Chapter 2

Introductory Material

2.1 Preliminaries

2.1.1 Metrics

Let $M$ be a set. A nonnegative function $f : M \times M \to \mathbb{R}^+$ is a metric if the following properties hold:

\[
\begin{align*}
  f(x, y) = 0 & \iff x = y \quad \text{(identity)} \\
  f(x, y) & = f(y, x) \quad \text{(symmetry)} \\
  f(x, y) & \leq f(x, z) + f(z, y) \quad \text{(triangle inequality)}
\end{align*}
\]

If only the symmetry and the triangle inequality condition are satisfied and the weaker condition $f(x, x) = 0$ holds, function $f$ is denoted a pseudo-metric.

2.1.2 Sequences

Let $\Sigma$ be a finite set, the alphabet. The elements of $\Sigma$ are characters. $\Sigma_{\text{RNA}} = \{A, C, G, U\}$ is the RNA alphabet consisting of the bases Adenin, Cytosin, Guanin and Uracil. Sequences or equivalently strings, or words are written by juxtaposition of characters. In particular, let $\lambda$ denote the empty
character, also referred to as the *gap character* which acts as the neutral element of the juxtaposition, i.e. $\lambda a = a\lambda = a$. The set $\Sigma^*$ of *strings over* $\Sigma$ is defined by

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i,$$

where $\Sigma^0 = \{\lambda\}$ and $\Sigma^{i+1} = \{aw \mid a \in \Sigma, w \in \Sigma^i\}$. The empty sequence that contains no characters or only gap characters is denoted by $\varepsilon$. I define the *tuple alphabet* as $\Sigma^n = \{(a_1, a_2, \ldots, a_n) \mid a_1, a_2, \ldots, a_n \in \Sigma\}$. For some $\sigma \in \Sigma^n$, $\sigma_i$ identifies the $i$th component of $\sigma$. The symbols $a, b, c, d$ refer to characters and $S, S_1, S_2, \ldots, S_n$ to sequences, unless stated otherwise.

The *length* of a string $S$, denoted by $|S|$, is the number of characters in $S$. I make no distinction between a character and a string of length one. If $S = uvw$ for some (possibly empty) strings $u, v$ and $w$, then

- $u$ is a *prefix* of $S$,
- $v$ is a *substring* of $S$, and
- $w$ is a *suffix* of $S$.

A prefix or suffix of $S$ is *proper* if it is different from $S$. $S[i]$ is the $i$-th character of $S$. $S[i, j]$ is the substring of $S$ beginning at $S[i]$ and ending at $S[j]$. If $i > j$, then $S[i, j]$ is the empty string.

### 2.1.3 Trees and Forests

Generally, a tree is an acyclic connected graph. I consider *rooted, ordered, node-labeled trees*, called trees for short. A distinguished node, the *root node*, imposes a partial ancestor-descendant relation on the tree nodes. Naturally, each path beginning at the root node whereas a node can be visited at most once ends in some node where it can not be further extended, a *leaf node*. A node $v$ is a *descendant* of a node $w$, if $v$ appears after $w$ on such a path. Conversely, $w$ is an *ancestor* of $v$. If $v$ and $w$ are directly connected by an
edge, \( w \) is the parent of \( v \) and \( v \) is a child of \( w \). Two nodes are siblings if they have the same parent node. The last common ancestor of \( v \) and \( w \), denoted by \( \text{lca}(v, w) \), is the node \( p \) that is an ancestor of \( v \) and \( w \) such that there is no descendant of \( p \) that satisfies the condition of being ancestor of \( v \) and \( w \). A tree is ordered if the order among sibling nodes matters, i.e. there exists an order relation for each set of sibling nodes. An ordered forest is a sequence of trees, called forest for short. A function label assigns a character from some alphabet \( \Sigma \) to each node in a forest. I use \( T(\Sigma) \) and \( F(\Sigma) \) for the set of \( \Sigma \)-labeled trees and forests, respectively. The empty tree and the empty graph which contain no nodes are denoted by \( \emptyset \). Where convenient, I identify a tree with the forest containing only this tree.

Since a tree is a special case of a forest, I give the following definitions in terms of forests: Let \( F \) be a forest. \( V(F) \) denotes the set of nodes in \( F \). The size of \( F \), denoted by \( |F| \), is the number of its nodes. The number of leaf nodes is referred to as \( \text{leaves}(F) \). The length of the longest path from a root to a leaf is the depth of \( F \), denoted by \( \text{depth}(F) \). The preorder index of a node in a tree is its position in the sequence of nodes that is obtained by the following procedure: First, visit the root node. Second, apply this procedure recursively to the trees induced by the children nodes according to their left-to-right order. For forests, the preorder index is defined by the same procedure assuming a virtual root node that is not counted in the indexing. \( \text{pre}_F(v) \) denotes the preorder index of node \( v \) in \( F \).

I now give definitions of substructures in trees and forests: A subtree at node \( v \) of \( F \) consists of node \( v \) and all its descendants. Two subtrees are siblings if their root nodes are siblings. A subforest is a sequence of sibling subtrees. A tree pattern is a subtree \( T' \) whereas arbitrary subtrees of \( T' \) can be removed.

### 2.1.4 The Sequence Edit Distance

A fundamental model for approximate string comparison is the model of edit distance \([113, 171, 213]\). It measures the distance between strings in terms
of edit operations, that is, deletions, insertions, and replacements of single characters. Two strings are compared by determining a sequence of edit operations that converts one string into the other and minimizes the sum of the costs of edit operations. Nowadays, the edit distance between strings is basic knowledge in computational biology and is an integral part of numerous textbooks, lectures and seminars. I give a brief introduction based on [108].

The notion of edit operations is the key to the edit distance model. I define the alignment alphabet $\Sigma^n_\lambda$ as the tuple alphabet where for each of its elements at least one component is different from $\lambda$. Formally, $\Sigma^n_\lambda = (\Sigma \cup \{\lambda\})^n \setminus \{\lambda\}^n$. An edit operation is a pair $(\alpha, \beta) \in \Sigma^2_\lambda$. $\alpha$ and $\beta$ are strings of length $\leq 1$. An edit operation $(\alpha, \beta)$ is usually written as $\alpha \rightarrow \beta$. This reflects the operational view which considers edit operations as rewrite rules transforming a source string into a target string, step by step. In particular, there are three kinds of edit operations:

- $\alpha \rightarrow \beta$ denotes the relabeling of the character $\alpha$ by the character $\beta$,
- $\alpha \rightarrow \lambda$ denotes the deletion of the character $\alpha$,
- $\lambda \rightarrow \beta$ denotes the insertion of the character $\beta$.

A relabeling $\alpha \rightarrow \beta$ where $\alpha = \beta$ is denoted a match. Notice that $\lambda \rightarrow \lambda$ is not an edit operation. Insertions and deletions are sometimes referred to collectively as indels.

Sometimes string comparison just means to measure how different strings are. Often it is additionally of interest to analyze the total difference between two strings into a collection of individual elementary differences. The most important mode of such analysis is an alignment of the strings. An alignment $A$ of $u$ and $v$ is a sequence

$$(\alpha_1 \rightarrow \beta_1, \ldots, \alpha_h \rightarrow \beta_h)$$

of edit operations, for short edit-sequence, such that $u = \alpha_1 \ldots \alpha_h$ and $v = \beta_1 \ldots \beta_h$. 

Note that the unique alignment of \( \varepsilon \) and \( \varepsilon \) is the empty alignment, that is, the empty sequence of edit operations. An alignment is usually written by placing the characters of the two aligned strings on different lines, with inserted dashes “-” denoting \( \lambda \). In such a representation, every column represents an edit operation.

The alignment \( A = (\lambda \rightarrow d, b \rightarrow b, c \rightarrow a, \lambda \rightarrow d, a \rightarrow a, c \rightarrow \lambda, d \rightarrow d) \) of the sequences \( u = bcacd \) and \( v = dbadad \) is written as follows:

\[
\begin{pmatrix}
- & b & c & - & a & c & d \\
- & d & b & a & d & a & - & d
\end{pmatrix}
\]

The notion of optimal alignment requires some scoring or optimization criterion. This is given by a cost function.

A cost function \( \delta \) assigns to each edit operation \( \alpha \rightarrow \beta, \alpha \neq \beta \) a positive real cost \( \delta(\alpha \rightarrow \beta) \). The cost \( \delta(\alpha \rightarrow \alpha) \) of an edit operation \( \alpha \rightarrow \alpha \) is 0. If \( \delta(\alpha \rightarrow \beta) = \delta(\beta \rightarrow \alpha) \) for all edit operations \( \alpha \rightarrow \beta \) and \( \beta \rightarrow \alpha \), then \( \delta \) is symmetric. \( \delta \) is extended to alignments in a straightforward way: The cost \( \delta(A) \) of an alignment \( A = (\alpha_1 \rightarrow \beta_1, \ldots, \alpha_h \rightarrow \beta_h) \) is the sum of the costs of the edit operations \( A \) consists of. More precisely,

\[
\delta(A) = \sum_{i=1}^{h} \delta(\alpha_i \rightarrow \beta_i).
\]

The unit cost function scores zero for matches and score one otherwise. The edit distance of \( S_1 \) and \( S_2 \), denoted by \( \delta_{SE}(S_1, S_2) \), is the minimum possible cost of an alignment of \( S_1 \) and \( S_2 \). That is,

\[
\delta_{SE}(S_1, S_2) = \min\{\delta(A) \mid A \text{ is an alignment of } S_1 \text{ and } S_2\}. \tag{2.1}
\]

An alignment \( A \) of \( S_1 \) and \( S_2 \) is optimal if \( \delta(A) = \delta_{SE}(S_1, S_2) \). Note that there can be more than one optimal alignment. If \( \delta \) satisfies the mathematical axioms of a metric, then \( \delta_{SE} \) is a metric.
2.2 Primary, Secondary and Tertiary Structure of RNA

RNA molecules can be formally described on different levels of abstraction. In messenger RNA (mRNA), coding regions of RNA molecules determine the sequence of amino acids in proteins which in turn determines the protein structure. This information, the primary structure of an RNA molecule, is carried as a sequence of nucleotides (bases) over the four letter alphabet \{A, C, G, U\}. RNA molecules have the tendency to form a three dimensional conformation, the tertiary structure. By folding back onto itself, an RNA molecule forms structure, stabilized by the forces of hydrogen bonds between certain pairs of bases, and dense stacking of neighboring base pairs. These base-pairs G–C, A–U and G–U, in order of their strength, are denoted canonical base-pairs. In fact, almost every other base-pair combination could exist, and has been observed in nature, but their contribution to the stability of the molecule are minor in comparison with the canonical base-pairs. External factors like cellular RNAs and proteins do also influence the structure. Crystallographic studies by X-ray diffraction and nuclear magnetic resonance (NMR) can reveal the tertiary structure of an RNA molecule with high accuracy [89, 100]. Although great progress has been made, crystallographic studies are still time consuming and expensive. Moreover, tertiary structure eludes from efficient algorithms for structure prediction and comparison. In particular, these problems are reported to be NP-hard for tertiary structures [94, 122]. From a biological viewpoint, RNA tertiary structure is likely formed hierarchically. First, stable stems are formed and afterward tertiary interactions are built. The strength of additional tertiary interactions is thought to be too small to significantly change the secondary structure conformation [13, 152, 156, 202]. For economical, biological and computational reasons, a subset of tertiary structures, the RNA secondary structures [36, 50], draw researchers attention.

An RNA secondary structure \((S, P)\) consists of a sequence \(S \in \Sigma_{RNA}\) and
a set of base-pairs $P = \{(i, j)\}$ such that $i, j \in [1, \ldots, |S|]$ and $i < j$. For all $(i, j), (i', j') \in P$ the following holds: W.l.o.g let $i < i'$,

1. $i = i' \iff j = j'$, i.e. there is a one-to-one relation between paired bases.

2. and it holds either:

   (a) $i < j < i' < j'$, i.e. $(i, j)$ precedes $(i', j')$, or

   (b) $i < i' < j' < j$, i.e. $(i, j)$ includes $(i', j')$.

$(S, P)$ is a tertiary structure if Condition 1 is satisfied. Figure 2.1 shows an example of the primary, secondary, and tertiary structure of an RNA molecule.

An intermediate between secondary and tertiary structures are pseudo-knotted structures which consider certain kinds of tertiary interactions. This is an emerging field but nowadays there is still a lack of algorithms and Bioinformatics tools that handle pseudo-knotted structures efficiently.

### 2.3 Representation and Visualization of RNA Structures

Understanding the macromolecular structure of an RNA molecule and its relation to function still requires expert knowledge and intuition from biologists. Visualization of RNA structures is a preliminary for this task. The topology of an RNA molecule is relevant to classify RNA structures or to search for structurally homologous RNA molecules. This typically involves the visualization of secondary and pseudo-knotted structures. A visualization of tertiary structures, based on the relative position of atoms, obtained by NMR spectroscopy or X-Ray diffraction, can give insights into macromolecular mechanisms.

The most common and biological informative drawing of RNA secondary structures is a 2d-plot, sometimes referred to as squiggle-plot. Embedded in a plane, paired bases are drawn adjacent to each other. Base-pair bonds
and the backbone of an RNA molecule are indicated as lines that do ideally not intersect. Several layout algorithms that generate 2d-plots have been proposed in [14, 109, 143, 179, 234]. The RNAviz [33, 34] software allows a manual fine tuning of drawings for producing publication-quality secondary structure drawings, e.g. the display of structural elements such as pseudo-knots or unformatted areas is possible. RNA_d2 [153], RNAdraw [132] and XRNA [233] are alternative tools within this scope. Recently, a layout algorithm for pseudo-knotted structures that produces non-overlapping drawings was proposed which is implemented in the tool Pseudoviewer [72, 74]. The visualization of the three dimensional structure of an RNA molecule belongs to the general field of three dimensional macromolecule visualization. Beside

Figure 2.1: [54] Primary, secondary and tertiary structures of yeast phenylalanine tRNA. A: The sequence was obtained from The Genomic tRNA Database [116, 117]. B: The secondary structure was inferred from an alignment of yeast tRNA-PHE sequences by RNAalifold [82], circled bases indicate neutral mutations with respect to the displayed secondary structure. Pseudo-knots and non-canonical base-pairs are indicated with a dashed line connecting squared bases [188]. C: A cartoon representation of tRNA tertiary structure, based upon tertiary structures obtained from the Protein Databank Bank (ID 6TNA,1EHZ) [99, 182].
freely distributed software like *RasMol* [175], there are many commercial
tools that offer visualization of macromolecules in the framework of drug
discovery.

Although 2d-plots are pleasant to read, it is difficult to compare them
or extract topological information. The dome-, circle- and mountain-plot
address this problem. In a *dome-plot*, base-pair bonds are drawn as arcs
above the sequence which is drawn as a straight line. In a *circle-plot*, the
sequence is arranged as a circle and chords inside the circle connect base-
pairs [151]. The *mountain-plot* draws the mountain-function of an RNA
secondary structure which intuitively assigns to each nucleotide the number
of base-pairs that enclose it [87]. Formally, we define the *mountain-function*
for an RNA secondary structure \((S, P)\) as follows:

\[
\begin{align*}
h(0) &= 0 \\
h(i) &= \begin{cases} 
  h(i-1) + 1 & \text{if } (i, j) \in P \text{ for some } i \in [i, |S|] \\
  h(i-1) - 1 & \text{if } (i, j) \in P \text{ for some } j \in [1, |S|] \\
  h(i-1) & \text{otherwise}
\end{cases} \\
\end{align*}
\]

(2.2)

A more technical representation are RNA secondary structure strings, for
their exhaustive use in the *Vienna RNA Package* referred to as *Vienna
strings* [84]. Vienna strings are sequences where, in order of the primary
structure, the characters ‘(‘ and ‘)’ denote the 5′ and 3′ bases of a base-pair,
respectively, while ‘.’ denotes an unpaired base. In addition, a second string
can hold the primary structure information. Vienna strings and *Zuker-CT*
files of the *mfold* software [244] are the most common formats to electroni-
cally store RNA secondary structures. In the era of web services, *RNAML*
is a suggestion of a XML based standardization which is designed for the
transmission of information among the RNA community [227]. An example
of RNA secondary structure drawings and representations is given in Figure
2.2.
UGGAAGAAGCUCUGGCAGCUUUUUAAGCGUUUAUAUAUGCGCGUUCCA
.(((.(((((......))))))....(((.(((((......))))))).))))..))

(a) Vienna string.

(b) 2d-plot generated by RNAplot from the Vienna RNA Package [84].

(c) dome-plot.

(d) circle-plot generated by the mfold server [244].

(e) mountain-plot. Hairpin loops appear as flat tops, interior loops and bulges as intermediate plateau, helices as sloping hillsides, and branching regions as valleys.

Figure 2.2: Visualization of a secondary structure for the Nanos 3' UTR translation control element taken from the Rfam database [67] (Id: RF00161, EMBL Id: U24695.1).
2.3 Representation and Visualization of RNA Structures

These visualization show a single structure of an RNA sequence. *Dot-plots* visualize the structure space of an RNA sequence with the potential to reveal suboptimal structures that are biologically relevant. Arranged in a matrix, the probabilities of base-pairs are plotted as dots whose diameter is proportional to their probability in the structure space. The base-pair frequency information has subsequently been included in single structure visualizations and likely base-pairs can be distinguished from unlikely base-pairs by a color gradient or some other indicator [245]. See Figure 2.3 for an example of a dot-plot and an annotated 2d-plot. *RNAmovies* is an interactive software for the visualization of secondary structure spaces [57]. It automatically generates animated 2d-plots where structures are morphed to explore the structure space of an RNA molecule.

From the viewpoint of computer scientists, RNA secondary structures are often represented as trees or forests. The parent and sibling relationship of nodes is determined by the nesting of base-pair bonds. The 5′ to 3′ nature of an RNA molecule imposes the order among sibling nodes. This produces a forest structure in general but a virtual root node can always turn a forest into a tree. Different tree representations that vary in their resolution have been proposed. A tree structure where base-pairs correspond to internal nodes while unpaired bases correspond to leaves in the tree was proposed in [173]. I refer to it as the *natural tree representation*. A *coarse grained tree representation* where nodes correspond to the structural components - stacking regions, hairpins, bulges, internal loops and multiloops - was proposed in [110, 178, 180]. Parse trees of grammar based prediction strategies for RNA secondary structures represent the structure such that the sequence information corresponds to the preorder sequence of leaves while the internal nodes correspond to productions of the grammar [167]. An example of tree representations of RNA structures is shown in Figure 2.4.
Figure 2.3: (a) shows the base-pair probabilities as predicted by RNAfold [84]. The lower triangle show only the bases included in the minimum free energy structure and the upper triangle contains the full base-pair probabilities where the diameter of a square is proportional to the probability of the corresponding base-pair. (b) shows the 2d plot of the structure annotated with the probabilities of a base-pair. The colors range from blue to red in correspondence to less and high frequent base-pairs.
Figure 2.4: (a) shows a secondary structure with colored components that indicate the relation between the representations. (b) shows the natural tree representation where internal nodes correspond to base-pairs and leaves correspond to unpaired bases. (c) shows the coarse grained tree representation. The red and cyan part are stacking region (S), the green part is a multiloop (M), the yellow part is an internal loop (I), and the blue and magenta parts are hairpin loops (H). A bulge left (L) and a bulge right (R) are internal loops that have only a left and right unpaired region, respectively. Note that single stranded regions at the root level of the tree and in multi-loops are omitted in this tree representation. (d) shows a simplified parse tree for some grammar describing RNA secondary structures. The internal nodes correspond to productions of the grammar and impose a structure on the sequence that resides at the leaves. A virtual root node $v$ is added in (b) and (d) to guarantee a tree structure.
2.4 RNA Secondary Structure Prediction

The structure of an RNA molecule can be crucial for its function (see Section 1.1). Accordingly, the automatic prediction of RNA structures from sequence information is an important problem. Today, there are two prediction strategies:

- **Thermodynamic approaches**: The conformation of paired and unpaired regions in an RNA structure can be associated with an energy value. Given some energy model, thermodynamic approaches find the energetically most stable structures among all possible secondary structures of an RNA sequence. Such a structure is denoted the *minimum free energy* (mfe) structure.

- **Comparative approaches**: In functional non-coding RNA, the structure of an RNA is conserved during evolution. Since a base-pair can be formed by different combinations of nucleotides, different sequences can have the same or a similar structure. If a family of structural homolog RNA molecules has a sufficient amount of sequence conservation, a multiple sequence alignment can emphasize regions of sequence variation. The regions containing structure-neutral mutations, denoted as *compensatory base changes*, give clues to the structure of an RNA molecule.

In 1978, Nussinov et al. introduced a first folding algorithm requiring a single sequence as input [151]. They determine the structure that maximizes the number of possible base-pairs for an RNA sequence. This problem is also known as the *maximum circular matching problem*. The incorporation of thermodynamics in this model assumes that the energy contribution of each base-pair is independent from adjacent base-pairs in the structure. This assumption is not realistic since the stability of RNA molecules is based on the stacking of base-pairs. Zuker & Stiegler proposed a dynamic programming algorithm that calculates the minimum free energy structure based on a
model that considers base-pair stacking and destabilizing loops [247]. Their algorithm uses thermodynamic parameters of Tinoco et al. [201]. The energy model and parameters were refined in [129, 209]. McCaskill introduced a statistically motivated model based on Boltzmann’s distribution and thermodynamic parameters, the *partition function* [133]. The most likely structure under this model is the mfe structure. The main contribution of McCaskill is the computation of probabilities for the individual base-pairs. Sakakibara et al. and Eddy & Durbin invented a generalization of hidden markov models, the *stochastic context-free grammars*, and formulated the RNA secondary structure prediction problem in this context [42, 167, 168].

Thermodynamic folding relies on parameters that were measured *in vitro* under fixed conditions which is a simplification of real conditions. The folding *in vivo* takes place in a dynamic, hence, more complex environment. From the inaccuracy of energy parameters (and even the model itself), it is possible that the mfe structure is not the biological correct one. The biological relevant prediction is often a *suboptimal solution* that has an energy close to the mfe structure. Thus, the generation of suboptimal structures is important for the practical impact of prediction algorithms based on thermodynamics [243, 246]. The assumption of equilibrium folding pathways is another common simplification of thermodynamic folding models. Studies of the folding of the *Tetrahymena group I intron* gave insights in the complexity of the folding process [8, 208]. It has been observed that RNA can fold during transcription, the folding process happens on a wide range of time scales, and ions and macromolecules guide the folding. Thus, the kinetics of RNA folding are important to understand the true folding pathway. Models and algorithms for kinetic folding prediction are provided in [49, 70, 138, 232]. A further challenge for mfe folding algorithms are RNA secondary structures that are known to have two conformations depending on some environmental parameters, known as *RNA switches* [126]. Recently, Giegerich et al. provided a structure prediction algorithm based on thermodynamics that compartmentalizes the suboptimal solution space into different *shapes* [59].
The different shapes of an RNA molecule give a compact overview of the structure space and are useful find the biological relevant prediction or to detect different conformational states.

The most popular tools for energy-based RNA secondary structure prediction from single sequences are \textit{mfold} \cite{243,244} and \textit{RNAfold} \cite{79,84}. The former implements the mfe algorithm and the latter implement additionally McCaskill’s partition function algorithm. Recently, the energy-based prediction of pseudoknotted structures received more attention \cite{35,158,161}.

Comparative approaches require a set of homologous RNA sequences that have a putative similar structure. The general idea is to exploit the covariance that is expected to occur in aligned stem regions. Until the early 80’s the structural inference from homologous RNA sequences had been hand-crafted. Noller & Woese described a procedure to detect compensatory changes in helical elements \cite{147}. An algorithm building upon this strategy was provided by Waterman \cite{224,225}. Given a multiple sequence alignment, the \textit{mutual information content} and \textit{sequence covariation} are measures that help to automatically identify conserved stem regions \cite{26,229}. These pure phylogenetic approaches assume that the sequences, in fact, share a common structure, which requires a careful choice of sequences. A combination of phylogenetic information and thermodynamics can further improve the results. A multiple sequence alignment is used to validate predicted structures in \cite{81,112,121}. Conversely, Han & Kim resolve ambiguities in the alignment by thermodynamics \cite{73}. As an extension of the minimum free energy approach, \textit{RNAalifold} \cite{83} calculates the best folding using an objective function that combines energy contributions and covariance. Ruan et al.’s \textit{ILM} (iterated loop matching) optimizes a similar objective function \cite{166}. As the name suggests, the structure is iteratively constructed by adding non-conflicting stem regions. \textit{ILM} is capable of returning pseudoknotted RNA structures. Knudsen & Hein predict a common RNA secondary structure by stochastic context-free grammars, implemented in the tool \textit{Pfold} \cite{104}. 
Sankoff opened a branch of comparative strategies considering the alignment and folding problem simultaneously [172]. The time complexity of Sankoff’s algorithm is $O(n^6)$ where $n$ is the length of RNA sequences. This is too high to be practical even for two sequences. Mathews & Turner’s DYNALIGN restricts the maximum distance of possible base-pairs to bound the parameters that affect time complexity in Sankoff’s algorithm [130, 131]. Gorotkin et al.’s FOLDALIGN implements a modification of Sankoff’s algorithm than does not allow branching structures, which reduces the time complexity. Tabaska & Stormo used a graph theoretic approach, the maximum weight matching to infer RNA secondary structures from different sources [189, 190]. They consider a set of base pairing scores that can be derived from a range of sources, such as free energy considerations, mutual information, and experimental data. Hofacker et al. provide a strategy that is based on aligning base-pair probability matrices, predicted by McCaskill’s partition function algorithm [80]. Their algorithm is implemented in the tool pmmulti in the Vienna RNA package.

According to Gardner & Giegerich, approaches that use phylogenetic information yield significant better predictions than pure thermodynamic approaches [55]. However, the quality of the multiple sequence alignment that should reveal the phylogeny depends on the degree of sequence homology of RNA molecules. The minimum homology that is necessary depends on the particular prediction strategy, i.e. the sources of information that are used to predict structures. Moreover, phylogenetic approaches require a large number of sequences which is a rare situation.

For families of RNA molecules with low sequence conservation, a strategy that was proposed by Shapiro and Konings & Hogeweg more than a decade ago is currently revitalized [105, 180]: First, structures are predicted based on thermodynamics and then a structural alignment, instead of a sequence alignment, is done. Recent progress in structural comparison models and algorithms make this strategy a promising candidate for low sequence homologous, but (putative) structural homologous RNA molecules. In par-
ticular, this strategy requires a model for structurally aligning multiple RNA secondary structures. I will provide a structure prediction strategy based on multiple structure alignment in Chapter 5.

### 2.5 RNA Structure Comparison

The field of RNA structure comparison emerged with the invention of RNA secondary structure prediction algorithms. Since then, the resulting pool of predicted structures, be they right or wrong, were available for analyzing structural properties. The prediction of structural motifs, the inference of a taxonomy based on structural similarity instead of sequence similarity, and the prediction of consensus structures for a set of functionally related RNA molecules are active research topics that involve the comparison of RNA structures. I distinguish the following approaches to compare RNA structures:

- **Base-pair distances**: Base-pair distances are classical mathematical metrics that operate on the base-pair sets of RNA structures.

- **Sequence alignment**: RNA secondary structures are represented as strings that in turn are compared in the sequence alignment model.

- **Edit distances between ordered rooted trees**: Since an RNA secondary structure can be represented as a tree, distances on trees can be applied to compare RNA secondary structures.

- **Arc annotated sequences**: Pure sequence alignment based approaches are extended to incorporate structural constraints that are induced by the structure of RNA. Constrained sequence edit models are generally studied in the context of arc annotated sequences.

- **Graphs**: Graphs can express any sort of RNA structures. Algorithms for the classification of graphs are applied to RNA structure analysis.
Distance and Similarity  The result of a comparison of RNA structures can be quantified in two different ways: The first is distance and the second is similarity. Distance measures satisfy the mathematical axioms of a metric (or at least pseudo-metric). A similarity measure assigns a numeric value to some pairs of structures such that the larger the value the more similar the structures are. Distances are non-negative and the distance between two structures is zero iff the structures are equal. In contrast, the similarity of equal structures is an arbitrary positive number. Accordingly, a small distance is equivalent to a large similarity.

In the following sections, I consider distance versions of models for RNA structure comparison. The corresponding similarity versions can be derived easily for distances that are based on optimization problems. For distance problems, optimal means minimal, while for similarity problems optimal means maximal. Throughout this section, \((S_1, P_1), (S_2, P_2), \ldots, (S_n, P_n)\) denote secondary structures.

2.5.1 Base-pair Distances

Base-pair distances are distance measures that are defined on the base-pair sets of RNA structures. An analysis of some properties of base-pair distances and their comparison with the tree edit distance is provided in [142].

Symmetric Set Difference

One of the simplest measures is defined by the symmetric set difference, that is:

\[
\delta_{SD}(P_1, P_2) = P_1 \setminus P_2 \cup P_2 \setminus P_1
\]  

(2.3)

Clearly, this simple measure is sensitive to the exact position of base-pairs and is therefore not suitable to compare structures of different length. Also if the structures have the same length, the measure is sensitive for shifted structures. Consider the following structures:
Intuitively, these structures should obtain a distance close to zero, but
\( \delta_{SD}(P_1, P_2) = 6 \) since there is no common base-pair. This discrepancy gets the larger the larger the shifted structures are. Still, for suboptimal structures of the same sequence, \( \delta_{SD} \) can be a useful *ad hoc* distance.

**Hausdorff Distance**

A more flexible metric is the *Hausdorff distance* which was applied by Zuker to filter out similar suboptimal foldings in the original *mfold* program [243]. The Hausdorff distance measures the distance between non empty point sets of some metric space. For the problem of RNA structure comparison, these are the sets of base-pairs. Intuitively, the Hausdorff distance between structures \( P_1 \) and \( P_2 \) is the maximum of the distances between all nearest base-pairs connecting \( P_1 \) and \( P_2 \). Formally, the distance between two base-pairs \((i, j) \in P_1 \) and \((i', j') \in P_2 \) is defined as \( \delta((i, j), (i', j')) = \max\{|i-i'|, |j-j'|\} \).

The distance of a base-pair to a set of base-pairs is defined as \( \delta((i, j), P) = \inf_{(i', j') \in P} \delta((i, j), (i', j')) \). Then the *Hausdorff distance between \( P_1 \) and \( P_2 \) is defined as

\[
\delta_H(P_1, P_2) = \max(\delta_{\text{asym}}(P_1, P_2), \delta_{\text{asym}}(P_2, P_1)) \quad \text{where} \quad (2.4)
\]

\[
\delta_{\text{asym}}(P_1, P_2) = \sup_{(i, j) \in P_1} \delta((i, j), P_2)
\]

Although this distance behaves reasonable for structure shifts, the distance between structures that differ only in one base-pair depends on the position of this base-pair. Consider the following structures:

\[
P_1 = \ldots \ldots \ldots (((\ldots \ldots)))
\]
\[
P_2 = (\ldots) \ldots \ldots (((\ldots \ldots)))
\]
\[
P_3 = \ldots (\ldots) \ldots (((\ldots \ldots)))
\]
2.5 RNA Structure Comparison

\( P_2 \) and \( P_3 \) are both one base-pair apart from \( P_1 \), but their Hausdorff distance is different, i.e. \( \delta_H(P_1, P_2) = 11 \) and \( \delta_H(P_1, P_3) = 7 \). Thus, isolated base-pairs can lead to high distance values depending on the distance to the next base-pair. Aware of this problem, Zuker et al defined a variant of \( \delta_H \) that ignores up to \( d \) bases to obtain a distance \( d \) [246]. This variant is a pseudo-metric, since the triangle inequality is not satisfied as exemplified in [142].

**Mountain Metric**

Another application of a classical mathematical metric to RNA structures is the *mountain metric* which is based on the \( l_p \)-norm of the difference of two mountain functions \( h_{P_1} \) and \( h_{P_2} \) (see Equation (2.2)) of RNA secondary structures of length \( n \) [142]:

\[
\delta_M^p(P_1, P_2) = \|h_{P_1} - h_{P_2}\|_p := \sqrt[p]{\sum_{i=1}^n |h_{P_1}(i) - h_{P_2}(i)|^p}
\]

For \( p = 2 \) this is the root mean square (RMS) distance between two functions which is, followed by \( p = 1 \), the most frequent choice. This metric is more flexible for shifted structures and isolated base-pairs and it can be computed in linear time. A property of this distance that one must be aware of is that the extension of stem regions does not have uniform costs. See the following example:

\[
P_1 = \ldots((.\ldots))\ldots \\
P_2 = (.((.\ldots)))). \\
P_3 = \ldots((.\ldots))\ldots
\]

\( P_1 \) differs from \( P_2 \) and \( P_3 \) in just one base-pair but their mountain distances (for simplicity I use \( p = 1 \)) do not reflect that. In particular, \( \delta_M^1(P_1, P_2) = 13 \) and \( \delta_M^1(P_1, P_3) = 5 \). See Figure 2.5 for an illustration. A variant of the mountain distance that re-scales mountain functions for structures of different length is proposed and applied in [44].
Figure 2.5: The difference between $P_2$ and $P_1$ is larger than the difference between $P_3$ and $P_1$, though both differ in exactly one base-pair.

2.5.2 Sequence Alignment

Shapiro and Konings & Hogeweg simultaneously proposed the idea to compare RNA secondary structures by well established sequence alignment algorithms [105, 180]. While Konings & Hogeweg focused on pairwise alignments, Shapiro considered multiple sequence alignments. In both approaches, the key idea is to use a string representation of RNA secondary structures, in flavor of the Vienna strings\textsuperscript{1}, which are the data structures that are further analyzed.

Konings & Hogeweg’s Encoding

Following Konings & Hogeweg, “A full linear representation is obtained by transforming the mountain structure into a linear array of symbols representing the direction of base-pairing at each of the single positions: upstream pairing ($>$), downstream pairing ($<$) or single strandedness ($+$) ... Extra information in terms of secondary structure can be included in the linear representation by distinct coding of hairpin loops ($^\circ$) and other types of single stranded positions ($+$)” . In this representation, the secondary structure in Figure 2.2 is written as:

\[
+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}+\underbrace{+\cdots+}_{30}
\]

\textsuperscript{1}The Vienna format was established later, building upon the results of Shapiro and Hogeweg & Konings.
A potential disadvantage of this representation for a topological classification is that basic secondary structure elements may be broken up in an alignment, i.e. matching of individual parts of one helix to parts of two different helices, not considering interruptions by internal loops and bulges.

**Shapiro’s Encoding**

Shapiro introduced a different string representation that circumvents this problem. The coarse grained tree representation of an RNA structure is transformed to a string by a left-to-right preorder traversal of the tree, putting subtrees into brackets. The components are encoded as single letters. In this representation the structure in Figure 2.2 is:

\[(S(M((H)(S(I(H))))))\]

To simplify the notation brackets are removed for non-branching subtrees:

\[(S(M(H (S I H))))\]

For a topological classification, this coarse grained representation is suitable. However, if the aim is an improved sequence alignment that incorporates structural constraints, it should be possible to match individual parts of one helix with two different parts of another helix. For instance, there could exist a larger helix that was broken during evolution resulting in two smaller helices that are separated by a bulge.

Beside these effects, both methods suffer from the same inherent problem: A pair of brackets is not treated as a unit by a sequence alignment and thus the tree nature of a secondary structure is not treated appropriately. Consider the following structures:

\[P_1 = (((((.))))))\]
\[P_2 = ((((.))).))\]

The following alignment is among the optimal alignments given a scoring scheme that favors matches in contrast to mismatches, insertions and deletions.
The opening brackets ‘(’ are not aligned to its corresponding closing brackets ‘)’ and in terms of structure this alignment is not meaningful. Shapiro was aware of such problems but appropriate, efficient algorithms for comparing RNA secondary structures as trees were just about to emerge.

2.5.3 Edit Distances between Rooted Ordered Trees

From the tree nature of RNA secondary structures, every distance measure on trees can be applied to RNA secondary structures. Inspired by the sequence edit distance [113, 171, 213], different edit models for trees have been invented [95, 118, 177, 191, 198] which result in various algorithms. Beside the fact that tree editing is a challenging theoretical problem dealing with a fundamental data structure, this field was (and is still) driven by the need for such algorithms in a broad spectrum of applications. This includes the comparison of RNA secondary structures [25, 110, 111, 178], the analysis of structured documents and text databases [18, 96, 127, 144, 159], script recognition [22, 118], fingerprint recognition [139], image analysis [165, 169], the analysis of parse trees [97, 235], the comparison of assembly rules [48], and the identification of common structural fragments among chemical structures [192]. The semantic of tree edit distances in the scope of RNA structure comparison depends on the choice of the tree representation and the edit model.

A review of tree edit models that are particularly interesting for document trees (but also for RNA secondary structures) was given in [7]. The authors provide implementations of tree edit algorithms in the programming language Turing [90]. A more recent survey on tree editing problems, including unrooted, unordered variants, and different notions of tree editing, was provided in [10, 11, 241]. The relation between tree-edit distances was studied in [216] resulting in a hierarchy of edit-models.

In the world of sequences, the terms edit distance and alignment distance are used synonymously. For each optimal sequence of edit operations,
an alignment achieving the same score can be constructed and vice versa. However, on a conceptional level the models are different. While the edit distance is an operational model of editing one sequence into another, an alignment is a declarative model, a data structure rather than a process. In the world of trees, these models turned out to be dual: The tree edit model constructs a largest common subforest, while the tree alignment distance constructs a smallest common supertree. Moreover, the higher complexity of trees (in comparison to sequences) leads to a multitude of problems that vary in the constraints that are imposed by the chosen model. The models that are interesting for the comparison of RNA structures are introduced in the following paragraphs, beginning with the most general model which is successively restricted. Throughout this chapter, $T, T_1, T_2$ are trees unless stated differently.

Tree Edit Distance

In the tree-to-tree correction problem [191], Tai introduced the generalization of the string-to-string correction problem [213] which is also known as the edit distance problem for strings. I refer to Tai’s model as the tree edit model\(^2\), following the mainstream of literature.

**Edit Operations** The edit operations *relabel*, *delete* and *insert* generalize from strings to trees (and forests) as follows:

- **relabel**: The label of a node $v$ in $T$ is changed. If a label is relabeled by itself, this is denoted a *match*.

- **delete**: Deleting node $v$ in $T$ means that the children of node $v$ become the children of the parent node of $v$. Moreover, if $v$ has any siblings, the deletion preserves the preorder relation of these node. Note, if $v$ is the root node, the result is the forest consisting of the children nodes of $v$.

\(^2\)The same model was also, independently, proposed by Lu [118]. However, Lu considered an algorithm for a special case of the general tree edit distance.
**Insert:** This operation is complementary to *delete*. Inserting a new node \( v \) into \( T \) results in a new tree \( T' \) such that the deletion of \( v \) in \( T' \) results in \( T \). Intuitively, a node \( v \) is inserted as a child of \( v' \) making \( v \) the parent of a consecutive subsequences of children of \( v' \).

According to the sequence edit model, I represent edit operations by \( \alpha \rightarrow \beta \) where \( (\alpha, \beta) \in \Sigma^2_\lambda \). \( \alpha \rightarrow \lambda \) and \( \lambda \rightarrow \beta \) denote the functions *delete* and *insert* of \( a \) and \( b \), respectively. Otherwise, \( a \rightarrow b \) is the *relabel* function, relabeling \( a \) with \( b \). An illustration of the tree edit operations is given in Figure 2.6. Note, the node that is affected by an edit operations is defined by the edit operation together with the tree to be edited and the resulting tree.

Let \( E \) be a sequence \( e_1, e_2, \ldots, e_n \) of edit operations, for short *edit-sequence*. Following Tai, \( E \) transforms \( T \) into \( T' \) if there is a sequence of trees \( T_0, T_1, \ldots, T_n \) such that \( T = T_0, T' = T_n \) and \( T_i \) results from the application of \( e_i \) to \( T_{i-1} \) for \( i \in [1, n] \). Let \( \delta \) be a metric defined on edit operations. The cost of an edit-sequence \( E \) is the sum of the costs of its edit operations, that is:
\[
\delta(E) = \sum_{i=1}^{n} \delta(e_i)
\]
which is also a metric [240]. The *edit distance* \( \delta_{TE} \) between trees \( T_1 \) and \( T_2 \) is the minimum cost that is necessary to transform \( T_1 \) into \( T_2 \):
\[
\delta_{TE}(T_1, T_2) = \min \{ \delta(E) \mid E \text{ is an edit sequence transforming } T_1 \text{ into } T_2 \}.
\]

(2.6)

Edit sequences are an intuitive, operational concept that accounts for the differences between trees. However, the infinite number of edit sequences that can transform one tree into another make theoretical observations intricate. Again inspired by the sequence edit model, Tai extended the concept of traces, known from the sequence edit model [213], to trees, commonly referred to as mappings.

**Mappings** A mapping establishes a one-to-one correspondence of nodes in \(T_1\) and \(T_2\) which preserves the sibling and ancestor relation of nodes. Formally, a mapping between trees \(T_1\) and \(T_2\) is defined by a triple \((M, T_1, T_2)\) where \(M \subseteq V(T_1) \times V(T_2)\) such that for all \((v_1, w_1), (v_2, w_2) \in M\) the following holds:

\[
\begin{align*}
    v_1 = v_2 & \iff w_1 = w_2 & \text{(one-to-one correspondence)} \\
    v_1 \text{ is ancestor of } v_2 & \iff w_1 \text{ is ancestor of } w_2 & \text{(ancestor preservation)} \\
    \text{pre}_{T_1}(v_1) < \text{pre}_{T_1}(v_2) & \iff \text{pre}_{T_2}(w_1) < \text{pre}_{T_2}(w_2) & \text{(sibling preservation)}
\end{align*}
\]

Let \(V(T_1) \setminus M\) and \(V(T_2) \setminus M\) be the nodes in \(T_1\) and \(T_2\) that are not mapped by \(M\), respectively. The cost of a mapping is given by:

\[
\delta(M) = \sum_{(v, w) \in M} v \to w + \sum_{v \in V(T_1) \setminus M} v \to \lambda + \sum_{w \in V(T_2) \setminus M} \lambda \to w \tag{2.7}
\]

The following lemma shows that mapping are equivalent to edit-sequences.

**Lemma 2.1.** Given an edit-sequence \(E\) transforming \(T_1\) into \(T_2\), there exists a mapping from \(T_1\) to \(T_2\) such that \(\delta_{TE}(M) \leq \delta_{TE}(E)\). Conversely, for any mapping \(M\), there exists an edit-sequence such that \(\delta_{TE}(E) = \delta_{TE}(M)\).

**Proof.** See Proof of Lemma 2 in [240].
Hence, the edit distance between trees can be defined likewise by

\[ \delta_{TE}(T_1, T_2) = \min \{ \delta(M) \mid M \text{ is a mapping from } T_1 \text{ to } T_2 \} \]. \quad (2.8)

**Isomorphic Subforests** A third definition of the edit distance between trees is more related to graph theory. Forests \( F_1 \) and \( F_2 \) are isomorphic, denoted by \( F_1 \cong F_2 \) if they can be transformed into each other simply by applying the relabel-function. For isomorphic forests, there exists a corresponding mapping \( M_i \) including all nodes in \( F_1 \) and \( F_2 \). Such a mapping \( M_i \) is denoted an isomorphism. For some \( D \subseteq V(T) \), \( T \setminus D \) denotes the forest that results from applying the delete-function to all nodes in \( D \) to \( T \). This definition, allowing isomorphic subforests instead of isomorphic subtrees, is important since a valid mapping between trees can correspond to an isomorphic subforest. The edit distance between \( T_1 \) and \( T_2 \) can then be defined as

\[
\delta_{TE}(T_1, T_2) = \min \{ \delta_{TE}(M_i) + \sum_{v \in D_1} v \rightarrow \lambda + \sum_{w \in D_2} \lambda \rightarrow w \mid D_1 \in V(T_1), D_2 \in V(T_2) \text{ such that } T_1 \setminus D_1 \cong T_2 \setminus D_2 \}. \quad (2.9)
\]

It is obvious that this definition is equivalent to the definition of a mapping (2.8) and, consequently, to the edit sequence based definition. Figure 2.7 shows an example of a mapping and the correspondence to isomorphic subforests.

**Algorithms** Algorithms that calculate the tree edit distance generally build upon the mapping concept since the number of mappings for given trees is finite. The first proposed algorithm is due to Tai and requires \( O(|T_1| \cdot |T_2| \cdot \text{leaves}(T_1)^2 \cdot \text{leaves}(T_2)^2) \) time and space. It follows the strategy of extending mappings from the root of a tree to its leaves. A faster and much simpler algorithm is due to Zhang & Sasha (Zhang-Shasha Algorithm) and improves the time complexity to \( O(|T_1| \cdot |T_2| \cdot \min\{\text{leaves}(T_1), \text{depth}(T_1)\}) \).
2.5 RNA Structure Comparison

![Diagram showing RNA structure comparison](image)

Figure 2.7: The dashed lines indicate the mapping \( M = \{(a,a), (b,b), (c,c), (d,d)\} \) of \( T_1 \) and \( T_2 \). \( T_3 \) shows the maximum isomorphic subforest (here a tree) that is obtained by deleting node \( x \) in \( T_1 \) and node \( y \) in \( T_2 \). The edit sequence \( x \rightarrow \lambda, \lambda \rightarrow y \) together with the sequence of trees \( T_1, T_3, T_2 \) determines the corresponding edit process.

\[
\min\{\text{leaves}(T_2), \text{depth}(T_2)\} \}
\]

In the worst case, which is a tree that grows linear in the number of leaves and its depth, the time complexity is in \( O(|T_1|^2 \cdot |T_2|^2) \). Special algorithms for the tree edit distance under a unit cost scheme are studied in [181]. The parallelization of tree edit algorithms is considered in [237, 239]. The average runtime of the Zhang-Shasha Algorithm for RNA secondary structure trees turned out to be \( O(|T_1|^3 \cdot |T_2|^3) \) which essentially means that it is cubic [39]. Klein improved the worst case runtime of the tree edit algorithm to \( O(|T_1|^2 \cdot |T_2| \cdot \log |T_2|) \) by applying a divide and conquer strategy (Klein’s Algorithm) [102]. An analysis of the Zhang-Shasha Algorithm and Klein’s Algorithm in a general framework of cover strategies is given by Dulucq & Touzet [40]. Moreover, they present an improvement of Klein’s strategy which can result in a better practical runtime. A different strategy is followed by Chen, the tree edit problem is reduced to a matrix multiplication problem and is solved by using results in this field [21]. This algorithm runs in \( O(|T_1| \cdot |T_2| + \min\{\text{leaves}(T_1)^2 \cdot |T_2| + \text{leaves}(T_1)^{2.5} \cdot |T_2|, \text{leaves}(T_2)^2 \cdot |T_1| \} + \)
leaves($T_2^{2.5} \cdot |T_1|$) and improves the time complexity for certain kind of trees in comparison to Klein’s algorithm, e.g. if one of $T_1$ and $T_2$ is thin and deep.

**Variants** Touzet gave a definition of gaps in a tree [207]. The idea is to consider contiguous gaps as a single large gap where the term contiguous is equivalent to our definition of a tree pattern. They study convex scoring functions for gaps, that is: \( \text{gapscore}(T_1 \circ T_2) \leq \text{gapscore}(T_1) + \text{gapscore}(T_2) \) where $T_1$ and $T_2$ are tree patterns and $T_1 \circ T_2$ means that $T_2$ is attached to a leaf node of $T_1$. They proved that the calculation of the tree edit distance with gaps for convex gap scores is a NP-hard problem.

**Tree Alignment Distance**

The tree alignment distance was introduced by Jiang et al. [95]. My central notion is the following generic view of an alignment: An alignment of two structures with labels from some alphabet $\Sigma$ is the same type of structure with labels from the alignment alphabet $\Sigma^2_\lambda$. Labels of the form $(\alpha, \beta), (\alpha, \lambda), (\lambda, \beta)$ where $\alpha, \beta \in \Sigma$ denote the edit operations relabel, delete, and insert, respectively. Applying this general concept to trees, a tree alignment $A$ is an element of $T(\Sigma^2_\lambda)$. Its component-wise projections $A|_1$ and $A|_2$ are elements of $T(\Sigma \cup \{ \lambda \})$. For some $T \in T(\Sigma \cup \{ \lambda \})$, $\pi(T) \in F(\Sigma)$ is the forest that results from the deletion of all nodes $v$ with $\text{label}(v) = \lambda$. Formally:

$$\pi(T) = T \setminus D \text{ where } D = \{ v \mid \text{label}(v) = \lambda \}$$

(2.10)

The following equation formally defines the notion of alignment of trees.

\[ A \in T(\Sigma^2_\lambda) \text{ is an alignment of trees } T_1, T_2 \in T(\Sigma) \text{ iff } \]

\[ T_1 = \pi(A|_1) \text{ and } T_2 = \pi(A|_2). \]

(2.11)

\[ ^3 \text{See the definition of } T \setminus D \text{ on Page 34.} \]
Note that this definition forbids elements of $T(\Sigma_3^\lambda)$ where the deletion of a root node results in a forest (A forest alignment model will be introduced in Section 3.2). Figure 2.8 shows an example of a pairwise tree alignment. The cost $\delta$ of an alignment $A$ is the sum of the costs of its node labels, that is:

$$\delta(A) = \sum_{v \in V(A)} \delta(label(v)).$$  \hfill (2.12)

The alignment distance between $T_1$ and $T_2$ is the minimum cost that an alignment of $T_1$ and $T_2$ can achieve. An alignment of $T_1$ and $T_2$ is optimal if it achieves this score. Formally, the alignment distance $\delta_{TA}$ between trees $T_1$ and $T_2$ is defined as:

$$\delta_{TA}(T_1, T_2) = \min\{\delta(A) \mid A \text{ is an alignment of } T_1 \text{ and } T_2\} \hfill (2.13)$$

For each alignment it is possible to construct a corresponding edit sequence and a mapping. The converse does not hold in general: Consider the mapping in Figure 2.7. In this mapping, nodes labeled with “c” are mapped to each
other. Thus, in a possible alignment there must exist a node labeled with 
“c, c”. Then, this node must be the son of the nodes labeled with “x, λ” and
“λ, y”. This is in contrast to the definition of a tree since a node can have at
most one parent node in a tree. From this observation, it is clear that tree
alignments form a subset of tree edit distance mappings. For trees $T_1$ and $T_2$
holds $\delta_{TE}(T_1, T_2) \leq \delta_{TA}(T_1, T_2)$.

Since the edit sequence definition is equivalent to the mapping definition,
it follows that not each edit sequence has a corresponding alignment. Jiang
et al. claimed that an “alignment of trees actually corresponds to a restricted
tree edit in which all the insertions precede all the deletions” [95]. This is
intuitive, but a formal proof is missing.

I now demonstrate that $\delta_{TA}$ does not satisfy the triangle inequality of
the metric axioms: An arbitrary edit sequence can be divided into two edit
sequences where the one includes all insert- and the other all delete- and
relabel-operations. Assuming Jiang et al.’s claimed property of alignment
compatible edit sequences (see above), the divided edit sequences are com-
patible with an alignment. From this and the fact that the tree edit distance
can be less than the tree alignment distance follows that it does not satisfy
the triangle inequality. Hence, the tree alignment distance is not a metric.
See Figure 2.9 for an example.

I am not aware of a constrained mapping definition that corresponds to
alignments, in literature.

**Isomorphic Supertree** A graph theoretical definition of the tree align-
ment distance is based on tree isomorphisms. In this context, the minimum
possible distance between isomorphic trees that result from the insertion of
“λ” labeled nodes in the original trees is sought. The forests that are con-
sidered by this procedure are *isomorphic supertrees*. Nodes that are labeled
with “λ, λ” should naturally score 0. Clearly, an overlay of such isomorphic
superforests and the deletion of possible “λ, λ” labeled nodes produces an
alignment and, hence, the models define the same distance.
Algorithms Together with the definition of the tree alignment distance, Jiang et al. proposed an algorithm that computes this distance in $O(|T_1| \cdot |T_2| \cdot (\text{degree}(T_1) + \text{degree}(T_2))^2)$ time which is still the asymptotical best algorithm [95]. For a fixed number $d$ of possible deletions and insertions, Jansson & Lingas presented an algorithm that calculates the tree alignment distance in $O(n^2 \cdot \log n \cdot k^3 \cdot d^2)$ where $n = \max\{|T_1|, |T_2|\}$ and $k = \max\{\text{degree}(T_1), \text{degree}(T_2)\}$ [92].

Variants Wang & Zhao make three interesting contributions considering the tree alignment distance for RNA structure comparison [221]:

1. They provide a model for the tree alignment distance including gaps where the notion of gaps in a tree corresponds to tree patterns as done in [207]. However, Wang & Zhao consider a simpler gap score function where the score of a gap is a constant function. They derive

---

4Precisely, the similarity version.
an algorithm from Jiang et al.’s algorithm that computes the alignment distance, involving gap scores, in the same time complexity.

2. They present a modified version of Jiang’s algorithm that improves the space complexity to $O(\text{degree}(T_1) \cdot \log |T_1| \cdot |T_2| \cdot (\text{degree}(T_1) + \text{degree}(T_2)))$ while having the same time complexity as the Jiang algorithm. However, an optimal alignment cannot be obtained by a straightforward backtracking procedure. As space is crucial in their application they use a naive algorithm that raises the time complexity to $O(|T_1|^2 \cdot |T_2| \cdot (\text{degree}(T_1) \cdot \text{degree}(T_2))^2)$ while achieving their improved space complexity.

3. They consider the problem of parametric tree alignment which was studied earlier for sequences [71] and gives clues to the parameter space of tree alignments. In particular, the scoring of edit operations is often not deducible from the problem and therefore somewhat arbitrary. Parametric alignment partitions the parameter space into regions such that in each region any alignment, that is optimal for some choice of parameters inside the region, is optimal throughout that entire region and nowhere else. A software to visualize and explore the parameter space is also provided.

**Isolated Subtree Distance**

The isolated subtree distance was first proposed in [198]$^5$ and is also referred to as the *structure respecting edit distance* or *structure preserving mapping distance*. Intuitively, it restricts mappings such that two separate subtrees in $T_1$ are mapped to two separate subtrees in $T_2$. Alternatively formulated, trees can only be mapped to trees and not to forests.

$^5$In [198], Tanaka & Tanaka refer to an earlier publication that introduce this distance [197]. As it is written in Japanese I was not able to validate this. Further early contributions in the field of tree editing, again in Japanese, are given in [1, 193–196].
Mappings  A mapping $M$ between trees $T_1$ and $T_2$ is an isolated subtree mapping if for all $(v_1, w_1), (v_2, w_2), (v_3, w_3) \in M$ holds:

$$lca(v_1, v_2) = lca(v_1, v_3) \iff lca(w_1, w_2) = lca(w_1, w_3)$$

(isolated subtree condition)

The isolated subtree distance $\delta_{T_1}$ between $T_1$ and $T_2$ is the minimum cost that an isomorphic subtree mapping between them can achieve. Formally,

$$\delta_{T_1}(T_1, T_2) = \min\{\delta(M) \mid M \text{ is an isolated subtree mapping} \text{ between } T_1 \text{ and } T_2\}. \quad (2.14)$$

Figure 2.10 shows an example of a mapping that is not an isolated subtree mapping, but corresponds to an alignment. The metric properties of the isolated subtree distance are proven in [236].

Algorithms  Tanaka & Tanaka proposed an algorithm that computes the isolated subtree distance in $O(|T_1| \cdot |T_2| \cdot \min\{|leaves(T_1)|, |leaves(T_2)|\})$ time and $O(|T_1| \cdot |T_2|)$ space [198]. Zhang improved the worst case complexity to $O(|T_1| \cdot |T_2|)$ time and space [236]. Later, Richter presented an algorithm that computes the isolated subtree distance in $O(|T_1| \cdot |T_2| \cdot \text{degree}(T_1) \cdot \text{degree}(T_2))$ time and $O(|T_1| \cdot \text{depth}(T_2) \cdot \text{degree}(T_2))$ space. For balanced trees of bounded degree $k$, i.e. each internal node has $k$ children, this algorithm consumes less space than Zhang’s Algorithm.

Top-Down Distance

Although I introduce the top-down distance at the end of this survey, its introduction by Selkow opened the discipline of tree edit distances in 1977 [177]. He considered a tree edit distance model where insertions and deletions are restricted to the leaves of a tree: Only leaves may be deleted, and a node may be inserted only as a son of a leaf.
Mappings In terms of mappings, this has the consequence that whenever w.l.o.g. a node \( v \) in \( T_1 \) is mapped to some node in \( T_2 \), all ancestor nodes of \( v \) must be included in the mapping. Given some mapping \( M \) between \( T_1 \) and \( T_2 \), let \( M|_1 \) and \( M|_2 \) be the nodes in \( T_1 \) and \( T_2 \) that are touched by \( M \), respectively. Let \( \text{ancs}_T(v) \) denote the set of all ancestor nodes of \( v \). Formally, a mapping \( M \) between trees \( T_1 \) and \( T_2 \) is a top-down mapping if the following holds:

\[
(v, w) \in M \Rightarrow \text{ancs}_{T_1}(v) \subseteq M|_1 \text{ and } \text{ancs}_{T_2}(w) \subseteq M|_2 \tag{2.15}
\]

The top-down distance \( \delta_{\text{TD}} \) between \( T_1 \) and \( T_2 \) is the minimum cost that an
top-down mapping between them can achieve:

$$\delta_{TD}(T_1, T_2) = \min\{\delta(M) \mid M \text{ is a top-down mapping between } T_1 \text{ and } T_2\}$$  \hspace{1cm} (2.16)

Recently, Valiente proposed a “dual” model, a bottom-up distance between Trees, where deletions and insertions must begin at the root level [210].

**Algorithms**  Selkows algorithm computes the top-down distance in $O(|T_1| \cdot |T_2|)$ time and space [170, 177]. The algorithm was implemented and applied to the problem of identifying syntactic differences in [235].

### 2.5.4 Related Problems

**Similar Consensus Problems**

The similar consensus problem is the problem of finding a largest approximately common substructure in trees. For strings, a substructure is a subword. For graphs, a substructure can be defined as a connected subgraph which for trees results in my definition of a tree pattern. Let $d$ be an integer, the similar consensus problem is to find pattern trees $T'_1$ of $T_1$ and $T'_2$ of $T_2$ such that the distance between $T'_1$ and $T'_2$ is within distance $d$ and there does not exists any other substructure $T''_1$ of $T_1$ and $T''_2$ of $T_2$ that satisfy the distance constraint and $|T'_1| + |T'_2| \leq |T''_1| + |T''_2|$. The similar consensus problem was studied for the different distances that were presented in this section:

<table>
<thead>
<tr>
<th>distance</th>
<th>time complexity</th>
<th>studied in</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree edit distance</td>
<td>$O(d^2 \cdot</td>
<td>T_1</td>
</tr>
<tr>
<td>tree alignment distance</td>
<td>$O(</td>
<td>T_1</td>
</tr>
<tr>
<td>isolated subtree distance</td>
<td>$O(d^2 \cdot</td>
<td>T_1</td>
</tr>
<tr>
<td>top-down distance</td>
<td>$O(d^2 \cdot</td>
<td>T_1</td>
</tr>
</tbody>
</table>
Tree Inclusion Problems

The tree inclusion problem is a variant of the general tree edit distance. In terms of a maximum isomorphic subtree, a tree pattern $T_p$ is included in a target tree $T$ if $T_p$ can be obtained from $T$ by node deletions. This corresponds to an edit model that only supports the functions relabel and insert where $T_p$ is the first and $T$ the second tree. Kilpäinen & Mannila presented an algorithm that solves this problem in $O(|T_p| \cdot |T|)$ [98]. Improvements and variations of their algorithm are proposed in [3, 20, 160]. The classic problem of tree pattern matching is a restricted version of the tree inclusion problem. The deletion of nodes in the target tree is only allowed for leaf nodes in $T$ (and the trees that result from such deletions), which is equivalent to subtree removals in $T$. This corresponds to the tree inclusion problem in the domain of Selkow’s top-down distance. Among others, substantial contributions are reported in [28, 38, 86, 106, 119, 125, 157].

Zhang et al. considered the approximate tree matching in the presence of variable length don’t cares (VLDC) [219]. The query tree can contain wildcards that may match multiple nodes. For example, symbol “|” substitutes for a part of a path from the root to a leaf in the target tree. Symbol “~” matches a path and all subtrees emanating from the nodes on that path. Building upon that wildcards, the authors introduced a querying language for inexact matching of trees.

2.5.5 Arc Annotated Sequences

The pure sequence based approaches to compare RNA secondary structures are known to have the problem of violating the tree structure (see Section 2.5.2). On the other hand, tree edit based approaches are so far limited to compare RNA secondary structures. Moreover, in the coarse grained tree representation the meaning of tree edit operations in the process of editing RNA structures is difficult to motivate biologically. In the natural tree representation, the tree edit model cannot account adequately for a deletion
of a base-pair bond. This gave rise to the idea of incorporating structural constraints into sequence alignment strategies.

The first structural refined sequence alignment algorithm was proposed by Sankoff [172], although for the more sophisticated problem of folding and aligning simultaneously. Bafna et al. introduced the concept of RNA strings which include both, the primary sequence and the secondary structure information [4]. Beside matching problems on RNA strings, they introduced an alignment model for RNA strings. Evans generally studied annotation schemes that add auxiliary information to a sequence. These can be taken into account when the sequences are analyzed [45]. Evans introduced the general notion of arc-annotated sequences. An arc is a link joining two different symbols of a sequence and can be used to represent a binary relation between them. The definition of an arc-annotated sequence complies to the definition of a tertiary structure (see Section 2.2). As a natural extension of the longest common subsequence problem, Evans introduced the longest arc-preserving common subsequence problem [45]. This problem is not only studied extensively due to its potential application for RNA structure comparison, but also because it has a compact definition, is easy to understand and turned out to be NP-hard even for RNA secondary structures [114]. Zhang et al. introduced a further edit model for RNA structures including tertiary interactions [242]. For RNA secondary structures, their model corresponds to the tree edit model in conjunction with the natural tree representation. Finally, Jiang et al. suggested a set of edit operations for RNA structures that are biological motivated and form a superset of edit operations of the formerly mentioned models [94]. I introduce this general edit model for RNA structures first and use its terminology to give a uniform description of the other models.

\footnote{A general arc-annotated structure additionally allows a connection of one to many characters. I neglect this case since complex interactions like base-triplets are beyond the scope of this thesis.}
Figure 2.11: Structural edit operations of Jiang et al.’s general edit model for RNA structures. Sequence edit operations that do not involve base-pairs are omitted in this figure.

A General Edit Model for RNA Structures

Jiang et al. proposed a set of edit operations for RNA structures that are motivated by the evolution of structural RNA [94].

Edit operations  An edit operation that affects the primary and the secondary structure transforms an RNA structure (S₁, P₁) into a structure (S₂, P₂) by modifying both, S₁ and P₁. Since a deletion or insertion of a base in S₁ requires to “adjust” the indexes of the base-pairs in P₁, the definition of edit operations is intricate on that level. I introduce a terminology for structural edit operations that is consistent with the terminology of the sequence and tree edit model. To uniquely define structural edit operations, the positions that are affected by the operation must be specified as well as the new base for base-replacements. For convenience, I define the rules in terms of their effect on sequence and structure. The parameterized edit operations can be derived from this description. Let be $u, v, w \in \Sigma_{RNA}^*$ and $a, b, c, d \in \Sigma_{RNA}$. Let the concatenated string $u'v'w'$ be a dot-bracket sequence in spirit of the Vienna strings that defines an RNA structure. Moreover, let the brackets “(“ and “)”) uniquely identify a base-pair. Note, the unique correspondence of a bracket string to an RNA structure requires different pairs of brackets in the presence of tertiary interactions. The symbol “.” denotes an unpaired base. I arrange structure and sequence such that the structure is shown on top of the sequence. The changes by an edit operation are indicated as bold
A family of structural conserved RNA molecules does often exhibit compensatory base mutations in stem regions. The replacement of a base-pair is modeled by the following edit operation:

\[
\begin{vmatrix}
    \text{u}' & \text{v}' & \text{w}' \\
    \text{u} & \text{a} & \text{v} & \text{b} & \text{w}
\end{vmatrix}
\rightarrow
\begin{vmatrix}
    \text{u}' & \text{v}' & \text{w}' \\
    \text{u} & \text{c} & \text{v} & \text{d} & \text{w}
\end{vmatrix}
\] (base-pair replacement)

This notation is read as follows: \((S_1, P_1)\) is edited to \((S_2, P_2)\) where \(S_1 = uavbw\), \(P_1 \simeq u'(v)w'\), \(S_2 = ucvdw'\), and \(P_2 \simeq u'(v')w\). The operator \(\simeq\) means that the lefthand set of base-pairs is compatible with the base-pair pattern given by the righthand string. If \(a = c\) and \(b = d\) then the operation is also referred to as a base-pair match, otherwise it is denoted a base-pair mismatch. The disappearance of a base-pair, i.e. two pairing bases are lost during evolution, is given by:

\[
\begin{vmatrix}
    \text{u}' & \text{v}' & \text{w}' \\
    \text{u} & \text{a} & \text{v} & \text{b} & \text{w}
\end{vmatrix}
\rightarrow
\begin{vmatrix}
    \text{u}' & \text{v}' & \text{w}' \\
    \text{u} & \text{v} & \text{w}
\end{vmatrix}
\] (base-pair deletion)

During the evolution of an RNA structure, it can happen that the bond between two bases becomes too weak due to mutations in other regions of the structure. Accordingly, the disappearance of a base-pair bond is among the structural edit operations:

\[
\begin{vmatrix}
    \text{u}' & \text{v}' & \text{w}' \\
    \text{u} & \text{a} & \text{v} & \text{b} & \text{w}
\end{vmatrix}
\rightarrow
\begin{vmatrix}
    \text{u}' & \text{v}' & \text{w}' \\
    \text{u} & \text{a} & \text{v} & \text{b} & \text{w}
\end{vmatrix}
\] (base-pair breaking)

The scenario where a base-pair bond disappears because one of the pairing bases is deleted is modeled by either of the following two edit-operations.

\[
\begin{vmatrix}
    \text{u}' & \text{v}' & \text{w}' \\
    \text{u} & \text{a} & \text{v} & \text{b} & \text{w}
\end{vmatrix}
\rightarrow
\begin{vmatrix}
    \text{u}' & \text{v}' & \text{w}' \\
    \text{u} & \text{v} & \text{b} & \text{w}
\end{vmatrix}
\] (base-pair altering right)
Bases that are not paired undergo the classical sequence edit operations:

$$u' \cdot v' \rightarrow u' \cdot v'$$

(base-replacement)

$$u' \cdot v' \rightarrow u' \cdot v'$$

(base-deletion)

Each of the edit operations can also be read and applied from right to left. For edit operations that involve the deletion of bases or base-pairs this defines the corresponding insert versions. Figure 2.11 shows the edit operations in an alignment on the sequence and structure level.

The concept of edit-sequences can be naturally applied: Let $E$ be an edit-sequence $e_1, e_2, \ldots, e_n$. $E$ transforms $(S, P)$ into $(S', P')$ if there is a sequence of structures $(S_0, P_0), (S_1, P_1), \ldots, (S_n, P_n)$ such that $(S, P) = (S_0, P_0)$, $(S', P') = (S_n, P_n)$ and $(S_i, P_i)$ results from the application of $e_i$ to $(S_{i-1}, P_{i-1})$ for $i \in [1, n]$. Let $\delta$ be a cost function defined on edit operations. The cost of an edit-sequence $E$ is the sum of costs of its edit operations, that is: $\delta(E) = \sum_{i=1}^{n} \delta(e_i)$. The general edit distance $\delta_{GE}$ between structures $(S_1, P_1)$ and $(S_2, P_2)$ is the minimum cost that is necessary to transform $(S_1, P_1)$ into $(S_2, P_2)$. Formally,

$$\delta_{GE}((S_1, P_1), (S_2, P_2)) = \min \{ \delta(E) \mid E \text{ is an edit sequence transforming } (S_1, P_1) \text{ into } (S_2, P_2) \}. \quad (2.17)$$

**Algorithms**  Jiang et al. provided algorithms and complexity results for a fixed scoring scheme, i.e. the cost of an edit operation does not account
2.5 RNA Structure Comparison

for the involved bases, or equivalently, it is a constant [94]. Computing \( \delta_{\text{GE}} \) between \((S_1, P_1) \) and \((S_2, P_2) \) where \( P_1 \) is a tertiary structure and \( P_2 = \emptyset \) is MAX SNP-hard. For a restricted model that omits the base-pair altering and base-pair deletion edit operations, they propose an algorithm that requires \( O(|S_1|^2 \cdot |S_2|^2) \) time. If \( P_1 \) is a secondary structure and \( P_2 = \emptyset \) the general (unrestricted) problem is solvable in \( O(|S_1| \cdot |S_2|) \) time. The case when both \( P_1 \) and \( P_2 \) are secondary structures is not considered in [94]. I will show in Section 2.5.5 that the general edit model with a certain scoring function is NP-hard.

**Bafna et al.’s Model**

Bafna et al. introduced a sequence alignment problem for RNA secondary structures that maximizes both, base and base-pair replacement scores [4]. Let \( \alpha(a, b) \) be the score for replacing base \( a \) by base \( b \) and let \( \beta(a \circ b, c \circ d) \) be the score for relabeling a base-pair \( a \circ b \) by base-pair \( c \circ d \). Given an alignment \( A \) of sequences \( S_1 \) and \( S_2 \), I define \( A_{S_i} \) to be the \( i \)th row in \( A \). Let \( \text{gap}_{S_i}[j] \) be the number of gaps that are inserted in \( S_i \) up to the \( j \)th position in \( A \). Formally:

\[
gap_{S_i}[j] = \begin{cases} 
  j & \text{if } A_{S_i}[j] = ‘\lambda’, \\
  |\{l \mid A_{S_i}[l] = ‘\lambda’ \text{ and } l \leq j\}| & \text{otherwise.}
\end{cases}
\]

Bafna et al. do the following trick to for a compact definition of their model: They define \( S_i[0] = ‘\lambda’ \). If there is a gap in \( S_1 \) at position \( i \), \( S_1[i - \text{gap}_{S_1}[i]] \) evaluates to “\( \lambda \)” which corresponds to an insertion. The corresponding holds for \( S_2 \). Let \( m \) be the number of columns in an alignment \( A \). The score of \( A \) is the sum of scores of the aligned bases, be they paired or unpaired, and the scores of the aligned base-pairs. The sequence score \( \alpha \) is defined as

\[
\alpha(A) = \sum_{1 \leq i \leq m} \alpha(S_1[i - \text{gap}_{S_1}[i]], S_2[i - \text{gap}_{S_2}[i]]).
\]
The base-pair scoring is defined as:

$$\beta(A) = \sum_{1 \leq i \leq j \leq m} \beta(S_1[i - \text{gap}_{S_1}[i]) \circ S_1[j - \text{gap}_{S_1}[j]], S_2[i - \text{gap}_{S_2}[i]) \circ S_2[j - \text{gap}_{S_2}[j])]$$

where $(i - \text{gap}_{S_1}[i], j - \text{gap}_{S_1}[j]) \in P_1$

and $(i - \text{gap}_{S_2}[i], j - \text{gap}_{S_2}[j]) \in P_2$.

Bafna et al.’s score $\sigma_{BAF}$ is the sum of these scores:

$$\sigma_{BAF}(A) = \alpha(A) + \beta(A) \tag{2.18}$$

The similarity score of secondary structures $(S_1, P_1)$ and $(S_2, P_2)$ is then given by:

$$\sigma_{BAF}((S_1, P_1), (S_2, P_2)) = \max\{\sigma_{BAF}(A) \mid A \text{ is an alignment of } S_1 \text{ and } S_2\} \tag{2.19}$$

Note that $S_1$ and $S_2$ are sequences and, thus, $A$ is a sequence alignment.

**Algorithms**  Bafna et al. provide an algorithm that computes $\sigma_{BAF}((S_1, P_1), (S_2, P_2))$ in $O(|S_1|^2 \cdot |S_2|^2)$.

**Bafna et al.’s Model Revisited**  Bafna et al.’s model has been criticized for not systematically treating base-pairs as basic units [45, 94]. I show that their model can be expressed in the general edit model with a special scoring scheme: Function $\alpha$ scores base replacements, base-insertions and base-deletions. The scoring contributions are $\alpha(a, b), \alpha(\lambda, b)$ and $\alpha(a, \lambda)$, respectively. Clearly, function $\beta$ in Equation (2.18) does only account for base-pair replacements. In this case, the function $\alpha$ contributes additionally to the overall score for the aligned base-pairs. Thus, the score for a base-pair replacement of $a \circ b$ with $c \circ d$ is $\beta(a \circ b, c \circ d) + \alpha(a, c) + \alpha(b, d)$. Otherwise, a base, be it paired or unpaired, can be aligned with any other base and the scoring contributions for aligning a base $a$ with a base $b$ is $\alpha(a, b)$. A scoring contribution of 0 for the base-pair breaking operation allows to align paired
bases to unpaired bases without a penalty. The deletion of a base-pair is composed of a base-pair breaking and two base-deletions. The corresponding holds for the base-pair insertion. A base-pair altering is composed of a base-pair breaking, a base-match and a base-indel. Summarizing these observations, $\sigma_{BAF}$ can be calculated by employing the following scoring scheme for Jiang et al.’s general edit model:

<table>
<thead>
<tr>
<th>edit operation</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>base replacement</td>
<td>$\alpha(a, b)$</td>
</tr>
<tr>
<td>base indel</td>
<td>$\alpha(a, \lambda)$ and $\alpha(\lambda, b)$</td>
</tr>
<tr>
<td>base-pair replacement</td>
<td>$\beta(a \circ b, c \circ d) + \alpha(a, c) + \alpha(b, d)$</td>
</tr>
<tr>
<td>base-pair breaking</td>
<td>0</td>
</tr>
</tbody>
</table>

I conclude that Bafna et al.’s model is a proper structural alignment model which means that it can be expressed in Jiang et al.’s general edit model. Whether the scoring of edit operations is a good choice or not remains to be analyzed.

The Longest Arc-Preserving Common Subsequence Problem

The longest arc-preserving common subsequence problem is an extension of the classic longest common subsequence problem. A sequence $S'$ is a subsequence of a sequence $S$ if $S'$ can be obtained from $S$ by deleting characters. Given a set of sequences $S_1, S_2, \ldots, S_n$, the longest common subsequence problem asks for the longest sequence $S'$ that is a subsequence of $S_1, S_2, \ldots, S_n$.

Mostly driven by the application of RNA structure comparison, including tertiary structures, Evans generalized the problem for arc-annotated sequences [45]. Let $(S_1, P_1)$ and $(S_2, P_2)$ be arc annotated sequences which means that $P_1$ and $P_2$ can be tertiary structures throughout this section. A longest common subsequence $S'$ of $S_1$ and $S_2$ induces a mapping between characters in $S_1$ and $S_2$ by associating the characters $i_k$ in $S_1$ and $j_k$ in $S_2$, that correspond to the $k$th position of $S'$. Suppose $M = \{(i_1, j_1), (i_2, j_2), \ldots, (i_{|S'|}, j_{|S'|})\}$ is such a mapping. The longest common subsequence $S'$ is arc-preserving if
the arcs touched by the mapping are preserved. That is, for any \((i_k, j_k), (i_l, j_l) \in M\) holds:

\[(i_k, i_l) \in P_1 \iff (j_k, j_l) \in P_2. \] (2.20)

The longest arc-preserving common subsequence (LAPCS) problem is to find a longest common subsequence \(S'\) that is arc-preserving.

Different instances of the problem, depending on the complexity of the arc set (here the complexity of RNA structures), are studied in the literature. The relevant instances in the context of RNA sequence and structure comparison are \(\text{LAPCS}(P_1, P_2)\) where \(P_i\) belongs to one of the following classes:

- **PLAIN**: no structure, i.e. \(P_i = \emptyset\)
- **NESTED**: \(P_i\) is a secondary structure
- **CROSSING**: \(P_i\) is a tertiary structure

I follow this terminology since it is established in the literature concerning LAPCS problems [2, 45, 93, 114]. I review the most important results and comment on the LAPCS(NESTED,NESTED) problem which is particularly interesting for comparing RNA secondary structures in the following.

**Algorithms**  LAPCS(PLAIN,PLAIN) is the well known longest common subsequence problem which can be solved in \(O(|S_1| \cdot |S_2|)\) [76]. If the number of sequences is unrestricted this problem is NP-complete [124]. Otherwise, if at least one structure is CROSSING, the problem is NP-hard [45]. A maximization optimization problem, such as the LAPCS problem, is \(\alpha\)-approximable if there exists a polynomial time algorithm \(\mathcal{A}\) and a positive number \(\alpha\) such that the output of \(\mathcal{A}\) is within a factor \(\frac{1}{\alpha}\) of the optimum. If at least one structure is CROSSING, the LAPCS problem is also MAX SNP-hard which has the consequence that it is not approximable within \(\alpha = 1 + \epsilon\) for some positive \(\epsilon\) [93]. A 2-approximation algorithm for these problems is proposed in [93].
The probably most relevant problem in the context of RNA structures is the LAPCS(NESTED,NESTED) problem to compare RNA secondary structures. The NP-hardness of this problem was shown in [114]. A LAPCS(NESTED,NESTED) that can be obtained by at most $k_1$ and $k_2$ character deletions (together with the corresponding arcs) can be calculated in $O(3.31^{k_1+k_2})$ [2]. A polynomial time algorithm for the LAPCS(NESTED, PLAIN) problem, running in $O(|S_1| \cdot |S_2|^3)$ time, is presented in [93].

**LAPCS(NESTED,NESTED) Revisited** A longest arc-preserving common subsequence of secondary structures $(S_1, P_1)$ and $(S_2, P_2)$ maps characters from $S_1$ to $S_2$. In the following, I observe which edit operations of the general edit model are compatible with such a mapping, resulting in an equivalent edit based description of the LAPCS(NESTED,NESTED) problem. The arc-preserving property (2.20) of a longest arc-preserving common subsequence guarantees that if both bases of a base-pair are mapped, then they must be mapped to bases that are also paired. In terms of the general edit model for RNA structures this means that there must exist a base-pair match operation but no base-pair breaking. The base-pair match adds two new characters to the longest arc-preserving common subsequence. The base-pair breaking operation can be excluded by assigning an infinite negative score to it. If only one base of a base-pair is mapped, then the other base must not exist in the mapping. This adds one new character to the longest arc-preserving common subsequence. The sequence edit operations base-match and base-indel handle these cases. Clearly, a longest arc-preserving common subsequence does not allow any mismatches and, hence, the scoring contribution for those cases must be $-\infty$. Summarizing these observations, the length of
a LAPCS can be calculated in Jiang et al.’s general edit model using the following scoring scheme:

<table>
<thead>
<tr>
<th>edit operation</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>base match</td>
<td>1</td>
</tr>
<tr>
<td>base mismatch</td>
<td>$-\infty$</td>
</tr>
<tr>
<td>base indel</td>
<td>0</td>
</tr>
<tr>
<td>base-pair match</td>
<td>2</td>
</tr>
<tr>
<td>base-pair mismatch</td>
<td>$-\infty$</td>
</tr>
<tr>
<td>base-pair indel</td>
<td>0</td>
</tr>
<tr>
<td>base-pair breaking</td>
<td>$-\infty$</td>
</tr>
<tr>
<td>base-pair altering</td>
<td>1</td>
</tr>
</tbody>
</table>

The LAPCS can be derived from the resulting alignment. The complexity of the LAPCS(NESTED,NESTED) problem was an important question until Lin et al. proved it to be NP-hard [114]. Since the computation of the general edit distance using the above scores solves the LAPCS problem, I conclude that the computation of the general edit distance for RNA secondary structures is a NP-hard problem for the above scoring scheme. I assume that the complexity results from the presence of the base-pair altering operations. If those must be considered explicitly, i.e. the score is not build from simpler edit operations, the number of resulting subproblems grows exponentially. This remains to be further analyzed.

**Zhang et al.’s Model**

Zhang et al. considered RNA secondary structure trees in the natural representation that are compared under the tree edit and alignment model in [238]. The entities of the tree nodes are bases and base-pairs (see Section 2.3). Thus, the classic edit operations replace, insert and delete can be applied to either an unpaired base or a base-pair. A replacement of a base by a base-pair is prohibited. Ma et al. extended this model for general RNA structures by extending the mapping concept of the tree edit model for general RNA
structures which is the central definition of this line of work \[123, 222, 242\].
The essential extension of the mapping is a new condition for “crossing” base-pairs. Intuitively, the crossing pattern of tertiary interactions should be conserved. I do not go into the details of their mapping definitions, since their model was constructed on the assumption of certain edit operations on structures. I will revisit their models in terms of Jiang et al.’s general model.

**Algorithms** Computing $\delta_{\text{ZHA}}((S_1, P_1), (S_2, P_2))$ where $P_1$ and $P_2$ are tertiary structures is MAX-SNP hard \[123\]. Ma et al. considered a simpler edit model for tertiary structures which restricts mappings between tertiary structures to preserve secondary structure. Essentially, their algorithm deletes tertiary structure interactions such that the resulting secondary structure alignment is optimal. Let $\text{stem}(P)$ be the number of stacking regions (stems) in an RNA structure $(S, P)$. Their algorithm requires $O(\text{stem}(P_1) \cdot \text{stem}(P_2) \cdot |S_1| \cdot |S_2|)$ time and $O(\text{stem}(P_1) \cdot \text{stem}(P_2))$ space. Collins et al. presented a variant of $\delta_{\text{ZHA}}$ with the constraint that bases and base-pairs can be specified that must be replaced by each other. They do not improve the complexity, but their technique reduces the search space and consequently the runtime \[29\]. Moreover, they propose a two step strategy for tertiary structures: In the first step, tertiary structures are ignored resulting in a secondary structure alignment. In the second step, the secondary structure alignment is used to restrict the tertiary structure alignment.

**Zhang et al.’s Model Revisited** The edit operations in Zhang et al.’s edit model can be applied to either unpaired bases or base-pairs. According to Jiang et al.’s model the structural edit operations are: base-pair replace and base-pair indel. The sequence counterparts are the operations base-replace and base-indel. An edit operation that works on both unpaired base and a base-pair is not defined in their model. Thus, there is no base-pair altering and base-pair breaking operation. An infinite negative score for these edit operations is sufficient to calculate Zhang et al.’s model under the general edit model for RNA structures:
<table>
<thead>
<tr>
<th>edit operation</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>base match</td>
<td>$\alpha_m$</td>
</tr>
<tr>
<td>base mismatch</td>
<td>$\alpha_{mm}$</td>
</tr>
<tr>
<td>base indel</td>
<td>$\alpha_{id}$</td>
</tr>
<tr>
<td>base-pair match</td>
<td>$\beta_m$</td>
</tr>
<tr>
<td>base-pair mismatch</td>
<td>$\beta_{mm}$</td>
</tr>
<tr>
<td>base-pair indel</td>
<td>$\beta_{id}$</td>
</tr>
<tr>
<td>base-pair breaking</td>
<td>$-\infty$</td>
</tr>
<tr>
<td>base-pair altering</td>
<td>$-\infty$</td>
</tr>
</tbody>
</table>

### 2.5.6 Graphs

The most general mathematical construct to model relations between certain objects is a graph. Clearly, an RNA tertiary structure can be modeled as a graph where the vertices are bases and the edges are interactions between them. Note, this concerns topological rather than geometric aspects of RNA molecules. For example, such a graph abstracts from the relative angles between stems.

#### Edit Models

Wan et al. considered the generalization of the tree edit model for graphs, these are approximate graph isomorphism and subgraph isomorphism [218]. Both are known to be NP-complete. They outline an application where RNA structures (not restricted to secondary structures) are compared under this model.

#### Eigenvalue Spectrum of the Laplacian Matrix

In the Schlicks’s group two simpler types of graphs are considered [47, 52, 53]. The one are tree graphs, corresponding to a collapsed form of the natural tree representation (see Section 2.3), where collapsed means that connected non-branching nodes are merged to one node (ignoring labels). The other
is dual graphs. A dual graph can represent all tree like RNA structures as well as pseudoknotted structures. They focus on the problem of quantitatively characterizing known structural motifs to identify missing or favored motif topologies. For the topological classification of structures they consider the eigenvalue spectrum of the Laplacian matrix obtained from the graph’s adjacency matrix. In particular, the second eigenvalue reflects the overall pattern of connectivity for a graph. Barash used the second eigenvalue to detect structural changes in RNA that are caused by single point mutations [6].

2.6 Discussion

The multitude of structure comparison models presented in Section 2.5 gives rise to the question why this thesis presents another RNA structure comparison model. Otherwise, this shows that RNA structure comparison is an active research field and the problem is not sufficiently solved.

Nowadays, the detection of locally structure conserved motifs in RNA molecules is a hot topic in molecular biology. On the algorithmic side, the problem of finding local similar structures, given RNA secondary structures, has not been studied thoroughly. The similar consensus problem for trees is the only contribution, I am aware of, to detect local similar regions in RNA secondary structures (see 2.5.4). However, this model calculates distance instead of similarity. As the distance between equal substructures is always zero, the size of substructures must be considered additionally. Hence, in the similar consensus problem, the largest subtree within some distance threshold is sought. A similarity version in spirit of the Smith-Waterman algorithm [184] for trees would be more convenient to calculate local similar structures. Moreover, the similar consensus problem considers subtrees. This is too restrictive since neighboring subtrees should be considered as local structures as well, i.e. two adjacent stems in a multiloop could be the most similar substructure which corresponds to two different subtrees.
Another problem that has not been addressed thoroughly is the problem of comparing multiple RNA secondary structures. As multiple sequence alignments emphasize sequence conserved regions, multiple structure alignments emphasize structural conserved regions. A multiple structural alignment is useful for phylogenetic analyses, identification of conserved motifs, and domain and structure prediction.

In the following, I motivate my choice of the tree alignment model to address the above problems. The model that I consider should have the following properties:

- a biologically reasonable edit model,
- suitable for a generalization to multiple structures,
- build upon an adequate data structure for local similarity problems,
- allow algorithms with a low computational complexity.

Base-pair distances are suitable to compare structures that have the same length, i.e. the same number of nucleotides. If the structures to be compared have a different length, edit based approaches provide a better distance measure.

The approach to apply classical sequence alignments to string representations of RNA secondary structures is more a historical remark. At the time these were invented, structural alignment strategies were just about to emerge. More elaborate models are edit and alignment models for trees and arc-annotated sequences.

Unlike sequences, trees are convenient to express substructures of RNA secondary structures as coherent parts of the data structure. In explanation, adjacent base-pairs are neighbored in a tree while in a sequence they are split in the 5′ and 3′ bases connected by arcs. From the viewpoint of being able to generalize the model to align multiple structures, the tree alignment model has an interesting property. Alignments of trees are trees. Thus, a tree alignments can, again, be aligned in the tree alignment model. This makes
virtually every progressive strategy known for the calculation of multiple sequence alignments applicable to the calculation of multiple tree alignments. Another property of tree based approaches is that the chosen tree representation can control the level of abstraction of RNA secondary structures. In the end, the time complexity for calculating tree alignments meets practical requirements.