructures are not necessarily preserved in the alignment and these methods are often very time consuming [84, 85].

State-of-the-art approaches dealing with pairwise RNA comparison can be distinguished by the given structural information and their representation. With the `ExpaRNA` approach presented in this chapter we follow the line of research that a nested RNA secondary structure is given (or predicted) for each RNA. The more advanced problem of starting with the full RNA structural ensemble is solved with the successor `ExpaRNA-P` presented in chapter 3.

Existing alignment-based comparison approaches employ the computation of edit distances between given RNA secondary structures [8, 85]. Generally speaking, the comparison of arc-annotated sequences boils down to find the longest arc-preserving common subsequence (LAPCS) [50]. However, even for two nested RNA secondary structures the problem remains NP-hard [15, 102]. Contrarily, RNA molecules are in general three-dimensional structures with complex base pairing interactions and they often contain crossing base pairs like pseudoknots. However, most comparison methods neglect them for the sake of described algorithmic complexity and applicability in practice. Fortunately, these limitations still allow to describe sequence-structure motifs with nested RNA secondary structures in a biological relevant way, as shown in Figure 2.1. Moreover, the comparison of nested RNA secondary structures can be lowered to polynomial-time algorithms with some restrictions to the scoring scheme [85].

Alternatively, the nested secondary structure can be represented as a tree. In this case comparison methods exist for the edit distance between two ordered labeled trees [190] as well as for the alignment of trees [84]. An improved version of the tree alignment method with extension to global and local forest alignments is implemented in the program `RNAforester` [73]. The MIGAL approach extends the tree edit distance model by two new tree edit operations and is especially efficient due to its usage of different abstraction layers [3]. General drawbacks of tree alignment methods are their lack of scoring arbitrary alignments and possible differences between edit distance and alignment distance.

With `ExpaRNA` we present a new lightweight, motif-based method for pairwise comparison of RNA molecules. Instead of computing a full sequence-structure alignment, our approach efficiently computes a significant arrangement of sequence-structure motifs, common to two RNAs. The complete, but overlapping set of exact common substructures for two RNAs of lengths $n$ and $m$ is determined by a fast $O(nm)$ time and space algo-

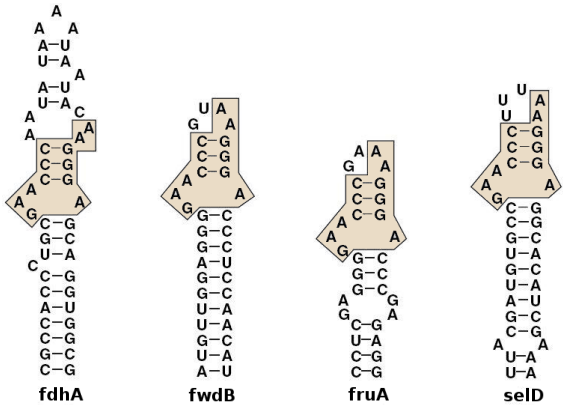


**Figure 2.1.** Putative SECIS elements in non-coding regions of *Methanococcus jannaschii* (according to Wilting et al. [185]). The highlighted substructure represents a common local motif, i.e. an exact matching substructure, of the shown RNA elements.

rithm as pre-processing step [5, 155]. `ExpaRNA` makes use of these common substructures and computes the longest collinear, non-overlapping sequence of substructures common to two RNAs in $O(H \cdot nm)$ time and $O(nm)$ space, where $H \ll n \cdot m$ for real RNA structures.

We successfully show the performance of `ExpaRNA` against existing methods. In addition we create a pipeline which uses `ExpaRNA`'s predicted motifs as anchor constraints in order to speed up state-of-the-art Sankoff-style algorithms for simultaneous alignment and folding [147]. We successfully show the benefit of this combination for a large and reliable benchmark set.

## 2.2 Exact Pattern Matchings in RNA Structures

In the following we introduce our concept of exact pattern matchings in RNA secondary structures. We define an exact pattern matching (EPM) as local matching between two RNAs that does not necessarily implicate a contiguous subsequence in each RNA, but implies a structural locality induced by backbone bonds or base pairs. Furthermore, we require an *exact* matching in order to achieve its fast algorithmic identification. This scheme allows us to use EPMs for the description of local RNA motifs which comprise a set of structure-local nucleotides as shown in Figure 2.1. It was shown that comparative RNA methods can profit from this notion of locality because it improves the detection of signals which are weak at the sequence level ([6, 125]).

According to Definition 1.2.1 from Section 1.2.1 for arc-annotated sequences, we fix two non-crossing RNA sequences $(A, P_A)$ and $(B, P_B)$ and define an EPM as follows.

**Definition 2.2.1** (EPM)**.** *An Exact Pattern Matching (EPM) is a tuple* $(\mathcal{M}, \mathcal{S})$ *with* $\mathcal{M} \subseteq \{(i \sim k) | i \in \{1, \ldots, |A|\}, k \in \{1, \ldots, |B|\}\}$ *and* $\mathcal{S} \subseteq \{(ij \sim kl) | (i, j) \in \{1, \ldots, |A|\}^2, i < j, (k, l) \in \{1, \ldots, |B|\}^2, k < l\}$ *such that*

- *for all* $(i \sim k) \in \mathcal{M} : A_i = B_k$
- *for all* $(i \sim k), (j \sim l) \in \mathcal{M} : (i < j \Rightarrow k < l \wedge i = j \Leftrightarrow k = l)$
- $(ij \sim kl) \in \mathcal{S} \Rightarrow \{(i \sim k), (j \sim l)\} \subseteq \mathcal{M}$
- *the structure* $\{(i,j)|(ij \sim kl) \in \mathcal{S}\}$ *is non-crossing (with the previous condition this implies* $\{(k,l)|(ij \sim kl) \in \mathcal{S}\}$ *is non-crossing).*
- *the matching is connected on the sequence or structure level, i.e. the graph* $(\mathcal{M}, E)$ *with* $E = \{(i \sim k, j \sim l)|(i = j + 1$ *and* $k = l + 1)$ *or* $(ij \sim kl) \in \mathcal{S}\}$ *is (weakly) connected.*

Please note that Definition 2.2.1 is according to the EPM definition presented in our `ExpaRNA-P` paper [150]. Although we require that an EPM maintains all base pairs, it does not necessarily preserve all backbone bonds. Therefore the resulting graph from the last condition is only weakly connected because not for any two matchings $(i \sim k), (j \sim l)$ with $i = j + 1$ it also holds that $k = l + 1$. In the view of the RNA secondary structure this is for example the case if an EPM matches a complete hairpin in one RNA but the hairpin has a different size in the second RNA.

In the following we will use $\mathcal{E}$ for an arbitrary EPM $(\mathcal{M}, \mathcal{S})$ between two RNAs. In addition we define $\mathcal{E}|_{\mathcal{M}}$ as the set of all base matchings of a single EPM and with $\mathcal{E}|_{\mathcal{S}}$ we denote the set of matchings involved in base pairs. The size of an EPM $\mathcal{E}$ is defined as $|\mathcal{E}| = |\mathcal{M}|$, i.e. the size corresponds to the number of base matchings.

### Matching parents

Since non-crossing RNA structures correspond to trees, we define, for any position $k$ of $A$, the *parent of* $k$ as the $(i,j) \in P_A$ with $i < k < j$ such that there does not exist any $(i',j') \in P_A$ with $i < i' < k < j' < j$. Analogously, the *parent of a base pair* $(i,j)$ is the parent of $i$ (which is also the parent of $j$). Intuitively, if a base or base pair has a parent $(i,j)$, it is located in the loop closed by $(i,j)$. For external positions $k$ that are not included in any loop, we define the parent to be an additional virtual base pair $(0, |A|+1)$ over the entire sequence.

As the correspondence between nested RNA structures and trees naturally generalizes to EPMs, we can define the parent of some element of $\mathcal{M} \cup \mathcal{S}$ according to [150] as

$$\text{parent}_{\mathcal{S}}(i \sim k) = \underset{(i'j' \sim k'l') \in \mathcal{S} \cup \{(0|A|+1 \sim 0|B|+1)\}, i' \leq i \leq j'}{\text{argmin}} |j' - i'| \tag{2.1}$$

$$\text{parent}_{\mathcal{S}}(ij \sim kl) = \underset{(i'j' \sim k'l') \in \mathcal{S} \cup \{(0|A|+1 \sim 0|B|+1)\}, i' < i < j < j'}{\text{argmin}} |j' - i'|. \tag{2.2}$$

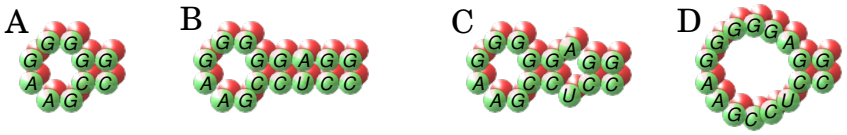


**Figure 2.2.** EPM A is not maximally extended if there exists a larger EPM like B or C. EPMs B, C, and D can all be maximally extended simultaneously since in each case some base matches have different parents. Each EPM is a matching between two RNAs which are denoted in red and green.

Note that every matched element that is not enclosed by matched base pairs has the *pseudo-parent* $(0|A|+1 \sim 0|B|+1)$ which is best understood as additional match of pseudo base pairs outside of the two sequences. Also note that for $\text{parent}_{\mathcal{S}}(ij \sim kl) \in \mathcal{S}$ $\text{parent}_{\mathcal{S}}(ij \sim kl) \neq \text{parent}_{\mathcal{S}}(i \sim k) = \text{parent}_{\mathcal{S}}(j \sim l) = (ij \sim kl)$, i.e. the parent of a matching base pair is always outside the pairs whereas the parent of a matching base can include the bases itself.

### Score of an EPM

The score of an EPM $(\mathcal{M}, \mathcal{S})$ consists of a score $\sigma(i,k)$ for each pair of matched unpaired bases and $\tau(i,j,k,l)$ for each pair of matched base pairs. For the purpose of simplification, we define the set of structure matches for two RNAs as $\mathcal{M}|_{\mathcal{S}} := \{(i \sim k), (j \sim l) | (ij \sim kl) \in \mathcal{S}\}$. We define the score of an EPM according to [150] as

$$\text{score}(\mathcal{M}, \mathcal{S}) = \sum_{(i \sim k) \in \mathcal{M} \setminus \mathcal{M}|_{\mathcal{S}}} \sigma(i,k) + \sum_{(ij \sim kl) \in \mathcal{S}} \tau(i,j,k,l). \tag{2.3}$$

### Maximally Extended EPM

In order to remove simple variants of large EPMs we consider only maximally extended EPMs. This not only maintains algorithmic complexity bounds, but also focus on important substructures. Maximally extended EPM is defined as follows [150].

**Definition 2.2.2** (maximally extended EPM)**.** *An EPM $(\mathcal{M}, \mathcal{S})$ is maximally extended, if there does not exist any $(\mathcal{M}', \mathcal{S}')$ with $\mathcal{M} \subset \mathcal{M}'$, $\mathcal{S} \subseteq \mathcal{S}'$ and such that for all $(i \sim k) \in \mathcal{M}$ $\text{parent}_{\mathcal{S}}(i \sim k) = \text{parent}_{\mathcal{S}'}(i \sim k)$.*

As shown in Figure 2.2, the last condition of this definition is required to ensure that we consider EPMs with different structures as being different. Due to this definition, the set of maximally extended EPMs does not contain proper substructures. For example, the EPM A in Figure 2.2 depicts a proper substructure of EPM B, but allowed are structural variants of the same set of matched positions as shown with EPMs C and D. Please note, the given definition for maximally extended EPMs can already handle structural variants of more complex structure representations like we use in Chapter 3, but which do not occur for fixed input structures in case of `ExpaRNA`.

**Set of EPMs**

Given two RNAs and their secondary structures, a large set of possible EPMs can exist. Clearly not all of them are relevant or meaningful. Hence we apply a score threshold $\gamma$ on all EPMs before we consider them for further comparison. We define the set of all maximally extended EPMs $(\mathcal{M}, \mathcal{S})$ over two RNAs $A$ and $B$ as

$$\mathbf{E}_\gamma = \left\{ \mathcal{E} \mid \mathcal{E} \text{ is EPM}(\mathcal{M}, \mathcal{S}) \ \wedge \ \text{score}(\mathcal{M}, \mathcal{S}) \geq \gamma \right\}. \tag{2.4}$$

If we use for example $\delta = 1$ and $\tau = 2$ then score$(\mathcal{M}, \mathcal{S})$ is defined similar to the minimal word size in BLAST [4] and the score threshold $\gamma$ designates the minimal number of matching nucleotides. A more advanced measure for score$(\mathcal{M}, \mathcal{S})$ could include energetic stability of the EPM and its matching substructures.

## 2.3 A Method for RNA Comparison with EPMs on Fixed Input Structures

The method presented in this chapter requires two nested RNA structures in input. According to Definition 2.2.1 each EPM is an arc-preserving common (but not longest common) subsequence as defined for the LAPCS problem [50]. Since EPMs have in addition the above described properties, the detection of all EPMs is a computationally light problem, compared to LAPCS, which is NP-complete even for nested sequences [15]. Using the dynamic programming approach described by Backofen and Siebert [5], the set of all EPMs for two nested RNA secondary structures can be found in $O(nm)$ time and $O(nm)$ space, making this approach applicable for fast sequence-structure comparisons.

$\mathbf{E}_\gamma$ can be seen as a "library" of all common motifs between two RNAs that can be utilized for a pairwise comparison method. Thus, the main idea of our approach will be to take a subset of EPMs from $\mathbf{E}_\gamma$ that in combination will cover a large portion of both RNAs. The EPMs in $\mathbf{E}_\gamma$ differ in their size and shape as well as in their structural positions in both RNAs. Simply selecting two or several of these substructures for combination would probably lead to overlapping or crossing structures (see Figure 2.3). Hence, the set of all EPMs is not a solution for the LAPCS problem since the combination of several EPMs is not necessarily arc-preserving. Clearly, a meaningful subset of common substructures excludes overlapping and crossing patterns. This guarantees that the backbone order of matched nucleotides as well as base pairs of the given RNAs are preserved. Compatible EPMs are non-crossing and non-overlapping. Formally, two EPMs $\mathcal{E}_1$ and $\mathcal{E}_2$ are *non-crossing* if $\mathcal{E}_1 \cup \mathcal{E}_2$ is an ordered matching. Note, this definition excludes overlapping EPMs as well. Figure 2.3 shows an example of a possible set $\mathbf{E}_\gamma$. If the EPMs indicated in red are excluded, then any other two EPMs are an ordered matching and they can be used to describe the similarity between two RNAs.

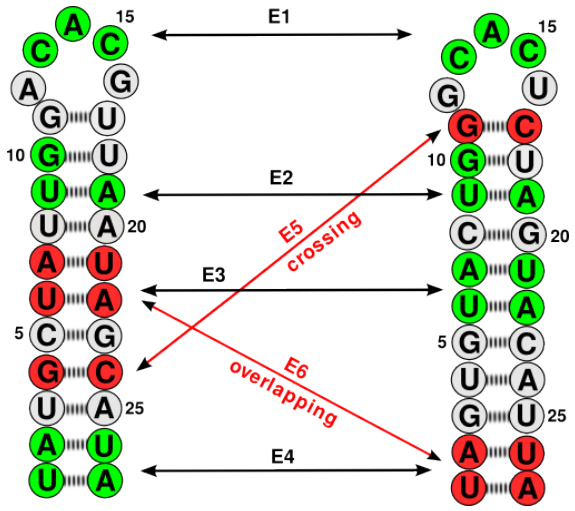


**Figure 2.3.** A possible set $\mathbf{E}_\gamma$ for two RNAs. The set $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$ can be used for a comparison, whereas $\{\mathcal{E}_5, \mathcal{E}_6\}$ should be excluded. $\mathcal{E}_5$ is crossing $\mathcal{E}_2$ and $\mathcal{E}_3$ whereas $\mathcal{E}_6$ is overlapping with $\mathcal{E}_3$ in the left RNA and with $\mathcal{E}_4$ in the right RNA. Note, not all possible EPMs are indicated.

### 2.3.1 Combining EPMs for Comparing RNAs: Problem Definition and Algorithm Overview

Our approach is motivated by the fact that similar RNAs with fixed secondary structures share identical structural elements in a similar arrangement. Examples are shown in our result section for the comparison of thermodynamically folded as well as experimentally verified secondary structures. The knowledge of such a "common core" of identical substructures in two RNAs is highly useful for comparative analysis tasks.

For our global approach, we are interested in a *maximal* possible arrangement of substructures shared by two RNAs. If the motifs are given in the form of exact pattern matchings, we call this the LCS-EPM problem (Longest Common Subsequence of Exact Pattern Matchings). Basically, we search for a maximal combination of EPMs that form a common subsequence. Note that albeit the problem shares some similarity with LAPCS, it is restricted in such a way that an efficient solution is possible.

Formally, LCS-EPM is defined as follows. Given two nested RNAs $A$ and $B$ and a set of exact pattern matchings $\mathbf{E}_\gamma$ of these two RNAs, find an ordered matching $\mathcal{M}_{\mathsf{EPM}}$ consisting of a subset of EPMs from $\mathbf{E}_\gamma$ that has maximal cardinality. Thus, $\mathcal{M}_{\mathsf{EPM}}$ is defined as the union $\mathcal{M}_{\mathsf{EPM}} = \bigcup \mathcal{C}$ of a subset $\mathcal{C} \subseteq \mathbf{E}_\gamma$, where all EPMs contained in $\mathcal{C}$ are mutually non-crossing. Note, this implies that the found subsequence is a common subsequence since $\mathcal{M}_{\mathsf{EPM}}$ is an ordered matching. Common base pairs are induced by the EPMs itself. We can now simply set $\sigma(i,k) = 1$ and $\tau(i,j,k,l) = 2$ to find the *longest* common subsequence of EPMs. In this case, the score corresponds to the number of matched nucleotides, i.e. $\text{score}(\mathcal{M}, \mathcal{S}) = |\mathcal{E}|$.

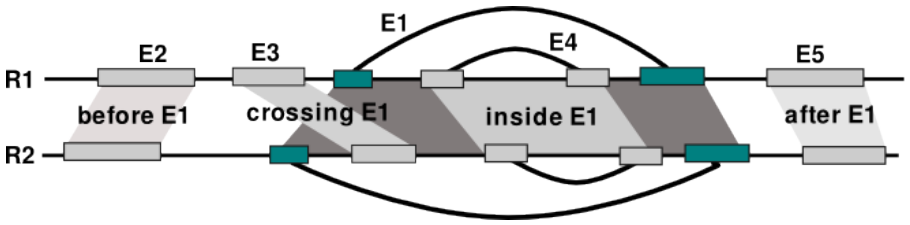


**Figure 2.4.** Ordering of exact pattern matchings relative to EPM $\mathcal{E}_1$ (indicated in green and dark gray). The cases *before, inside* and *after* do not violate the non-crossing condition. Only EPM $\mathcal{E}_3$ crosses $\mathcal{E}_1$. An arc denotes that an EPM contains at least one base pair.

Given an EPM library $\mathbf{E}_\gamma$, our algorithm works by singling out the best combination of compatible EPMs. This task is performed efficiently by dynamic programming. The main idea is to recursively reduce the EPM puzzle for EPMs enclosed in subsequences $A_{i...j}$ and $B_{k...l}$ to a problem of smaller subsequences. For our recursion scheme, we exploit the special structure of EPMs, which span matchings of certain subsequences of consecutive nucleotides. Between the boundaries of these matched consecutive subsequence, EPMs can omit subsequences; thereby they contain holes.

Figure 2.4 illustrates this structure of EPMs and shows, given a single EPM $\mathcal{E}$, how the relative position of other EPMs to $\mathcal{E}$ can be distinguished. Formally, this is defined via boundaries and holes of a single EPM.

### 2.3.2 Boundaries and Holes of an EPM

According to the definition of an EPM $(\mathcal{M}, \mathcal{S})$ as ordered matching, its base matchings $(i \sim k) \in \mathcal{M}$ with $|\mathcal{M}| = n$ can be denoted as two increasing sequences $\langle i_1, i_2, ..., i_n \rangle$ and $\langle k_1, k_2, ..., k_n \rangle$ of nucleotide positions.

#### Boundaries of EPMs

In the view of the secondary structure, the elements $(i_1, i_n)$ and $(k_1, k_n)$ determine the outside borders of the EPM. Therefore we call them *outside-boundaries* and write them as $\mathsf{OUT}_\mathcal{E} = \langle (i_1, i_n), (k_1, k_n) \rangle$. In the view of an arc-annotated sequence, we call $(i_1, k_1)$ *left-outside-boundaries* and $(i_n, k_n)$ *right-outside-boundaries* and denote them as $\mathsf{LEFT}_\mathcal{E}$ and $\mathsf{RIGHT}_\mathcal{E}$.

In case an EPM contains base pairs, the structural shape is more complex and the outside-boundaries are not sufficient to describe all structural borders. If not all enclosed nucleotides of a base pair are part of the EPM, then there exist two positions in each RNA that form an additional structural border *inside* the range of the outside-boundaries. In addition, if a pattern contains several independent base pairs (e.g. in a multi-loop), there can be several inside borders (cf. Figure 2.5). The set of all such borders is called *inside-boundaries* and is defined as

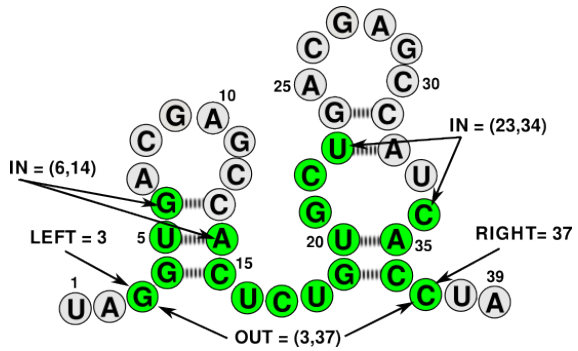


**Figure 2.5.** A pattern of an EPM in one RNA (green nucleotides). The different boundaries are indicated with arrows: a pair of outside boundaries and a two pairs of inside boundaries.

$$\mathsf{IN}_{\mathcal{E}} = \left\{ \left\langle (i_r, i_{r+1}), (k_r, k_{r+1}) \right\rangle \mid i_{r+1} > i_r + \beta \wedge k_{r+1} > k_r + \beta, \beta \geq 1 \right\}. \tag{2.5}$$

With the parameter $\beta$ we can control the required sequence length between a pair of inside-boundaries in each RNA. This will be useful later when we filter out holes which cannot contain other EPMs. Note that *outside-boundaries* always exists, whereas the set *inside-boundaries* can be empty. For example, assume an EPM that comprises only unpaired nucleotides or a complete hairpin including the closing bond. If an EPM consists of only one base pair in each sequence, then inside and outside boundaries are identical. With a superscript index to a specific boundary we indicate the position in a single RNA, for example $\mathsf{LEFT}^A_{\mathcal{E}} = i_1$.

### Holes

Holes describe the region between inside-boundaries of an EPM $\mathcal{E}$ which are not part of the subsequences $A[\mathsf{LEFT}^A_{\mathcal{E}} \,..\, \mathsf{RIGHT}^A_{\mathcal{E}}]$ and $B[\mathsf{LEFT}^B_{\mathcal{E}} \,..\, \mathsf{RIGHT}^B_{\mathcal{E}}]$. For a given EPM $\mathcal{E}$ with its set of inside-boundaries $\mathsf{IN}_{\mathcal{E}}$, the set of holes with minimal size $\beta$ is defined as

$$\begin{aligned}\mathsf{HOLES}_{\mathcal{E}} = \{\ \langle (h^{LA}, h^{RA}), (h^{LB}, h^{RB}) \rangle \mid h^{RA} - h^{LA} + 1 \geq \beta \ \wedge\ h^{RB} - h^{LB} + 1 \geq \beta \wedge \\ \langle (h^{LA} - 1, h^{RA} + 1), (h^{LB} - 1, h^{RB} + 1) \rangle \in \mathsf{IN}_{\mathcal{E}}, \beta \geq 1\ \}\end{aligned} \tag{2.6}$$

Each hole $h \in \mathsf{HOLES}_{\mathcal{E}}$ corresponds to a pair of inside-boundaries in $\mathsf{IN}_{\mathcal{E}}$ and a hole spans over the two subsequences $A[h^{LA} \,..\, h^{RA}]$ and $B[h^{LB} \,..\, h^{RB}]$; with '$R$' and '$L$' indicating the rightmost or leftmost position. With the parameter $\beta$ we omit holes that are too small for other EPMs and thus we can skip these holes in our algorithm. In general, we can set $\beta$ to the size of the smallest EPM found in $\mathbf{E}_\gamma$, i.e. $\beta = \mathrm{argmin}_{\mathcal{E} \in \mathbf{E}_\gamma} |\mathcal{E}|$.

In order to enhance the recursion scheme and avoid recalculation of some matrix entries, all holes are sorted according to their size w.r.t. one RNA. Let $h_i \in \mathsf{HOLES}_{\mathcal{E}_i}$ and $h_j \in \mathsf{HOLES}_{\mathcal{E}_j}$ two holes for any two $\mathcal{E}_i, \mathcal{E}_j \in \mathbf{E}_\gamma$. We define an ordering $h_i \preceq_{\mathsf{HOLES}} h_j$ w.l.o.g. for the first RNA if and only if $h_i$ is of smaller size than $h_j$ or of equal size, i.e.

$$h_i \preceq_{\mathsf{HOLES}} h_j \iff (h_i^{RA} - h_i^{LA}) \leq (h_j^{RA} - h_j^{LA}). \tag{2.7}$$

### 2.3.3 Dynamic Programming Recursion for LCS-EPM

The essential difference of LCS-EPM to other, alignment-based RNA comparison problems (including LAPCS) is that it treats a common substructure (i.e. an exact pattern matching) as a whole, unbreakable unit. This means that a solution of LCS-EPM either completely includes or completely excludes all matchings $(i \sim k) \in \mathcal{M}$ of an EPM. Following this idea, we want to compute the longest collinear sequence of EPMs which does not contain any crossing and overlapping EPMs. Note that there is also a relation to fragment chaining methods, which can be used for example for multiple genome comparison or to identify regions with conserved synteny [1]. However the algorithm presented here is better described as *structural* fragment chaining.

The overall solution for LCS-EPM is constructed by a bottom-up approach from the comparison of substructures that are covered by the subsequences $A[i\,..\,j]$ and $B[k\,..\,l]$. In principle, this requires a four-dimensional matrix, denoted as $D(i, j, k, l)$, which contains the maximal score for combining EPMs with matchings only in $A[i\,..\,j]$ and $B[k\,..\,l]$. However, we can restrict ourselves to two-dimensional matrices using the notions of boundaries and holes in combination with the ordering $\preceq_{\mathsf{HOLES}}$. For each hole, we introduce one two-dimensional matrix of entries $\mathbf{D}^h(j, l)$, such that $\mathbf{D}^h(j, l)$ is $D(h^{LA}, j, h^{LB}, l)$ of our imaginary four-dimensional matrix.

Finding non-crossing regions relative to an EPM $\mathcal{E}$ is achieved as follows: all nucleotides before $\mathsf{LEFT}_{\mathcal{E}}$, i.e $A[1\,..\,\mathsf{LEFT}^A_{\mathcal{E}}-1]$ and $B[1\,..\,\mathsf{LEFT}^B_{\mathcal{E}}-1]$, as well as all nucleotides after $\mathsf{RIGHT}_{\mathcal{E}}$, i.e. $A[\mathsf{RIGHT}^A_{\mathcal{E}}+1\,..\,|A|]$ and $B[\mathsf{RIGHT}^B_{\mathcal{E}}+1\,..\,|A|]$ fulfill the non-crossing condition. This means that any EPM with its outside-boundaries $\mathsf{OUT}_{\mathcal{E}}$ in these regions is non-crossing relative to the considered EPM. Similar we handle EPMs that contain base pairs with the introduced notion of $\mathsf{HOLES}_{\mathcal{E}}$. All EPMs that are located inside any hole of $\mathcal{E}$ cannot cross or overlap with $\mathcal{E}$.

The recursion scheme for a dynamic programming algorithm is now straightforward. Any $\mathcal{E}$ is handled only once at its right-outside-boundary $\mathsf{RIGHT}_{\mathcal{E}}$. The score of $\mathcal{E}$ is composed of the score *before* $\mathcal{E}$ (see Figure 2.4), given at the position $\mathsf{LEFT}_{\mathcal{E}} - 1$, plus the score of $\mathcal{E}$ itself and the additional contributions obtained from inside-boundaries, recursively computed for all holes $h \in \mathsf{HOLES}_{\mathcal{E}}$. This last recursion case recurses to possible substructures and therefore suggests the use of a four-dimensional matrix. However, it suffices to use only quadratic space, since a) all the scores for EPMs are stored in a vector

with entries $\mathbf{S}_{\mathcal{E}}$ and b) the score of each hole of an EPM can be computed using only a two-dimensional matrix. By ordering all holes according to $\preceq_{\mathsf{HOLES}}$, we guarantee that all necessary scores are already computed and stored, whenever an EPM is considered. Thus the recursion starts with the smallest holes and goes on to larger ones. Note, that two holes of the same size can be treated in any order. After a hole score is computed, $\mathbf{S}_{\mathcal{E}}$ is updated.

For the formal description of the recursion, a hole $h$ is fixed. The following recursion scheme works for any $j, l$ with $h^{LA} \leq j \leq h^{RA}$ and $h^{LB} \leq l \leq h^{RB}$.

$$\mathbf{D}^h(j,l) = \max \begin{cases} \mathbf{D}^h(j-1,l) \\ \mathbf{D}^h(j,l-1) \\ \mathbf{D}^h(i-1,k-1) + \mathbf{S}_{\mathcal{E}}, \\ \quad \text{if } \exists \mathcal{E} \in \mathbf{E}_\gamma \text{ with } \mathsf{RIGHT}_{\mathcal{E}} = (j,l) \text{ and} \\ \qquad \mathsf{LEFT}_{\mathcal{E}} = (i,k), i \geq h^{LA}, k \geq h^{LB} \end{cases}$$

$$\mathbf{S}_{\mathcal{E}} = \omega(\mathcal{E}) \; + \sum_{h \in \mathsf{HOLES}_{\mathcal{E}}} \mathbf{D}^h(h^{RA}, h^{RB}).$$

After filling all matrices $\mathbf{D}^h(j,l)$ we have the final vector $\mathbf{S}_{\mathcal{E}}$. The best score is then computed from treating the whole sequence as hole. With a standard traceback technique the set of EPMs that form the LCS-EPM are found.

### 2.3.4 Complexity Analysis

Let $n = |A|$ and $m = |B|$ denote the lengths of the input sequences. Given that we have only nested RNA secondary structures in input and each EPM is maximally extended, the number of EPMs contained in $\mathbf{E}_\gamma$ is bound by $n \cdot m$. Maximal EPMs imply that any two EPMs are disjoint and therefore any matching $(i \sim k) \in \mathcal{E} \in \mathbf{E}_\gamma$ is unique in $\mathbf{E}_\gamma$ and part of at most one EPM.

Accordingly, the set $\mathbf{E}_\gamma$ contains maximal $n \cdot m$ different holes. This can be estimated with $O(nm)$ and is the primary influence of the time complexity of the algorithm. For each hole, we fill a two-dimensional matrix with a size of at most $|A[h^{LA}, h^{RA}]| \leq |A| = n$ and $|B[h^{LB}, h^{RB}]| \leq |B| = m$. Consequently, for all holes we need $O(n^2m^2)$ time as worst case complexity.

However, a more appropriate time complexity can be given as $O(H \cdot nm)$ where $H$ denotes the number of holes. For real RNAs it es very likely that $H \ll n \cdot m$. This explains the fast running time of our algorithm on RNAs. The space complexity is only $O(nm)$ because for each hole, after computing its score contribution and adding the score to its EPM, the space for the corresponding matrix $\mathbf{D}^h$ is recycled.

We summarize the complexity of solving the LCS-EPM problem as follows. Given two nested RNAs $A$ and $B$. The problem to determine the longest common subsequence of exact pattern matchings (LCS-EPM), including computation of $\mathbf{E}_\gamma$, is solvable in total $O(n^2\,m^2)$ time and $O(nm)$ space.

## 2.4 Speeding Up RNA Alignment by EPMs

In the following we introduce an important application of LCS-EPM which uses the predicted chain of non-crossing EPMs as anchor constraints for sequence structure alignment methods like `FoldAlign`, `LocARNA` and related methods [10, 67, 179]. The idea of this combined alignment approach is to first solve the LCS-EPM for two given RNAs and then hand over the obtained result to an (usually much more expensive) sequence structure alignment algorithm. This second algorithm is used to fill the unaligned space between the exact pattern matchings in order to produce a complete alignment, i.e. an alignment that also includes all the bases that do not occur in EPMs. This procedure is illustrated in Figure 2.6.

In general, anchor constraints restrict the search space of possible alignments which in turn can accelerate alignment algorithms. Consequently, one expects a speed up of the existing sequence structure alignment tools that support anchor constraints, when combining them with a pre-processing by `ExpaRNA` that generates anchor constraints. Thus, the proposed combination will result in an accelerated RNA alignment approach compared to the underlying RNA alignment approach alone. The combination will work for any available alignment method. Since all anchor constraints are based on EPMs, which inherit structural properties of the RNAs, the quality of the final alignment should not be affected largely.

In particular, we modified the `LocARNA` algorithm for simultaneous folding and alignment of two RNA sequences $A$ and $B$ in order to profit from anchors [71, 179]. As a Sankoff-style algorithm, `LocARNA` essentially evaluates the recursion

$$M_{i\,j;k\,l} = \max \begin{cases} M_{i\,j-1;k\,l-1} + \sigma(j,l) \\ M_{i\,j-1;k\,l} + \alpha \\ M_{i\,j;k\,l-1} + \alpha \\ \max\limits_{j'l'} M_{i\,j'-1;k\,l'-1} + D_{j'\,j;l'\,l} \end{cases}$$

$$D_{i\,j;k\,l} = M_{i\,j-1;k\,l-1} + \tau_{ij;kl},$$

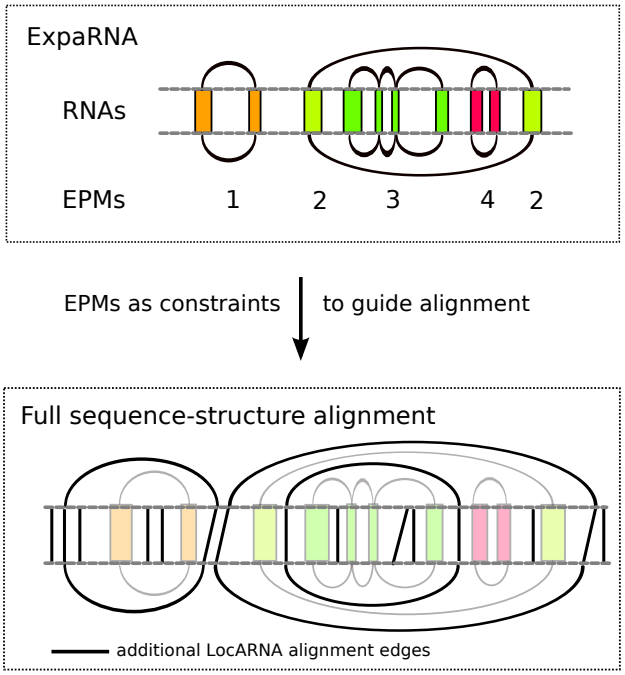


**Figure 2.6.** Workflow for combining `ExpaRNA` with `LocARNA`. First, `ExpaRNA` is called on the input RNAs to predict EPMs (1-4). This information is used as anchor constraints for a complete sequence-structure alignment by `LocARNA`. EPM numbers and colours correspond to the comparison of two RNaseP secondary structures shown in Figure A.4.

where $i, j, k, l$ are sequence positions, i.e. $1 \leq i < j \leq n = |A|$ and $1 \leq k < l \leq m = |B|$, $\alpha$ is the gap cost, $\sigma$ is a base similarity function, and $\tau$ is a base pair similarity function $\tau$, which reflects Turner's RNA energy model [78, 113]. An entry $M_{i\,j;k\,l}$ contains the maximal score of alignments of $A[i..j]$ with $B[k..l]$, whereas for the entries $D_{i\,j;k\,l}$ the alignments additionally have to match the base pairs $(i, j)$ and $(k, l)$. In consequence, $D_{i\,j;k\,l}$ are only required when $(i, k)$ and $(j, l)$ can be alignment edges of some alignment at all. For computing all entries $D_{i\,j;k\,l}$ with a common $(i \sim k)$, the algorithm fills the matrix slice $M_{i\,\cdot;k\,\cdot}$, which is the main load of the algorithm.

When given anchors, the algorithm can be modified to require less entries in $D_{i\,j;k\,l}$, namely only those where $(i \sim k)$ and $(j \sim l)$ are compatible with the anchors. Particularly this implies that it needs to compute only entries $M_{i\,j;k\,l}$ where $(i, k)$ is compatible with the anchor constraints.

For example assume that we have a single anchor constraint $(n/2 \sim m/2)$ (w.l.o.g. $n$ and $m$ even). Because only alignment edges $(i \sim k)$ with $i \leq n/2$ and $k \leq m/2$ or $i > n/2$ and $k > m/2$ are compatible with the anchor, the algorithm computes only entries in $M_{i\,j;k\,l}$ for those $(i, j)$, i.e. only 50% of the entries compared to the unconstrained algorithm.

## 2.5 Results

The algorithm for finding the longest common subsequence of exact pattern matchings (i.e. LCS-EPM) is implemented in C++ in the tool ExpaRNA (Exact pattern alignment of RNA) along with the algorithm to determine all EPMs [5].

We see at least two main application areas for ExpaRNA. First, given two RNAs along with their known or predicted secondary structure, the result of ExpaRNA comprises the optimal set of compatible exact common substructures. In biology, this can be used to get a good overview of existing similarities to support a functional determination. Second, due to the fast running time of ExpaRNA, it is very attractive to use ExpaRNA for high throughput RNA analysis tasks. We designed scenarios for both applications to study the different uses of our tool in detail.

### 2.5.1 Comparative Structural Analysis of Large RNAs

Here, we study the application of ExpaRNA for studying large RNAs that are very costly to compare by other sufficiently accurate tools. In this case, ExpaRNA elucidates information about identical structural motifs, which is not directly addressed by other tools and therefore may remain hidden. To enable an evaluation of our results, the experiments are performed on medium-sized and large RNAs where sequence structure alignment tools are still applicable.

We have chosen two pairs of RNAs: **a)** two IRES RNAs from Hepatitis C virus, which belong both to the Rfam family HCV_IRES for internal ribosomal entry sites (IRES); GenBank: AF165050 (bases 1-379) and D45172 (bases 1-391) [63]; and **b)** two 16S rRNAs. The first 16S is from *Escherichia coli* and is 1541 bases long; the second rRNA with 1551 nucleotides stems from *Dictyostelium discoideum* (GenBank codes: J01859 and D16466). The secondary structures for the IRES RNAs were predicted by RNAfold [76], whereas secondary structures for the 16S rRNAs were taken from the Comparative RNA Web (CRW) site [25].

Table 2.1 shows the results for both pairs of RNAs. The solution of LCS-EPM is depicted as colored annotations of the secondary structures - in Figure 2.7 for the IRES RNAs and in Figure 2.8 for the 16S rRNAs. These figures are directly produced by ExpaRNA with the help of the Vienna RNA Package for the structural layout [76]. For the IRES RNAs, the numbers mark the five largest EPMs from the set $\mathbf{E}_\gamma$ and correspond to the manually marked EPMs in the Backofen and Siebert article [5]. ExpaRNA predicts all of them automatically. In the case of the 16S rRNAs, the result of ExpaRNA shows significant similarities in nearly all stem and loop regions. Note that the EPM library $\mathbf{E}_\gamma$ was computed with $\gamma = 2$ and we used $\delta = 1$, $\tau = 2$ and $\beta = 2$ for both examples.

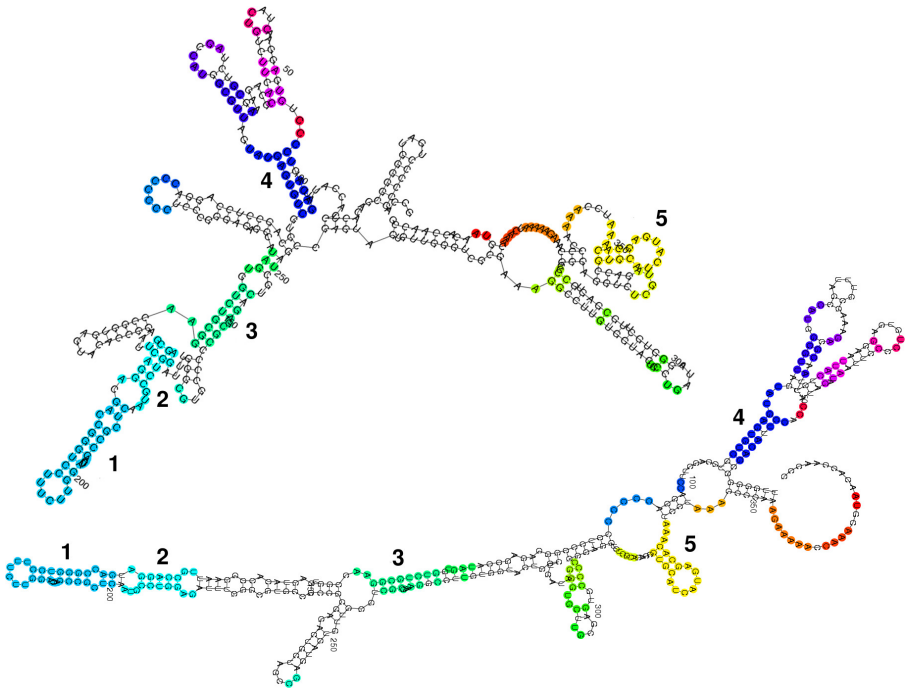


**Figure 2.7.** LCS-EPM approach applied to two Hepatitis C virus IRES RNAs. The colored nucleotides represent the optimal set of exact matching substructures (LCS-EPM) with a coverage of 45% (175 nt). Each EPM is shown in a different color. The numbers indicate the five largest EPMs from $\mathbf{E}_\gamma$. GenBank: D45172 (upper RNA), AF165050 (lower RNA).

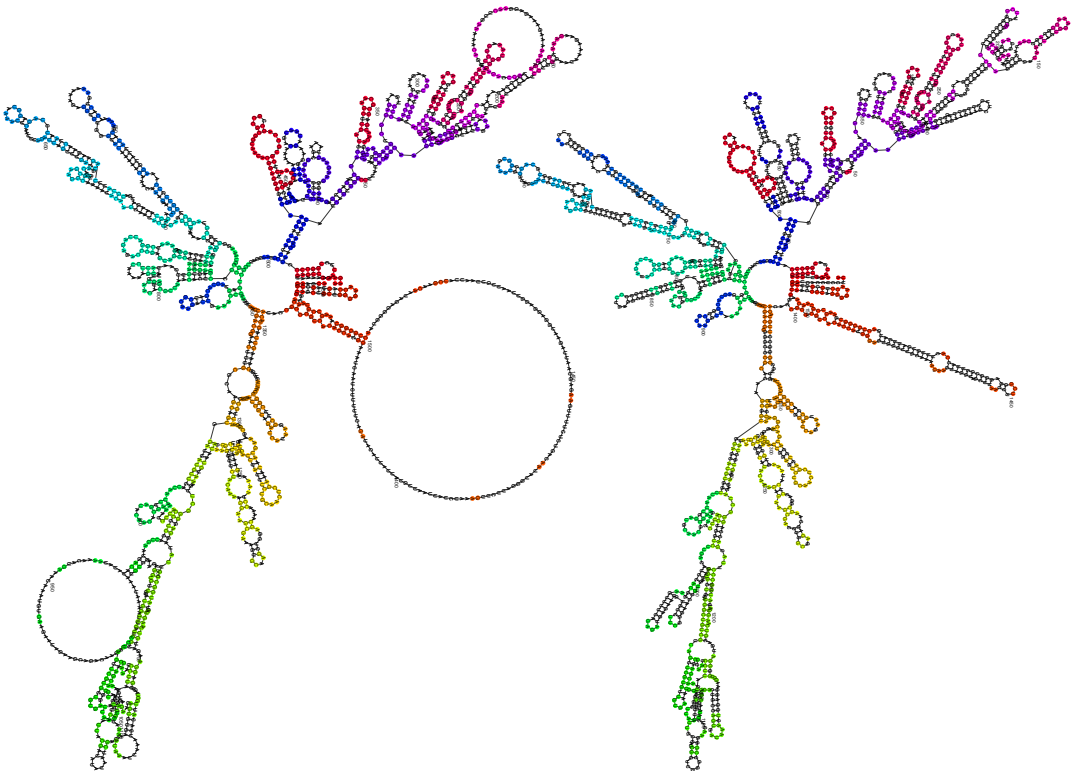

**Figure 2.8.** LCS-EPM approach applied to two 16S ribosomal RNAs. The colored nucleotides represent the found LCS-EPM with a coverage of 57% (875 nt). Each EPM is shown in a different color. Left: *D. discoideum* 16S rRNA (D16466) Right: *E. coli* 16S rRNA (J01859)

An interesting detail in Figure 2.7 is, for example, the included small blue hairpin in the top structure between number three and four. In the bottom RNA, this hairpin is opposite to the small yellow stem with number five, whereas in the top structure this stem is situated in another region.

We compare our results with the output of `RNA_align` and `RNAforester`. The first method computes sequence structure alignments according to the general edit distance algorithm [85]. The `RNAforester` program is build upon the tree editing algorithm for ordered trees and extends it to calculate forest alignments [73, 84]. We compare us with these tools since both tools cover the state-of-the-art in RNA alignment that is based on fixed structures. The general edit distance algorithm is a classic editing type algorithm for RNA comparison, whereas `RNAforester` represents the class of tree-alignment based algorithms, which can be due to their working principle much faster, but are less accurate than editing algorithms.

We compared the methods by the number of common realized alignment edges. For this purpose, we have first computed the alignments for both RNA pairs. Next, we have counted all positions with exact sequence structure matchings in these alignments and also determined the intersections with LCS-EPM. Note that the time for `ExpaRNA` in Table 2.1 includes the time to determine all EPMs for the two IRES RNAs ($0.44s$) and for the two 16S rRNAs ($1.2s$). The given sequence coverage rate is twice the number of predicted exact matches divided by the sum of the two sequence lengths.

| | IRES RNAs | | | 16S rRNAs | | |
|---|---|---|---|---|---|---|
| | #matches | coverage | time | #matches | coverage | time |
| **ExpaRNA** | 175 | 45% | $0.97s$ | 875 | 57% | $16.9s$ |
| `RNA_align` | 192 | 50% | $62.1s$ | 861 | 56% | $1h\,35m$ |
| `RNAforester` | 128 | 33% | $5.41s$ | 847 | 55% | $7m\,25s$ |

| comparison | IRES RNAs #common matches | 16S rRNAs #common matches |
|---|---|---|
| **ExpaRNA** & `RNA_align` | 159 (82.8%) | 688(79.9%) |
| **ExpaRNA** & `RNAforester` | 103 (80.5%) | 700(82.6%) |

**Table 2.1.** Comparison of the number of found exactly matching alignment edges by LCS-EPM and two alignment methods. In the lower part, *#common matches* defines the number of identical aligned nucleotides of `ExpaRNA` and the other methods.

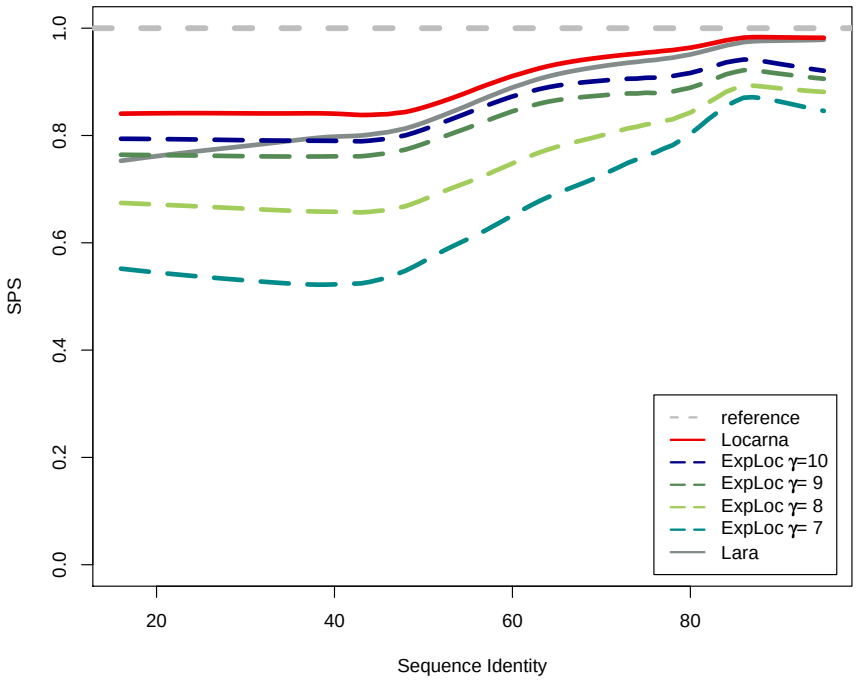


**Figure 2.9.** Obtained alignment qualities from the combined approach of `ExpaRNA` and `LocARNA` (`ExpLoc`). The alignment quality is measured as sum of pair score (SPS) for different minimal EPM sizes $\gamma$ in comparison to `LocARNA` and `Lara` on `BRAliBase 2.1` k2 dataset.

### 2.5.2 Speeding Up RNA Alignment for Large Scale Analysis

Here, we study the performance of `ExpaRNA` for high throughput RNA analysis. In Section 2.4 we have shown by which means sequence structure alignment algorithms can profit from anchor constraints. In consequence, we suggest combining complex RNA alignments methods with tools like `ExpaRNA` that predict EPM-like anchor constraints during a pre-computation step.

In order to assess the speedup of this approach, we exemplarily combine `ExpaRNA` with the sequence structure alignment tool `LocARNA` [125, 179]. We evaluate the accuracy of our combined approach (called `ExpLoc`) with the `BRAliBase 2.1` benchmark, which consists of a collection of hand-curated RNA alignments derived from `Rfam` [54, 184]. Because we are interested in the performance of pairwise alignment, we choose the k2 benchmark dataset with 8976 pairwise alignments. For each reference alignment we compute the corresponding `ExpLoc` alignment. As quality measure we use the compalign score which evaluates the accuracy of reproducing a reference alignment and refers to a sum-of-pairs score (SPS) [9, 54, 184]. A compalign score of 1 means the obtained alignment matches the

reference alignment perfectly whereas a compalign score of 0 refers to the case where the calculated alignment has no correspondence with the reference alignment. Furthermore, we record the runtime of ExpLoc and LocARNA for each k2 alignment.

For the computation of a single ExpLoc alignment we first determined for both sequences the minimum free energy structure via RNAfold and feed these two RNAs into ExpaRNA. Next, the ExpaRNA output is used as anchor constraint for LocARNA in order to obtain the complete alignment of the two RNAs.

In order to test the performance of the two approaches, we carried out five experiments. First, we examined the accuracy of LocARNA alone. The other four experiments evaluate the performance of the combined approach ExpLoc. Here, we assessed the resulting alignment quality for different score thresholds $\gamma = 7, 8, 9, 10$ when we use $\delta = 1$ and $\tau = 2$ in the ExpaRNA scoring function. Note, $\gamma$ corresponds in this case to an EPM size threshold.

Figure 2.9 shows the achieved SPS scores at different levels of sequence identity for all five experiments. In addition, we included the performance of the Lara sequence structure alignment algorithm [10]. Figure A.1 shows a boxplot (also called box-and-whisker plot) visualizing min-, max-values, medians and quartils of the SPS/Compalign score distribution for varying pairwise sequence identities.

The obtained speedup factors shown in Figure 2.10 are calculated relative to the LocARNA algorithm for different EPM score thresholds $\gamma$. With decreasing $\gamma$ the speedup increases because more anchor constraints can be predicted. The overall running time of LocARNA was $19h26min$. All computations were carried out on a Pentium 4 with 3.2 GHz.

## 2.6 A Web Server for ExpaRNA

We have set up a web server for ExpaRNA as part of the Freiburg RNA tools web server for advanced RNA analysis tasks [157]. Therein ExpaRNA can be used via an integrated, easy to use interface which supports the combination with other RNA analysis tools. The ExpaRNA web server is available at http://rna.informatik.uni-freiburg.de.

The input of ExpaRNA consists of a pair of RNA sequences and secondary structures in dot-bracket notation using an extended FASTA format. These sequences can be either entered directly or uploaded. If no secondary structure is available then the sequences are automatically folded by RNAfold [76]. Furthermore, the input page provides program-specific options with reasonable default settings in order that the user can configure ExpaRNA to their needs. The input is validated and the user is informed of inconsistencies as early as possible. An example input page is shown in Figure 2.11.

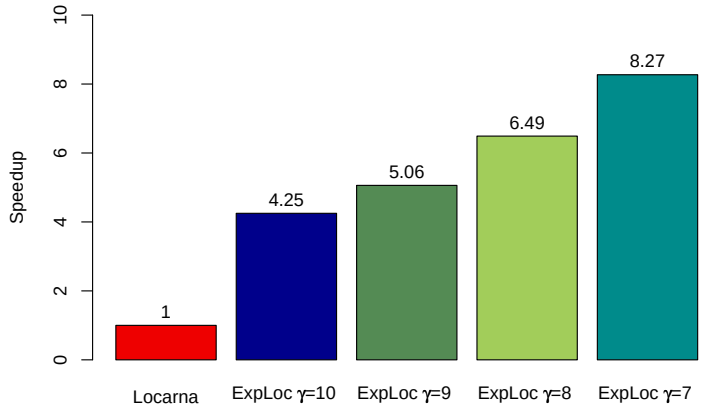


**Figure 2.10.** Speedup of the combined approach ExpLoc. The plot shows the achieved speedup relative to the run time of LocARNA for different minimal EPMsizes $\gamma$. With decreasing $\gamma$ the speedup increases because more anchor constraints can be predicted. In general, anchor constraints restrict the alignment search space and thus the runtime of the final alignment is reduced. Total times were measured against all alignments of the BRAliBase 2.1 k2 dataset.

ExpaRNA outputs the optimal set of exact pattern matches between the input RNAs. The result is presented graphically in the browser as coloured secondary structure plots (see e.g. Figures 2.7 and 2.8). Figures are displayed as PNG graphics and offered for download in postscript and PDF format. Additionally, the web server allows the user to download results in different text file formats, for example as structure annotated alignment or list of (all) found exact pattern matchings.

The webserver provides example input and a video tutorial for demonstration purposes. Online help is provided for general tool overview, its input, available options and output. Finally, the server provides a link to the source code of ExpaRNA. The stand-alone command-line version is more convenient and appropriate for large scale studies, however there are no input size restrictions by our web server.

**Combining ExpaRNA and LocARNA** In addition to the EPM-based comparison, the web server supports the direct usage of ExpaRNA's exact matches as anchor constraints for a full sequence structure alignment. This allows the calculation of a constraint alignment by LocARNA, hence enabling alignment of very large RNAs that otherwise could not be aligned in reasonable time. This procedure is supported by the web server with a direct link from the ExpaRNA results page to the LocARNA input page as shown in Figure 2.12.

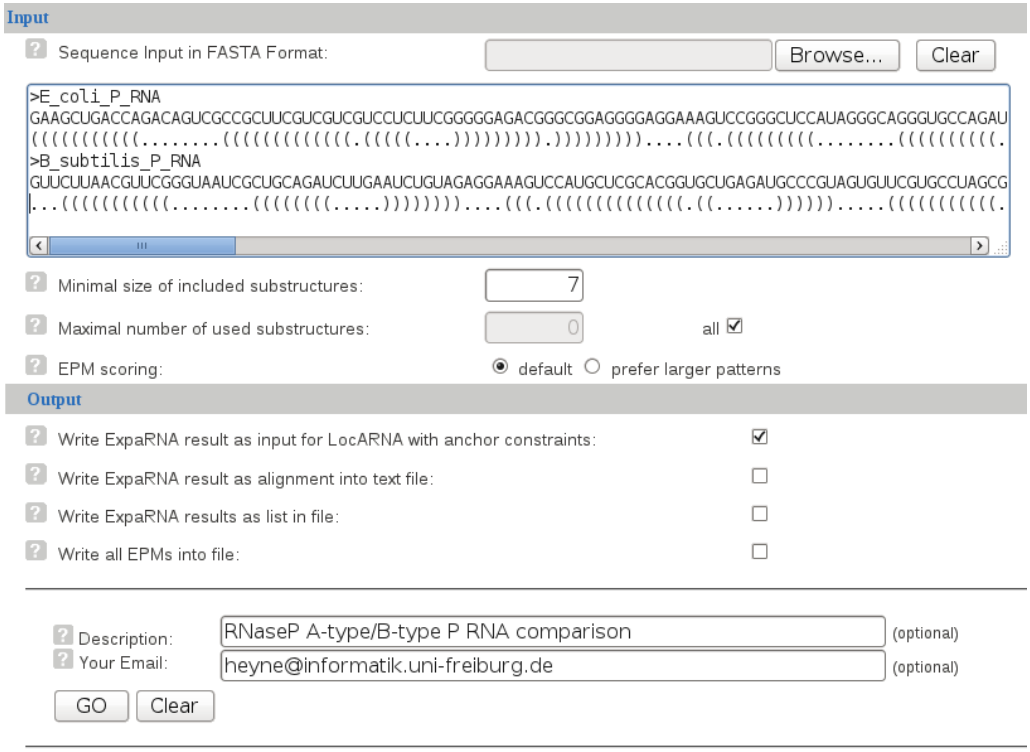


**Figure 2.11.** Screenshot of the `ExpaRNA` web server input page for a pair of example sequences. The input form allows to set important parameters and to choose between different output options. RNA secondary structures can be given in an extended FASTA format (as shown) or they will be automatically predicted with `RNAfold`.

**Technical aspects** The `ExpaRNA` web server is based on a general framework developed in the Freiburg Bioinformatics group and has been continuously improved [110]. XHTML is served by Apache Tomcat that supports the use of JavaServer Pages and Java Servlets consequently allowing a large deal of dynamically generated content to be provided. JavaScripting is used to aid the user in providing well formed input (sequences and parameters), which is then stored in a Java Bean and processed by a Java Servlet.

`ExpaRNA` together with all other tools from the Freiburg RNA tools web server are processed following a general scheme: jobs with valid input parameters are scheduled to a computing cluster managed by Sun Grid Engine in order that jobs can be computed in parallel and resources flexibly adapted to the server load. After submission the current status of the job is reported and the user receives a URL allowing access to the job status or output. It is also possible for the user to set a job description and to provide an email address to receive a notification when the job is finished. Upon job completion the result page is displayed online in the web browser and the user can access result details. Figure 2.12 shows an example result page. The whole web server is run on a virtual machine hosted on a server running Scientific Linux.

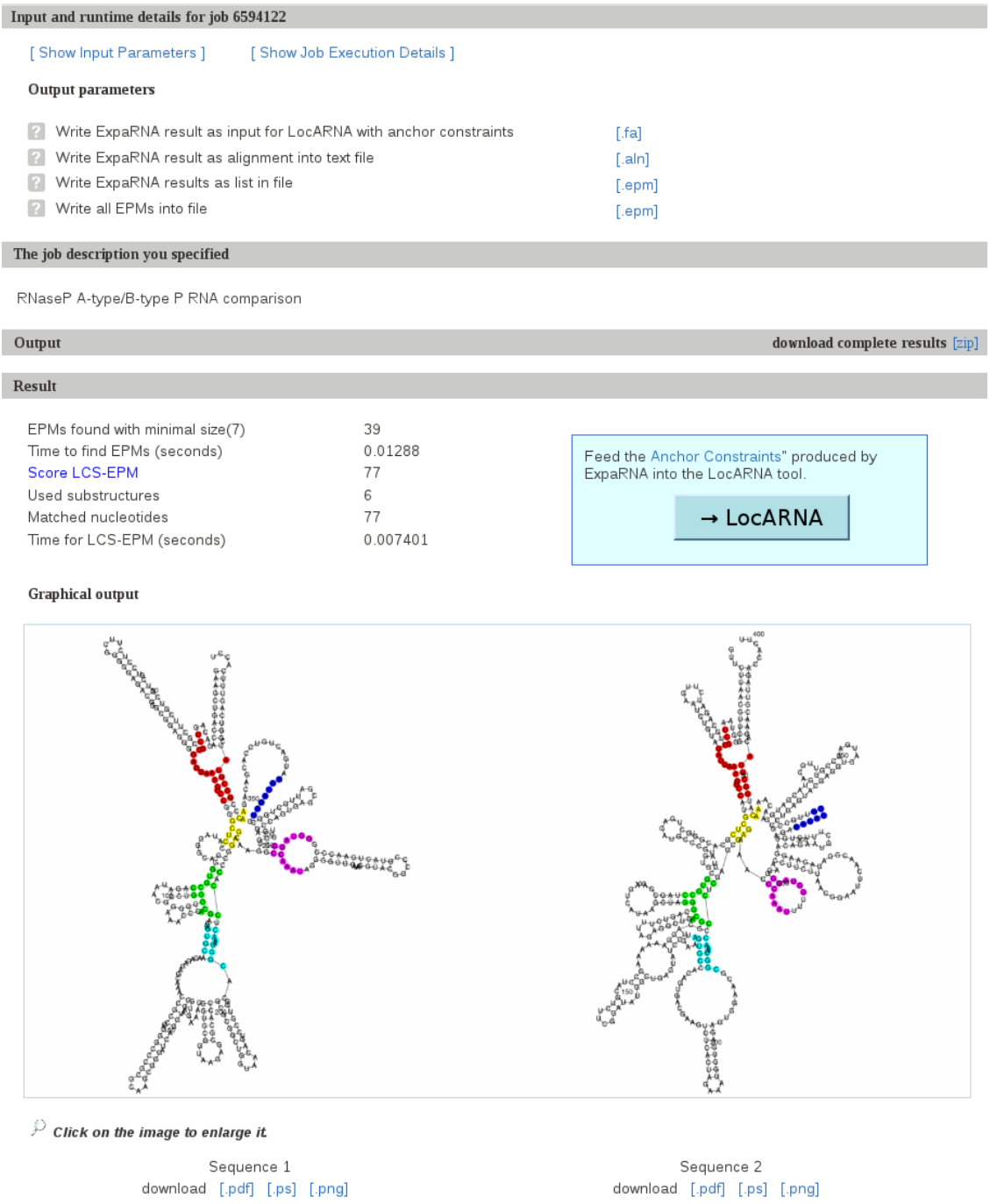


**Figure 2.12.** Screenshot of the ExpaRNA webserver result page. The result of the comparison is presented via colored secondary structure plots as well as in a summarizing table. A full screen view of the plots is possible to better analyze details. A direct link from the result page allows to feed the result as anchor constraints into the input page of the LocARNA web server. In addition, the page allows access to all job details as well as to download result files in various formats.

## 2.7 Discussion

In this chapter we presented ExpaRNA, a new and fast approach for pairwise comparison of RNA structures based on exact local matchings. Instead of computing a full sequence structure alignment, ExpaRNA efficiently computes the best non-crossing arrangement of sequence structure motifs common to two RNAs. As motifs we consider exact matching substructures, called exact pattern matchings or simply EPMs. We also introduced a general scheme to formalize exact matching substructures in RNA secondary structures and their usage for RNA comparison algorithms.

The presented results show that the developed ExpaRNA algorithm is useful for different applications in Bioinformatics and Biology. For comparative RNA analysis tasks, the result of ExpaRNA illustrates in nice way existing similarities between RNA structures. Existing relationships can be detected in a fraction of runtime without using a full alignment procedure. In addition, ExpaRNA can be used as a fast filtering method for large-scale datasets from modern sequencing techniques. High throughput analysis methods for thousands of RNAs are needed which incorporate sequence and structure. We analyzed the contribution of ExpaRNA for such tasks with the BRAliBase benchmark. In general, our combined approach yields comparable results like other sequence-structure alignment algorithms. We observed a scaleable trade-off between speedup and resulting alignment quality according to the selected minimal EPM size $\gamma$ (see Figure 2.9 and 2.10). By using different $\gamma$ parameters our combined approach ExpLoc can be nicely balanced. This is important for problems with large datasets in which often a lower quality setting is sufficient. Moreover, our results show that anchor constraints are able to speedup Sankoff-style alignment algorithms in general (cf. section 2.4).

A more fine-grained picture of the achieved accuracy of ExpLoc with $\gamma = 10$ is shown in Figure A.1. In the area below 70% sequence identity the differences are small. The lowered quality especially in the region with a high sequence identity can be explained by the used minimum free energy structures for ExpaRNA predicted by RNAfold. Only slight differences in the sequence result in wide changes of the secondary structure which in turn leads to wrong predicted anchors. However, pure sequence alignment programs are sufficient here. For benchmark instances with a low sequence identities ($\leq 30\%$) we observe nearly no differences. Here, ExpaRNA often does not find anchors which result in a standard LocARNA alignment. However, these cases are rare in the used benchmark set as indicated by the width of the boxes in Figure A.1. Finally, we also observe 336 alignments for ExpLoc with $\gamma = 10$ (447 alignments for $\gamma = 7$) resulting in a better SPS score than LocARNA alone indicating there are cases where our ExpaRNA is superior to existing methods.

The different speedups of `ExpLoc` for different $\gamma$ values can be explained by the number of predicted anchor points. For $\gamma = 7$ there exists more anchors than for $\gamma = 10$. Furthermore, we observe from our data speedups for short as well as for long alignments as shown in Figures A.2 and A.3. In particular, the speedup for long alignments is higher than for small ones, but also the majority of small alignments are accelerated. For longer RNAs we observe speedups around 100. We also look into the distribution of the speedups over different sequence identity classes. In general, sequences with a high sequence identity gain a higher speedup, but we also observe high speedups for classes between 35% and 65% sequence identity. This range is especially relevant for sequence structure alignment methods, as pure sequence alignment methods will fail here.

`ExpaRNA` can be extended by considering different matchings. It is known that many RNA families comprising members that share a only a very low sequence similarity. For this reason `ExpaRNA` could be extended by allowing mismatches in EPMs. However, only the usage of exact matchings permits a fast identification of common substructures and hence to maintain a fast overall runtime.

On the other hand, `ExpaRNA` uses only a fixed input structure for each RNA which limits largely the space of possible EPMs. Furthermore, the correct folding is often unknown and thus existing EPMs may remain hidden. Therefore, the usage of the structural ensemble for each RNA is necessary to identify EPMs which are not part of the minimum free energy structure. To this end, we developed the `ExpaRNA-P` approach presented in the following chapter. In addition, `ExpaRNA` can be extended by applying a different EPM scoring scheme which incorporates thermodynamic stability information of the underlying substructure.

# Chapter 3

# `ExpaRNA-P`: Exact Matchings in RNA Structure Ensembles

In this chapter we present `ExpaRNA-P` that efficiently solves the problem of finding exact matching substructures in RNA structure ensembles. First, we introduce novel in-loop probabilities and define significant EPMs in RNA structures. Second, we outline a dynamic programming algorithm that efficiently computes significant EPMs by means of a novel sparsification technique and analyze its complexity. Finally, we evaluate and discuss the benefits of our ensemble-based approach. This chapter is based on the publication Schmiedl et al. [150].

## 3.1 The `ExpaRNA-P` Approach

The large amount of recent studies like genome-wide transcriptomics point out the crucial role of RNAs in living cells [13, 27, 162]. It has become clear that the majority of novel transcripts are non-protein coding RNAs and they performing important primarily regulatory functions [116]. However, their functional annotation is strongly lagging behind and reliable automated annotation pipelines exist only for subclasses of ncRNAs such as tRNAs, miRNAs or snoRNAs [17]. Hence there is a constant need for fast and accurate methods for RNA analysis [60].

With `ExpaRNA` we have already presented a promising approach to enhance RNA analysis (cf. Chapter 2) and it can be used e.g. to speed up RNA alignments. In `ExpaRNA`, sequence-structure similarities are based on a single, fixed secondary structure for each RNA sequence. This strategy is working for well-conserved RNAs where the structure can be for example inferred due to their functional importance. This fact is also used by *a priori* RNA-gene finders like QRNA [138], RNAz [168], and Evofold [128], which detect conserved RNA-structures in whole-genome alignments. However, inferring a conserved structure on sequence-based alignments of less-conserved RNAs easily fails because the

secondary structure is not considered during alignment and thus gets misaligned [183]. Clearly, this strategy also fails if no comparative information is available at all. In these cases one have to resort to structure prediction methods like `RNAfold` or `mfold` and often the non-functional minimum free energy structure is used for tools which can only handle a fixed structure [76, 192].

Recently, a more qualified strategy towards the automatic annotation of non-coding RNAs has emerged, which identifies RNAs with similar sequence and common secondary structure on a genomic scale [86, 126, 179]. This can be used to determine remote members of RNA families as defined in the `Rfam` database [23, 62]. Albeit this approach is appealing, a wide-spread, or even automated, application of these methods has been hindered by the huge complexity of the underlying sequence-structure alignment approach for detecting similarity in both sequence *and* structure. The first practical approaches for multiple structural alignment, such as `RNAforester` and `MARNA` depend on predicted or known secondary structures [73, 154]. In practice, however, these approaches are limited by the low accuracy of non-comparative structure prediction. Sankoff's algorithm [147] provides a general solution to the problem of simultaneously computing an alignment and the common secondary structure of two aligned sequences. In its full form, the problem requires $O(n^6)$ CPU time and $O(n^4)$ memory (for RNA sequences of length $n$). This complexity is already limiting for most practical problems such as routinely scanning remote members of RNA families. For detecting novel RNA classes in the plethora of newly discovered RNA transcripts, this complexity becomes plainly prohibitive, since this task requires clustering based on quadratically many all-against-all pairwise RNA comparisons.

For that reason, many variants of the Sankoff algorithm with different optimizations have been introduced. `FoldAlign` [66] and `Dynalign` [114] implement a full energy model for RNA that is evaluated during the alignment computation. In contrast, `PMcomp` [78] and `LocARNA` [179] use a lightweight energy model, which assigns energies to single base pairs. This simplification reduces the computational cost significantly. They achieve their accuracy by precomputing the energy contributions of base pairs from their probabilities in a full-featured energy model [117]. Whereas approaches like `FoldAlign` and others [18, 59, 66, 114] have to compensate their computational demands by strong, often sequence-based, heuristics, `LocARNA` [179] takes advantage of structural sparsity in the RNA structure ensembles to reduce its complexity to $O(n^4)$ time and $O(n^2)$ space. This successful approach is consequently found in other Sankoff-like methods [10, 39, 163].

With `ExpaRNA-P` we introduce a strategy that reduces the computational demands further, but differs fundamentally from heuristic improvements (as e.g used in `RAF` [39]), that restrict the search space based on sequence alignments. The presented approach

computes sequence-structure-conserved elements that form highly probable local substructures in the RNA structure ensemble of both input RNAs. Subsequently we use these elements as anchor constraints in a full sequence-structure alignment by LocARNA.

In our previous approach ExpaRNA we have successfully shown how to use conserved elements in pairs of fixed RNA secondary structures for RNA comparison [71]. Therein finding of conserved substructures (EPMs) is based on an fast algorithm with quadratic time and space complexity [5]. Albeit ExpaRNA reduces the overall computation time significantly, we face similar problems as the first generation of RNA alignment methods [73, 154], due to the use of a single predicted input structure for each sequence. Since predicting minimum free energy (MFE) structures from single sequences is unreliable, this strategy fails frequently and causes severe misalignments.

Overcoming the limitations of the previous approach, our novel algorithm for determining exact sequence-structure patterns is based on probabilities in the RNA structure ensembles. We point out that a straight-forward extension of the fixed input structure algorithm to RNA structure ensembles, would result in a complexity of $O(n^4)$ time and $O(n^2)$ space. This complexity is as high as the one of a full sequence-structure alignment method like LocARNA, which would nullify the benefits of exact matching.

Thus, our main technical contribution is to solve the ensemble based problem in quadratic time and space. This advancement is comparable to the leap from first generation RNA alignment to efficient Sankoff-style alignment. For this achievement, we introduce a method of sparsification that uses the ensemble properties of the input sequences. Previous sparsification approaches reduced the number of computations required for each entry [7, 67, 146, 178, 191] or the number of matrices to be considered [163, 179]. In addition, we identify sparse regions of each matrix *a priori* such that, in total, only quadratically many entries remain; each of these entries is calculated in constant time. The *a priori* identification of sparse regions is based on the joint probability that a sequence position occurs as part of a particular loop. Since the sum of these probabilities is bound by one, we can control the complexity on a global scale by setting a probability threshold. As a further benefit over sparsification methods that filter non-optimal solutions [7, 67, 146, 178, 191], our sparsification allows us to enumerate suboptimal solutions.

We successfully evaluate the practical benefits of these algorithmic innovations by a novel pipeline ExpLoc-P for sequence-structure alignment. In its first stage, it enumerates suboptimal exact matchings of local sequence-structure patterns due to the introduced algorithm ExpaRNA-P. We also enhance ExpaRNA's chaining algorithm in order to deal with suboptimal matchings derived from structure ensembles. Finally, we utilize an optimal subset of structurally compatible matchings as anchor constraints in a subsequent Sankoff-style alignment by LocARNA. In benchmarks on BRAliBase we show the effective improvement of ensemble-based anchor constraints, both in quality and speed.

## 3.2 Significant EPMs in RNA Structure Ensembles

In the previous chapter we have introduced an EPM as exact matching substructure between two RNAs. More precisely, we only require that each EPM itself is a non-crossing or nested structure independently of any underlying structure space. For our ExpaRNA-P approach we use similar notions and definitions (cf. Section 2.2), but extend them with features required to define EPMs on top of structural ensembles. In detail, we extend the previous EPM definition (cf. Definition 2.2.1) to *significant* EPMs since we only want to match substructures that are probable in the structural ensemble of the given RNA sequences. Considering only significant EPMs is crucial for both the quality of the results and the complexity of the algorithm, which will be discussed later. In order to define significant EPMs we fix two RNAs $A$ and $B$ and introduce the following novel in-loop probabilities according to [150] over the Boltzmann weighted ensemble of RNA structures.

a) $\Pr\{(i,j)|X\}$ denotes the probability, that a structure in the ensemble of $X \in \{A,B\}$ contains the base pair $(i,j)$,

b) $\Pr^{\text{loop}}_{(i,j)}(k|X)$ denotes for $i<k<j$ and $X \in \{A,B\}$ the joint probability that the structure of $X$ contains the base pair $(i,j)$ and the unpaired base $k$ such that $(i,j)$ is the parent of $k$.

c) $\Pr^{\text{loop}}_{(i,j)}((i',j')|X)$ denotes for $i<i'<j'<j$ and $X \in \{A,B\}$ the joint probability that the structure of $X$ contains the base pairs $(i,j)$ and $(i',j')$ and that $(i,j)$ is the parent of $(i',j')$.

In the special case of a pseudo base pair (or pseudo parent; cf. Section 2.2) where $(i,j) = (0,|A|+1)$ we define

$$\Pr^{\text{loop}}_{(0,|A|+1)}((i',j')|X) := \Pr\{(i',j')|X\} \tag{3.1}$$

$$\Pr^{\text{loop}}_{(0,|A|+1)}(k|X) := 1 - \sum_{j<i} \Pr\{(j,i)|X\} - \sum_{i<j} \Pr\{(i,j)|X\} \tag{3.2}$$

whereas the latter describes the probability that base $k$ of $X$ is unpaired [150]. Please note that these probabilities include the cases where $(i,j)$ or $k$ are covered by some base pair. This is reasonable as EPMs are structurally local. Thus, they can be enclosed by other structure or be external. In Section 3.3 we show how to compute these probabilities efficiently. The advantage over the ExpaRNA approach is that these probabilities allows for selecting only relevant local structures instead of treating any base or base pair equally. Additionally, we can maintain the maximally-extended property of EPMs in structural

ensembles because we allow for EPMs which are structural variants. Figure 2.2 in the previous chapter shows three EPMs B, C and D which are all varying EPMs of the same set of positions.

For significant EPMs we introduce three different thresholds $\theta_1$, $\theta_2$ and $\theta_3$. We require that all matched base pairs have a probability of at least $\theta_1$ and that the probabilities of all matched unpaired bases and matched base pairs to occur as part of the loop of their respective parent is at least $\theta_2$ and $\theta_3$, respectively. This leads to the following definition for significant EPMs [150].

**Definition 3.2.1** (significant EPM)**.** *Given the thresholds $\theta_1$, $\theta_2$, $\theta_3$, an EPM is significant iff*

- *for all* $(ij \sim kl) \in \mathcal{S}$*:* $\Pr\{(i,j)|A\} \geq \theta_1$ *and* $\Pr\{(k,l)|B\} \geq \theta_1$
- *for all* $(i \sim k) \in \mathcal{M} \setminus \mathcal{M}|_{\mathcal{S}}$ *with* $(i'j' \sim k'l') = \text{parent}_{\mathcal{S}}(i \sim k)$*:* $\Pr^{loop}_{(i',j')}(i|A) \geq \theta_2$ *and* $\Pr^{loop}_{(k',l')}(k|B) \geq \theta_2$
- *for all* $(ij \sim kl) \in \mathcal{S}$ *with* $(i'j' \sim k'l') = \text{parent}_{\mathcal{S}}(ij \sim kl)$*:* $\Pr^{loop}_{(i',j')}((i,j)|A) \geq \theta_3$ *and* $\Pr^{loop}_{(k',l')}((k,l)|B) \geq \theta_3$

## 3.3 Precomputing Likely Loops

In a preprocessing step we compute the above introduced probabilities required to determine significant EPMs. This can be performed for each sequence separately. Hence, in clustering scenarios, for example, where all pairs from a set of sequences need to be matched, this preprocessing needs to be done only once for each sequence and not for all quadratically many pairs.

For the precomputation of likely loops we use the McCaskill algorithm [117] as introduced in Section 1.3.1. This algorithm can be used to compute the base pair probabilities $\Pr\{(i,j)|A\}$. We extend the McCaskill algorithm to calculate the probabilities $\Pr^{\text{loop}}_{(i,j)}(k|A)$ and $\Pr^{\text{loop}}_{(i,j)}((i',j')|A)$. To this end, we use the original McCaskill matrices $Q_{ij}$, $Q^b_{ij}$, $Q^m_{ij}$, and $Q^{m1}_{ij}$. In addition, we compute a matrix $Q^{m2}_{ij} = \sum_{i<k<j-1} Q^m_{ik} Q^{m1}_{k+1\,j}$ that represents parts of a multiloop with at least two outermost base pairs. Then we can compute the unpaired probability $\Pr^{\text{loop}}_{(i,j)}(k|A)$ as joint probability of $\Pr\{(i,j)|A\}$ and the probability of the set of all structures where $(i,j)$ is parent of $k$, i.e. by summing up all Boltzmann weights of such structures divided by the corresponding partition function $Q^b_{ij}$. Accordingly, we can compute the probability of base pairs in loops, $\Pr^{\text{loop}}_{(i,j)}((i',j')|A)$, by computing the probability of all structures where $(i,j)$ is parent of $(i',j')$.

Given an RNA $A$, both joint probabilities, i.e. the base pair probabilities and the unpaired probabilities in loops, can be computed within the same asymptotic time complexity of $O(|A|^3)$ as the McCaskill algorithm. More details and formulas are given in our paper [150].

## 3.4 A Method for Computing the Significant EPMs

In the following we outline the developed dynamic programming algorithm and its recursion scheme that computes significant EPMs. Basically, we fill a matrix $D$ whose entries $D((ij),(kl))$ store the maximum score of an significant EPM under a base pair match $(ij \sim kl)$. The matrix $D$ is filled in increasing order to the size of the base pair matches. This ensures that all base pairs within $(i,j)$ or $(k,l)$ are already computed when an entry $D((ij),(kl))$ is filled.

In addition, for each entry in $D$ we need to compute the matrices $L^{ijkl}(j',l')$, $G_A^{ijkl}(j',l')$, $G_{AB}^{ijkl}(j',l')$, and $LR^{ijkl}(j',l')$ for each $i < j' < j$ and $k < l' < l$. The scheme of our algorithm is that we match bases and closed substructures from left to right below the loops $(i,j)$ and $(k,l)$. The matrix $L$ denotes the part of the matching that is connected to the left ends $i$ and $k$ of the base pairs. Then the matching can continue with a gap in both sequences which is modeled by the matrices $G_A$ and $G_{AB}$. Finally, the matrix $LR$ denotes the part that is connected to the right ends $j$ and $l$.

In the following we describe the main principles of the recursion scheme and give details for the $L$ matrix. All other exact recursions will be part of a journal version for `ExpaRNA-P` (Otto, C. et al., in preparation). To ease understanding of all recursions and their cases, Figure 3.1 shows a visualization of the recursion scheme. For matrix $L$, the recursion is given as follows.

$$L^{ijkl}(j',l') = \max \begin{cases} -\infty \\ \text{if } A_{j'} = B_{l'},\ \Pr^{\text{loop}}_{(i,j)}(j'|A) \geq \theta_2 \text{ and } \Pr^{\text{loop}}_{(k,l)}(l'|B) \geq \theta_2 \\ \quad L^{ijkl}(j'-1,l'-1) + \sigma(j',l') \\ \text{for all } (i',j') \in P_A, (k',l') \in P_B \\ \quad \text{with} \begin{array}{l} \Pr\{(i',j')|A\} \geq \theta_1, \Pr\{(k',l')|B\} \geq \theta_1, \\ \Pr^{\text{loop}}_{(i,j)}((i',j')|A) \geq \theta_3 \text{ and } \Pr^{\text{loop}}_{(k,l)}((k',l')|B) \geq \theta_3, \end{array} \\ \quad L(i'-1,k'-1) + D((i'j'),(k'l')) \end{cases} \tag{3.3}$$

The base cases are $L^{ijkl}(i,k) = 0$, $L^{ijkl}(j',k) = -\infty$ for all $j' > i$ and $L^{ijkl}(i,l') = -\infty$ for all $l' > k$. The recursion for $L$ always matches the last positions $j'$ and $l'$ or returns $-\infty$ if $j'$ and $l'$ do not match. The matching $j' \sim l'$ can be an extension of an unpaired

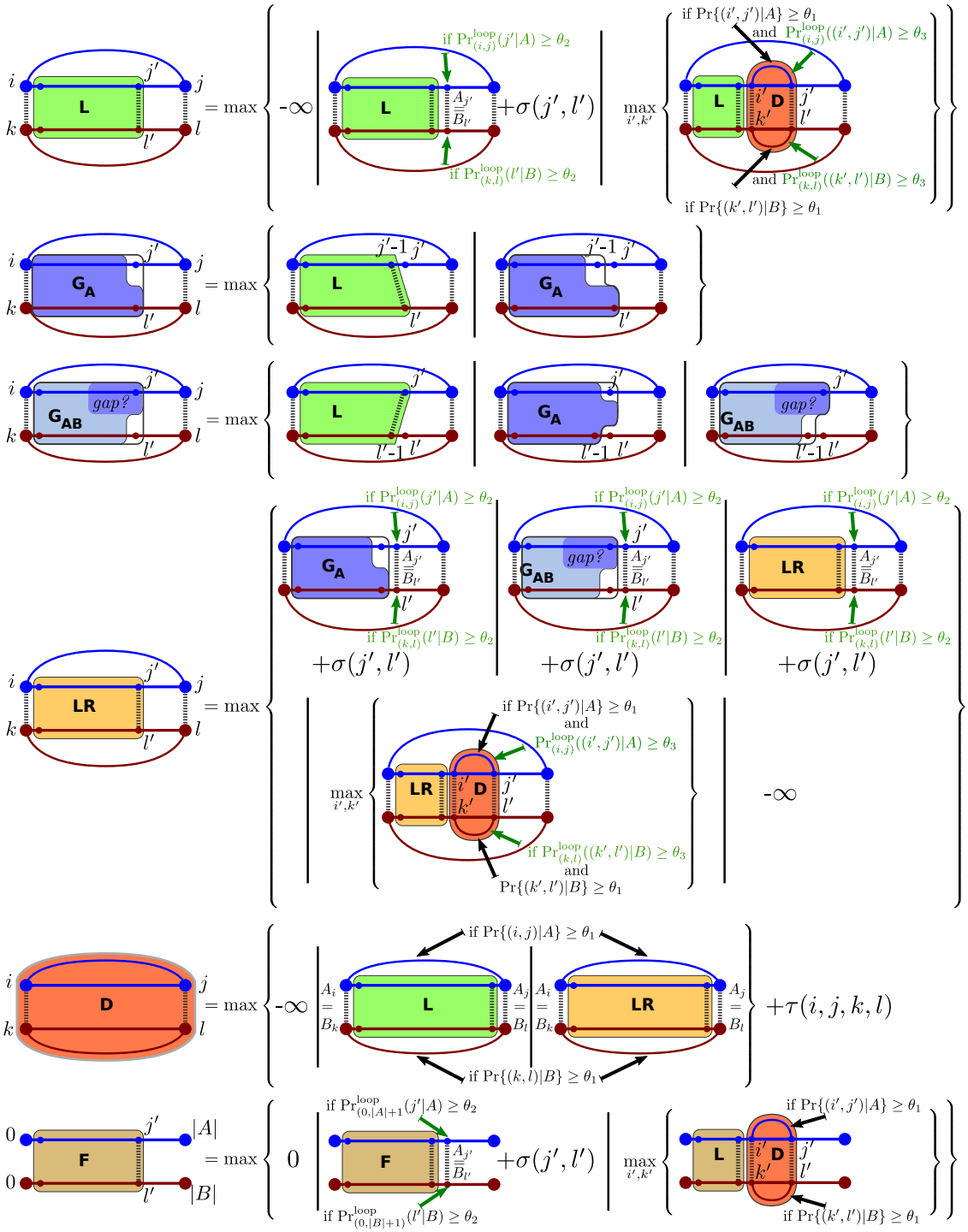


**Figure 3.1.** Visualization of the recursions to compute the matrix entries $L^{ijkl}(j', l')$, $G_A^{ijkl}(j', l')$, $G_{AB}^{ijkl}(j', l')$, $LR^{ijkl}(j', l')$, $D((ij), (kl))$, and $F(j', l')$. The recursion equations for matrix $L$ is given in Equation 3.3.

match or of two base pairs. In addition, we check if $j' \sim l'$ ensures all probability thresholds $\theta_1, \theta_2$ and $\theta_3$ to make use of our novel in-loop probabilities. The recursion for $LR$ is similar to $L$ but it also recognizes a gap ending at $j'-1$ or $l'-1$. The gap itself is handled in the matrices $G_A$ and $G_{AB}$. To obtain only unambiguous solutions during the suboptimal traceback, we always use gaps in RNA $A$ first and then skip positions in $B$. The case that a matching goes completely from the left to the right below a matching base pair $(ij \sim kl)$ is handled in $D$ by considering $L^{ijkl}$, as well.

Finally, we compute the matrix $F$ where each $F(j', l')$ indicates the best score of an EPM ending at $(j', l')$. As we want to find all significant EPMs, we initialize $F$ with 0 instead of $-\infty$. This allow EPMs to start at any point in the matrix likewise to local sequence alignment. However, we only want to enumerate maximally extended EPMs. This can be ensured if we start tracebacks in $F$ only for entries where $A_{j'+1} \neq B_{l'+1}$, i.e. for non-matching bases. This leads to the following lemma about maximally extended EPMs [150].

**Lemma 1.** *A maximally extended EPM $(\mathcal{M}, \mathcal{S})$ of $A[1\,..\,j']$ and $B[1\,..\,l']$ with $(j' \sim l') \in \mathcal{M}$ is also a maximally extended EPM of $A$ and $B$, iff $A_{j'+1} \neq B_{l'+1}$.*

For the proof we refer to our paper [150], but the idea is as follows. If there is a matching $(j'+1 \sim l'+1)$ then clearly the smaller EPM ending at $(j', l')$ is not maximally extended. On the other hand, two different EPMs can only exist if they have different parents for some $(j' \sim l')$, but this contradicts Definition 2.2.2 for maximally extended EPMs.

In general, the recursion scheme uses the introduced in-loop probabilities and therefore the matrices are sparse. Note that we precompute these probabilities and whenever we recurse, for example, into $L$ or $LR$, we can skip unlikely entries completely. Please see Sections 3.4.1 and 3.4.2 below for more details on the sparsification and complexity. Suboptimal EPMs are retrieved by doing a standard traceback, but enumerating all EPMs up to a given score threshold.

### 3.4.1 Sparsification

We need to compute matrices $L^{ijkl}$, $G_A^{ijkl}$, $G_{AB}^{ijkl}$, and $LR^{ijkl}$ only for $i, j, k, l$ with $\Pr\{(i,j)|A\} \geq \theta_1$ and $\Pr\{(k,l)|B\} \geq \theta_1$. For each of these matrices, we further reduce the number of entries as follows. We call each $j'$ a *candidate* of $(i,j)$ if $\Pr^{\text{loop}}_{(i,j)}(j'|A) \geq \theta_2$ or if for some $i'$ $\Pr^{\text{loop}}_{(i,j)}((i',j')|A) \geq \theta_3$. Analogously, $l'$ is a candidate of $(k,l)$ if $\Pr^{\text{loop}}_{(k,l)}(l'|B) \geq \theta_2$ or if for some $k'$ $\Pr^{\text{loop}}_{(k,l)}((k',l')|B) \geq \theta_3$. Note that if $j'$ or $l'$ is no candidate, the recursion directly implies that $L^{ijkl}(j', l') = LR^{ijkl}(j', l') = -\infty$ and hence we neither have to explicitly compute nor to store these entries. This allows to skip the corresponding entries $G_A^{ijkl}(j', l')$ and $G_{AB}^{ijkl}(j', l')$, because for $L^{ijkl}(j', l') = -\infty$ their value is identical to their respective neighboring entry. In total, this optimization allows to skip in $L^{ijkl}$, $G_A^{ijkl}$,

$G_{AB}^{ijkl}$, and $LR^{ijkl}$ each complete row or column whose index is no candidate. Since we can compute (in a preprocessing step and for each sequence separately) a mapping from sequence positions to candidate positions, the recursion can be implemented on matrices that only contain the candidate rows and columns. In the following complexity analysis, we show that this optimization reduces the entries from $O(|A|^3|B|^3)$ to only $O(|A||B|)$ across all matrices.

### 3.4.2 Complexity Analysis

The central aspect for the complexity reduction by in-loop probabilities is given by the following lemma according to [150].

**Lemma 2.** *For a fixed $j'$, there are only $O(1)$ base pairs $(i, j)$, such that $j'$ is a candidate of $(i, j)$ (and analogously for $l'$ and $(k, l)$ in sequence B).*

For the full proof we refer to our paper [150]. Basically, we can argue that $j'$ is a candidate in RNA $A$ only if $p_{j'}(i, j) \geq \theta^* := \min\{\theta_2, \theta_3\}$, i.e $j'$ is above a probability threshold. With $p_{j'}(i, j)$ we indicate the probability that a structure of $A$ contains the base pair $(i, j)$ and $j'$ is unpaired or the right end of a base pair below the loop closed by the base pair $(i, j)$. But $j'$ is always part of exactly one loop $(i, j)$ and therefore the sum of probabilities for all structures in the loop is bound by 1. Consequently, there are at most $\frac{1}{\theta^*} \in O(1)$ base pairs for that $j'$ is a candidate.

Clearly, this argumentation is only possible due to the new probabilities for single loops instead of, for example, global base pair probabilities. We make use of these in-loop probabilities in our DP-algorithm with a new sparsification scheme. In consequence of Lemma 2, there exist only a linear number of possibilities $i, j, j'$ for candidate positions in RNA $A$, and thus, $O(n^2)$ combinations for two RNAs. Hence, we only fill $O(n^2)$ many entries in the matrices $L^{ijkl}(j', l')$, $G_A^{ijkl}(j', l')$, $G_{AB}^{ijkl}(j', l')$, and $LR^{ijkl}(j', l')$ where $j'$ and $l'$ is a candidate of $(i, j)$ and $(k, l)$, respectively.

The complexity analysis for the matrices $D$ and $F$ can be done similar to the argumentation for the `LocARNA` algorithm [125, 179]. For each base $j$ in $A$ there exists only a fixed number of base pairs because we require for all base pairs $\Pr\{(i', j')|A\} \geq \theta_1$ and analogously for $B$. This threshold can be ensured only for a constant number of base pairs $\frac{1}{\theta_1} \in O(1)$ as the sum of probabilities is limited to one by the fact that there is only one structure that ends in $j$. Then we can compute each entry in all matrices in constant time for some fixed $j'$ and $l'$ for a base pair $(i', j')$ from RNA $A$ and a base pair $(k', l')$ from RNA $B$.

Consequently, the algorithm to compute significant EPMs has a time and space complexity of $O(n^2)$. Note that the McCaskill algorithm has the dominating complexity, albeit the calculation of base pair probabilities can be done during a preprocessing step.

## 3.5 Chaining

The identified set of significant EPMs is useful for different RNA comparison tasks as shown in Section 2.5. In virtue of the LCS-EPM algorithm presented in Section 2.3, we developed a chaining algorithm for structural ensembles that selects from the computed suboptimal EPMs a non-crossing and non-overlapping subset that can be extended to an alignment. This new EPM chaining variant generalizes the LCS-EPM algorithm used in `ExpaRNA` to cope with more than one EPM ending at the same position [71, 150].

Similarly, the new algorithm recursively fills the holes of all EPMs with other EPMs. For each of the holes a matrix of size $O(|A||B|)$ is computed. In contrast, at each of its entries all EPMs are considered and simply the best score over all EPMs ending at this position is taken. Although we have multiple EPMs ending at the same position, we can still create an ordering $\preceq_{\mathsf{HOLES}}$ over all holes. This restricts the algorithm to two-dimensional matrices as we can guarantee that all smaller holes are processed before a larger one is considered. For EPMs in structural ensembles we can additionally exploit that fact that different (suboptimal) EPMs contain equal holes, i.e holes with similar inside-boundaries. In order to avoid redundant calculation of such holes, we order all holes by size and their boundaries, i.e.

$$h_i \preceq_{\mathsf{HOLES}} h_j \iff (h_i^{RA} - h_i^{LA}) \leq (h_j^{RA} - h_j^{LA}) \wedge h_i^{RA} \leq h_j^{RA}. \tag{3.4}$$

This allows for a simple check whether the current hole is equal to the previous hole and thus we can inherit the score from the previous hole without any recalculation.

Since each EPM ends at exactly one position, the complexity is $O(H \cdot (|A||B| + E))$, where $E$ is the number of input EPMs and $H$ the total number of their gaps.

If we guarantee that $E$ is in $O(|A||B|)$, i.e. there is only a constant number of EPMs ending at each position, the complexity of the chaining is $O(H|A||B|)$ (as in `ExpaRNA`). Whereas the suboptimal traceback does not *guarantee* $E \in O(|A||B|)$, we also implemented and evaluated a heuristic strategy that satisfies the assumption by considering only the best EPM ending at each position.

## 3.6 Results

Our `ExpaRNA-P` approach combines all described steps, i.e. the computation of in-loop probabilities, the enumeration of significant EPMs and the identification of the best non-crossing arrangement of EPMs for two RNA sequences. We implemented all steps of `ExpaRNA-P` in C++. In order to calculate the additional in-loop probabilities we modified the partition function algorithm in a recent version of the Vienna RNA 2.0 library [105]. The dynamic programming algorithm to find all significant EPMs together with the new chaining algorithm is implemented as part of the `LocARNA` package [179].

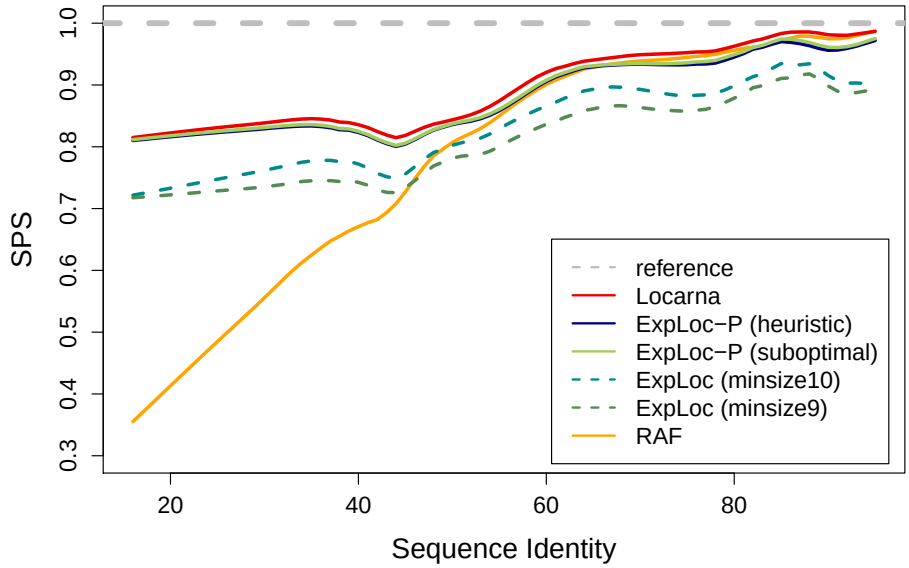


**Figure 3.2.** Alignment quality of ExpLoc-P with respect to the sequence identity on the k2 dataset of `BRAliBase`.

Furthermore we implemented two versions of the traceback of `ExpaRNA-P`: the suboptimal one as described in Section 3.4 and a heuristic version where we consider from each position $(i, j)$ in the $F$ matrix only the optimal EPM ending at that position.

In order to assess the performance of `ExpaRNA-P` in comparison to other alignment tools, we designed the following pipeline like the one we used for `ExpaRNA`: in a first step we compute the significant EPMs with `ExpaRNA-P` and use the chaining algorithm to extract from these EPMs an optimal non-overlapping and non-crossing subset. Then we compute a sequence structure alignment that includes all matches of the chained EPMs. For this purpose, we apply `LocARNA` using the EPMs as anchor constraints. This is faster than computing an unconstrained alignment since each anchor reduces the alignment space (cf. Section 2.4). We refer to this pipeline, i.e. the combination of `ExpaRNA-P` and `LocARNA`, as ExpLoc-P.

We performed a benchmark test on the k2 dataset of `BRAliBase` which contains only pairwise alignments [54, 184]. To measure the quality of the calculated alignment in comparison to the reference alignment, we utilized the compalign score which refers to a sum-of-pairs score (SPS) introduced in this specific form with `BRAliBase` [184]. See Section 2.5.2 for more details. Besides the quality of the results, we also compared the runtime of the different methods. We compared our new approach ExpLoc-P with three other approaches: `LocARNA` without any anchor constraints, `ExpLoc` [71], and `RAF` [39]. Obviously, we include a comparison with the similar `ExpLoc` approach (cf. Section 2.5.2) in order to evaluate the potential of using EPMs resulting from structural ensembles instead of EPMs identified on fixed minimum free energy structures. `RAF` is the currently fastest Sankoff-style sequence structure alignment approach due to its heuristic filtering

**Table 3.1.** Runtime comparison of the different approaches. The speedup factor is measured relative to the speed of LocARNA. The runtime is the total runtime for computing the entire benchmark dataset on a single Opteron 2356 processor (2.3 GHz). For ExpLoc-P the first value in brackets is the time for computing and chaining the EPMs and the second one the runtime for the subsequent LocARNA alignments.

| | LocARNA | ExpLoc-P (heuristic) | ExpLoc-P (suboptimal) | ExpLoc (minsize 10) | ExpLoc (minsize 8) | RAF |
|---|---|---|---|---|---|---|
| speedup | 1 | 6.0 | 4.9 | 4.4 | 5.4 | 15.6 |
| total time | 14.3h | 2.4h (0.4h+2h) | 2.9h (0.4h+2.5h) | 3.2h | 2.6h | 0.9h |

based on sequence alignments. We instantiated the scoring of ExpaRNA-P (see Equation 2.3) by $\sigma(i,k) = 1$ and $\tau(i,j,k,l) = 5(\Pr\{(i,j)|A\} + \Pr\{(k,l)|B\}) + 2$. In addition to the presented scoring, we add a reward of $5\Pr\{(i,j)|(i+1,j-1)|X\}$ $(X \in \{A,B\})$ for each stacking in the EPM. In the suboptimal traceback, we enumerate EPMs that have a score of at least 90 and a score difference of less than 20 to the optimal EPM. Furthermore, we set $\theta_1 = \theta_2 = 0.01$ and $\theta_3 = 0$.

Figure 3.2 shows the result for the compalign score with respect to the sequence identity on the k2 dataset of BRAliBase. LocARNA achieves the best results at the expense of the highest computation time. Table 3.1 lists the speedups of the other approaches compared to LocARNA. Our novel combined approach ExpLoc-P achieves with both the heuristic and the subobtimal traceback almost the same quality as LocARNA but is 6 and 4.9 times faster, respectively. The best alignment quality that could be obtained with ExpLoc was achieved with parameter minsize $= \gamma = 10$ which refers to the minimal size of used EPMs. Even for this optimal setting the quality of the result is significantly lower than the one for LocARNA alone and ExpLoc-P. Additionally, the speedup for this setting is only 4.4 which is also less than both speedups for ExpLoc-P. With minsize $= 9$, the speedup of ExpLoc is comparable to ExpLoc-P but the quality declines much more. RAF achieves the best speedup of 15.6 but the drawback of the sequence alignment based heuristic filtering, which causes this speedup, is clearly visible: for sequence similarities below 50% the quality drops tremendously. This indicates that RAF is only successful on instances where sequence information alone is sufficient to get already reasonable alignments. In summary, this means that our novel tool ExpLoc-P finds the best trade-off between alignment quality and speedup and is robust regarding the alignment quality for the whole range of sequence identities.

In order to analyze the quality of ExpLoc-P further, we investigated whether the compalign scores of ExpLoc-P and LocARNA without constraints do correlate well. We found a high correlation of 0.85. This indicates that the six-times faster ExpLoc-P pipeline can replace LocARNA in RNA clustering approaches [86, 126, 179].

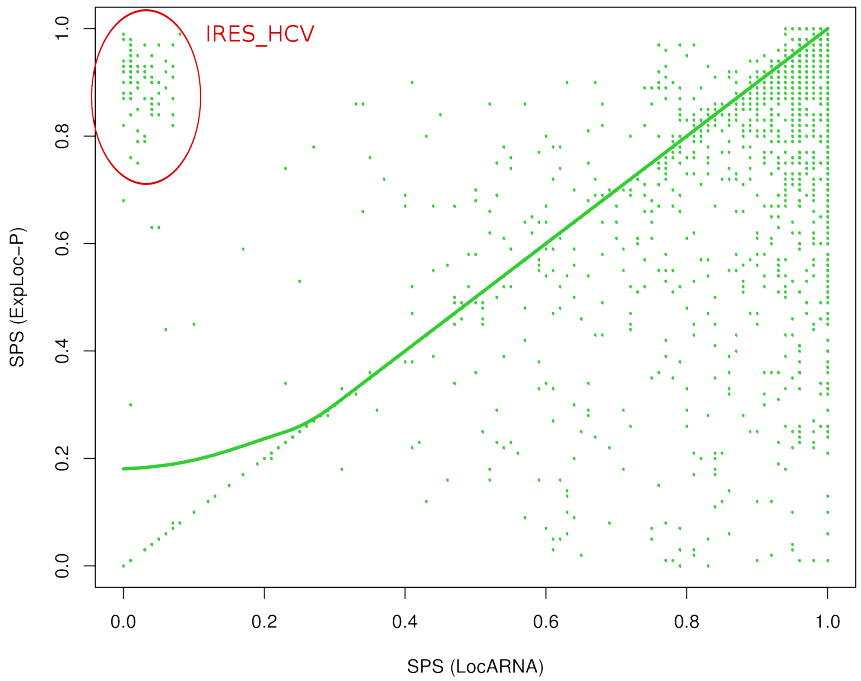


**Figure 3.3.** Correlation between the compalign score of `LocARNA` and ExpLoc-P. The green line indicates a lowess fit for the `BRAliBase` dataset. A subset of the IRES_HCV family shows significantly better alignments with ExpLoc-P than with `LocARNA` alone.

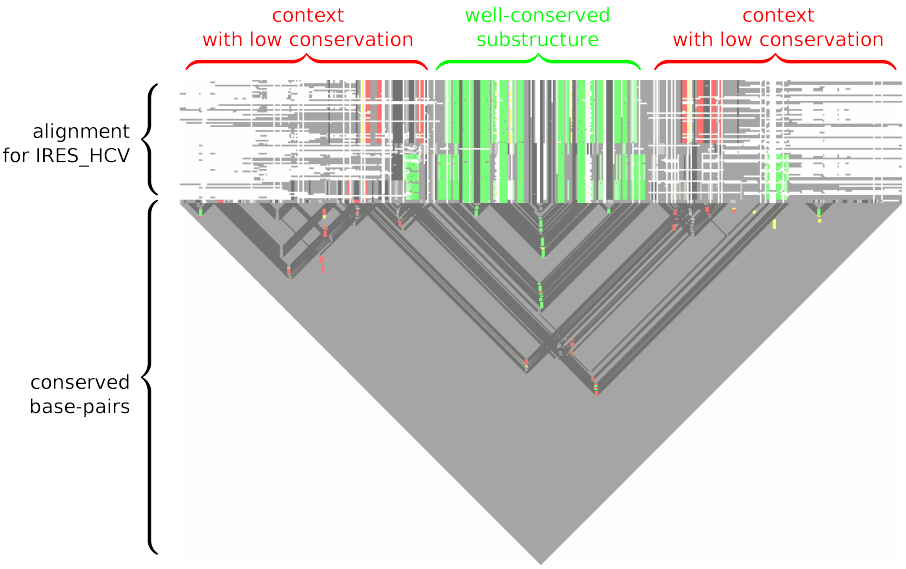


**Figure 3.4.** IRES_HCV (RF00061) seed alignment. The visualization is taken from SARSE (http://sarse.ku.dk/Rfam_sarse/sno.html). It displays a highly conserved with a low conserved context.

Moreover, we observe several benchmark instances where the ExpLoc-P pipeline significantly outperforms LocARNA. Notably, as can be seen in Figure 3.3, there is a prominent cluster of improved benchmark instances. Surprisingly, all these alignments belong to the family IRES_HCV (RF00061) leading to an overall improvement for this family (compalign score 0.89 compared to 0.82 on average). For the subset of alignments where ExpLoc-P was better than LocARNA, we found a significant drop (compalign score 0.22 for LocARNA compared to 0.89 ExpLoc-P), indicating that LocARNA was not able to align these sequences without the help of anchor constraints. Manual inspection of the multiple alignment revealed that this family is one of the rare cases where there is no global conservation. Instead, there is a highly conserved substructure and a less conserved context (see Figure 3.4). In this case, determining a well-conserved structure first and using this subsequently as an anchor constraint is clearly a strategy to improve the overall alignment quality.

## 3.7 Discussion

We presented in this chapter ExpaRNA-P, a novel method for RNA comparison based on significant motifs derived from RNA structural ensembles. ExpaRNA-P lifts the concept of exact matching substructures (EPMs) introduced with ExpaRNA in Chapter 2 from fixed input structures to RNA structural ensembles. We call EPMs over structural ensembles significant EPMs (cf Section 3.2) if they are both similar and very likely. This generalization is highly advantageous for many applications since the correct or functional RNA secondary structure is often unknown. Reverting to a single fixed structure, e.g. the minimum free energy structure, is erroneous and leads to wrong results whereas an ensemble based approach includes very likely as well as different structural conformations. However, allowing structural variants usually implies a runtime complexity of at least $O(n^4)$ (e.g. in FoldAlign or LocARNA) which in turn would eliminate most of the benefits obtained by ExpaRNA. Thus the main technical challenge we solved with ExpaRNA-P is to keep ExpaRNA's fast quadratic runtime complexity while dealing with a much larger structure space for each RNA. We achieved this by introducing novel in-loop probabilities for RNA secondary structures along with a novel sparsification technique.

As part of ExpaRNA-P we developed a fast dynamic programming algorithm which identifies significant EPMs for two given RNA structure ensembles (cf. Section 3.4). Our algorithm maintains important EPM features like finding only maximally extended EPMs. Furthermore we can use a suboptimal traceback to identify structural varying EPMs for a specific local region. Hence, our method supersedes the previous Backofen and Siebert algorithm used in ExpaRNA to determine EPMs [5]. Due to the additional probabilities we only compute matrix entries for likely loops and likely unpaired position resulting in a reduced quadratic complexity (cf. Section 3.4.1 and 3.4.2). The novel in-

loop probabilities are calculated with a modified version of McCaskill's partition function algorithm (cf. Section 3.3). Although the partition function algorithm has a dominating runtime complexity of $O(n^3)$, we can neglect this in scenarios where it is only done once as precomputation step, such as clustering of RNA sequences or scanning a query sequence against a database.

Our evaluation evidences the high quality of EPMs based on RNA structure ensembles because our new pipeline ExpLoc-P outperforms in all benchmarks EPMs from fixed RNA structures. We designed a benchmark similar to `ExpaRNA` that utilizes EPMs as anchor constraints for structure alignments and we achieved both higher quality and higher speedups with `ExpaRNA-P`. First, increased speedups indicate that a higher number of EPMs are used as anchor constraints. Secondly, the improved alignment quality shows that the used EPMs are very accurate and highly similar to `LocARNA` alignments without constraints. This as also shown by the high correlation between the compalign scores of `LocARNA` and ExpLoc-P in Figure 3.3.

Furthermore, we can even illustrate cases where ExpLoc-P alignments are superior to standard `LocARNA` alignments. For example, our ExpLoc-P strategy leads to a clear improvement of the overall alignment quality for the Rfam family IRES_HCV. Since our used benchmark is biased towards global alignment, this situation probably occurs more often in practice than seen in `BRAliBase`. Thus, it is conceivable that the ExpLoc-P approach is an improvement in case where less well-defined families have to be aligned, as in cluster-based approaches for detecting new structural RNA families on a genome-wide scale. Here the boundaries of ncRNAs are typically loosely defined and pure global comparison approaches can fail. For such cases, it can be beneficial to determine well-conserved local structures first and use them subsequently as anchors. Hence, our `ExpaRNA-P` approach can improve the comparison of RNA families that share local and global similarity.

As future work, we will investigate relaxations of the notion of exact patterns to further improve the results. In particular, the same recursions can be used to detect patterns that allow mismatches of base pairs or unpaired bases. In addition, we will explore further parameter optimizations, especially for the used thresholds of in-loop probabilities. Another useful extension could be a method for multiple RNA comparison that is based on EPMs. For example, this could be achieved by feeding significant EPMs into a `T-Coffee`-like primary library, similarly to `LocARNAte` [123, 125]. The additional local EPM information is then used to construct improved multiple alignments with a standard progressive alignment approach.

Furthermore, EPM based anchor constraints could be used to improve other alignment tools, like `RAF`. While for `LocARNA` the constraints yield a considerable speed-up, in `RAF` they could improve the quality that is poor for low sequence similarity. The score of the

chained EPMs could also be used as a distance measure for clustering approaches. This would speedup the clustering process since the expensive computation of full structure alignments can be avoided.

Above all, we want to stress the impact of the presented state-of-the-art techniques for the field of RNA bioinformatics in general. There are already constitutive methods like SPARSE which build upon the described structure sparsification in order to reduce the time complexity from $O(n^4)$ down to $O(n^2)$ for Sankoff-style simultaneous alignment and folding methods [181].

# Chapter 4

# GraphClust: Efficient Structural Clustering of Local RNA Secondary Structures

With `GraphClust` we present a novel and very efficient solution for sequence and structure-based clustering of thousands of RNA sequences. We first review the need and difficulties of clustering strategies for RNA sequences. In the following, we introduce our efficient, alignment-free, approximate nearest neighbor search procedure that combines several algorithmic enhancements to achieve the required linear time complexity. Moreover, we give details on the used graph kernel, RNA structure and feature encoding and cluster refinement procedures. In Section 4.3, we present our `GraphClust` pipeline that combines all improvements in a ready-to-use clustering pipeline. Finally, we evaluate and discuss our approach and investigate its potential on finding novel clusters of ncRNAs. This chapter is based on the publication Heyne et al. [72].

## 4.1 The `GraphClust` approach

Tiling arrays and high-throughput sequencing have strikingly driven the discovery of novel non-protein coding RNAs [45]. Complementing advances in the computational *de-novo* prediction of ncRNAs have uncovered a wealth of signals for potentially novel ncRNAs in genomic sequences from basically all kingdoms of life. Examples include metazoans with thousands of predicted ncRNA signals in human, fish or insects [140, 141, 170]. Consequently, the prediction, comparison, and (functional) annotation of ncRNAs are major tasks of current RNA research. Annotation of ncRNAs, however, is still not part of generic annotation pipelines of genome or next-generation sequencing data and precise functional annotations of the majority of identified and predicted RNA transcripts remain elusive. There are reports of up to 450.000 predicted ncRNAs only in the human

genome [132]. The exact numbers and even the magnitude are of course matter of discussion, but it is reflecting the current problems in the analysis of whole transcriptome data. Thus, there is an urgent need for efficient algorithms identifying novel ncRNAs in the exponentially growing bulk of sequence data, including complete genomic sequences as well as whole transcriptomes.

One of the major reasons for the multitude of unannotated ncRNAs is that in contrast to protein-coding genes, ncRNAs belong to a diverse array of classes with vastly different structures, functions, and evolutionary patterns [17]. Albeit their heterogeneity, ncRNAs can be divided into *RNA families* according to inherent functional, structural, or compositional similarities. Today, the `Rfam` 11.0 database already lists 2,208 different RNA families [23]. In contrast to RNA families, an *RNA class* groups together ncRNAs whose members have no discernible homology at the sequence level, but still share common structural and functional properties. Prominent examples are the two distinct classes of snoRNAs (box H/ACA and box C/D) and micro RNAs (miRNAs).

Since an RNA class consists of RNA molecules with similar structure and function, clustering according to sequence-structure similarity has become a generally accepted scheme for ncRNAs annotation. The quality and complexity of the clusters is however largely determined by the pairwise sequence comparison method. The most generic methods, as introduced with `LocARNA` [179] and `FoldAlign` [163], use derivatives of the full Sankoff algorithm [147] of simultaneous alignment and folding. They suffer, however, from a very high computational complexity (at least $O(n^4)$ in time), and thus, they are limited to relative small sequence sets. As truly stated by Gorodkin et al. [60]: *"Even using substantially more sophisticated techniques, genome-scale ncRNA analyses often consume tens to hundreds of computer years. These high computational costs are one reason why ncRNA gene finding is still in its infancy."* For this reason, many approaches use different heuristics to achieve a reasonable trade-off between time and quality. Oversimplifying and without completeness, given the variety of approaches present in literature, one can distinguish two main classes of clustering approaches. The first class uses simplifications in the representation of structures and consider only single predicted structures [91, 135]. These comparison approaches heavily depend on the correctness of the structure, although computational prediction of secondary structures are known to be error prone. Other approaches use simplified structural models [148]. The `EvoFam` work can also be listed to some extend under this class since they use an approximate measure between two structural RNAs' SCFG models for clustering hits found by `EvoFold` [126, 128].

The second class uses sequence information as prior knowledge to speed up the computation. In the simplest case, sequences are first clustered by sequence-alignment [96, 153]. These alignments are then used to predict conserved consensus structures using approaches like `RNAalifold` [12] or `PETfold` [151]. A similar overall scheme is e.g. applied in `CMfinder` [188], which determines a consensus motif from a cluster of unaligned

sequences. Yet another set of tools employs a sequential encoding of structural information [164]. Finally, one can use the information from an ensemble of sequence alignments to speed up the computation [144]. The major problem, however, is that the sequence of an ncRNA evolves much faster than its structure, which implies that in most of the cases, no homology on the sequence level is detectable. Indeed, it has been shown that family assignments of structured RNAs obtained from sequence alignments at pairwise sequence identities below 60% are often wrong [54].

With GraphClust we propose an efficient approach for clustering very large sets of RNA sequences according to sequence and structure information. The size of current datasets exclude the use of alignment-based techniques. Hence, to achieve the required efficiency, we propose an alignment-free hashing technique over a novel encoding for RNA structures. Although there are alignment-free methods for comparison of RNA with respect to sequence and structure (e.g. Gan et al. [53], Liu and Wang [103]), we are not aware of any alignment-free method capable to perform RNA sequence-structure comparisons on hundred of thousands sequences. For this purpose, we extend a fast graph kernel technique that has been recently developed for chemoinformatics applications and we adapt it to detect similarities between RNA secondary structures [33]. The key novelty that we introduce lies in an explicit representation of the associated sequence-structure information which we encode as sparse vectors in a very high dimensional space. This allows us to use efficient locality sensitive hashing methods to accurately retrieve dense data regions with a complexity that is linear in the number of sequences $N$, rather than quadratic as it would be with both alignment methods or standard kernels that work with implicit representations.

We have integrated the approach in a ready-to-use pipeline for large-scale clustering of putative ncRNAs. After the efficient clustering step, we increase the quality of the resulting clusters by employing alignment methods to filter away inconsistent elements. GraphClust has been successfully evaluated on known ncRNA classes and compared against existing approaches, i.e. the complete LocARNA-pipeline and RNAsoup [86]. We show that our clustering is of high quality yielding reliable clusters of homologous RNA sequences. Due to the algorithmic improvements presented here, we achieve a striking performance speedup (from years to days for serial computation and even hours when parallelized) outperforming any of the existing approaches. We applied our method to six heterogeneous large-scale data sets containing more than 220,000 sequence fragments. First, we analyzed computationally derived predictions of short ncRNAs lacking reliable class assignments. Next, we searched for local structural elements specific to experimentally validated lincRNAs. We observed enriched GO-terms for lincRNAs containing predicted local motifs likely functionally "linc"-ing these transcripts to vital processes of the human nervous system. In general, both application scenarios aim at the detection of novel structural non-coding RNAs.

### Related Work

Concerning the calculation of sequence-structure similarity, which is the basis for the clustering of ncRNA in putative RNA-classes, one can roughly distinguish alignment and kernel methods. For the alignment-based methods, several different algorithmic approaches have been introduced in the past to determine structural similarities and to derive consensus structure patterns for RNAs that are too diverse to be alignable at sequence level. One class, with `MARNA` and `RNAforester` [74] being their main representatives, uses a given or predicted secondary structure as additional input aligning the sequences. However, these approaches heavily depend on the correctness of the given structure, and computational prediction of secondary structures are known to be error prone. In contrast, derivatives of the Sankoff algorithm [147] solve the problem of simultaneous folding and alignment. In its full form, the problem requires $O(n^6)$ CPU time and $O(n^4)$ memory, where $n$ is the length of given RNA sequences. This complexity is prohibitive for most practical problems. There are two variants of the Sankoff algorithms. Programs such as `FoldAlign` [59, 66], `Dynalign` [114], and `Stemloc-AMA` [18] implement an energy model for RNA that is evaluated during the alignment computation. In contrast, `PMcomp` [78] and `LocARNA` [179] use a full-featured energy model in their pre-computation step by determining a matrix of base pair probabilities using McCaskill's algorithm [117] for each input sequence. During the alignment process, base pair probabilities are used to assess the similarity of the secondary structures. This strategy reduces the time complexity to $O(n^4)$ for pairwise alignments, and thus, improves the overall time needed for the clustering. The approaches `FoldAlignM` [163] and `RAF` [39] followed the same filtering principle. The latter approach is interesting because it combines sparsity on the structure and sequence level (this combination first seen in `Stemloc` [18]) with a lightweight scoring scheme that significantly improves its efficiency over other Sankoff-style methods [39].

There are several kernel method that have been used for RNA (see e.g. Wang and Wu [167] for a review). Basically, RNA kernel methods that compare RNA sequence and structure without resorting to a single predicted structure can be divided into three classes. The most complex one resembles more full sequence-structure alignment [145]. On the other hand, the complexity of $O(n^4)$ is same as `LocARNA`. Later, a simplified representation of RNA structures as DAG was introduced that gave rise to a $O(n^2)$ comparison method [148]. The second class consist of approaches, which perform pure structure comparison without using sequence information at the same time [89]. The last class use an extended sequence alphabet that encodes also some structural properties. This trick was introduced by the alignment method `STRAL` [36] and is shared with approaches that use predicted structures like `Triplet-SVM` [187], an SVM-based method to predict miRNAs, and others [92, 119].

## 4.2 Efficient Alignment-Free Structural Clustering of RNAs

In the following, we provide the algorithmic details necessary for an efficient structural clustering of RNA sequences using kernel and hashing techniques. First, we propose to sample a small number of probable, but sufficiently different, structures for each RNA sequence. We then encode each structure as a labeled graph preserving all information about the nucleotide type and the bond type (i.e. backbone, binding and stacking). In this way, structural variants of an RNA sequence can be represented as a graph with several disconnected components. This would be already sufficient to compute the similarity between the representative graphs using a graph kernel. However, to avoid a quadratic number of comparisons, we first extract an explicit vector representation for each graph, and then build an inverse index on a compressed representation obtained via hashing techniques. This allows us to retrieve in constant time the nearest neighbours sequences for any given query structure. We evaluate each neighbourhood and select as candidate clusters those that contain very similar elements.

### 4.2.1 Graph Encoding for RNA Secondary Structures.

The encoding of structural information is crucial for finding clusters of RNAs with similar sequence and structure. However, verified structure information is not available in most cases and one have to resort to RNA structure prediction methods. In case of single sequences, minimum free energy based secondary structure prediction has been shown to be error prone [40]. For this reason we want to use a representation of the entire ensemble of low energy conformations. Resorting to a complete enumeration of near-optimal structures would yield a tremendously large number of structures [186]. Instead, we apply the abstract shape analysis method by using `RNAShapes` [56]. Here one analyzes the complete folding space using McCaskill's partition function but classifies the structures *a priori* into folding topologies, called shape types. This allows us to represent each RNA by sufficiently different, but probable, RNA secondary structures. Within each shape type, the most stable structure, called shrep, is selected. Furthermore, shreps can be sampled within a small energy difference to the optimal energy to ensure their stability. `RNAShapes` provides five shape types that describe the different levels of abstraction for an RNA secondary structure. In the most accurate type (shape type 1), all loops and unpaired regions of a secondary structure are denoted, whereas in the most abstract level (shape type 5), only branches of multiloops and external loops are depicted, but no unpaired regions. Figure 4.1 gives an example of different shreps for an RNA molecule.

In addition we address an important issue regarding true sequence boundaries. In benchmark tests for sequence-structure alignments like `BRAliBase`, sequences are usually given as full-length transcripts with correct boundaries. However, this setting is quite unlikely in practical application scenarios, where one often has either a partial transcript

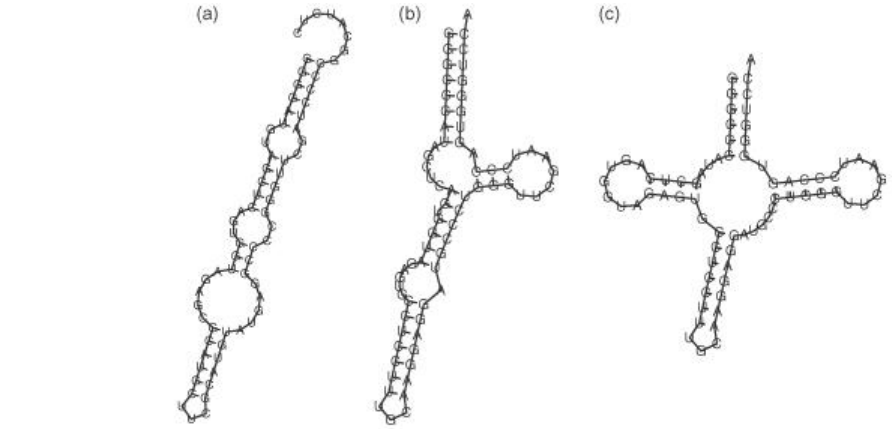


**Figure 4.1.** Shape representation of an RNA secondary structure obtained by `RNAShapes`. The figure shows the three most stable shreps derived from a tRNA sequence at the most abstract shape type. Interestingly, the mfe structure (a) as well as the second-best shrep (b) do not fold into a correct tRNA. The biological relevant cloverleaf structure (c) shows up at the 3rd position (d) with only a small energy difference to the mfe structure. This approach allows us to encode correct structural features from a very limited number of folding hypothesis.

(e.g., from RNA-seq data), or just transcripts with wrong boundaries. The fact that secondary structure prediction may change when additional context is considered, adds considerable noise to the task of identification of functional families. In order to deal with this issue we consider several subsequences, obtained from the original sequence, as overlapping windows of different sizes (parametrized by a set of values $W$) and at different starting locations (parametrized by the overlap value $O$). This facilitates the capturing of both local hairpins and larger multi-loop structures as the used window sizes influences the locality of the structural features that we encode, and thus, our method is aware of. This also reduces the problem that global folding becoming inexact for longer sequences [11, 99]. A local folding window is also better suited to capture the correct folding of e.g. cis-regulatory elements in UTRs or miRNAs in introns (mirtrons) [142].

Finally, for each subsequence in each window we consider the set of $l$ most representative structures (shreps) obtained by `RNAShapes` and encode them as disconnected components. The vertex set for each of the graph components is derived from the sequence of nucleotides, while the edge set encodes both the nucleotide sequence adjacency information (i.e. the RNA backbone) and the pairwise binding status (i.e RNA base pairs). In addition, an extra set of vertices (and corresponding edges) is introduced to better match biological knowledge on important RNA motifs, namely, for each stacking

base pairs quadruplet, an additional vertex is added (and linked to each of the four nucleotides involved; in this way the notion of neighbourhood subgraph (see next Section) coincides with that of a sequence of stacking base pairs (see Figure 4.2).

The graph encoding of RNA secondary structures via `RNAShapes` was developed in collaboration with Sita Lange and Daniel Maticzka.

### 4.2.2 Graph Kernel for Local RNA Motifs

In order to deal with entities represented as graphs, graph kernels of several types have been proposed [166]. Graph kernels compute a similarity between graphs using the so-called kernel trick, i.e they first employ an implicit mapping of graphs or subgraphs into a very high-dimensional vector space and then utilizing an appropriate dot product function as similarity measure. For our approach we start from the recently introduced fast kernel called "Neighborhood Subgraph Pairwise Distance Kernel" (`NSPDK`) [33], since this kernel is suitable for large datasets of sparse graphs with discrete vertex and edge labels.

We now review the `NSPDK` graph kernel and describe the similarity notion that it induces. The `NSPDK` is an instance of a decomposition kernel [65], i.e., a composite kernel that operates over all possible "parts" defined by a given relation. In this case, the parts are *pairs* of special subgraphs, called "neighborhood" subgraphs. More formally, for a given graph $G = (V, E)$, and an integer $r \geq 0$, let $N_r^v(G)$ denote the neighborhood subgraph, i.e. the subgraph of $G$ rooted in $v$ induced by the set of vertices at distance not greater than $r$. The *neighborhood-pair* relation $R_{r,d}$, is defined to hold when the distance between the roots of two neighborhood subgraphs of radius $r$ is exactly equal to $d$. Note, the distance between two vertices $v, u$ is the number of edges of the shortest path between $u$ and $v$. `NSPDK` defines a kernel $\kappa_{r,d}$ as the decomposition kernel on the relation $R_{r,d}$, i.e.

$$\kappa_{r,d}(G, G') = \sum_{\substack{A,B \in R_{r,d}^{-1}(G) \\ A',B' \in R_{r,d}^{-1}(G')}} \mathbf{1}(A \cong A')\mathbf{1}(B \cong B') \tag{4.1}$$

where $R_{r,d}^{-1}$ is the inverse of the relation $R_{r,d}$ and indicates all the possible pairs of neighborhood subgraphs of radius $r$, whose root vertices are at distance $d$ in the given graph $G$, $\mathbf{1}$ denotes the indicator function and $\cong$ the isomorphism between rooted graphs [33].

Please note that the indicator function $\mathbf{1}(Q)$ returns 1 when its argument $Q$ evaluates to true, and zero otherwise. Furthermore, the rooted isomorphism requires, in addition to standard isomorphism, that the roots of the two graphs are mapped to each other.

The (non-normalized) `NSPDK` is finally defined as the sum of all the kernels for all radii and all distances:

$$K(G, G') = \sum_r \sum_d \kappa_{r,d}(G, G'). \tag{4.2}$$

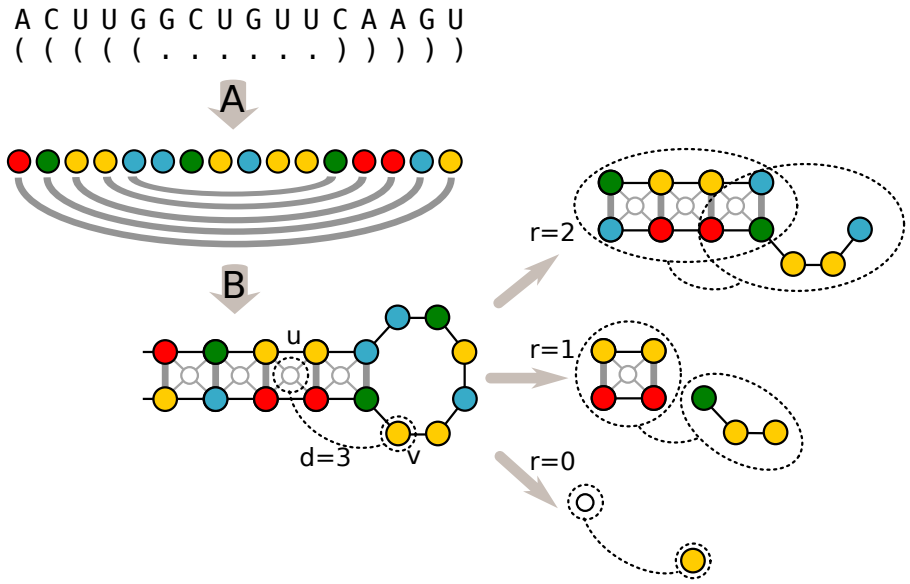


**Figure 4.2.** RNA secondary structure encoding and Graph Kernel Features: Top A) The graph encoding preserves the nucleotide information (vertex labels) and the base pairs (edge labels), here depicted with different colors. B) Additional vertices are inserted in order to induce features related to stacking base-pairs quadruplets (thin light gray vertices at the center of each stacking pair). Right: Example of features induced by the graph kernel `NSPDK` for a pair of vertices $u, v$ at distance 3 with radius 0,1,2. Neighbourhood graphs are enclosed in dashed ovals.

For efficiency reasons `NSPDK` considers the zero-extension of $K$ obtained by imposing an upper bound on the radius and the distance parameter:

$$K_{r^*,d^*}(G,G') = \sum_{r=0}^{r^*} \sum_{d=0}^{d^*} \kappa_{r,d}(G,G'), \tag{4.3}$$

i.e. they limit the sum of the $\kappa_{r,d}$ kernels for all increasing values of the radius (distance) parameter up to a maximum given value $r^*$ ($d^*$) [33]. Furthermore a normalized version of $\kappa_{r,d}$ is suggested, that is:

$$\hat{\kappa}_{r,d}(G,G') = \frac{\kappa_{r,d}(G,G')}{\sqrt{\kappa_{r,d}(G,G)\kappa_{r,d}(G',G')}}, \tag{4.4}$$

to ensure that the features induced by all values of radii and distances are equally important regardless of the overall dimensionality of the induced feature space [33].

In our case of graphs originating from encoded RNA secondary structures, the type of features that the `NSPDK` is considering is depicted in Figure 4.2. The neighborhood subgraphs nicely represent local RNA structures that can be relevant for the biological function of the corresponding RNA (see e.g. Figure 2.1). Moreover, the obtained neighborhood subgraphs are highly descriptive to model effects of RNA evolution. In case distance $d$ is bigger than radius $r$, the neighboring subgraphs do not overlap which can

be treated as a structural "don't care" region where mutations (mismatches or indels) are allowed. Figure B.1 in the appendix give further details on the induced features by NSPDK. In order to get a meaningful graph encoding of the RNA secondary structure, we distinguish between backbone bonds and base-pairs by using different edge labels. Furthermore, we introduce a special stacking node type to specifically obtain subgraphs with stable stacking base pairs. For each stack of two base pairs, a node that connects to each of the four participating nucleotides is integrated (see Figure 4.2). We also use directed edges to better reflect the RNA orientation. As typical values for RNA clustering we consider $r \leq 3$ and $d \leq 4$.

### 4.2.3 Fast Subgraph Encoding

Once we can decompose a graph in its neighborhood subgraphs, we are faced to the problem of running an exact isomorphism test on all parts to either count similar subgraphs of a single graph $G$ or to compute the similarity $\hat{\kappa}_{r,d}(G, G')$. This test is usually computationally expensive, however, NSPDK substitutes the test with a more efficient graph invariant computation. The core idea is to devise an efficient graph serialization procedure, such that two isomorphic graphs can be reduced to an identical string. The encoding is achieved using a technique based on the distance information between pairs of vertices and can be computed in linear time w.r.t. the vertex size of the component on sparse graphs with bounded degree. Finally, an iterative hashing procedure can map the string encoding into an integer code (see Costa and Grave [33] for further details). Figure 4.3 illustrates the procedure of the graph encoding.

Clearly, this encoding is beneficial because the isomorphism test between two graphs can be reduced to the equality test between their integer codes. Note that the string encoding introduce potential feature "collisions" (i.e. different subgraphs can induce the same strings or features), although in practice the collision event is negligible by using a fixed, but high dimensional feature space (e.g. 30 bit integer numbers).

### 4.2.4 Explicit Feature Representation

In contrast to standard kernel approaches as well as the original NSPDK, here we choose to materialize the implicit feature encoding. This will turn out to be the key step to obtain an efficient, sub-linear clustering procedure. The benefit shows up in successive phases, where we need to build an index over these features to collect nearest neighbors.

More precisely, we make use of the integer code for the invariant graph encoding as a feature indicator and explicitly list all features. In this way we can interpret the integer associated to each feature (i.e. each pair or neighborhood subgraphs of radius $r$ whose roots are at distance $d$) as the feature key and the (normalized) count of occurrences as its value. This allows us to obtain an explicit feature encoding for a given graph $G$ as

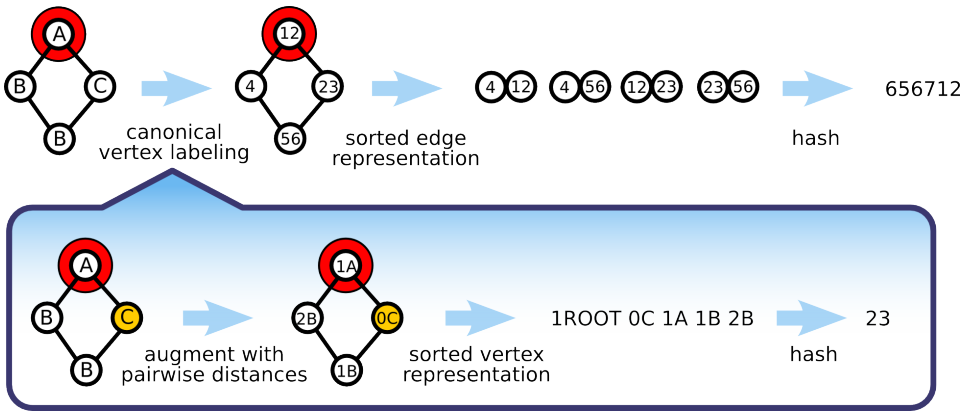


**Figure 4.3.** Graph hashing procedure used by NSPDK. Top: First, all vertex labels are replaced by canonical labels. Then all edeges are sorted according to their vertex labels. The obtained serial representation can be hashed into an integer number. Bottom: The key step in this procedure is the canonical relabeling. The figure was kindly provided by F. Costa.

a sparse vector in $\mathbb{R}^m$ (with a very high dimension $m$). The feasibility of the approach lies in the fact that the encoding does not produce an exponential number of features, as it would happen with most graph kernels that enumerate all possible general subgraphs. Instead NSPDK limits the number of features to $O(r^* d^* |V(G)|^2)$ pairs of neighborhood subgraphs, i.e. one feature for each pair of vertices times each possible combination of values for the radius and the distance. Note that typically $r^* \in [0, 5]$ and $d^* \in [0, 10]$ and hence the multiplicative factor ($\approx 50$) is reasonable.

Moreover, for sparse graphs the number of vertices that are reachable within fixed small distance is typically small (depending on the average degree) so that the dependency on the vertex set size can be more tightly approximated by $O(|V(G)|)$. As a result, each graph is mapped into a sparse vector that lives in a very high dimensional feature space but that has a number of non-zero features which is linear in the number of the graph's vertices. As the computation of the encoding for each neighbor subgraph can be precomputed and cached, the practical complexity for the overall encoding of a graph is linear in its vertex set size with small hidden multiplicative coefficient. These properties make NSPDK one of the fastest graph kernels available and suitable for our RNA structural clustering approach [33].

### 4.2.5 Efficient Nearest Neighbor Determination Using Locality Sensitive Hashing

In recent bioinformatics application scenarios dataset sizes become large easily and contain $> 10^4$ or $> 10^5$ instances. Algorithms that directly make use of pairwise distance or similarity information become infeasible as they inevitably exhibit a quadratic complexity. For example, approaches like RNAclust [179], which use an $O(n^4)$ sequence-structure

alignment step to compute one of the $O(m^2)$ matrix entries, can be reasonably applied only for dataset in the order of $\leq 10^3$ RNAs. Thus, the main goal of our `GraphClust` approach is to avoid computation of the full similarity matrix by e.g. expensive pairwise sequence structure alignments, while still obtaining high quality clusters of structured RNAs.

The key idea of our solution to this problem is to formulate the clustering problem in terms of approximate nearest neighbor queries which can be answered efficiently (sub-linearly). In detail, given a set of $n$ instances $P = \{p_1, \ldots, p_n\}$ in a metric space $X$ with a distance function $d$, a neighborhood query is a procedure that returns the instance in $P$ closest to a query instance $q \in X$. The nearest neighbor search problem is formulated as a dataset pre-processing that allows nearest neighbors queries to be answered efficiently. This problem, first posed in the 1960s by Minsky and Papert [118], admits well understood solutions when $X = \mathbb{R}^m$ with small $m$ [44]. However, for high dimensional cases, an efficient solution was proposed by Indyk and Motwani [83]. The key idea in high dimensional space is to relax the requirements, ask for $\epsilon-$approximate nearest neighbor queries, and use *locality-sensitive hashing* techniques. The $\epsilon-$approximate nearest neighbor query returns an instance $p$ for a given query $q$ such that $\forall p' \in P$,

$$d(p, q) \leq (1 + \epsilon)\, d(p', q). \tag{4.5}$$

A locality-sensitive hash function is a hash function such that the probability of collision is higher for objects that are close to each other than for those that are far apart. As locality-sensitive hash function we choose the min-hash function as it approximates the natural similarity notion defined by the Jaccard index [21].

However these techniques require instances to be represented as sparse binary vectors rather than sparse real vectors. We therefore binarize all instances from $\mathbb{R}^m \mapsto \{0, 1\}^m$ setting to 1 all non-null components, i.e. all occurring feature indicators of an instance (RNA). Let $x, z \in \{0, 1\}^m$ be two instances of binary vectors, the Jaccard similarity between the two instances is defined as

$$s(x, z) = \frac{|x \cap z|}{|x \cup z|}, \tag{4.6}$$

i.e. the ratio of the number of features that the instances have in common over the overall number of features. We build a min-hash function starting from a set of random hash functions $f_i : \mathbb{N} \mapsto \mathbb{N}$, i.e. functions that map integers randomly (but consistently) to integers. In our case the integers of the domain/co-domain represent feature indicators for subgraphs. These functions must be independent and satisfy: $\forall x_j \neq x_k, f_i(x_j) \neq f_i(x_k)$, and $\forall x_j \neq x_k, \Pr(f_i(x_j) \leq f_i(x_k)) = \frac{1}{2}$. The min-hash function derived from $f_i$ is defined as $h_i(x) = \arg\min_{x_j \in x} f_i(x_j)$, i.e. the first feature indicator under a random permutation of the feature order. With other words, the min-hash functions returns that element which

has the minimal value after a consistent re-mapping. With a *random* hash function we denote the fact that there are different functions which hash a specific feature indicator to different, independent values.

A rather surprising (and for us very useful) fact is that a min-hash collision is an unbiased estimator of the Jaccard similarity:

$$\Pr(h_i(x) = h_i(z)) = \frac{|x \cap z|}{|x \cup z|} = s(x, z), \tag{4.7}$$

i.e. the probability to select as the minimum feature indicator a non-null feature that belongs to both $x$ and $z$ is exactly the fraction of features that $x$ and $z$ have in common over the total number of non-null features of $x$ and $z$. In order to decrease the (high) variance of this estimate one can take $N$ independent min-hash functions and compute the number $n$ of times that $h_i(x) = h_i(z)$. The estimated value $n/N$ is the average of $N$ different 0-1 random variables, which evaluates to one when $h_i(x) = h_i(z)$ and zero in all other cases. The average of these unbiased estimators of $s(x, z)$ is also an unbiased estimator, with an expected error bounded by $O(1/\sqrt{N})$. Equivalently, for any constant $\gamma > 0$ we can compute a constant $N = O(1/\gamma^2)$ such that the expected error of the estimate is at most $\gamma$. For example, with 400 hash functions the estimate of $s(x, z)$ would have an expected error $\leq 0.05$. Note that the expected error relation can be obtained by standard Chernoff bounds for sums of 0-1 random variables [31].

We collect the results of the entire set of min-hash functions in an *instance signature* as the tuple $\langle h_1(x), \ldots, h_N(x) \rangle$. This tuple act like a compressed representation of an RNA and its features. In order to obtain an efficient neighbor search procedure, we build an inverse index that returns all instances with the same min-hash value in $O(1)$. Basically, we build $N$ different inverse indices (for each of the N hash functions) where the keys represent the min-hashed feature indicators. With each key we store a list of instances, representing those instances which have the same min-hash value under the same hash function.

Now we can retrieve the neighborhood of an instance $x$, i.e. of one RNA, by querying the inverse index as follows. Given the $i-$th hash function and a value $\bar{h} = h_i(x)$, i.e. the $i-$th value of the signature of $x$, we collect the set of instances $Z_i(\bar{h}) = \{z \in P : h_i(z) = \bar{h}\}$ stored under the key $\bar{h}$. The approximate neighbourhood $Z$ of an instance $x$ is then induced from the multi-set $Z = \{Z_i\}_{i=1}^{N}$ by combining all $N$ hash functions. Note that when $\gamma$ (or equivalently $N$) is fixed, the complexity to build a single signature is constant and therefore the complexity for building the index is linear in the size of the dataset. Similarly, the approximate neighborhood $Z$ of an instance $x$ can be retrieved in constant time from the inverse index as we combine only a fixed number of lists, each one stored under the key given by the instance signature, from the inverse index. Note that no sorting is required for lists stored in the inverse index.

### 4.2.6 Neighborhood Refinement and Candidate Clusters

Given the described efficient $\epsilon-$approximate nearest neighbor search procedure offered by the min-hash technique, there exists a large variety of methods for clustering [41]. Clearly, in our case not all approximate neighborhoods are equally good and we want to select those that contain clusters of similar RNAs. To this end, we first refine all neighborhoods and employ then a density-based ranking to retrieve only clusters of high quality.

Obviously, a single instance $z \in Z$ can occur multiple times in the approximate neighborhood of $x$ which indicates $x$ and $z$ have multiple min-hash results (and thus multiple features) in common. We use this information to improve the quality of the returned neighbors further. For this purpose, we sort all instances in $Z$ according to their frequency and consider only the $k'$ most frequent elements in $Z$. Furthermore, we re-rank the $k'$ most frequent elements according to their normalized `NSPDK` similarity in respect to $x$ (see Equation 4.4). The refined $k'-$neighborhood $N_{k'}(x)$ contains the $k'$-closest elements of $x$ that are kept for further processing.

Clearly, all refinement steps need to maintain the constant time complexity for each neighborhood. We manage this by retrieving only lists from the inverse index that are limited in size, i.e. where $|Z_i(\bar{h})| \leq \alpha$. The size restriction is necessary as we frequency-sort all instances in the multi-set $Z$. This allows us, in addition, to skip feature indicators that are present in many RNAs, and thus, are rather unspecific. Furthermore, we apply the re-ranking according to the non-approximate `NSPDK` similarity only to the first $k'$ elements, with $k' \leq |Z|$. Reasonable thresholds are, for example, $\alpha := 1000$ and $k' := 100$.

Finally, we define candidate clusters by using a $k-$neighborhood $N_k(x)$ and a notion of data density. Intuitively, we prefer as a candidate cluster a set of closely related RNAs. Choosing a good $k$ is important but not crucial when we use a relatively small $k$, i.e. $k \ll P$. Therefore, we use a $k$ with $k < k'$ to compute the density of $x$. The idea is to rank each instance $x$ according to the density $D_k$ of its $k-$neighborhood $N_k(x)$. We define $D_k$ as

$$D_k(x) = \frac{1}{k} \sum_{z \in N_k(x)} K_{r^*,d^*}(x, z), \tag{4.8}$$

i.e. the average pairwise similarity between $x$ and all elements in its $k-$neighborhood. In practise, we can use the kernel similarities already obtained during the re-ranking for the neighborhood refinement. The candidate clusters are finally obtained from the most dense neighborhoods according to $D_k(x)$. Note that we choose $k < k'$ in order rank a larger fraction of approximate neighbors according to the true `NSPDK` similarity, and thus, to compensate for inaccuracies of the frequency-based ranking.

Additional measures can be used to improve the ranking of high quality neighborhoods. For example, we exclude neighborhoods with a high mutual overlap. Returning the top ranking dense neighborhoods would produce highly redundant sets as the densest instances are likely to be part of the same cluster. To tackle the redundancy issue, we adopt a simple yet effective strategy: the candidate clusters are chosen as the top ranking $k$-neighborhoods provided that the size of their overlap is below a specified threshold $th$ with $th < k$. More specifically, we sort all candidate clusters $c_i$ in decreasing order so that $\forall i < j, D(c_i) > D(c_j)$. We then iteratively build the union of the candidate clusters as $C_j = \bigcup_{i=1}^{j} c_i$ but we greedily discard a candidate cluster $c_k$ if $|C_j \cap c_k| > th$.

In order to achieve further speedups for the clustering step, we utilize only a random sample of neighborhoods. Instead of ranking the entire set of sequences according to their approximate density, we work on a smaller sample extracted uniformly at random. The intuition is that the larger the cluster, the higher is the probability that it will be hit by the sample. In this way, samples of 50% or 25% allow a 2-4 fold speedup while having a high probability to identify at least one of the instances of the underlying high density clusters of size greater than 2 or 4 respectively. Note that the neighborhood queries, the density estimates and the returned neighborhood are computed on the complete dataset, not on the sample.

Furthermore we weight a density $D_k(x)$ by its shared neighborhood similarity, which is defined for two instances as

$$\text{simcos}_k(x, y) = \frac{|N_k(x) \cap N_k(y)|}{k}, \tag{4.9}$$

i.e. the normalized intersection size of the k-neighborhoods $x$ and $y$ [47]. This is also called cosine measure as it is equivalent to the cosine of the angle between the zero-one set membership vectors for $N_k(x)$ and $N_k(y)$. Specifically, shared neighborhood measures are reported to be effective for clustering high-dimensional data [47]. In our case, we obtain the weighting factor for $D_k(x)$ with

$$\omega_k(x) = \frac{1}{k} \sum_{y \in N_k(x)} \text{simcos}_k(x, y), \tag{4.10}$$

i.e. by computing the average shared neighborhood similarity of $N_k(x)$.

In the following, we summarize the procedure to obtain the final ranked list of candidate clusters. First, we select either the complete set or a random sample of instances. We apply then the following steps on each selected instance $x$ (steps 1-4) or on the list of instances (steps 5-8) respectively:

1. collect the approximate neighborhood $Z$ for $x$ from the inverse index,
2. sort all elements in $Z$ according to their frequency in $Z$,

3. rank the $k'$ most frequent elements in $Z$ by their normalized NSPDK similarity and obtain the $k'$-neighborhood $N_{k'}(x)$ where $k' \ll P$,
4. compute the weighted density $\omega_k(x)D_k(x)$ based on the $k$-neighborhood $N_k(x)$ where $k \leq k'$,
5. rank all selected instances according to their weighted density,
6. select the most dense instance and its $N_k(x)$ neighborhood as first candidate cluster,
7. proceed with next ranked cluster, select it only if size of overlap with already selected candidate clusters is below a threshold,
8. continue with 7. until a required number of candidate clusters is found.

Note that we keep the $k'-$neighborhood $N_{k'}(x)$ for all candidate clusters as extended $k-$neighborhood to improve the quality of candidate models later on (cf. Section 4.2.7).

### 4.2.7 Candidate Model and Refinement

A candidate cluster contains a set of $k$ sequences that are deemed similar under the min-hash and NSPDK similarity measure given by the refined $k$-neighborhood $N_k(x)$. Moreover, a candidate cluster contains a fixed number of sequences, whereas the true cluster size is unknown and also variable. To solve this problem, we employ a two step solution. First, we identify a candidate model from a given candidate cluster. Afterwards, we use this model to search the full dataset and find missing cluster members.

In our approach, we create a high quality candidate model by performing a sequence structure alignment procedure of the candidate cluster with existing state-of-the-art tools. This allows to incorporate further domain specific knowledge like compensatory mutations and consensus structure information. For this purpose, we use mainly the LocARNA package that identifies a common secondary structure of a set of RNA sequences and uses e.g. the RIBOSUM scoring to deal with compensatory mutations [93, 179].

More precisely, a cluster or guide tree is created by applying an average-linkage algorithm (UPGMA) to the pairwise distance matrix induced by the LocARNA alignment score. Alternatively, we can also use the pairwise distance matrix obtained from the NSPDK similarity by computing $1 - \hat{\kappa}_{r,d}(G, G')$. This will remarkably speedup the guide tree creation as we get the NSPDK-based distance matrix in only $O(k^2)$ time where $k$ is the neighborhood size. Instead, a LocARNA-based guide tree needs $O(k^2n^4)$ time where $n$ refers to the length of the RNAs. We denote the pairwise distance matrix of a candidate cluster as $A_{ij}$ with $1 \leq i, j \leq k$.

In addition, we employ a strategy to handle overlapping instances that originate from the same input sequence. This happens if input sequences are extracted in a sliding-window approach over long RNA sequences with a shift size lower than the window size.

In this case, overlapping instances share a common sequence part that biases the pairwise distance matrix, and hence, the guide tree. We resolve this by setting all entries $A_{mn}$ to a very high distance if the sequences that correspond to $m$ and $n$ are overlapping.

The guide tree of a candidate cluster is denoted as $T = (V, E)$. With $T(v)$ we indicate the subtree rooted at $v$ with $v \in V$. With $|T(v)|$ we denote the number of leafs in subtree $T(v)$. Given a guide tree $T$, we first identify all subtrees with a certain number of leafs, i.e. we find $M \subseteq V$ such that $min_M \leq |T(m)| \leq max_M$ for all $m \in M$. Each subtree induces a multiple alignment $\mathcal{A}$ with a number of $|T(m)|$ sequences that is computed with the `LocARNA`-package. The pairwise distance matrix for all sequences in $\mathcal{A}$ is denoted with $A^m_{ij}$, where $1 \leq i, j \leq n$ and $n = |T(m)|$. We can rank all subtree alignments according to their quality, which is defined as

$$Q(\mathcal{A}) = \left( \frac{n(n-1)}{2} \sum_{i=1, i<j, i \neq j}^{n} A^m_{ij} \right) \text{mpi}(\mathcal{A}) \ \text{sci}(\mathcal{A}), \tag{4.11}$$

i.e. the alignment quality is the average pairwise alignment score of its leafs in combination with the sequence identity (MPI) and structure conservation index (SCI) of the subtree alignment. Note that the average alignment score is not necessarily inversely proportional to the subtree size, and thus, large subtrees can be ranked higher than smaller ones [143]. As additional quality weights we consider the MPI and SCI, which were successfully used, for example, by `RNAz` to detect conserved ncRNAs [64, 170]. Only the top ranked subtree alignment, which exhibits the best model quality $Q(\mathcal{A})$, is retained.

The multiple alignment $\mathcal{A}$ of each subtree is computed with the sequence-structure alignment tool `LocARNA`. To better accommodate for models that exhibit a local similarity we employ a strategy based on `LocARNA-P` [180]. This method allows for an accurate prediction of ncRNA boundaries via column-wise reliability scores. Based on the reliability signal, `LocARNA-P` then identifies a trusted region as common local motif (i.e. a conserved ncRNA) and we use this region as final candidate alignment. The consensus secondary structure of the alignment is determined with `RNAalifold` [12]. Figure 4.4 shows a guide tree with reliability profiles for two subtrees.

The candidate alignment, i.e. the top-ranked alignment, contains a small number sequences according to the chosen thresholds $min_M$ and $max_M$. This candidate alignment can be interpreted as the core of a conserved ncRNA present in the dataset. The selection of the best subtree is necessary to correctly identify alignments of high quality (data not shown) as $N_k(x)$ can contain instances that do not fit into a common secondary structure. On the other hand, alignments with many sequences increase usually the diversity, and thus, models derived from such alignments can be beneficial during the search for remotely related sequences. To this end, we optionally allow a retraining of the candidate alignment by using additional RNA candidates from the extended neighborhood $N_{k'}(x)$. In general,

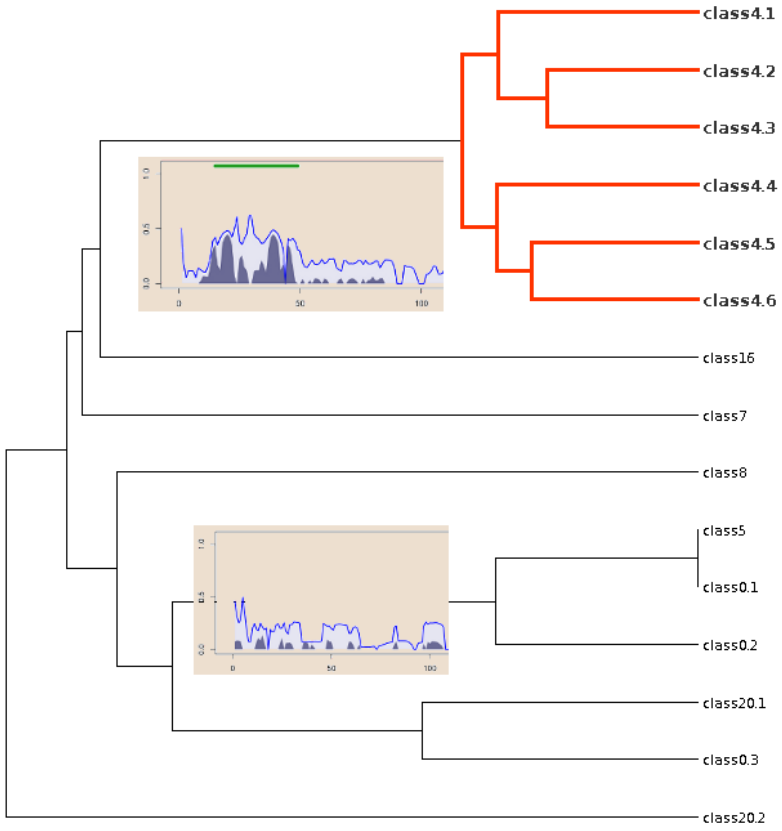


**Figure 4.4.** Guide tree example for a candidate cluster with $k = 15$ RNA fragments. For each candidate cluster, we first create a guide tree (UPGMA) based on the distance matrix derived either from pairwise `LocARNA` alignments or from the pairwise `NSPDK` similarity. Then, for all subtrees with a certain number of leafs, we compute the corresponding multiple sequence-structure alignment with `LocARNA-P`. This method also computes a reliability signal that is used to identify a trusted (local) region with a common RNA motif or ncRNA. The subtree indicated in red has a reliable region (green bar) in contrast to the other subtree. Leaf lables are replaced by external class assignments that clearly support the tree structure and the found signal. Finally, the alignment with the best quality, for example the red subtree, is used to build a cluster model.

expectation-maximization (EM) algorithms are suited for finding a maximum likelihood solution. In our case we use `CMfinder` that implements an EM-algorithm based on covariance models of RNA sequences [188]. We use the alignment with the best quality from the $k$-neighborhood guide tree as input for `CMfinder`. During the iterative EM-algorithm, `CMfinder` tries to include more sequences from the $k'$-neighborhood into the candidate alignment.

Finally, we build a covariance model (CM) from the candidate alignment using the `Infernal` package [122]. Such a cluster model can be used to search large-scale datasets for related RNA sequences, as done in `Rfam`. In our approach, we use the cluster model to scan all input sequences with `CMsearch`. This step allows to populate a cluster with more remote cluster members that are not part of the candidate alignment. Based on

the candidate model, several sane quality checks can be applied. For example, candidate models with a very low MPI or SCI can be discarded. Moreover, a candidate model is probably wrong if `CMsearch` not reliably recovers sequences from the corresponding candidate alignment.

### 4.2.8 Iterative Clustering

`GraphClust` is a *de-novo* clustering approach and thus it is difficult to determine beforehand the number of clusters to be found. In addition, candidate clusters are only similar under the min-hash and graph kernel similarity and therefore the clustering can be different by using additional RNA domain specific knowledge. We solve these problems with an iterative clustering procedure. During one cycle we only process a small number of very dense neighborhoods and continue not before identified clusters are removed from the dataset. This approach is beneficial under various aspects. Candidate clusters are determined by the neighborhood of elements in high density regions. These clusters have a spherical shape in the kernel feature space. By first filtering a cluster via RNA alignment procedures and then expanding it using the covariance model, we remove this bias and obtain non-spherical clusters. Finally, we iteratively remove all instances of the non-spherical clusters which alters the density distribution in the kernel feature space. This allows novel clusters to emerge in the next clustering phase. Moreover, an iterative procedure allows to adapt clustering parameters (e.g. number of hash functions) between iterations. For example, the algorithm first uses parameters sufficient to detect well-defined clusters. Then each following iteration change parameters to detect also less-conserved clusters of RNAs. As stated in Section 4.2.5, the number of hash functions directly controls the quality of the obtained approximate neighborhoods. In addition, altered clustering parameters can avoid recurrent local minima, for example, candidate clusters that are discarded due to quality measures.

Note, in high-dimensional feature space it is not feasible to use absolute values or their distributions (e.g. to determine a density threshold), because values tend to contract or their range change easily for a different dataset. Therefore, it is more reliable to process only very likely candidates instead of applying, for example, significance measures.

## 4.3 `GraphClust` Pipeline

In the following we describe in detail our `GraphClust` pipeline for an efficient clustering of large scale RNA datasets. All necessary steps, from preprocessing to cluster refinement, are integrated in an easy-to-use pipeline which finally outputs clusters of similar RNAs for further analysis. We distinguish nine phases of our `GraphClust` pipeline: initially, input is preprocessed and near-duplicates are filtered away (1), sub-optimal structures for each sequence are computed in parallel (2); the sparse feature encoding is extracted

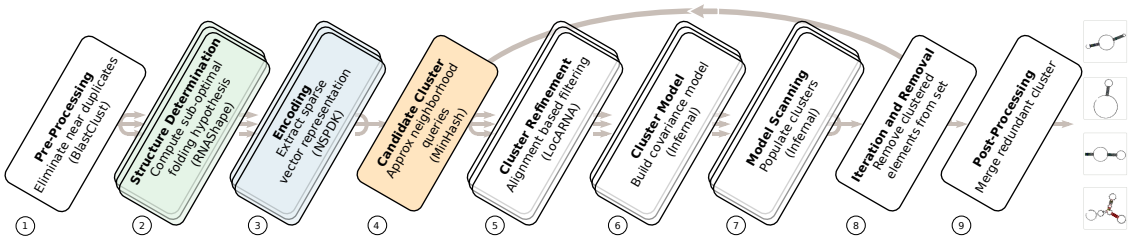


**Figure 4.5.** Complete clustering pipeline diagram. Phases that are executed in parallel are represented in stacked boxes. 1) filter near-duplicates, 2) compute sub-optimal structures, 3) compute sparse vector encoding, 4) compute global feature index and return top dense sets, 5) refine clusters with structural alignment procedure, 6) build covariance model with remaining high quality instances, 7) populate each cluster with retrieved instances, 8) remove clustered instances and iterate from step 4, 9) merge redundant clusters.

for each structure (3); the feature encoding is then used to build the min-hash based feature index for fast similarity searches; the top dense approximate neighborhoods are returned as candidate clusters (4) and they are subsequently refined using structural alignment procedures (5); the final candidate alignments are used as high quality seeds to build covariance models (6) with which additional instances are retrieved to further populate the clusters (7); before re-iterating starts from step 4, the clustered elements are eliminated from the working set (8). Finally, redundant clusters are merged and all instances receive a unique cluster assignment (9).

### 4.3.1 Pre-Processing and RNA Structure Encoding

Phases 1 to 3 are required to convert all input sequences into sparse vectors, necessary for our efficient feature based clustering. These steps are only executed once in our pipeline.

#### Phase 1: Pre-Processing (Sequential)

The `GraphClust` pipeline is able to cluster RNA sequences which originate from different sources like RNA-seq or computational methods like `RNAz` [170] or `EvoFold` [128]. Therefore, a solid pre-processing of the input sequences is essential.

In case of genomic sequences, repeats are masked with 'N's or excluded beforehand in order to avoid clusters made of genomic repeats. Contiguous strings with more than 15 'N's are deleted and the resulting fragments are treated as independent sequences.

Next, we split long sequences into smaller fragments to enable the detection of local signals. Reasonable fragment sizes in the range of $W' = [75, 250]$ nucleotides with an overlap of e.g. $O' = \{33, 50, 75\}$ in percentage of the window size. The actual overlap is slightly adapted to ensure fragments of nearly similar length. We also require that all fragments are longer than a required minimal length.

Obvious relationships of near identical sequences were removed using `BLASTclust` to prevent a bias towards sequential clusters [4]. Such sequences would form very dense neighborhoods and derived candidate clusters would overshadow more subtle sequence-structure relationships. We identify clusters of sequences which are more than 90% identical over 90% of the sequence length. From each such cluster we keep only one sequence at random and remove all the others. The removed sequences can be however included in the final clustering results (see Phase 9). Filtering with `BLASTclust` is applied iteratively until no sequence duplicates are found.

**Phase 2: Structure Determination (Parallel)**

In this phase we extract a set of structures for each sequence employing the `RNAShapes` tool as detailed in Section 4.2.1. Each fragment of window size $W'$ is split into further subsequences of different sizes $W$ with an overlap of $O$. Different windows sizes $W$ are helpful to capture both local hairpins and larger multi-loop structures. As default setting we select two window sizes $W = \{30, 150\}$ and a fixed overlap of $O = 20\%$. For each such subsequence we encode the top $l = 3$ shreps within 20% energy difference to the minimal energy under the most abstract shape level 5.

As a rule of thumb, a sequence of 150 nt is encoded in a set of disconnected graphs of $\approx 2500$ vertices and is obtained in approximately one second on a Xeon 5160, 3.0 GHz ($O = 20\%$, $W = \{30, 150\}$, $l = 3$). The structure finding can be done in parallel on chunks of fragments. For large window sizes we use the structure sampling approach of `RNAShapes` to speedup the shrep finding.

**Phase 3: Structure Encoding (Parallel)**

In this phase we manipulate the set of structures encoded as graphs in Phase 2, and produce an explicit sparse feature encoding as detailed in Sections 4.2.2 - 4.2.4. Reasonable values for radius are $r^* = [1, 4]$ and for distance $d^* = [1, 4]$. For a 150 nt long sequence this yields a sparse vector with $\approx 8000$ features when using the default values of $r^* = 2$ and $d^* = 4$. This can be obtained in approximately one second on a Xeon 5160, 3.0 GHz. Figure 4.6 illustrates exemplarily the behavior of the number of features on up to 100,000 RNA fragments of size 150nt from human 3' UTRs. It is important to note that the feature increase rate shows a sub-linear behavior. In the pipeline the feature encoding can be executed in parallel on chunks of disconnected components.

### 4.3.2 Iterative Clustering

Phases 4 to 8 constitute the iterative part of the GraphClust pipeline. Each iteration starts with the efficient clustering procedure. Then we select a limited number of top ranked candidate clusters and each of them undergoes subsequent steps of refinement and populating the cluster. After scanning of all candidate models is finished, we collect all hits and blacklist them for the next iteration.

#### Phase 4: Candidate Cluster (Sequential)

Here we employ our efficient $\epsilon-$approximate nearest neighbor search procedure offered by the min-hash technique to define candidate clusters (cf. Sections 4.2.5 and 4.2.6). First, we combine all parallel encoded features into a single file with all sparse feature vectors. Second, we generate the min-hash signature of each instance by using $n$ different hash functions and build the inverse index. Finally we compute the density of an instance by using its $k-$approximate neighborhood and rank all instances accordingly. We also apply all neighborhood refinements steps like NSPDK-based re-ranking and weighting each density with its shared neighborhood similarity. As output we provide a given number of $c$ top dense candidate clusters together with their extended $k'$-neighborhoods (cf. Section 4.2.6). We also output the kernel similarity matrix on each $k-$neighborhood, necessary for building the cluster tree in the next phase.

As this phase constitutes the bottleneck of the entire pipeline (since we go from a parallel flow to a sequential one) we use additional procedures to trade-off accuracy with speedup. For example we only compute densities on a smaller random sample of instances (e.g. 25% or 50%). Nevertheless, we want to emphasize the linear time complexity of the clustering phase even when using the full set of instances. For example, we measured the time necessary to finish phase 4 on a growing set of RNA sequences, ranging from 10,000 to 100,000 fragments from human 3' UTRs of size 150nt. Figure 4.6 shows nicely the linear dependency between clustering time and dataset size . In general, the size of the sparse vector depends linearly on the number of instances. For example, 10,000 sequences of length 150nt require ≈1GB RAM, 20,000 sequences require ≈2GB RAM. In contrast, the number of keys for the inverse index increase only sub-linearly. The parameters to determine candidate clusters are dataset dependent, but can be chosen according to some reasonable estimations. As we stated in Section 4.2.5, with 400 min-hash functions we can reduce the expected error $\leq 0.05$. The number of reported candidate clusters $c$ is set according to the number expected clusters. The neighborhood size $k$ is set according to the expected cluster size. The extended neighborhood size $k'$ is linked to $k$ by a given neighborhood excess factor $e$, i.e. $k' = k \cdot e$, which is set by default to $e = 5$. We use always $\alpha := 1000$ as maximal list size to collect the approximate neighborhood from the inverse index. Fragments that overlap on the original input sequences are handled in the

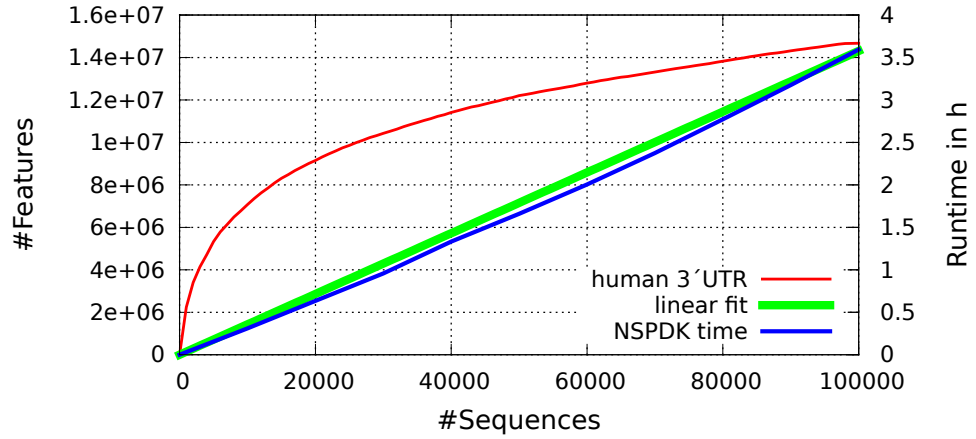


**Figure 4.6.** Dependency between dataset size and the number of features or runtime. The red curve indicates the total number of different features observed for different dataset sizes. The blue curve shows the measured runtime for our approximate nearest-neighbor clustering procedure. The linear fit to these times (green line) clearly indicates the linear runtime. For the plot, a dataset from human 3' UTRs with a total number of $100,000$ sequences of length 150nt was used. Used `NSPDK` parameters: $r = 2, d = 4$, 400 hash functions.

next phase and we assume here a similar influence on all densities. The neighborhood size $k$ should be increased for large overlap sizes ($\geq 50\%$) to ensure the expected cluster size, i.e that a candidate cluster contains enough sequences originating from different input regions.

### Phase 5: Cluster Refinement (Parallel)

Candidate cluster reaching phase 5 are composed of RNA candidates from the k-neighborhood and the extended $k'$-neighborhood. We enhance the quality of the candidate cluster by filtering away inconsistent elements using alignment based techniques as described in Section 4.2.7. Within this phase we accomplish the switch from the `NSPDK` similarity measure to a more domain specific metric that is, for example, aware of compensatory mutations or consensus structure information. To this end, we perform a sequence-structure alignment of the set of $k$ candidate sequences with the tool `LocARNA`. A cluster tree is created by applying an average-linkage algorithm (UPGMA) to the pairwise distance matrix induced by the `LocARNA` alignment score. Alternatively, we can speed up this step and generate the cluster tree from the `NSPDK` similarity. Note, as we split input sequences into windows with a given overlap (e.g. 50%), it is very likely that a candidate cluster contains overlapping fragments. We avoid any bias in the cluster tree by simulating a

large distance between overlapping fragments. In addition, we filter out subtrees which still contain overlapping RNA candidates to avoid any influence on the quality ranking. This phase is done in parallel on each candidate cluster.

### Phase 6: Candidate Model (Parallel)

Next, we traverse the cluster tree and compute multiple alignments with `LocARNA-P` for subtrees of a given size (default is $min_M = 3$, $max_M = 7$). These alignments are then ranked by their quality $Q(\mathcal{A})$. Only the top ranked alignment, which contains the RNA candidates that exhibit the best quality, is retained. With the help of an alignment reliability score provided by `LocARNA-P`, we re-estimate the boundaries of a common ncRNA signal. A covariance model (CM) is finally build by using `Infernal` on the identified subsequences. Further refinement can be done by retraining the CM with `CMfinder` on the RNA candidates from the extended $k'-$neighborhood. This phase is done in parallel on each candidate cluster.

### Phase 7: Model Scanning (Parallel)

Each candidate cluster induces a CM model which is used to search the full dataset for residual cluster members with `CMsearch`. Hits are considered significant on the basis of their bit score (default threshold is 20) or E-value and they are added to the final cluster. Scanning the original, unfragmented sequences, is also beneficial to find cluster members which span over fragment boundaries and hence are difficult to identify. Note that, as every time we perform the search on the entire dataset, a sequence can be assigned to multiple clusters. This ambiguity is allowed in this phase as, until all clusters are available, there is not enough information to decide unambiguously for the best cluster assignment. The search also retrieves RNA candidates taken out by the initial `BLASTclust` filtering. The entire dataset is scanned in parallel by each candidate model.

### Phase 8: Iteration and Removal (Sequential)

All RNA fragments assigned to clusters in the previous phase are removed from the dataset and a new iteration starts from phase 4. The termination condition is given either by a pre-determined maximal number of iterations, a time limit or when the remaining dataset is exhausted. At this point we also apply simple filters on found clusters. For example, clusters with too few members (e.g. $< 5$) are ignored. A further plausibility check is conducted whether the list of significant hits comprise RNA candidates initially used for building the CM model. Finally, all valid clusters are collected and we generate a list of blacklisted RNA candidates that are skipped during the next iteration. Note that we can avoid any recomputation of sparse vectors by saving them to disk. In the next

iteration, blacklisted instances can be skipped while `NSPDK` reads this sparse vector file of all instances. The instance signatures and the inverse index are currently recomputed in every iteration.

### 4.3.3 Post-Processing and Technical Aspects

#### Phase 9: Post-Processing (Sequential)

Redundant clusters are merged and instances that belong to multiple clusters are assigned unambiguously. For every pair of clusters we compute the relative overlap (i.e. the fraction of instances that occur in both clusters) and merge them if the overlap exceeds 50%. Cluster members are finally ranked by their CM bitscore. We also indicate members already found by `BLASTclust` to focus analysis on interesting members. For each cluster we provide example alignments of the top cluster members and various output files for further analysis. A table summarizes all found clusters and promising motifs can be identified by their SCI, MPI or further cluster measures.

#### Technical Aspects

The feature encoding and graph kernel are part of `NSPDK` which is implemented in C++. For the `GraphClust` project we extended `NSPDK` by the min-hash clustering procedure and the retrieval of density-based clusters. The pipeline scripts are implemented in Perl. For parallelization the *Sun Grid Engine* (SGE) is used. A more recent version of `GraphClust` (v0.7+) and `NSPDK` (v9.2+) can also be used with threads. In order to achieve the required linear efficiency during clustering, we use the hash map implementation from `std::unordered_map` (e.g. for the inverse index), which offers a constant average time complexity on element insert and access.

## 4.4 Performance on Clustering of Known ncRNAs

In the following we evaluate the clustering performance of `GraphClust` on datasets comprising known ncRNAs. First, we give details about the measures we use to assess the clustering quality. Then we describe the compiled benchmark datasets and how we compare `GraphClust` with existing clustering methods.

### 4.4.1 Evaluation Measures and Parameters

As the number of clusters cannot be determined beforehand `GraphClust` uses an iterative clustering procedure on a limited number of most dense clusters per iteration. In our evaluation we follow this procedure and measure the clustering quality at the end of each round on all clusters found so far. We use the $F$ measure and the adjusted Rand index as

quality measures [81, 134]. The $F$ measure is the harmonic mean of precision and recall and measures the quality of a single cluster, defined as $F = 2(Prec \cdot Recall)/(Prec + Recall)$. Precision is defined as $Prec = TP/(TP + FP)$, where $TP$ is the number of correct cluster members and $FP$ is the number of wrong cluster members. Recall is defined as the fraction of correctly clustered members over the full family size. To each cluster we assign the family with the majority of members. We report the average $F$ measure over all clusters at the end of each round. The Rand index compares two clustering hypotheses taking into account all possible pairs of instances. The similarity measure is based on the fraction of times when the two clustering hypothesis agree that the elements in each pair belong to the same or to different clusters. We use the adjusted Rand index, which uses a hypergeomeric model to correct for chance effects, so that the value range is $[0, 1]$, with random partitioning scoring 0 and perfect agreement scoring 1. Note that we measure only the quality of clustered RNA sequences, i.e. we ignore elements which are not part of any cluster.

We calculate the quality measures on the basis of different clustering hypotheses, called partition types. The initial clustering is build upon all significant hits of each candidate model and is called *SOFT* partition. In this type a specific RNA candidate could belong to more than one cluster. Based on the *SOFT* partition we generate the two partitions types *BEST* and *MERGED*. In addition we consider for evaluation purposes a theoretical *ORACLE* partition. Partition *BEST* assigns an RNA candidate to the model with the best score (without any merging). *MERGED* uses the described cluster merging strategy during phase 9 (see section 4.3.3) of overlapping clusters and applies *BEST* afterwards. The *MERGED* hypothesis is used as main results for all benchmarks as well as for all application scenarios presented in Section 4.5. The *ORACLE* partition assumes a supervised or perfect merging strategy and shows the maximum theoretical performance. In this case, all initial clusters (*SOFT* partition) with the same majority true class were merged, using *BEST* as final partitioning strategy. The overall running time is an important measure of our pipeline. Here we measure the time of each phase and provide a total time after each iteration and an average time per predicted cluster.

#### `GraphClust` Parameters

Some default parameters are given in Section 4.3. If not stated differently in a section, we applied the following parameters while running `GraphClust`. Phase 2: `RNAShapes` abstraction level: 3. Phase 4: Neighborhood subgraphs use radius $r = 2$ and distance $d = 4$. The approximate neighborhood size is set to $k = 15$ with an excess factor of 5. All benchmark sets and human lincRNAs are clustered without using a downsampling during the calculation of densities. For `RNAz` screens and the human `EvoFold` set a random sample of 50% is used. For human 3'UTR set and Fugu lincRNAs a sample of 20% is used. The initial hash signature uses 300 hash functions. For every subsequent iteration we increase

the signature size by 50 hash functions. No overlap between candidate clusters is allowed. Phase 7: For benchmark sets hits with a bitscore $\geq 15$ are considered as significant, for all the other datasets the threshold is $\geq 20$.

#### `GraphClust` Space and Memory Requirements

A memory limit of 3.5GB is set for all `GraphClust`-phases except phase 4. The memory requirement in phase 4 depend directly on the size of the sparse vector. For example, the sparse vector of the Rfam benchmark is ≈500MB, for the Fruit fly RNAz screen ≈1,7GB and for the human 3'UTR ≈15GB. Assuming sequence fragments of similar size, the sparse vector increases linear in the number of sequences. Datasets up to 30.000 sequence fragments (≈150nt) are therefore possible to cluster on a normal machine (4 GB RAM).

#### Used Software and Hardware

For all runs we use the following tools: `GraphClust` (v0.5), `LocARNA` (v1.6.2), Vienna RNA package (v1.8.5), `RNAsoup` (v1.2.5), `RNAShapes` (v2.1.6), `Infernal` (v1.0.2), `BLASTclust` (v2.2.15). Runtime is measured on Opteron 2356 (2.3 GHz) machines. For parallelization the *Sun Grid Engine* (SGE) is used.

### 4.4.2 Benchmark Datasets

In order to test the performance on clustering known ncRNA classes we use two benchmarks sets from two different sources. (1) We cluster a set of 503 families obtained from the `Rfam` database which has previously been successfully used to benchmark `LocARNA`-based structural clustering [179]. The original set consists of all `Rfam` seed sequences (v7.1), filtered for 80% sequence similarity and lengths $< 400$ nt. Application of `BLASTclust` to remove trivial sequence clusters leaves 3,900 sequences. The majority of families (252/503), however, has less than three members. Only 124 families, comprising ∼80% (3,118) of all sequences, have more than five members. (2) We collected a comprehensive set of 49 bacterial small ncRNA families (941 sequences) from the NCBI Genome Database. Non-coding RNAs present in at least 10 species were considered. Removing sequences exceeding 400 nt in length and similar sequences using `BLASTclust` leaves 363 bacterial ncRNAs. These were randomly embedded in 50 nt genomic context sequence to harden the classification problem. The set contains 6 families with less than 3 members and 37 families with more than 3 members. Sequences of both benchmark sets are not split into smaller fragments. In addition, in order to recover the structural classes presented in [126], we clustered 725 `EvoFold` hits that form 220 `EvoFam` families [126].

### 4.4.3 Comparison to Other Methods

Alignment based RNA clustering methods which take into account structural properties need to calculate a pairwise distance matrix first. This information can be used to get a clustering hypothesis using different methods, e.g. by creating a guide tree. We compare our clustering to a `LocARNA`-based clustering. The idea of the comparison is to show that our clustering approach achieves a similar and high clustering quality but with the discussed benefits especially in complexity and therefore run-time. We use `RNAsoup` in order to partition the `LocARNA` cluster tree into an optimal number of clusters and evaluate them with the given quality measures. A cluster is reported as optimal cluster according to a variant of the Duda rule [41]. It checks if the sum-of-squared error for two clusters is not significantly smaller than expected by chance. The significance level of `RNAsoup` can be controlled by $k$ and authors give a range $0.8 \leq k \leq 1.2$ for `Rfam` sequences. The error of a cluster is determined via the free energy of its consensus structure and the minimum free energies of its individual sequences. Clearly, this procedure gives a full clustering whereas our pipeline only clusters a subset. Therefore we eliminate all clusters with less than 3 members from the `RNAsoup` partition. We also measure the `LocARNA` run-time as aggregated serial time. We use `LocARNA` without any speedup heuristics to stress the inherent complexity issue of existing structure based clustering methods. Using speedup heuristics would give a much lower overall `LocARNA` run-time, but not in the order of several orders of magnitudes.

### 4.4.4 Rfam Benchmark

We run the `GraphClust` pipeline for 15 iterations, retrieving 10 candidate clusters at each iteration. Table 4.1 gives an overview of the result for the clustering of 3,901 `Rfam` sequences. See Appendix Table B.2 and Table B.1 for full details on cluster quality and runtimes. After 15 iterations we identified 130 clusters (MERGED partition). The high F measure (0.834) and Rand index (0.984) indicates a correct clustering. The result reflects the fact that only 124 of 502 families have more than 5 members. Prior to the merging phase we identified 148 clusters (SOFT partition) with a quality of F=0.796 (R=0.483). This clearly indicates that the overall cluster quality can be significantly improved employing a merging strategy. Increasing the number of iterations does not produce additional meaningful clusters, resulting rather in a slightly decreased overall quality.

We report the aggregated run-time for all serial and parallel phases. Running the entire pipeline took ~36 h (129626 s) when viewed as serial process on one CPU core. The parallelized version however, took ~3 h. Note that the clustering step in phase 4 took only between 1 and 8 minutes (serial time), which is the main bottleneck in previous RNA clustering approaches. See also Figure B.2 for a more general runtime comparison.

| | | | **Quality** (MERGED) | | **Time** in secs | | |
|---|---|---|---|---|---|---|---|
| $i$ | #Seq | #C | F | Rand | Phase 4 | Time$_i$ | Time$_{ALL}$ |
| *Rfam benchmark* | | | | | | | |
| 0 | | | | | | 8314 | 8314 |
| 1 | 271 | 5 | 0.882 | 0.888 | 458 | 14995 | 23309 |
| 2 | 629 | 14 | 0.834 | 0.932 | 416 | 19962 | 43272 |
| 3 | 1076 | 23 | 0.868 | 0.956 | 334 | 15108 | 58380 |
| 7 | 2181 | 58 | 0.877 | 0.985 | 154 | 11964 | 104940 |
| 15 | 2821 | 130 | **0.834** | **0.984** | 77 | 2491 | **129626** |
| *Small ncRNA benchmark* | | | | | | | |
| 0 | | | | | | 720 | 720 |
| 1 | 140 | 10 | 0.942 | 0.945 | 42 | 2434 | 3154 |
| 2 | 232 | 20 | 0.926 | 0.939 | 27 | 3395 | 6549 |
| 3 | 270 | 26 | 0.936 | 0.935 | 17 | 7681 | 14230 |
| 7 | 329 | 35 | 0.890 | 0.897 | 5 | 250 | 23186 |
| 15 | 360 | 43 | **0.858** | **0.866** | 1 | 92 | **24301** |

**Table 4.1.** Results for `Rfam` and small ncRNA benchmark set. Results for each iteration $i$ on the MERGED partition. Clustering quality is given as F measure and Rand index. The total number of clustered sequences is indicated with #Seq. The total number of clusters after merging is given by #C. Time$_i$ denotes the total time for iteration $i$, Time$_{ALL}$ is the total serial time up to iteration $i$.

In order to compare our results to state-of-the-art sequence-structure clustering, we applied `RNAsoup` to the cluster tree obtained from `LocARNA` alignment scores. We chose the partition with $k = 0.8$ which gave a quality of $F = 0.588$ ($R = 0.586$) for 160 predicted clusters containing 3,569 sequences. We considered only cluster with at least 3 members for a fair match. Other $k$ values give similar (although slightly worse) results. Clustering 3,901 sequences with `LocARNA` without any speedup heuristics took ~370 days, yielding a theoretical 246-fold speedup for our method. Clearly, it is possible to employ parallelization and effective heuristics also for `LocARNA`. We also analyzed the impact of using a sample of 50% and 25% in phase 4 and observe a similar quality (see Table B.5).

### 4.4.5 Small ncRNAs Benchmark

We run our pipeline for 15 iterations, retrieving 10 candidate clusters at each iteration. Table 4.1 gives an overview of the results for the clustering of 363 small ncRNAs (see also Tab. B.3 and Tab. B.4). After 15 iterations we identified 43 clusters (MERGED partition) from 38 unique families. The overall clustering quality is high with $F = 0.858$ and $R = 0.866$. The additional flanking sequences do not disturb the quality. Please note, although we ask for 10 candidate clusters at each iteration, less clusters are reported. This happens for example in case that further clusters would overlap higher ranked clusters,

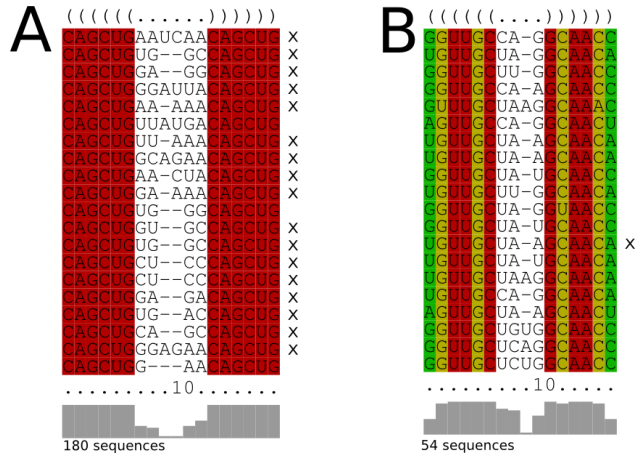


**Figure 4.7.** Two exemplary cluster identified by `GraphClust` when processing `EvoFold` hits. For each cluster the top 20 sequences are given. The consensus secondary structures of both clusters are small hairpins. Cluster (A) contains many sequences that belong to the same `EvoFam` family (as indicated by 'x'). Contrary, only one of the depicted sequences of cluster (B) is a member of a previously described `EvoFam` family. Interestingly, this novel cluster contains several compensatory mutations that support the structural clustering. This demonstrates that `GraphClust` can identify relevant local structural clusters. It may not only help to improve existing family assignments, it can also be used to define new ones.

which is disallowed in the used setting. `LocARNA` applied to this set results in a quality of $F = 0.729$ and $R = 0.88$ using `RNAsoup` with $k = 0.4$ (other $k$ have lower qualities). The serial run-time of the pipeline is ~6.8 h. Using only `LocARNA` on the same dataset takes ~7 days.

### 4.4.6 EvoFam Family Benchmark

Application of `GraphClust` (5 iterations) to the 725 `EvoFam`-annotated sequences yields 37 structural classes. We recover 14 known families with $F \geq 0.5$. In particular, even raising the threshold ($F \geq 0.7$), we identify 5 out of 8 families with $\geq 10$ members. Applying `GraphClust` on all 37,381 human `EvoFold` hits (20 iterations, see Tab. 4.2 as overview and Fig. 4.7 for an exemplary clusters) recovered the same amount (5/8) of `EvoFam` families (albeit different in type) having $\geq 10$ members. `Infernal` annotates ~38% (14/37) of the `GraphClust`-derived `EvoFam` clusters as known ncRNA classes. For example, we identified the Histone 3'UTR stem-loop motif and the let-7 miRNA family. The vast majority of clusters (11/14) were known miRNAs. For 10 of the 11 miRNA cluster the E-value of the best `Infernal` hit was $< 10^{-15}$ indicating a reliable class annotation. Interestingly, these `GraphClust` results can also be used to identify novel human miRNA candidates as shown in Figure 4.9.

## 4.5 GraphClust Predicts Novel Local Structural Motifs

In the following we predict clusters with novel RNA motifs and apply GraphClust to large scale and genome-wide datasets with several thousands of RNA fragments. With previous methods a similar setting for sequence structure based clustering would have been impossible in reasonable time. First we give details on the used datasets and then we summarize found motifs for RNAz-screens and lincRNAs.

### 4.5.1 Datasets

We analyze different sets of predicted ncRNAs. We apply GraphClust to 37,381 human EvoFold hits [128], 16,377 RNAz ncRNA candidates of the fruit fly *Drosophila melanogaster* [140], and 11,536 ncRNA candidates of the teleost *Takifugu rubripes* [141]. EvoFold predictions are generally short and were not split before clustering. For both RNAz screens, sequences are fragmented into stretches of 150 nt (min. length 50 nt). Removing nearly similar sequences using BLASTclust left 17,765 fragments for fruit fly and 11,287 fragments for teleosts. Moreover, we search for novel local structural motifs in long ncRNAs. As first set we cluster the collection of 8,195 human lincRNAs described in [24]. Splitting (150 nt windows, min length 50 nt) and BLASTclust filtering resulted in 31.418 fragments. Secondly, we cluster the set of 1,133 lincRNAs expressed in zebrafish embryos recently reported in [127]. Preprocessing yields 5,877 fragments, ready for clustering.

Resulting structural clusters were annotated using Infernal (v1.0.2) [122]. Using CMsearch we compared our clusters to all Rfam seed models (v10.1) that have an average seed sequence length $\leq 500$ nt. Clusters that contain CMsearch hits with an E-value of $E < 10^{-5}$ were considered as known, others as novel.

### 4.5.2 Structural Motifs in RNAz Screens

Most of the known families, including tRNAs, snRNAs (U2, U5) and miRNAs, are recovered in the fruit fly RNAz screen (6 annotable clusters). Throughout all *de-novo* discovery screens, miRNAs were most abundantly detected (four clusters in fugu- and two clusters in the fruit fly-RNAz screen; two clusters in fugu lincRNAs). Clustering the EvoFam dataset has shown that GraphClust can recover known UTR elements. Therefore, we decided to analyze and search for novel *cis*-regulatory UTR elements on a broader scale by clustering all RefSeq 3'UTRs. Beyond a single box H/ACA snoRNA, this search again returned the Histone 3'UTR stem-loop motif. Furthermore, it resulted in up to 116 candidates for novel *cis*-regulatory elements. The majority of generated clusters, however, can not be annotated by existing Rfam models and are candidates for novel ncRNA classes. As

depicted in Table 4.2 and Figure 4.8, our obtained motifs have comparably low sequence similarity (measured by MPI). Nevertheless, we predicted 186 novel clusters that have an SCI $> 0.5$ indicating that these are indeed novel structural clusters.

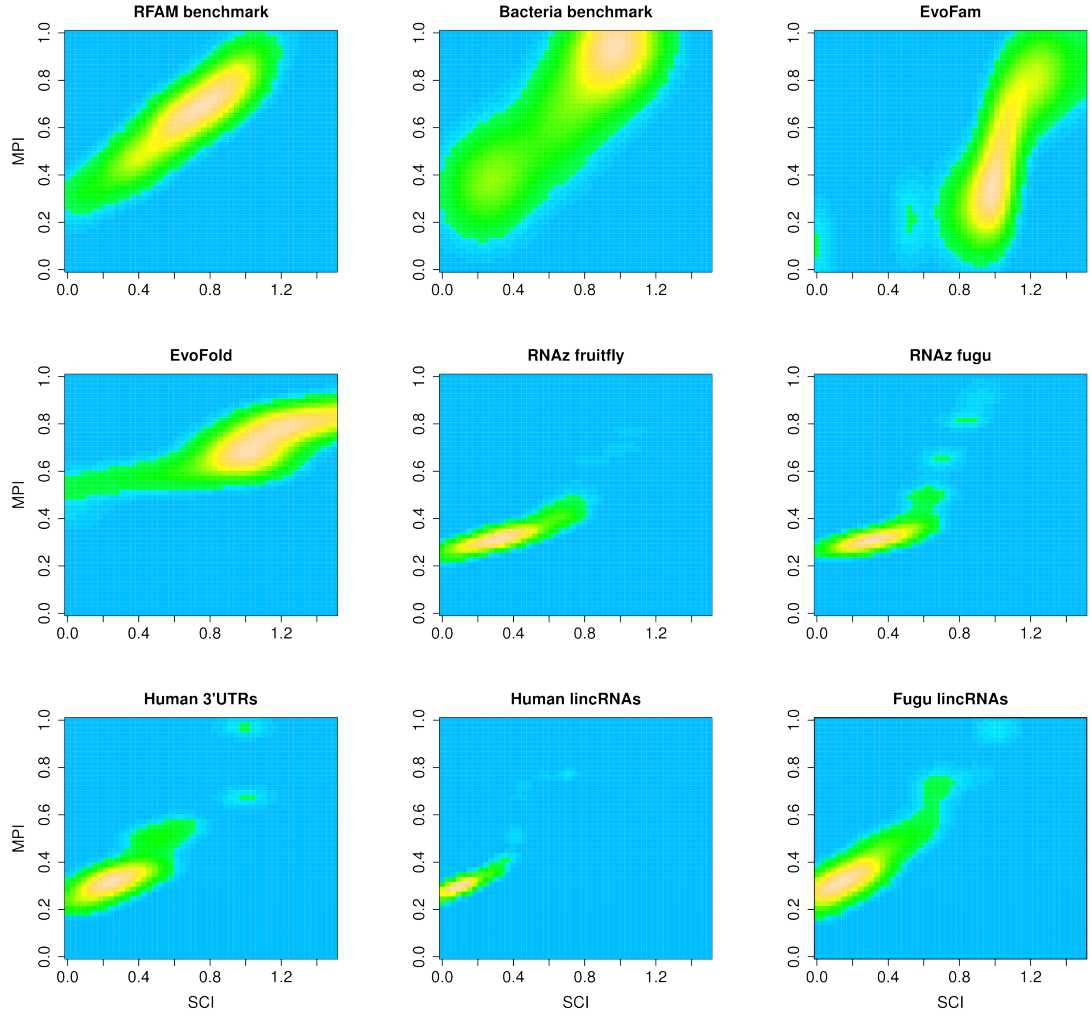


**Figure 4.8.** SCI/MPI density heat-maps of GraphClust-generated clusters. The heat-maps illustrate that GraphClust can indeed identify local structural clusters. We present heat-maps for different benchmark and application scenarios. Recall that for local motifs the structure conservation index (SCI) can only be used as a measure of "structured-ness" in case it is high. Low SCIs are known to be uninformative for local structural elements and no conclusion can be drawn. Thus, although the mean pairwise sequence identity (MPI) is low for many structural clusters, we still observe clusters with reasonable high SCIs indicating conserved secondary structural elements.

| Species | Type | Method | Input | Size [Mb] | Time* | Cluster | $MPI_{avg}$ | $SCI_{>0.5}$ | Reference |
|---|---|---|---|---|---|---|---|---|---|
| *benchmark* | | | | | | | | | |
| Bacteria | small ncRNAs | misc | 363 | 0.06 | 6.8h | 39 | 0.75 | 29 | NCBI ftp** |
| Human | predicted RNA elements | `EvoFam` | 699 | 0.03 | 0.3h | 37 | 0.52 | 36 | [126] |
| Misc | small ncRNAs | `Rfam` | 3,900 | 0.51 | 36h | 130 | 0.64 | 98 | [55, 179] |
| *de-novo discovery* | | | | | | | | | |
| Fugu | lincRNAs | RNA-seq | 5,877 | 0.09 | 10.3h | 99 | 0.39 | 16 | [127] |
| Fugu | predicted RNA elements | `RNAz` | 11,287 | 1.36 | 13.3h | 97 | 0.39 | 22 | [141] |
| Fruit fly | predicted RNA elements | `RNAz` | 17,765 | 2.15 | 20.4h | 95 | 0.34 | 23 | [140] |
| Human | lincRNAs | RNA-seq | 31,418 | 5.40 | 3.6d | 95 | 0.34 | 3 | [24] |
| Human | predicted RNA elements | `EvoFold` | 37,258 | 1.37 | 5.7d | 117 | 0.75 | 109 | [128] |
| Human | 3'UTRs | RefSeq | 118,514 | 21.91 | 12.8d | 106 | 0.34 | 13 | [129] |
| $\sum$ | | | 227,081 | 32.88 | 25.7d | 815 | - | 349 | |

**Table 4.2.** Overview. The table summarizes the datasets used in this study and gives an overview on `GraphClust`-detected clusters. For each screen, we provide the number of input instances and the sum of their lengths (*size*). We denote the required serial running time to process the input and list the number of obtained clusters. Next, we report the mean pairwise identity *(MPI)* and the number of clusters that have a structure conservation index *(SCI)* above 0.5. These are prime candidates for structured ncRNA classes. * Please see text for different parameters influencing the run-times. **`ftp://ftp.ncbi.nih.gov/genomes/Bacteria/`

### 4.5.3 Structural Motifs of lincRNAs

With `GraphClust` we extract 95 local motifs from the 8,195 human lincRNAs recently reported by Cabili et al. [24]. In 55% (52/95) of all cases the majority of transcripts underlying our structural clusters are consistently expressed in the same tissue. The vast majority of clusters (49/52) contains transcripts specifically expressed in testes. This, however, is expected, since it has already been shown that most of the lincRNAs from this dataset are expressed in testes [24]. Nevertheless, we also obtain structural motifs from transcripts that are consistently and specifically expressed in either skeletal muscle, kidney and brain. Next, half of our clusters (47/95) contain transcripts with enriched GO-FAT biological process terms and hence have a putative functional link to their nearest protein-coding gene [80]. We find 17 clusters that contain at least two different lincRNAs with enriched GO terms. Of these, ∼53% (9/17) of structural lincRNA motifs are associated with exactly the same GO term. The actual number of structural motifs with a specific biological function is likely higher, since different GO terms can still convincingly refer to similar biological processes. For example, we obtain clusters whose transcripts are described by the obviously related GO terms "neuron differentiation", "regulation of neurogenesis", "regulation of nervous system development" and others. Manual inspection has shown that most of the cluster-associated GO terms deal with aspects of neuron differentiation and development, neuronal signaling, cognition and related processes. This is in-line with recent findings that long ncRNAs are functionally linked to the nervous system, neuronal diseases and brain function [28, 130].

Furthermore, we extract 99 local motifs from the 1,133 teleost lincRNAs recently reported by Pauli et al. [127]. Interestingly, these contain up to 5 times more novel structural classes than their human counterpart. This might be explained by the fact that teleost fish underwent an additional whole genome duplication which increases the likelihood to identify paralogous genes [29]. Found structural clusters from human lincRNAs and teleost lincRNA are summarized in Figure 4.8 and Table 4.2.

## 4.6 Discussion

With `GraphClust` we introduced for the first time an ultra-fast approach for large-scale comparison and clustering of RNAs according to sequence *and* structure, which is key to the functional annotation of ncRNAs. Strikingly, our clustering step is alignment-free to tackle the inherent complexity issue of sequence structure alignment methods as well as to skip the computation of a full similarity matrix.

As clearly indicated by the results, our approach yields high quality clusters while at the same time it is linear in time and thus scales to sets of hundreds of thousands of sequences. To this end, we introduce a clustering procedure based on the recently introduced graph kernel (`NSPDK`) in combination with a fast approximate neighborhood search

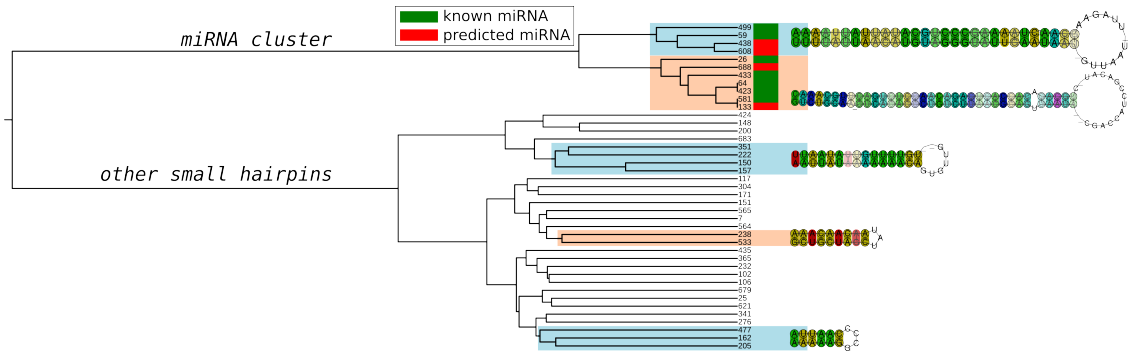


**Figure 4.9.** `GraphClust` identifies novel human miRNA candidates. A hierarchical `LocARNA`-based structural clustering based on one representative sequence selected from `GraphClust`-derived clusters of `EvoFam` sequences reveals two main structural classes. Apart from several small hairpins, we observe a prominent miRNA cluster consisting of known miRNAs (annotated by miRBase v.17, highlighted in green) and structurally related sequences lacking any annotation (red). These are promising candidates for novel miRNAs.

query using hashing techniques. We incorporate structure information by encoding a set of low energy RNA secondary structures as labeled graphs which preserve important information like nucleotides, bond types and stacking base pairs. By using `RNAShapes` for secondary structure prediction, we ensure the encoding of stable but sufficiently different conformations from the RNA structural ensemble. We simply handle different structures as a graph with disconnected components. This is advantageous because we can encode as many different structures as necessary to cover the most probable conformations of an RNA. Thereby we also avoid using the complex structural ensemble as well as solely the erroneous minimum free energy structure. Furthermore, we can address the problem of unknown signal boundaries by using different folding windows. For feature extraction we use the neighborhood subgraph pairwise distance kernel (`NSPDK`). The extracted subgraphs (e.g. shown in Figure 4.2) correspond to local substructures in the RNA and the neighborhood relation is very suitable to model regions in the RNA which evolve differently. In addition, `NSPDK` employs a fast graph isomorphism check which we exploit to obtain an explicit feature representation. Each neighborhood subgraph is hashed into a high dimensional feature space which allows us to represent an RNA as sparse feature vector. This explicit feature representation turns out to be the key for an efficient clustering step. Instead of computing a full similarity matrix, we apply an efficient $\epsilon-$approximate nearest neighbor search query in order to define candidate clusters. First, we generate a compressed signature of each sparse vector by using $n$ different min-hash functions. Then we build an inverse index over these signatures which allows us to retrieve the approximate neighborhood of one instance in constant time, and thus, in linear time for the whole dataset. In order to retrieve high quality clusters, we refine all neighborhoods and rank them by their density. We continue only with the top dense clusters. Each candidate

cluster consists only of a small number of RNAs, and thus, we can apply more complex RNA sequence structure alignment methods to build a candidate model of the cluster. Remote cluster members we obtain by searching with the candidate model against the full dataset. We also established a novel iterative clustering procedure to address challenges of both high dimensional data and large RNA datasets. The total number of RNA clusters is unknown beforehand and therefore we process only a small number of very dense neighborhoods each time. New dense clusters can emerge after each iteration because we remove found clusters which alters data densities.

The feasibility of the presented RNA clustering approach we confirmed with several benchmark studies and application scenarios. Eventually we also implemented our approach as `GraphClust` pipeline, which is an ready-to-use tool for efficient clustering of large-scale RNA datasets.

The largest data set we considered consists of ~118 thousand sequences, and they can be clustered by the proposed pipeline on a single computer in ~13 days. Furthermore we have parallelized 5/9 phases of the pipeline. This allows to reduce the run-time for clustering the 3,901 `Rfam` seed sequences from 36 h to ~3h. When compared with the time required by an efficient pairwise sequence-structure alignment, namely `LocARNA`, we observe a ~250-fold speedup. It indeed took us 370 days to perform the clustering based on this state-of-the-art complete all-against-all sequence/structure comparison.

Our integrated pipeline uses `LocARNA` and `Infernal` to improve the candidate clusters found by a neighborhood search. The latter, now allows us for the first time to compile RNA-classes of ncRNA and determine associated consensus structures for large-scale datasets without resorting to alignment-based clustering. This is important as it is known that sequence alignments often fail at pairwise sequence identities below ~60%. In addition, our pipeline exhibits an anytime characteristics, since we do not need to produce a complete hierarchical cluster tree, which is a computational bottleneck for large datasets. In contrast, we output as many best clusters as wanted by the user. More cluster can be found by simply running an additional clustering iteration. The overall complexity of our pipeline is to a large extend determined by the number of reported clusters.

We have evaluated the approach on several benchmark sets consisting of `Rfam` seed alignments, `EvoFam` families and known bacterial ncRNA. As presented in Section 4.4, we achieve a high overall clustering quality, even if the known RNA signal is embedded in flanking context. In addition, the high clustering quality supports the used iterative procedure. The quality after each iteration decreases very slowly. In contrast, the overall quality is lower when using only one round with many clusters (data not shown). To further elucidate the capacity of our approach, we have also clustered datasets where no clustering approach has been applied so far, for example for `RNAz` screens and lincRNAs. By processing the complete dataset to generate its density landscape our method in

particular enables us to likely detect previously missed structural classes. The screens of this pilot study only consisted of sequences from a single genome. Thus, we can cluster only RNA genes that are present in multiple copies within a genome. This implies that most of the found clusters consist of paralogs, structures from repeat-associated RNAs, mobile elements, i.e. transposon-derived ncRNAs, and maybe also pseudogenes. Even under this setting, we can show that we find many structured classes, when we use the commonly accepted structure conservation index (SCI) to determine structured-ness of a cluster. This can easily be improved by using additional information on orthologs, as it is for instance done in the `EvoFam` approach, where a 41-way multiple alignment is used. Our structure encoding procedure is flexible enough to combine features of a multi-species alignment block as single sparse vector. Furthermore it would be beneficial to first apply a structure-based whole genome realignment with tools like `REAPR` [183].

Since the lincRNA dataset contains GO annotation, we have used this information for further evaluation. Albeit the GO enrichment analysis is limited by the low number of transcripts that are associated with GO terms (overall, GO terms are only available for 12% (1044/8195) of the Cabili et al. [24] lincRNAs), we found nevertheless that the GO terms for the majority of clusters (53% but likely more) are consistent and support our clustering approach.

Our current focus for evaluation is based on the complete approach to show its feasibility in general rather then analyzing and optimizing each individual step. Therefore the current `GraphClust` pipeline uses several preset parameters. Most of them are set according to default values of external tools but we also provide reasonable estimations for parameters like the number of min-hash functions. For other parameters we have used external knowledge like for folding windows or `Infernal` parameters. Some parameters are also dataset dependent and cannot be optimized. Clearly this yields several starting points for further investigations and optimizations. For example, it would be beneficial to obtain robust estimations for the neighborhood size $k$, the number of candidate clusters as well as the used window size along with optimized `RNAShapes` parameters. On the other hand, instead of relying to fixed parameters we could base the parameter optimization onto machine learning techniques, i.e. let a support vector machines select the optimal parameters.

Although our approach is suited for *de-novo* clustering, we have integrated only simple quality measures which improve our predictions but do not have the capacity to finally annotate a found cluster as functional. This is a general problem in ncRNA research and is highly related to the ongoing discussion about important features of "functional" alignments [158]. By integrating advanced quality measures we could for example improve the ranking of subtree alignments for each candidate cluster. This would allow us to

filter out much earlier clusters which are likely non-functional. Furthermore it might be beneficial to replace the `Infernal` scanning phase with tools like `LocARNAscan` to discover more structural related RNAs [182].

The presented `GraphClust` approach constitutes a major improvement for structure based RNA clustering. The steadily enhanced pipeline has already a broad user base and future applications of graph kernel based alignment-free clustering approaches will likely result in the detection of additional functionally relevant structural ncRNA classes.

# Chapter 5

# Conclusion

In this thesis, we addressed the problem of comparative RNA sequence analysis and annotation by computational methods. We developed two pairwise RNA comparison methods and an efficient approach for sequence-structure-based RNA clustering. The presented algorithms established novel principles for the incorporation of both RNA sequence and structure, resulting in efficient *and* accurate approaches. Especially in the domain of clustering of RNA sequences, our approach is the first computationally feasible solution for large-scale datasets that takes RNA structure into account.

In the first part of this thesis, we developed the method `ExpaRNA`, which uses a novel comparison principle for RNA sequences based on exact matching substructures, called exact pattern matchings (EPMs). Although structure-based comparison algorithms are usually computational expensive, it is possible to compute the set of all EPMs for two given RNAs with fixed secondary structure in quadratic time [5, 155]. This fast algorithm accommodates to the high demand for efficient RNA comparison methods due to constantly expanding dataset sizes. The low time complexity of the EPM detection in combination with the intrinsic structural locality of EPMs qualify exact matching substructures for a fast and structure-based RNA comparison. Moreover, functionally important motifs are often part of conserved substructures and, thus, should be recognized during the comparison. However, classic alignment-based algorithms can break conserved substructures as they score only a single nucleotide or base pair. Consequently, the correct alignment of conserved substructures is achieved only indirectly via an appropriate gap-cost function. Our method `ExpaRNA` addresses these issues and enhances previous RNA comparison methods by efficiently computing an optimal, non-crossing (i.e. secondary structure-aware) arrangement of EPMs instead of generating a full sequence-structure alignment. This optimal set of EPMs for two RNA molecules is called LCS-EPM and can be computed by our developed dynamic programming algorithm. We showed that our result is in good agreement with existing comparison approaches, but can be computed in a fraction of runtime. The identified EPMs can, for example, guide a manual

inspection of RNA similarities. In addition, we investigated a promising approach that integrates conserved EPMs in sequence-structure alignments. To this end, we use the `ExpaRNA`-generated EPM set as anchor point of a full alignment. We showed the benefits of this approach by combining `ExpaRNA` with `LocARNA` and observed the following three main improvements: a) `ExpaRNA`'s EPM set used as anchor constraints for an full alignment speeds up expensive sequence-structure alignment methods by reducing the alignment search space, b) EPMs are able to improve the overall alignment quality and c) the integration of EPMs as anchor constraints is a generally applicable scheme for sequence-structure alignment methods in order to profit from conserved substructures. Our evaluation showed that there is a trade-off between overall quality and speedup that can be controlled by the minimal size of EPMs used. This allows for the right balancing in large-scale application scenarios.

The second part of this thesis is dedicated to lifting the concept of EPMs from fixed RNA secondary structures to energy-based RNA structure ensembles. Using a more flexible structural model is very beneficial for structure prediction as shown by various RNA structure alignment methods, but unfortunately this also increases the computational complexity to at least $O(n^4)$. Thus, our main contribution is not only to provide a solution to find ensemble-based EPMs in quadratic time, but to have the first $O(n^2)$-time RNA comparison algorithm that uses the full RNA energy model. For this purpose, we developed the `ExpaRNA-P` approach that combines two major novelties to achieve the desired quadratic runtime on RNA structural ensembles: 1) the introduction of in-loop probabilities, and 2) a new sparsification scheme based on these in-loop probabilities. The key feature is that we can decide *a priori*, based on a probability threshold, if a sequence position is part of a particular loop. We have used this innovation to identify significant EPMs in structural ensembles and designed an efficient DP algorithm for this task. To this end, we also extended the McCaskill algorithm to compute the novel in-loop probabilities by maintaining its complexity bounds. Furthermore, we have extended the chaining algorithm to deal with EPMs from structural ensembles. The advantages of these algorithmic improvements are supported by our evaluation, in which we observed both higher speed-ups and better quality than with `ExpaRNA` in a similar benchmark setting. These results prove that ensemble-based EPMs are superior to mfe-based EPMs and, hence, more EPMs can be included as anchor constraints. The data also suggests using `ExpaRNA-P` as a general filter, with the `ExpaRNA-P` score acting as proxy for an alignment-based similarity score. A method for the comparison of multiple RNA sequences using significant EPMs is also in preparation (Meinzer et *al.*). For this purpose, it seems promising to construct `T-Coffee`-like libraries based on EPM information to build a multiple alignment that is aware of conserved substructures. It should also be highlighted that the presented state-of-the-art structural sparsification scheme has been already adapted

to reduce the high complexity of simultaneous alignment and folding approaches. It can be expected, that it will influence further structure-based RNA comparison methods as well [181].

Structure-based RNA clustering is an accepted, but difficult technique for the annotation of RNA sequences as well as for the *de-novo* prediction of RNA genes or functional RNA elements. However, its application to large-scale datasets was impossible in the past due to an inherent complexity issue resulting from the computation of the full distance matrix by expensive sequence-structure alignment methods. With the `GraphClust` approach presented in the final part of this thesis, we eliminated this bottleneck and provided the first feasible solution for structure-based RNA clustering. In extensive benchmark studies on already annotated RNA sequences (from, e.g., `Rfam`), we proved the very high quality of our clustering in general as well as in comparison to previous RNA clustering methods. Convincingly, we achieved a significant runtime reduction from approximately one year to 36 hours for a dataset of 3900 RNA sequences using a single processor. This major improvement was possible due to an alignment-free, linear-time clustering step using a fast graph kernel in combination with an efficient approximate nearest neighbor search query. Structural properties of the RNA sequences are integrated by local substructures, which are, however in this case derived from the neighborhood subgraph pairwise distance kernel [33]. Structural variants are recognized by using a set of relevant structures obtained via `RNAShapes` together with a novel graph encoding of RNA secondary structures. The key for the fast clustering is the explicit representation of the kernel subgraphs in a high-dimensional feature space. This eventually allowed us to retrieve dense candidate clusters from an inverse feature index in linear time. To demonstrate the power of this approach, we set up a ready-to-use clustering pipeline integrating the clustering algorithm together with several novel refinements steps, which allows to predict high quality clusters of sequence-structure related RNAs even for large scale datasets. In a pilot study, we processed novel datasets with up to 118,000 RNA sequences and predicted several structural RNA clusters in human lincRNAs and in `RNAz` screens from fly and fish. Based on GO annotations, we predicted structural motifs in human lincRNAs that are specifically expressed in either testes, skeletal muscle, kidney or brain. These findings are also consistent with recent studies which showed that lincRNAs are often linked to functions in the nervous system [28, 130]. With the extension to multi-species input data, our `GraphClust` pipeline will probably allow to identify additional RNA motifs in long RNA transcripts and RNA-seq data in future studies.

Many functional RNA molecules have a specific three-dimensional structure that is ignored by the majority of existing computational methods for either structure prediction, comparison or clustering. However, several RNA motifs or modules are already known to be important for the overall topology of an RNA molecule and its three-dimensional folding [101, 177]. The incorporation of such information into RNA structure prediction

and comparison methods is supposed to be the next major leap to enhance their accuracy [173]. The presented principles for the identification of similar local substructures as well as the integration of local substructures into comparison algorithms could be used for 3D structure-aware algorithms as well. For example, with an appropriate structure encoding, a graph kernel could also be used to identify similar substructures and the best chain of such motifs can act as skeleton for a full three-dimensional structure prediction.

To conclude, the presented methods in this thesis suggest that local substructures from RNA secondary structures are a powerful bioinformatical concept especially for RNA comparison. Based on exact matching substructures, we developed three RNA comparison approaches. With `ExpaRNA`, we have established a motif-based comparison concept for RNA secondary structures that can be also used to speedup complex sequence-structure RNA alignments. The second approach `ExpaRNA-P` lifts this concept to RNA structure ensembles, which highly improves the quality of the predicted exact matching substructures. We also introduced a novel and general applicable structural sparsification scheme. The method presented last provides a fast and very efficient solution for structure-based RNA clustering by exploiting a fast graph kernel and hashing techniques. The provided `GraphClust` pipeline constitutes the first practical solution for the problem of clustering of large-scale RNA datasets.

# Appendix A

# Exparna Results

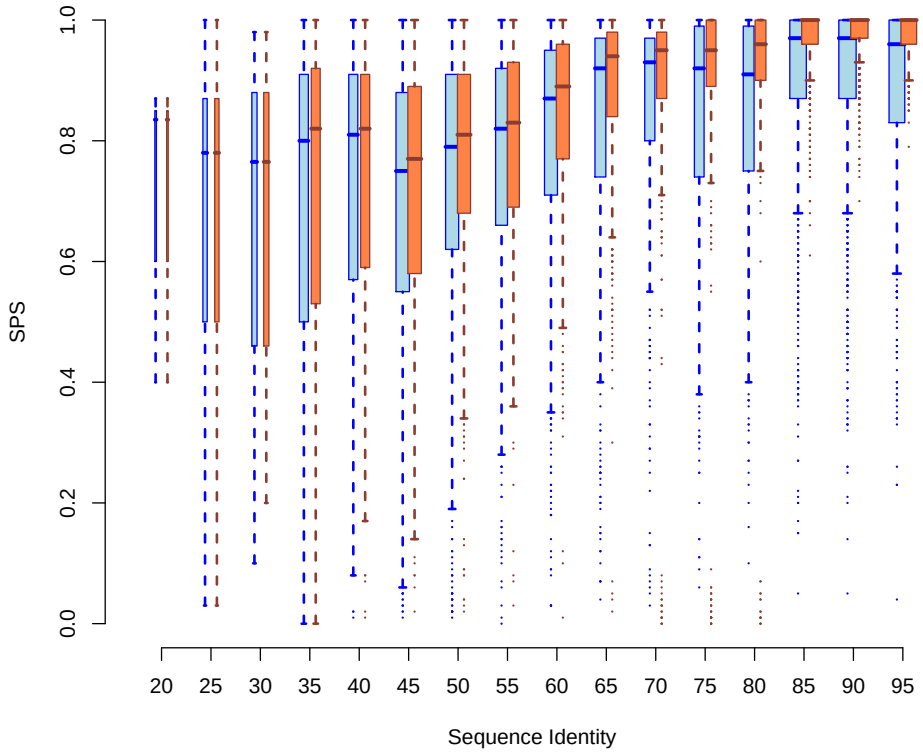


**Figure A.1.** Comparison of the quality of obtained results for `ExpLoc` (light blue) and `LocARNA` (orange). The boxplot shows distributions of sum of pair scores (SPS) (y-axis) for different sequence identities (x-axis) for all 8976 pairwise alignments from `BRAliBase 2.1` and an minimal EPM size of $\gamma = 10$. To compute distributions, alignments were grouped according to their mean pairwise identity in intervals of width 5.

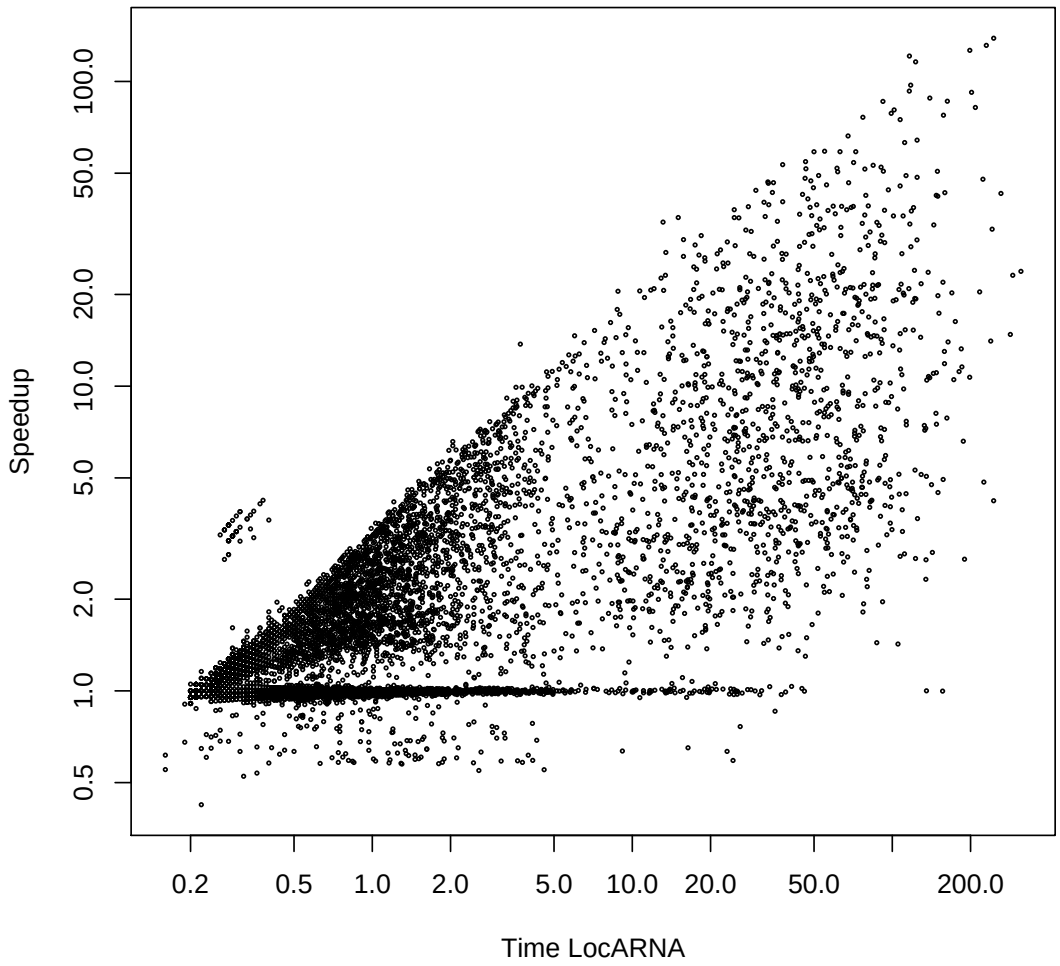


**Figure A.2.** Log-log scatterplot of `LocARNA` runtimes (x-axis) versus obtained speedup of `ExpLoc` with $\gamma = 10$.

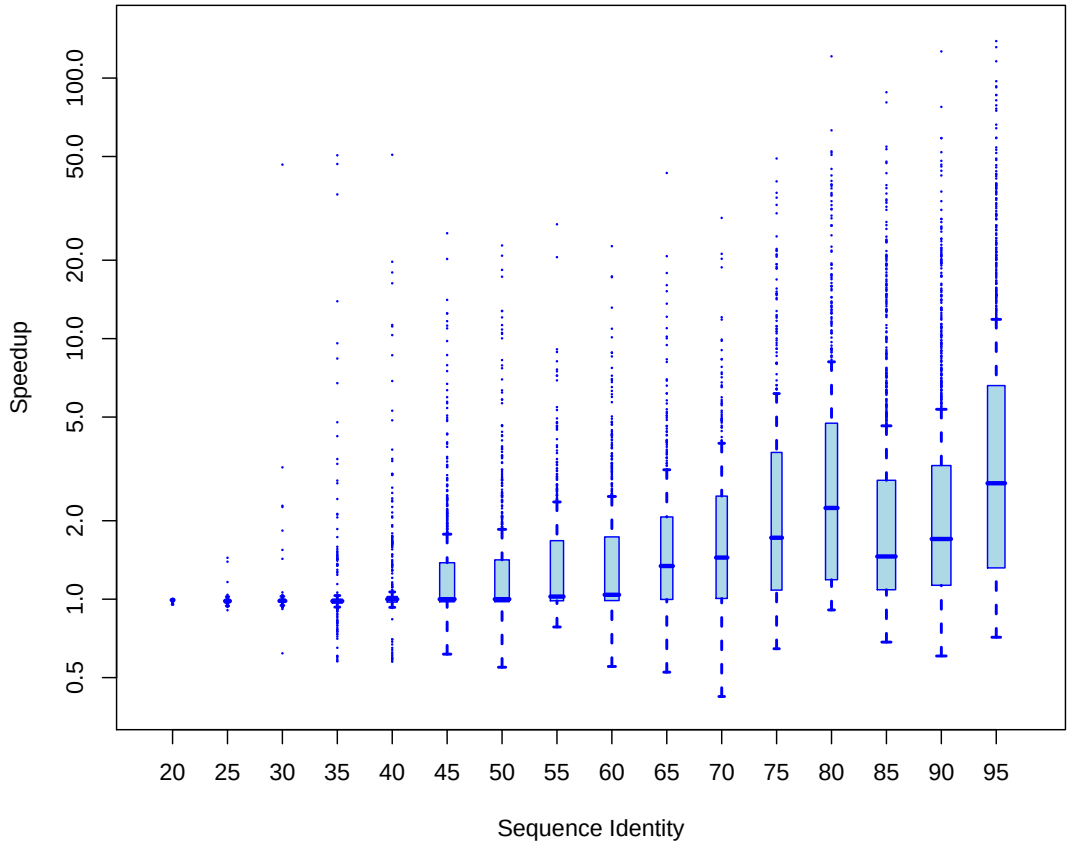


**Figure A.3.** Distribution of obtained speedup for `ExpLoc` on log-scale on `BRAliBase` k2 dataset with $\gamma = 10$ for different sequence identity values. For distribution computation, alignments were grouped according to their mean pairwise identity in intervals of width 5.

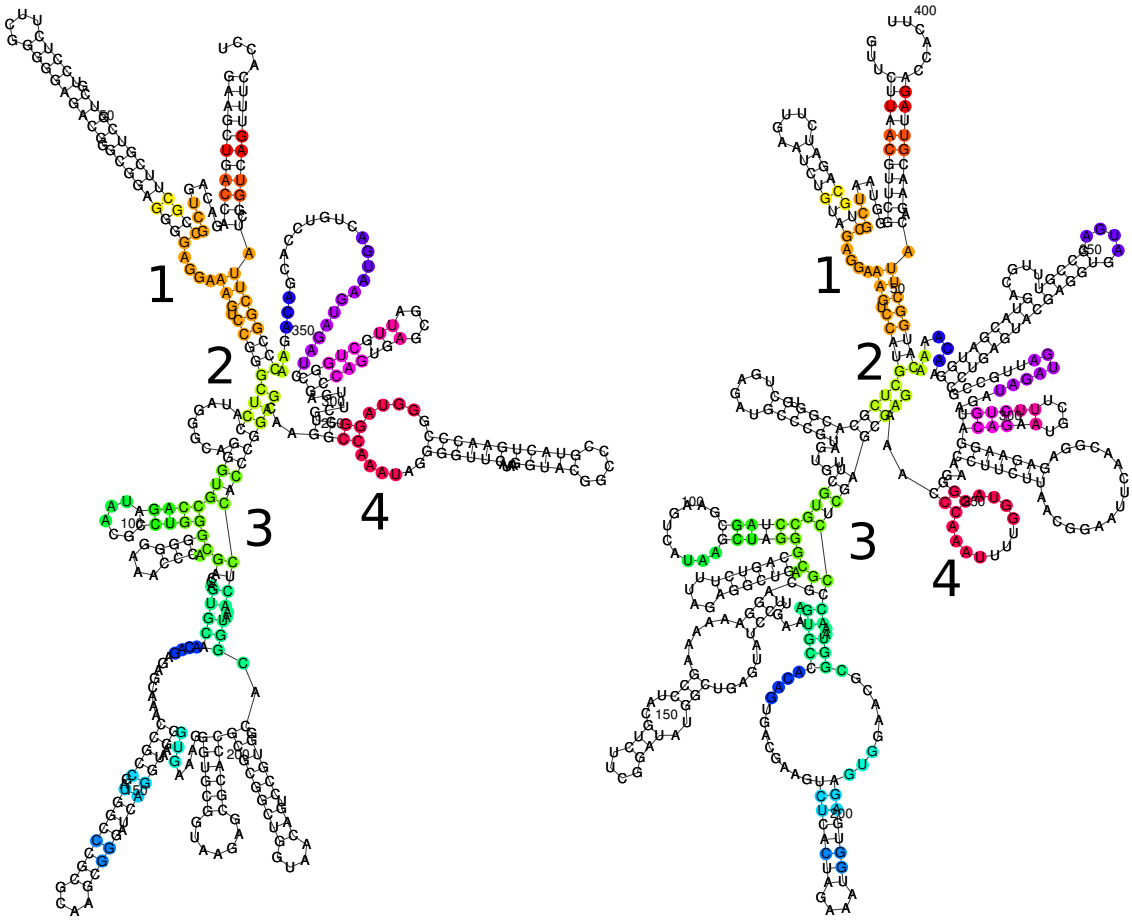


**Figure A.4.** Annotated structures from the `ExpaRNA` output. Regions with exact pattern matchings (EPMs) are indicated with a similar colour. Shown are two bacterial RNase P RNAs (left: A-type P RNA from *Escherichia coli*; right: B-type P RNA from *Bacillus subtilis*). Structures are taken from RNase P database [22]. `ExpaRNA` was called with a minimal EPM size of 3. The numbers indicate four large EPMs. Numbers and colours correspond to theq `ExpLoc` workflow shown in Figure 2.6. To get a full alignment, `ExpaRNA` is called on the input RNAs to predict EPMs. This information is used as anchor constraints for a complete sequence-structure alignment by `LocARNA`.

# Appendix B

# GraphClust

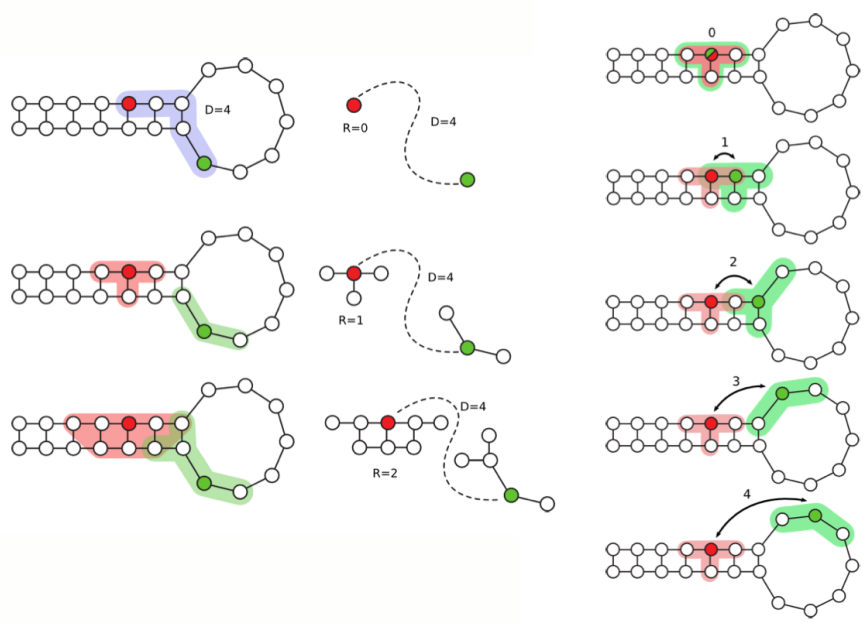


**Figure B.1.** Features induced by NSPDK on RNA secondary structure graphs. The left part shows the influence of the radius $r$ for a fixed distance $d = 4$. On the right the different features for a fixed radius $r = 1$, but varying distances $d$, are shown. NSPDK always induces a pairwise subgraph rooted at two vertices $u, v$ (indicated as red and green filled circles) within a distance $d$ (shown in purple). The radius determines the size of the neighborhood subgraph for one vertex. For $r = 0$ only the root vertices are used. With an increasing radius, each subgraph is expanded to all vertices within distance $r$ (indicated by red and green shaded regions). Right: the distance $d$ determines the number of vertices between the two roots $u$ and $v$. For certain combinations of $r$ and $d$, for example $r = 1$ and $d = 4$, the two subgraphs do not overlap which is beneficial to model different evolutionary constraints in the RNA structure. Please note that this figure omits the additional vertices introduced to model stacking base pairs.

| $i$ | $\#C$ | Phase 2 | Phase 3 | | $\text{Time}_i$ | $\text{Time}_{ALL}$ | $\text{Time}_C$ |
|---|---|---|---|---|---|---|---|
| 0 | | 4169 | 4145 | | 8314 | 8314 | |
| | | Phase 4 | Phase 5-6 | Phase 7 | | | |
| 1 | 10 | 458 | 10633 | 3904 | 14995 | 23309 | 2331 |
| 2 | 20 | 416 | 17980 | 1564 | 19962 | 43272 | 2163 |
| 3 | 30 | 334 | 13239 | 1533 | 15108 | 58380 | 1946 |
| 4 | 40 | 260 | 11694 | 752 | 12708 | 71088 | 1777 |
| 5 | 50 | 186 | 11351 | 783 | 12321 | 83409 | 1668 |
| 6 | 60 | 171 | 8667 | 726 | 9566 | 92975 | 1549 |
| 7 | 70 | 154 | 11069 | 739 | 11964 | 104940 | 1499 |
| 8 | 80 | 133 | 3671 | 597 | 4402 | 109342 | 1366 |
| 9 | 90 | 120 | 3841 | 620 | 4581 | 113924 | 1265 |
| 10 | 100 | 114 | 2768 | 707 | 3590 | 117515 | 1175 |
| 11 | 110 | 108 | 2544 | 492 | 3145 | 120660 | 1096 |
| 12 | 120 | 99 | 1917 | 712 | 2728 | 123388 | 1028 |
| 13 | 130 | 90 | 1197 | 640 | 1929 | 125318 | 964 |
| 14 | 140 | 83 | 1220 | 512 | 1817 | 127135 | 908 |
| 15 | 150 | 77 | 1776 | 636 | 2491 | **129626** | 864 |

**Table B.1.** Aggregated serial time for `Rfam` benchmark set. $\text{Time}_C$ denotes the average time per predicted candidate cluster $C$ up to iteration $i$. $\text{Time}_{ALL}$ is the total serial. All times are given in seconds.

| | | Partition | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | BEST | | | MERGED | | | | | | ORACLE | | | SOFT | | |
| Iteration | #Seq | #C | F | Rand | #C | F | Rand | Accuracy | Precision | Recall | #C | F | Rand | #C | F | Rand |
| 1 | 271 | 10 | 0.545 | 0.376 | 5 | 0.882 | 0.888 | 0.998 | 0.912 | 0.869 | 5 | 0.882 | 0.888 | 10 | 0.938 | 0.215 |
| 2 | 629 | 20 | 0.649 | 0.699 | 14 | 0.834 | 0.932 | 0.997 | 0.891 | 0.814 | 13 | 0.877 | 0.932 | 20 | 0.888 | 0.372 |
| 3 | 1076 | 30 | 0.743 | 0.864 | 23 | 0.868 | 0.956 | 0.997 | 0.928 | 0.841 | 22 | 0.894 | 0.957 | 30 | 0.879 | 0.429 |
| 4 | 1737 | 40 | 0.778 | 0.956 | 33 | 0.872 | 0.984 | 0.996 | 0.950 | 0.834 | 31 | 0.908 | 0.985 | 40 | 0.878 | 0.612 |
| 5 | 1844 | 50 | 0.774 | 0.943 | 42 | 0.860 | 0.984 | 0.997 | 0.924 | 0.839 | 39 | 0.906 | 0.985 | 50 | 0.862 | 0.609 |
| 6 | 2019 | 60 | 0.783 | 0.780 | 50 | 0.879 | 0.985 | 0.998 | 0.927 | 0.867 | 47 | 0.918 | 0.986 | 60 | 0.867 | 0.549 |
| 7 | 2181 | 69 | 0.786 | 0.783 | 58 | 0.877 | 0.985 | 0.998 | 0.922 | 0.875 | 55 | 0.910 | 0.986 | 69 | 0.865 | 0.554 |
| 8 | 2321 | 79 | 0.787 | 0.781 | 67 | 0.873 | 0.985 | 0.998 | 0.911 | 0.883 | 62 | 0.912 | 0.986 | 79 | 0.853 | 0.558 |
| 9 | 2391 | 89 | 0.782 | 0.782 | 77 | 0.856 | 0.985 | 0.998 | 0.892 | 0.872 | 70 | 0.908 | 0.986 | 89 | 0.838 | 0.559 |
| 10 | 2440 | 99 | 0.773 | 0.781 | 86 | 0.845 | 0.985 | 0.998 | 0.881 | 0.868 | 77 | 0.904 | 0.983 | 99 | 0.824 | 0.555 |
| 11 | 2503 | 109 | 0.767 | 0.779 | 94 | 0.846 | 0.984 | 0.998 | 0.878 | 0.872 | 84 | 0.907 | 0.983 | 109 | 0.821 | 0.555 |
| 12 | 2587 | 118 | 0.765 | 0.659 | 102 | 0.843 | 0.985 | 0.999 | 0.876 | 0.869 | 91 | 0.907 | 0.984 | 118 | 0.815 | 0.481 |
| 13 | 2671 | 128 | 0.768 | 0.661 | 112 | 0.839 | 0.985 | 0.999 | 0.868 | 0.869 | 100 | 0.902 | 0.984 | 128 | 0.807 | 0.482 |
| 14 | 2742 | 138 | 0.771 | 0.661 | 122 | 0.837 | 0.984 | 0.999 | 0.868 | 0.867 | 109 | 0.900 | 0.984 | 138 | 0.801 | 0.482 |
| 15 | 2821 | 148 | 0.766 | 0.649 | 130 | 0.834 | 0.984 | 0.999 | 0.856 | 0.876 | 117 | 0.892 | 0.983 | 148 | 0.796 | 0.483 |

**Table B.2.** `GraphClust` results for the `Rfam` benchmark set after each iteration for all predicted clusters for different partition types. The F measure is the average over all cluster predicted until the given iteration. The Rand index is based on the partition for all clustered RNA candidates. For the MERGED partition we provide in addition accuracy, precision and recall.

| | | Partition | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | BEST | | | MERGED | | | | | | ORACLE | | | SOFT | | |
| Iteration | #Seq | #C | F | Rand | #C | F | Rand | Accuracy | Precision | Recall | #C | F | Rand | #C | F | Rand |
| 1 | 140 | 10 | 0.942 | 0.945 | 10 | 0.942 | 0.945 | 0.996 | 0.924 | 0.974 | 10 | 0.942 | 0.945 | 10 | 0.942 | 0.897 |
| 2 | 232 | 20 | 0.926 | 0.939 | 20 | 0.926 | 0.939 | 0.996 | 0.903 | 0.971 | 20 | 0.926 | 0.939 | 20 | 0.916 | 0.892 |
| 3 | 270 | 26 | 0.936 | 0.935 | 26 | 0.936 | 0.935 | 0.996 | 0.920 | 0.970 | 26 | 0.936 | 0.935 | 26 | 0.928 | 0.894 |
| 4 | 298 | 30 | 0.925 | 0.922 | 30 | 0.925 | 0.922 | 0.996 | 0.902 | 0.971 | 29 | 0.929 | 0.906 | 30 | 0.907 | 0.874 |
| 5 | 305 | 32 | 0.909 | 0.918 | 32 | 0.909 | 0.918 | 0.996 | 0.880 | 0.973 | 30 | 0.925 | 0.900 | 32 | 0.892 | 0.872 |
| 6 | 319 | 34 | 0.897 | 0.904 | 34 | 0.897 | 0.904 | 0.996 | 0.861 | 0.974 | 32 | 0.911 | 0.887 | 34 | 0.881 | 0.862 |
| 7 | 329 | 35 | 0.890 | 0.897 | 35 | 0.890 | 0.897 | 0.995 | 0.851 | 0.975 | 33 | 0.903 | 0.881 | 35 | 0.875 | 0.857 |
| 8 | 332 | 36 | 0.883 | 0.898 | 36 | 0.883 | 0.898 | 0.995 | 0.844 | 0.966 | 34 | 0.895 | 0.882 | 36 | 0.866 | 0.856 |
| 9 | 335 | 37 | 0.882 | 0.901 | 37 | 0.882 | 0.901 | 0.995 | 0.840 | 0.967 | 35 | 0.894 | 0.884 | 37 | 0.863 | 0.854 |
| 10 | 339 | 38 | 0.867 | 0.891 | 38 | 0.867 | 0.891 | 0.995 | 0.831 | 0.948 | 36 | 0.878 | 0.875 | 38 | 0.852 | 0.846 |
| 11 | 345 | 39 | 0.871 | 0.899 | 39 | 0.871 | 0.899 | 0.995 | 0.839 | 0.946 | 37 | 0.881 | 0.888 | 39 | 0.848 | 0.828 |
| 12 | 349 | 40 | 0.868 | 0.900 | 39 | 0.873 | 0.894 | 0.995 | 0.848 | 0.940 | 38 | 0.876 | 0.889 | 40 | 0.839 | 0.815 |
| 13 | 353 | 41 | 0.868 | 0.902 | 39 | 0.871 | 0.891 | 0.994 | 0.848 | 0.934 | 38 | 0.892 | 0.893 | 41 | 0.837 | 0.794 |
| 14 | 357 | 42 | 0.854 | 0.809 | 39 | 0.872 | 0.886 | 0.995 | 0.848 | 0.936 | 38 | 0.893 | 0.888 | 42 | 0.836 | 0.646 |
| 15 | 360 | 43 | 0.843 | 0.794 | 39 | 0.858 | 0.866 | 0.993 | 0.841 | 0.916 | 38 | 0.893 | 0.874 | 43 | 0.827 | 0.636 |

**Table B.3.** `GraphClust` results for the small ncRNA benchmark set after each iteration for all predicted clusters for different partition types. The F measure is the average over all clusters predicted until the given iteration.

| $i$ | $\#C$ | Phase 2 | Phase 3 | | $\text{Time}_i$ | $\text{Time}_{ALL}$ | $\text{Time}_C$ |
|---|---|---|---|---|---|---|---|
| 0 | | 225 | 495 | | 720 | 720 | |
| | | Phase 4 | Phase 5-6 | Phase 7 | | | |
| 1 | 10 | 41.76 | 2192 | 200 | 2434 | 3154 | 315 |
| 2 | 20 | 27.01 | 3132 | 236 | 3395 | 6549 | 327 |
| 3 | 26 | 17.02 | 7548 | 116 | 7681 | 14230 | 547 |
| 4 | 30 | 13.33 | 3618 | 163 | 3795 | 18025 | 601 |
| 5 | 32 | 8.63 | 3792 | 75 | 3876 | 21902 | 684 |
| 6 | 34 | 6.78 | 984 | 43 | 1034 | 22936 | 675 |
| 7 | 35 | 4.81 | 221 | 24 | 250 | 23186 | 663 |
| 8 | 36 | 3.65 | 230 | 21 | 255 | 23441 | 651 |
| 9 | 37 | 3.07 | 135 | 16 | 155 | 23596 | 638 |
| 10 | 38 | 2.74 | 102 | 24 | 129 | 23725 | 624 |
| 11 | 39 | 2.17 | 73 | 33 | 108 | 23832 | 611 |
| 12 | 40 | 1.73 | 62 | 23 | 87 | 23920 | 598 |
| 13 | 41 | 1.42 | 114 | 27 | 143 | 24063 | 587 |
| 14 | 42 | 1.15 | 115 | 30 | 146 | 24209 | 576 |
| 15 | 43 | 0.88 | 73 | 18 | 92 | **24301** | 565 |

**Table B.4.** Aggregated serial time for small ncRNA benchmark set. $\text{Time}_C$ denotes the average time per predicted candidate cluster $C$ up to iteration $i$. $\text{Time}_{ALL}$ is the total serial. All times are given in seconds.

| | | | BEST | | | Partition MERGED | | | | | | ORACLE | | | SOFT | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sample | Iteration | #Seq | #C | F | Rand | #C | F | Rand | Accuracy | Precision | Recall | #C | F | Rand | #C | F | Rand |
| *Rfam benchmark* | | | | | | | | | | | | | | | | | |
| 100% | 15 | 2821 | 148 | 0.766 | 0.649 | 130 | 0.834 | 0.984 | 0.999 | 0.856 | 0.876 | 117 | 0.892 | 0.983 | 148 | 0.796 | 0.483 |
| 50% | 15 | 2707 | 150 | 0.758 | 0.655 | 127 | 0.850 | 0.984 | 0.999 | 0.888 | 0.881 | 117 | 0.909 | 0.988 | 150 | 0.809 | 0.479 |
| 25% | 15 | 2754 | 149 | 0.758 | 0.636 | 130 | 0.835 | 0.990 | 0.998 | 0.867 | 0.877 | 120 | 0.888 | 0.995 | 149 | 0.799 | 0.518 |
| *small ncRNAs benchmark* | | | | | | | | | | | | | | | | | |
| 100% | 15 | 360 | 43 | 0.843 | 0.794 | 39 | 0.858 | 0.866 | 0.993 | 0.841 | 0.916 | 38 | 0.893 | 0.874 | 43 | 0.827 | 0.636 |
| 50% | 15 | 299 | 38 | 0.854 | 0.812 | 35 | 0.880 | 0.913 | 0.994 | 0.913 | 0.890 | 33 | 0.920 | 0.925 | 39 | 0.856 | 0.698 |
| 25% | 15 | 269 | 34 | 0.854 | 0.905 | 33 | 0.846 | 0.880 | 0.991 | 0.918 | 0.843 | 30 | 0.907 | 0.907 | 34 | 0.837 | 0.838 |

**Table B.5.** `GraphClust` results for the `Rfam` benchmark set and the small ncRNA benchmark using different sample sizes during phase 4. Shown is the result after 15 iterations. For the Rfam set the final quality is nearly identical between all used samples sizes. The small ncRNA set shows the same quality in terms of F measure and Rand index. Only the final number of clusters and clustered sequences is decreased. This is probably due to the small size of the dataset. For practical purposes this would mean to run 1-2 additional iterations of the pipline.

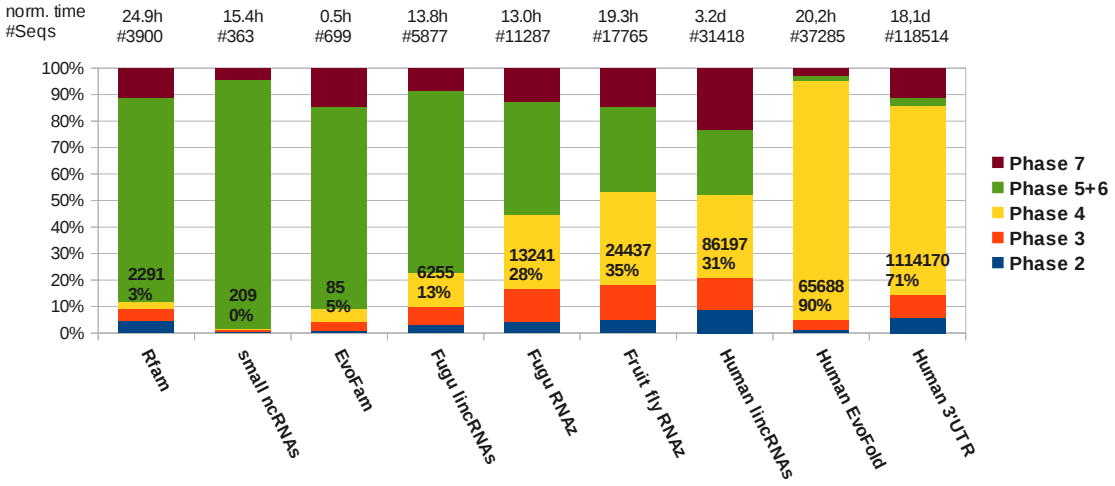


**Figure B.2.** Runtime comparison for all analyzed datasets. Shown is the relative runtime spent in phases 2-7 of the `GraphClust`-pipeline. Pre- and post processing phases are skipped. For the ease of comparison, we normalized all times to 5 iterations and 100 clusters. The time for phase 4 (clustering) is normalized to 100% sample size and is indicated on each bar (percentage and in seconds). On top of each dataset the normalized serial time and the number of sequences is given. For small datasets, the runtime is dominated by the cluster refinement step (Phase 5+6) which uses costly sequence-structure alignment. Please note that we do not normalize the influence of the sequence length which effects all phases, e.g. the RNA folding. The Rfam set and the small ncRNA set contain sequences up to 400 nt, whereas the `EvoFam` and `EvoFold` set contain mainly short sequences (average length = 36nt). `RNAz` screens have an average length of 120 nt and all other sets have an averge sequence length of 150 nt.

# Bibliography


[1] Abouelhoda, M. I. and Ohlebusch, E. Chaining algorithms for multiple genome comparison. *J. Discrete Algorithms*, 3(2-4):321–341, 2005.

[2] Addess, K. J., Basilion, J. P., Klausner, R. D., Rouault, T. A., and Pardi, A. Structure and dynamics of the iron responsive element RNA: implications for binding of the RNA by iron regulatory binding proteins. *J Mol Biol*, 274(1):72–83, 1997.

[3] Allali, J. and Sagot, M.-F. A new distance for high level RNA secondary structure comparison. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2(1):3–14, 2005.

[4] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–402, 1997.

[5] Backofen, R. and Siebert, S. Fast detection of common sequence structure patterns in RNAs. *Journal of Discrete Algorithms*, 5(2):212–228, 2007.

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

[7] Backofen, R., Tsur, D., Zakov, S., and Ziv-Ukelson, M. Sparse RNA folding: Time and space efficient algorithms. In Kucherov, G. and Ukkonen, E., editors, *Proc. 20th Symp. Combinatorial Pattern Matching*, volume 5577 of *LNCS*, pages 249–262. Springer, 2009.

[8] Bafna, V., Muthukrishnan, S., and Ravi, R. Computing similarity between RNA strings. In Galil, Z. and Ukkonen, E., editors, *Proc. 6th Symp. Combinatorial Pattern Matching*, volume 937 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1995.

[9] Bahr, A., Thompson, J. D., Thierry, J. C., and Poch, O. BAliBASE Benchmark Alignment dataBASE: enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acids Res*, 29(1):323–6, 2001.

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

[11] Bernhart, S. H., Hofacker, I. L., and Stadler, P. F. Local RNA base pairing probabilities in large sequences. *Bioinformatics*, 22(5):614–5, 2006.

[12] Bernhart, S. H., Hofacker, I. L., Will, S., Gruber, A. R., and Stadler, P. F. RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinformatics*, 9:474, 2008.

[13] Bertone, P., Stoc, V., Royce, T. E., Rozowsky, J. S., Urban, A. E., Zhu, X., Rinn, J. L., Tongprasit, W., Samanta, M., Weissman, S., Gerstein, M., and Snyder, M. Global identification of human transcribed sequences with genome tiling arrays. *Science*, 306:2242–2246, 2004.

[14] Blanchette, M., Kent, W. J., Riemer, C., Elnitski, L., Smit, A. F. A., Roskin, K. M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E. D., Haussler, D., and Miller, W. Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res*, 14(4):708–15, 2004.

[15] Blin, G., Fertin, G., Rusu, I., and Sinoquet, C. RNA sequences and the EDIT(NESTED,NESTED) problem. Technical Report RR-IRIN-03.07, IRIN, Université de Nantes. Submitted for publication, 2003.

[16] Bompfünewerer, A. F., Backofen, R., Bernhart, S. H., Hertel, J., Hofacker, I. L., Stadler, P. F., and Will, S. Variations on RNA folding and alignment: lessons from Benasque. *Journal of Mathematical Biology*, 56(1-2):129–144, 2008.

[17] Bompfünewerer Consortium, A. F., Backofen, R., Bernhart, S. H., Flamm, C., Fried, C., Fritzsch, G., Hackermüller, J., Hertel, J., Hofacker, I. L., Missal, K., Mosig, A., Prohaska, S. J., Rose, D., Stadler, P. F., Tanzer, A., Washietl, S., and Will, S. RNAs everywhere: genome-wide annotation of structured RNAs. *J Exp Zoolog B Mol Dev Evol*, 308B(1):1–25, 2007.

[18] Bradley, R. K., Pachter, L., and Holmes, I. Specific alignment of structured RNA: stochastic grammars and sequence annealing. *Bioinformatics*, 24(23):2677–83, 2008.

[19] Breaker, R. R. Riboswitches and the RNA world. *Cold Spring Harb Perspect Biol*, 4(2):a003566, 2012.

[20] Brion, P. and Westhof, E. Hierarchy and dynamics of RNA folding. *Annual Review of Biophysics and Biomolecular Structure*, 26(1):113–137, 1997. PMID: 9241415.

[21] Broder, A. Z. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–29, Washington, DC, USA, 1997. IEEE Computer Society.

[22] Brown, J. W. The ribonuclease P database. *Nucleic Acids Res*, 27(1):314, 1999.

[23] Burge, S. W., Daub, J., Eberhardt, R., Tate, J., Barquist, L., Nawrocki, E. P., Eddy, S. R., Gardner, P. P., and Bateman, A. Rfam 11.0: 10 years of RNA families. *Nucleic Acids Res*, 41(Database issue):D226–32, 2013.

[24] Cabili, M. N., Trapnell, C., Goff, L., Koziol, M., Tazon-Vega, B., Regev, A., and Rinn, J. L. Integrative annotation of human large intergenic noncoding RNAs reveals global properties and specific subclasses. *Genes Dev.*, 25:1915–1927, 2011.

[25] Cannone, J. J., Subramanian, S., Schnare, M. N., Collett, J. R., D'Souza, L. M., Du, Y., Feng, B., Lin, N., Madabusi, L. V., Muller, K. M., Pande, N., Shang, Z., Yu, N., and Gutell, R. R. The Comparative RNA Web (CRW) Site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs: Correction. *BMC Bioinformatics*, 3(1):15, 2002.

[26] Cech, T. R. The RNA worlds in context. *Cold Spring Harbor Perspectives in Biology*, 2011.

[27] Cheng, J., Kapranov, P., Drenkow, J., Dike, S., Brubaker, S., Patel, S., Long, J., Stern, D., Tammana, H., Helt, G., Sementchenko, V., Piccolboni, A., Bekiranov, S., Bailey, D. K., Ganesh, M., Ghosh, S., Bell, I., Gerhard, D. S., and Gingeras, T. R. Transcriptional maps of 10 human chromosomes at 5-nucleotide resolution. *Science*, 308:1149–1154, 2005.

[28] Chodroff, R., Goodstadt, L., Sirey, T., Oliver, P., Davies, K., Green, E., Molnár, Z., and Ponting, C. Long noncoding RNA genes: conservation of sequence and brain expression among diverse amniotes. *Genome Biol*, 11:R72, 2010.

[29] Christoffels, A., Koh, E., Chia, J., Brenner, S., Aparicio, S., and Venkatesh, B. Fugu genome analysis provides evidence for a whole-genome duplication early during the evolution of ray-finned fishes. *Mol Biol Evol*, 21:1146–51, Jun 2004.

[30] Clark, M. B., Amaral, P. P., Schlesinger, F. J., Dinger, M. E., Taft, R. J., Rinn, J. L., Ponting, C. P., Stadler, P. F., Morris, K. V., Morillon, A., Rozowsky, J. S., Gerstein, M. B., Wahlestedt, C., Hayashizaki, Y., Carninci, P., Gingeras, T. R., and Mattick, J. S. The reality of pervasive transcription. *PLoS Biol*, 9(7):e1000625; discussion e1001102, 2011.

[31] Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J. D., and Yang, C. Finding interesting associations without support pruning. In *Proceedings of the 16th International Conference on Data Engineering*, ICDE '00, pages 489–, Washington, DC, USA, 2000. IEEE Computer Society.

[32] Cook, K. B., Kazan, H., Zuberi, K., Morris, Q., and Hughes, T. R. RBPDB: a database of RNA-binding specificities. *Nucleic Acids Res*, 39(Database issue): D301–8, 2011.

[33] Costa, F. and Grave, K. D. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26 th International Conference on Machine Learning*, pages 255–262. Omnipress, 2010.

[34] Couzin, J. Breakthrough of the year. Small RNAs make big splash. *Science*, 298 (5602):2296–7, 2002.

[35] Crick, F. Central dogma of molecular biology. *Nature*, 227(5258):561–3, 1970.

[36] Dalli, D., Wilm, A., Mainz, I., and Steger, G. STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time. *Bioinformatics*, 22(13):1593–9, 2006.

[37] Deigan, K. E., Li, T. W., Mathews, D. H., and Weeks, K. M. Accurate SHAPE-directed RNA structure determination. *Proc Natl Acad Sci USA*, 106(1):97–102, 2009.

[38] Do, C. B., Woods, D. A., and Batzoglou, S. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–8, 2006.

[39] Do, C. B., Foo, C.-S., and Batzoglou, S. A max-margin model for efficient simultaneous alignment and folding of RNA sequences. *Bioinformatics*, 24(13):i68–76, 2008.

[40] Doshi, K. J., Cannone, J. J., Cobaugh, C. W., and Gutell, R. R. Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for RNA secondary structure prediction. *BMC Bioinformatics*, 5:105, 2004.

[41] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification.* John Wiley & Sons, INC., 2001.

[42] Eddy, S. R. A memory-efficient dynamic programming algorithm for optimal alignment of a sequence to an RNA secondary structure. *BMC Bioinformatics*, 3(1):18, 2002.

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

[44] Edelsbrunner, H. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Heidelberg, Germany, 1987.

[45] ENCODE Project Consortium. Identification and analysis of functional elements in 1genome by the ENCODE pilot project. *Nature*, 447:799–816, Jun 2007.

[46] ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome, 2012.

[47] Ertöz, L., Steinbach, M., and Kumar, V. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *SIAM International Conference on Data Mining (SDM)*, 2003.

[48] Esteller, M. Non-coding RNAs in human disease. *Nat Rev Genet*, 12(12):861–74, 2011.

[49] Evans, P. A. Finding common subsequences with arcs and pseudoknots. In *CPM '99: Proceedings of the 10th Annual Symposium on Combinatorial Pattern Matching*, pages 270–280, London, UK, 1999. Springer-Verlag.

[50] Evans, P. A. *Algorithms and Complexity for Annotated Sequence Analysis.* PhD thesis, University of Alberta, 1999.

[51] Fernández-Suárez, X. M. and Galperin, M. Y. The 2013 nucleic acids research database issue and the online molecular biology database collection. *Nucleic Acids Research*, 2012.

[52] Fire, A., Xu, S., Montgomery, M. K., Kostas, S. A., Driver, S. E., and Mello, C. C. Potent and specific genetic interference by double-stranded RNA in caenorhabditis elegans. *Nature*, 391(6669):806–811, 1998.

[53] Gan, H. H., Pasquali, S., and Schlick, T. Exploring the repertoire of RNA secondary motifs using graph theory; implications for RNA design. *Nucleic Acids Research*, 31(11):2926–2943, 2003.

[54] Gardner, P. P., Wilm, A., and Washietl, S. A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Res*, 33(8):2433–9, 2005.

[55] Gardner, P. P., Daub, J., Tate, J., Moore, B. L., Osuch, I. H., Griffiths-Jones, S., Finn, R. D., Nawrocki, E. P., Kolbe, D. L., Eddy, S. R., and Bateman, A. Rfam: Wikipedia, clans and the "decimal" release. *Nucleic Acids Res*, 39(Database issue): D141–5, 2011.

[56] Giegerich, R., Voss, B., and Rehmsmeier, M. Abstract shapes of RNA. *Nucleic Acids Res*, 32:4843–51, 2004.

[57] Giegerich, R., Voss, B., and Rehmsmeier, M. Abstract shapes of RNA. *Nucleic Acids Res*, 32(16):4843–51, 2004.

[58] Gorodkin, J. and Hofacker, I. L. From structure prediction to genomic screens for novel non-coding RNAs. *PLoS Comput Biol*, 7(8):e1002100, 2011.

[59] Gorodkin, J., Heyer, L., and Stormo, G. Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Res*, 25 (18):3724–32, 1997.

[60] Gorodkin, J., Hofacker, I. L., Torarinsson, E., Yao, Z., Havgaard, J. H., and Ruzzo, W. L. De novo prediction of structured RNAs from genomic sequences. *Trends Biotechnol*, 28(1):9–19, 2010.

[61] Gottesman, S. and Storz, G. Bacterial small RNA regulators: versatile roles and rapidly evolving variations. *Cold Spring Harb Perspect Biol*, 2010. doi:10.1101/cshperspect.a003798.

[62] Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., and Eddy, S. R. Rfam: an RNA family database. *Nucleic Acids Res*, 31(1):439–41, 2003.

[63] Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S. R., and Bateman, A. Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res*, 33 Database Issue:D121–4, 2005.

[64] Gruber, A. R., Findeiss, S., Washietl, S., Hofacker, I. L., and Stadler, P. F. RNAz 2.0: improved noncoding RNA detection. In *PSB10*, volume 15, pages 69–79, 2010.

[65] Haussler, D. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, 1999.

[66] Havgaard, J. H., Lyngso, R. B., Stormo, G. D., and Gorodkin, J. Pairwise local structural alignment of RNA sequences with sequence similarity less than 40%. *Bioinformatics*, 21(9):1815–24, 2005.

[67] Havgaard, J. H., Torarinsson, E., and Gorodkin, J. Fast pairwise structural RNA alignments by pruning of the dynamical programming matrix. *PLoS Comput Biol*, 3(10):1896–908, 2007.

[68] Hentze, M. W. and Kuhn, L. C. Molecular control of vertebrate iron metabolism: mRNA-based regulatory circuits operated by iron, nitric oxide, and oxidative stress. *Proc Natl Acad Sci USA*, 93(16):8175–82, 1996.

[69] Hertel, J., Hofacker, I. L., and Stadler, P. F. SnoReport: Computational identification of snoRNAs with unknown targets. *Bioinformatics*, 24:158–164, 2007.

[70] Heyne, S., Will, S., Beckstette, M., and Backofen, R. Lightweight comparison of RNAs based on exact sequence-structure matches. In *Proceedings of the German Conference on Bioinformatics (GCB'2008)*, volume P-136 of *Lecture Notes in Informatics (LNI)*, pages 189–198. Gesellschaft für Informatik (GI), 2008.

[71] Heyne, S., Will, S., Beckstette, M., and Backofen, R. Lightweight comparison of RNAs based on exact sequence-structure matches. *Bioinformatics*, 25(16):2095–2102, 2009.

[72] Heyne, S., Costa, F., Rose, D., and Backofen, R. GraphClust: alignment-free structural clustering of local RNA secondary structures. *Bioinformatics*, 28(12): i224–i232, 2012.

[73] Höchsmann, M., Töller, T., Giegerich, R., and Kurtz, S. Local similarity in RNA secondary structures. In *Proceedings of Computational Systems Bioinformatics (CSB 2003)*, volume 2, pages 159–168. IEEE Computer Society, 2003.

[74] Hochsmann, M., Voss, B., and Giegerich, R. Pure multiple RNA secondary structure alignments: a progressive profile approach. *IEEE/ACM Trans Comput Biol Bioinform*, 1(1):53–62, 2004.

[75] Hofacker, I. L. and Stadler, P. F. RNA secondary structures. In *Bioinformatics-From Genomes to Therapies*, pages 439–489. Wiley-VCH Verlag GmbH, 2008.

[76] Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, S., Tacker, M., and Schuster, P. Fast folding and comparison of RNA secondary structures. *Monatshefte Chemie*, 125:167–188, 1994.

[77] Hofacker, I. L., Schuster, P., and Stadler, P. F. Combinatorics of RNA seondary structures. *Discrete Applied Mathematics*, 88:207–237, 1998.

[78] Hofacker, I. L., Bernhart, S. H., and Stadler, P. F. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 20(14):2222–7, 2004.

[79] Holmes, I. Accelerated probabilistic inference of RNA structure evolution. *BMC Bioinformatics*, 6:73, 2005.

[80] Huang, d., Sherman, B., and Lempicki, R. Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nat Protoc*, 4:44–57, 2009.

[81] Hubert, L. and Arabie, P. Comparing partitions. *Journal of Classification*, 2: 193–218, 1985. 10.1007/BF01908075.

[82] Huttenhofer, A., Westhof, E., and Bock, A. Solution structure of mRNA hairpins promoting selenocysteine incorporation in Escherichia coli and their base-specific interaction with special elongation factor SELB. *RNA*, 2(4):354–66, 1996.

[83] Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[84] Jiang, T., Wang, J., and Zhang, K. Alignment of trees - an alternative to tree edit. *Theoretical Computer Science*, 143(1):137–148, 1995.

[85] Jiang, T., Lin, G., Ma, B., and Zhang, K. A general edit distance between RNA structures. *J Comput Biol*, 9(2):371–88, 2002.

[86] Kaczkowski, B., Torarinsson, E., Reiche, K., Havgaard, J. H., Stadler, P. F., and Gorodkin, J. Structural profiles of human miRNA families from pairwise clustering. *Bioinformatics*, 25(3):291–4, 2009.

[87] Kaikkonen, M. U., Lam, M. T., and Glass, C. K. Non-coding RNAs as regulators of gene expression and epigenetics. *Cardiovascular Research*, 90(3):430–440, 2011.

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

[89] Karklin, Y., Meraz, R. F., and Holbrook, S. R. Classification of non-coding RNA using graph representations of secondary structure. In *Proc. of the Pacific Symposium on Biocomputing 2005 (PSB 2005)*, pages 4–15, 2005.

[90] Katoh, K. and Toh, H. Recent developments in the MAFFT multiple sequence alignment program. *Brief Bioinform*, 9(4):286–98, 2008.

[91] Khaladkar, M., Bellofatto, V., Wang, J. T. L., Tian, B., and Shapiro, B. A. RADAR: a web server for RNA data analysis and research. *Nucleic Acids Res*, 35(Web Server issue):W300–4, 2007.

[92] Kin, T., Tsuda, K., and Asai, K. Marginalized kernels for RNA sequence data analysis. *Genome Inform*, 13:112–22, 2002.

[93] Klein, R. J. and Eddy, S. R. RSEARCH: finding homologs of single structured RNA sequences. *BMC Bioinformatics*, 4(1):44, 2003.

[94] Kolbe, D. L. and Eddy, S. R. Local RNA structure alignment with incomplete sequence. *Bioinformatics*, 25(10):1236–43, 2009.

[95] Kozomara, A. and Griffiths-Jones, S. miRBase: integrating microRNA annotation and deep-sequencing data. *Nucleic Acids Research*, 39(Database-Issue):152–157, 2011.

[96] Kunin, V., Sorek, R., and Hugenholtz, P. Evolutionary conservation of sequence and secondary structures in CRISPR repeats. *Genome Biol*, 8(4):R61, 2007.

[97] Lagesen, K., Hallin, P., Rødland, E. A., Stærfeldt, H.-H., Rognes, T., and Ussery, D. W. RNAmmer: consistent and rapid annotation of ribosomal RNA genes. *Nucleic Acids Research*, 35(9):3100–3108, 2007.

[98] Lagos-Quintana, M., Rauhut, R., Lendeckel, W., and Tuschl, T. Identification of novel genes coding for small expressed RNAs. *Science*, 294:853–857, 2001.

[99] Lange, S. J., Maticzka, D., Mohl, M., Gagnon, J. N., Brown, C. M., and Backofen, R. Global or local? Predicting secondary structure and accessibility in mRNAs. *Nucleic Acids Res*, 40(12):5215–26, 2012.

[100] Leontis, N. B. and Westhof, E. Geometric nomenclature and classification of RNA base pairs. *RNA*, 7(4):499–512, 2001.

[101] Leontis, N. B. and Westhof, E. Analysis of RNA motifs. *Current Opinion in Structural Biology*, 13(3):300 – 308, 2003.

[102] Lin, G., Chen, Z.-Z., Jiang, T., and Wen, J. The longest common subsequence problem for sequences with nested arc annotations. *J. Comput. Syst. Sci.*, 65(3): 465–480, 2002.

[103] Liu, N. and Wang, T. A method for rapid similarity analysis of RNA secondary structures. *BMC Bioinformatics*, 7(1):493, 2006.

[104] Livny, J. and Waldor, M. K. Identification of small RNAs in diverse bacterial species. *Curr Opin Microbiol*, 10(2):96–101, 2007.

[105] Lorenz, R., Bernhart, S. H., Honer Zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. ViennaRNA Package 2.0. *Algorithms Mol Biol*, 6:26, 2011.

[106] Lowe, T. M. and Eddy, S. R. tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Res*, 25(5):955–64, 1997.

[107] Lyngso, R. B. and Pedersen, C. N. S. Pseudoknots in RNA secondary structures. In *Proc. of the Fourth Annual International Conferences on Computational Molecular Biology (RECOMB'00)*. ACM Press, 2000. BRICS Report Series RS-00-1.

[108] Lyngso, R. B., Zuker, M., and Pedersen, C. N. Fast evaluation of internal loops in RNA secondary structure prediction. *Bioinformatics*, 15(6):440–5, 1999.

[109] Manak, R. J., Dike, S., Sementchenko, V., Kapranov, P., Biemar, F., Long, J., Cheng, J., Bell, I., Ghosh, S., Piccolboni, A., and Gingeras, T. R. Biological function of unannotated transcription during the early development of Drosophila melanogaster. *Nature Genetics*, 38(10):1151–1158, Sept. 2006.

[110] Mann, M., Smith, C., Rabbath, M., Edwards, M., Will, S., and Backofen, R. CPSP-web-tools: a server for 3D lattice protein studies. *Bioinformatics*, 25(5): 676–7, 2009.

[111] Martin, K. C. and Ephrussi, A. mRNA localization: gene expression in the spatial dimension. *Cell*, 136(4):719–30, 2009.

[112] Martineau, Y., Le Bec, C., Monbrun, L., Allo, V., Chiu, I.-M., Danos, O., Moine, H., Prats, H., and Prats, A.-C. Internal Ribosome Entry Site Structural Motifs Conserved among Mammalian Fibroblast Growth Factor 1 Alternatively Spliced mRNAs. *Mol Cell Biol*, 24(17):7622–35, 2004.

[113] Mathews, D., Sabina, J., Zuker, M., and Turner, D. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol*, 288(5):911–40, 1999.

[114] Mathews, D. H. and Turner, D. H. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J Mol Biol*, 317(2):191–203, 2002.

[115] Mattick, J. S. and Makunin, I. V. Non-coding RNA. *Hum Mol Genet*, 15 Spec No 1:R17–29, 2006.

[116] Mattick, J. S., Taft, R. J., and Faulkner, G. J. A global view of genomic information - moving beyond the gene and the master regulator. *Trends in Genetics*, 2009.

[117] McCaskill, J. S. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–19, 1990.

[118] Minsky, M. and Papert, S. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[119] Morita, K., Saito, Y., Sato, K., Oka, K., Hotta, K., and Sakakibara, Y. Genome-wide searching with base-pairing kernel functions for noncoding RNAs: computational and expression analysis of snoRNA families in Caenorhabditis elegans. *Nucleic Acids Res*, 37(3):999–1009, 2009.

[120] Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L., and Wold, B. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat Methods*, 5(7):621–8, 2008.

[121] Nawrocki, E. P. and Eddy, S. R. Query-dependent banding (qdb) for faster RNA similarity searches. *PLoS Comput Biol*, 3(3):e56, 03 2007.

[122] Nawrocki, E. P., Kolbe, D. L., and Eddy, S. R. Infernal 1.0: inference of RNA alignments. *Bioinformatics*, 25(10):1335–7, 2009.

[123] Notredame, C., Higgins, D. G., and Heringa, J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–17, 2000.

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

[125] Otto, W., Will, S., and Backofen, R. Structure local multiple alignment of RNA. In *Proceedings of German Conference on Bioinformatics (GCB'2008)*, volume P-136 of *Lecture Notes in Informatics (LNI)*, pages 178–188. Gesellschaft für Informatik (GI), 2008.

[126] Parker, B. J., Moltke, I., Roth, A., Washietl, S., Wen, J., Kellis, M., Breaker, R., and Pedersen, J. S. New families of human regulatory RNA structures identified by comparative analysis of vertebrate genomes. *Genome Res*, 2011.

[127] Pauli, A., Valen, E., Lin, M. F., Garber, M., Vastenhouw, N. L., Levin, J. Z., Fan, L., Sandelin, A., Rinn, J. L., Regev, A., and Schier, A. F. Systematic identification of long non-coding RNAs expressed during zebrafish embryogenesis. *Genome Research*, 22(3):577–59, 2011.

[128] Pedersen, J. S., Bejerano, G., Siepel, A., Rosenbloom, K., Lindblad-Toh, K., Lander, E. S., Kent, J., Miller, W., and Haussler, D. Identification and Classification of Conserved RNA Secondary Structures in the Human Genome. *PLoS Comput Biol*, 2(4):e33, 2006.

[129] Pruitt, K. D., Tatusova, T., Klimke, W., and Maglott, D. R. NCBI Reference Sequences: current status, policy and new initiatives. *Nucleic Acids Res*, 37(Database issue):D32–6, 2009.

[130] Qureshi, I., Mattick, J., and Mehler, M. Long non-coding RNAs in nervous system function and disease. *Brain Res*, 1338:20–35, Jun 2010.

[131] Rabani, M., Kertesz, M., and Segal, E. Computational prediction of RNA structural motifs involved in posttranscriptional regulatory processes. *Proc Natl Acad Sci USA*, 105(39):14885–90, 2008.

[132] Rederstorff, M., Bernhart, S. H., Tanzer, A., Zywicki, M., Perfler, K., Lukasser, M., Hofacker, I. L., and Huttenhofer, A. RNPomics: defining the ncRNA transcriptome by cDNA library generation from ribonucleo-protein particles. *Nucleic Acids Res*, 38(10):e113, 2010.

[133] Reuter, J. S. and Mathews, D. H. RNAstructure: software for RNA secondary structure prediction and analysis. *BMC Bioinformatics*, 11:129, 2010.

[134] Rijsbergen, C. J. V. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[135] Ritchie, W., Legendre, M., and Gautheret, D. RNA stem-loops: to be or not to be cleaved by RNAse III. *RNA*, 13(4):457–62, 2007.

[136] Rivas, E. and Eddy, S. R. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J Mol Biol*, 285(5):2053–68, 1999.

[137] Rivas, E. and Eddy, S. R. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, 16(7):583–605, 2000.

[138] Rivas, E. and Eddy, S. R. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, 2(1):8, 2001.

[139] Rivas, E., Klein, R. J., Jones, T. A., and Eddy, S. R. Computational identification of noncoding RNAs in *E. coli* by comparative genomics. *Curr Biol*, 11(17):1369–73, 2001.

[140] Rose, D., Hackermuller, J., Washietl, S., Reiche, K., Hertel, J., Findeiss, S., Stadler, P. F., and Prohaska, S. J. Computational RNomics of drosophilids. *BMC Genomics*, 8:406, 2007.

[141] Rose, D., Jöris, J., Hackermüller, J., Reiche, K., Li, Q., and Stadler, P. Duplicated RNA genes in teleost fish genomes. *J Bioinform Comput Biol*, 6:1157–75, Dec 2008.

[142] Ruby, J. G., Jan, C. H., and Bartel, D. P. Intronic microRNA precursors that bypass drosha processing. *Nature*, 448(7149):83–86, July 2007.

[143] Sadreyev, R. and Grishin, N. COMPASS: a tool for comparison of multiple protein alignments with assessment of statistical significance. *J Mol Biol*, 326(1):317–36, 2003.

[144] Saito, Y., Sato, K., and Sakakibara, Y. Fast and accurate clustering of noncoding RNAs using ensembles of sequence alignments and secondary structures. *BMC Bioinformatics*, 12 Suppl 1:S48, 2011.

[145] Sakakibara, Y., Popendorf, K., Ogawa, N., Asai, K., and Sato, K. Stem kernels for RNA sequence analyses. *J Bioinform Comput Biol*, 5(5):1103–22, 2007.

[146] Salari, R., Möhl, M., Will, S., Sahinalp, S. C., and Backofen, R. Time and space efficient RNA-RNA interaction prediction via sparse folding. In Berger, B., editor, *Proc. of RECOMB 2010*, volume 6044 of *Lecture Notes in Computer Science*, pages 473–490. Springer-Verlag Berlin Heidelberg, 2010.

[147] Sankoff, D. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, 45(5):810–825, 1985.

[148] Sato, K., Mituyama, T., Asai, K., and Sakakibara, Y. Directed acyclic graph kernels for structural RNA analysis. *BMC Bioinformatics*, 9:318, 2008.

[149] Sato, K., Kato, Y., Hamada, M., Akutsu, T., and Asai, K. IPknot: fast and accurate prediction of RNA secondary structures with pseudoknots using integer programming. *Bioinformatics*, 27(13):i85–93, 2011.

[150] Schmiedl, C., Möhl, M., Heyne, S., Amit, M., Landau, G. M., Will, S., and Backofen, R. Exact pattern matching for RNA structure ensembles. In *Proceedings of the 16th International Conference on Research in Computational Molecular Biology (RECOMB 2012)*, volume 7262 of *LNCS*, pages 245–260. Springer-Verlag, 2012.

[151] Seemann, S. E., Gorodkin, J., and Backofen, R. Unifying evolutionary and thermodynamic information for RNA folding of multiple alignments. *Nucleic Acids Res*, 36(20):6355–62, 2008.

[152] Serganov, A. and Patel, D. J. Ribozymes, riboswitches and beyond: regulation of gene expression without proteins. *Nat Rev Genet*, 8(10):776–90, 2007.

[153] Shi, Y., Tyson, G. W., and DeLong, E. F. Metatranscriptomics reveals unique microbial small RNAs in the ocean's water column. *Nature*, 459(7244):266–9, 2009.

[154] Siebert, S. and Backofen, R. MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, 21 (16):3352–9, 2005.

[155] Siebert, S. and Backofen, R. A dynamic programming approach for finding common patterns in RNAs. *J Comput Biol*, 14(1):33–44, 2007.

[156] Siomi, M. C., Sato, K., Pezic, D., and Aravin, A. A. Piwi-interacting small RNAs: the vanguard of genome defence. *Nature Reviews Molecular Cell Biology*, 12(4): 246–258, 2011.

[157] Smith, C., Heyne, S., Richter, A. S., Will, S., and Backofen, R. Freiburg RNA Tools: a web server integrating IntaRNA, ExpaRNA and LocARNA. *Nucleic Acids Res*, 38 Suppl:W373–7, 2010.

[158] Smith, M. A., Gesell, T., Stadler, P. F., and Mattick, J. S. Widespread purifying selection on RNA structure in mammals. *Nucleic Acids Research*, 41(17):8220–8236, 2013.

[159] Sperschneider, J. and Datta, A. KnotSeeker: heuristic pseudoknot detection in long RNA sequences. *RNA*, 14(4):630–40, 2008.

[160] Taft, R. J., Pang, K. C., Mercer, T. R., Dinger, M., and Mattick, J. S. Non-coding RNAs: regulators of disease. *The Journal of Pathology*, 220(2):126–139, 2010.

[161] Thanbichler, M. and Bock, A. The function of SECIS RNA in translational control of gene expression in Escherichia coli. *EMBO J*, 21(24):6925–34, 2002.

[162] The FANTOM Consortium. The transcriptional landscape of the mammalian genome. *Science*, 309(5740):1559–63, 2005.

[163] Torarinsson, E., Havgaard, J. H., and Gorodkin, J. Multiple structural alignment and clustering of RNA sequences. *Bioinformatics*, 23(8):926–32, 2007.

[164] Tseng, H.-H., Weinberg, Z., Gore, J., Breaker, R. R., and Ruzzo, W. L. Finding non-coding RNAs through genome-scale clustering. *J Bioinform Comput Biol*, 7 (2):373–88, 2009.

[165] Ulitsky, I. and Bartel, D. lincRNAs: Genomics, evolution, and mechanisms. *Cell*, 154(1):26 – 46, 2013.

[166] Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *J. Mach. Learn. Res.*, 99:1201–1242, August 2010.

[167] Wang, J. T. L. and Wu, X. Kernel design for RNA classification using Support Vector Machines. *Int J Data Min Bioinform*, 1(1):57–76, 2006.

[168] Washietl, S. and Hofacker, I. L. Identifying structural noncoding RNAs using RNAz. *Curr Protoc Bioinformatics*, Chapter 12:Unit 12.7, 2007.

[169] Washietl, S., Hofacker, I. L., Lukasser, M., Hüttenhofer, A., and Stadler, P. F. Mapping of conserved RNA secondary structures predicts thousands of functional noncoding RNAs in the human genome. *Nat Biotechnol*, 23(11):1383–90, 2005.

[170] Washietl, S., Hofacker, I. L., and Stadler, P. F. Fast and reliable prediction of noncoding RNAs. *Proc Natl Acad Sci USA*, 102(7):2454–9, 2005.

[171] Washietl, S., Pedersen, J. S., Korbel, J. O., Stocsits, C., Gruber, A. R., Hackermuller, J., Hertel, J., Lindemeyer, M., Reiche, K., Tanzer, A., Ucla, C., Wyss, C., Antonarakis, S. E., Denoeud, F., Lagarde, J., Drenkow, J., Kapranov, P., Gingeras, T. R., Guigo, R., Snyder, M., Gerstein, M. B., Reymond, A., Hofacker, I. L., and Stadler, P. F. Structured RNAs in the ENCODE selected regions of the human genome. *Genome Res*, 17(6):852–64, 2007.

[172] Washietl, S., Hofacker, I. L., Stadler, P. F., and Kellis, M. RNA folding with soft constraints: reconciliation of probing data and thermodynamic secondary structure prediction. *Nucleic Acids Res*, 40(10):4261–72, 2012.

[173] Washietl, S., Will, S., Hendrix, D. A., Goff, L. A., Rinn, J. L., Berger, B., and Kellis, M. Computational analysis of noncoding RNAs. *Wiley Interdiscip Rev RNA*, 3(6): 759–78, 2012.

[174] Washietl, S., Will, S., Hendrix, D. A., Goff, L. A., Rinn, J. L., Berger, B., and Kellis, M. Computational analysis of noncoding RNAs. *Wiley Interdiscip Rev RNA*, 3(6): 759–78, 2012.

[175] Watson, J. D. and Crick, F. H. C. Molecular structure of nucleic acids. A structure for deoxyribose nucleic acid. *Nature*, 171:737–738, 1953.

[176] Weinberg, Z., Barrick, J. E., Yao, Z., Roth, A., Kim, J. N., Gore, J., Wang, J. X., Lee, E. R., Block, K. F., Sudarsan, N., Neph, S., Tompa, M., Ruzzo, W. L., and Breaker, R. R. Identification of 22 candidate structured RNAs in bacteria using the CMfinder comparative genomics pipeline. *Nucleic Acids Res*, 35(14):4809–19, 2007.

[177] Westhof, E., Masquida, B., and Jossinet, F. Predicting and modeling RNA architecture. *Cold Spring Harbor Perspectives in Biology*, 3(2), 2011.

[178] Wexler, Y., Zilberstein, C., and Ziv-Ukelson, M. A study of accessible motifs and RNA folding complexity. *J Comput Biol*, 14(6):856–72, 2007.

[179] Will, S., Reiche, K., Hofacker, I. L., Stadler, P. F., and Backofen, R. Inferring non-coding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput Biol*, 3(4):e65, 2007.

[180] Will, S., Joshi, T., Hofacker, I. L., Stadler, P. F., and Backofen, R. LocARNA-P: Accurate boundary prediction and improved detection of structural RNAs. *RNA*, 18(5):900–914, 2012.

[181] Will, S., Miladi, C. S. M., Möhl, M., and Backofen, R. SPARSE: Quadratic time simultaneous alignment and folding of RNAs without sequence-based heuristics. In Sun, F., editor, *Proceedings of the 17th International Conference on Research in Computational Molecular Biology (RECOMB 2013)*, LNCS, 2013. To appear.

[182] Will, S., Siebauer, M. F., Heyne, S., Engelhardt, J., Stadler, P. F., Reiche, K., and Backofen, R. LocARNAscan: Incorporating thermodynamic stability in sequence and structure-based RNA homology search. *Algorithms Mol Biol*, 8(1):14, 2013.

[183] Will, S., Yu, M., and Berger, B. Structure-based whole genome realignment reveals many novel non-coding RNAs. *Genome Res*, page [Epub ahead of print], 2013.

[184] Wilm, A., Mainz, I., and Steger, G. An enhanced RNA alignment benchmark for sequence alignment programs. *Algorithms Mol Biol*, 1:19, 2006.

[185] Wilting, R., Schorling, S., Persson, B. C., and Böck, A. Selenoprotein synthesis in archaea: Identification of an mRNA element of Methanococcus jannaschii probably directing selenocysteine insertion. *J Mol Biol*, 266(4):637–41, 1997.

[186] Wuchty, S., Fontana, W., Hofacker, I. L., and Schuster, P. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49(2):145–65, 1999.

[187] Xue, C., Li, F., He, T., Liu, G.-P., Li, Y., and Zhang, X. Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. *BMC Bioinformatics*, 6:310, 2005.

[188] Yao, Z., Weinberg, Z., and Ruzzo, W. L. CMfinder – a covariance model based RNA motif finding algorithm. *Bioinformatics*, 22(4):445–52, 2006.

[189] Yao, Z., Barrick, J., Weinberg, Z., Neph, S., Breaker, R., Tompa, M., and Ruzzo, W. L. A computational pipeline for high- throughput discovery of cis-regulatory noncoding RNA in prokaryotes. *PLoS Comput Biol*, 3(7):e126, 2007.

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

[191] Ziv-Ukelson, M., Gat-Viks, I., Wexler, Y., and Shamir, R. A faster algorithm for RNA co-folding. In Crandall, K. A. and Lagergren, J., editors, *WABI 2008*, volume 5251 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 2008.

[192] Zuker, M. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res*, 31(13):3406–15, 2003.

[193] Zuker, M. and Stiegler, P. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res*, 9(1):133–48, 1981.

# Abbreviations

| | |
|---|---|
| bp | base pair |
| DAG | directed acyclic graph |
| DP | dynamic programming |
| EM | expectation maximization |
| EPM | exact pattern matching |
| lincRNA | long intergenic non-coding RNA |
| miRNA | micro RNA |
| mRNA | messenger RNA |
| MFE | minimum free energy |
| MPI | mean pairwise identity |
| ncRNA | non-coding RNA |
| nt | nucleotide(s) |
| NSPDK | neighborhood subgraph pairwise distance kernel |
| rRNA | ribosomal RNA |
| RNA | ribonucleic acid |
| SCFG | stochastic context-free grammar |
| SCI | structure conservation index |
| SECIS | selenocystein insertion sequence |
| SVM | support vector machine |
| tRNA | transfer RNA |
| tmRNA | transfer-messenger RNA |
| UTR | untranslated region |