Reconstruction of ancestral non-coding RNA sequences using sequential and structural information with Tree-decomposition

by

Songdi Hu

A thesis sumbitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Computer Science
University of Manitoba
Winnipeg

Reconstruction of ancestral non-coding RNA sequences using sequential and structural information with Tree decomposition

Songdi Hu

Abstract

Ancestral sequence reconstruction aims to infer what was the content of certain biological sequences of interest for ancestral species that do not exist anymore. This is accomplished by comparing and extracting similarities and differences from the sequences in extant (*i.e.* living) species. Since the search space is quite large, a lot of research has been devoted to the design of efficient and accurate methods to solve different variations of this problem. However, ancestral sequence reconstruction becomes even more complex when the goal is to reconstruct the ancestors of sequences that are not well conserved in extant species. This is the case with non-coding RNA (ncRNA) sequences, for which the structure (formed by base pairing) is more conserved than the actual sequences. One recent approach to tackle the ancestral reconstruction of ncRNA sequences involved considering the sequences of two related ncRNA families simultaneously [43]. Although this helped avoid biases in the reconstruction, some cost calculations had to be simplified for efficiency. In this thesis, the goal was to improve the cost calculation of that approach by using a more advanced structural model and tree decomposition to partition the cost calculation into subproblems. Our results demonstrate an important gain in accuracy

and a significant reduction in the number of optimal sequences inferred.

Acknowledgement

I would like to express my deepest gratitude to my thesis supervisor, Dr. Olivier Tremblay-Savard, for his expert guidance and valuable support throughout this research. His wisdom, knowledge, and commitment to the highest standards inspired and motivated me.

I am also thankful to Dr. Stephane Durocher and Dr. Maxime Turgeon for their insightful comments and encouragement, which have been invaluable on this journey.

My sincere thanks go to Dr. Vladimir Reinharz for his helpful advice and invaluable perspectives, which have significantly enhanced the quality of my research.

I am also grateful to Dr. Razvan Romanescu for providing additional research opportunities, which have enriched my academic experience and expanded the scope of my work.

I would like to extend my gratitude to the staff of Computer Science/University of Manitoba, Lynne Hermiston and Tetyana Pavlyuk, for their assistance in adminis-

tration. Their support played a crucial role in the completion of this thesis.

I am deeply grateful to my family and friends, particularly my parents, who provided me with endless support and understanding throughout my studies.

Finally, I would like to acknowledge Natural Sciences and Engineering Research Council of Canada (NSERC) for funding this research, making it possible to achieve the results presented.

Songdi Hu January 17, 2025

Contents

Chapter 1

Introduction

Bioinformatics is a multifaceted field that encompasses a variety of sub-disciplines, including medical informatics, transcriptomics, proteomics, functional genomics, structural bioinformatics, and comparative genomics. Comparative genomics specifically focuses on the analysis of genomes or genomic sequences to address questions related to the evolutionary processes responsible for the current diversity of genomes. It also aims to reconstruct ancestral genomes or sequences as they might have existed millions of years ago, which involves inferring everything from large ancestral chromosome sequences to ancestral gene orders or gene sequences[42, 6, 27].

Traditionally, the inference of ancestral sequences relies on having a phylogeny—typically a binary tree depicting the evolutionary relationships among the species under study. Various methods exist for building phylogenies based on different types of data, such as genetic or morphological relationships[13, 40]. Starting from the tree's leaves,

which are labelled with the current sequences of each species, ancestral reconstruction methodologies are applied to infer the sequences at the internal nodes, which represent ancestors. This reconstruction is done in a manner that minimizes a specific cost, which is defined by a given evolutionary model to penalize various mutations that alter the sequences along the tree branches. In comparative genomics, this challenge is known as the "small parsimony problem," and numerous strategies have been devised to tackle it[14, 36, 43].

However, these techniques are generally applicable only to sequences exhibiting a significant level of conservation. For example, when reconstructing ancestral non-coding RNA (ncRNA) gene sequences—genes that are transcribed into RNA but not translated into a protein—these methods face limitations. This is because ncRNA gene sequences are typically not well-conserved; rather, it is the structure of the base pairs, which imparts functionality to the ncRNA sequences, that remains stable[20]. Consequently, more innovative approaches that consider both the sequential and structural data are required.

This current study focuses on the challenge of inferring ancestral ncRNA sequences using the achARNement methodology introduced in 2016 by Tremblay-Savard et al.[43]. We will refer to it as `achARNement1.0` in the following sections. This method utilizes both sequential and structural information from two homogeneous families of ncRNAs to perform the inference. However, to avoid the extensive computational load caused by potential interconnections among the base pairs of different structures, the original algorithm was restricted to considering only one level of such dependen-

cies. In contrast, in this study, we designed and implemented a different strategy that employs tree decomposition and tree optimization techniques to manage the high time complexity induced by the inter-dependencies of the structures. We will refer to our new approach as `achARNement2.0` in the following sections. As a result, the refined algorithm (`achARNement2.0`) can account for more dependencies during the reconstruction process while maintaining a reasonable running time. A comparison is made between the results from `achARNement2.0` and three other existing algorithms, including `achARNement1.0`. Our proposed algorithm has demonstrated improved performance over all other methodologies regarding both accuracy and the breadth of the solution space.

From the results, `achARNement2.0` has been proven to be more accurate and produces fewer ancestral sequences (reduction of solution space) than some classic algorithms (`Fitch` and `Sankoff`), as well as its predecessor (`achARNement1.0`). A smaller set of ancestral sequences is beneficial since it is more interpretable and actionable compared to an overwhelmingly large set. It also contributes to the field of molecular biology and evolutionary studies. This new approach enables researchers in the field to pinpoint evolutionary changes with greater precision, offering clearer insights into the functional and structural evolution of RNA molecules. By generating fewer but more accurate ancestral sequences, the algorithm reduces the computational load and resource use, making it more feasible to analyze large datasets. This study has potential applications in biotechnology and medicine, such as in the design of stable RNA sequences with target structures. Using modern RNA sequences carrying desired structures, we can construct a phylogenetic tree with these sequences as the

leaves. Our proposed algorithm can then reconstruct hypothetical common ancestors (which may not represent actual historical ancestors) capable of folding into these target structures.

In Chapter 2, all the background knowledge essential for understanding this work is presented. Chapter 3 reviews existing methods and their limitations in ancestral sequence reconstruction. Chapter 4 defines the specific problem that we focus on in this study. Chapter 5 presents the core contribution of this study, describing the methodology of the algorithm we propose (`achARNement2.0`). In Chapter 6, we evaluated our algorithm on both simulated and biological data and compared them with three similar algorithms. Chapter 7 summarizes our findings and their significance. Limitations of the current study are discussed, followed by directions for future research and enhancements.

Chapter 2

Background

This chapter starts with a presentation of all the biological background that is necessary to understand the domain-specific terminology used in this thesis. Then, it describes several techniques from bioinformatics and graph theory that will be used in the context of this thesis.

2.1 Biology Background

2.1.1 DNA, RNA, Bases, Genes and Genomes

DNA, or deoxyribonucleic acid, is the molecule that carries all the genetic information in living beings, from humans to animals and plants. Every cell in an organism has

DNA, which is structured as a long, spiral staircase known as a double helix. This structure consists of two strands twisted around each other, with pairs of bases connecting the strands like steps on the staircase.

There are four types of bases (also called nucleotides) in DNA: adenine (A), thymine (T), cytosine (C), and guanine (G). Adenine always pairs with thymine, and cytosine always pairs with guanine. The sequence of these base pairs encodes the genetic instructions used in the development, functioning, and reproduction of organisms.

The primary role of DNA is to store this genetic information and pass it from one generation to the next. When cells divide, DNA is copied so that each new cell inherits a complete set of genetic information. This process ensures that cells in an organism's body can function properly and contribute to the organism's overall health. Additionally, segments of DNA are used to make RNA (ribonucleic acid), which is crucial for producing proteins. Proteins are necessary for many functions in the body, including acting as enzymes and hormones and building tissues.

RNA, or ribonucleic acid, is another crucial molecule in cellular biology, closely related to DNA, which is the primary genetic material. Unlike DNA, RNA is single-stranded and not double-helical. It contains the sugar ribose (rather than deoxyribose in DNA) and uses the base uracil (U) instead of thymine (T). One of the main roles of RNA is to convert the genetic information held in DNA into proteins, which perform vital functions in the body. This process begins with transcription, where an RNA copy of a DNA segment is created. This RNA copy is called messenger RNA (mRNA), and it carries the genetic code from the DNA in the nucleus of a cell to

the ribosomes, the cell's protein-making machinery. In the ribosomes, transfer RNA (tRNA) reads the sequence of mRNA and helps assemble the appropriate amino acid sequence to form proteins (amino acids are the building blocks of proteins). This process is known as translation. Another type of RNA, ribosomal RNA (rRNA), is part of the structure of ribosomes and plays a role in protein synthesis. In summary, RNA is essential for taking genetic blueprints from DNA and guiding the synthesis of proteins, which are needed for countless functions in living organisms.

A gene is a specific sequence of bases in DNA that encodes functional products, typically a protein or RNA molecule, which plays a pivotal role in the regulation of physiological functions and the manifestation of physical characteristics. Genes serve as the fundamental units of heredity and are located on chromosomes, where they occupy specific positions. The sequence within a gene dictates the assembly of amino acids to form proteins, which are critical for the maintenance of cellular function and the organism's development.

A genome, on the other hand, comprises the entire set of genetic material within an organism, or in other words, the entire DNA content. This includes not only the genes themselves but also the non-coding regions of the DNA that have roles in regulating gene expression. The genome embodies the full blueprint for the construction and maintenance of a living organism. In eukaryotic organisms, this genetic content is primarily contained within the cell nucleus, while prokaryotic organisms have their genetic material more directly accessible within the cell.

2.1.2 DNA Mutations

Small-scale DNA mutations refer to changes that occur at the level of individual nucleotides or small sequences within an DNA molecule. Such mutations can arise due to errors in DNA replication, the influence of mutagenic chemicals, or exposure to radiation. The consequences of small-scale mutations vary widely; they might be benign, alter protein function, or even lead to significant changes in an organism's phenotype. There are several types of small-scale DNA mutations:

(a) **Substitution/Point Mutations:** is the mutation when segments of the sequence (one or more bases) are replaced by the same number of bases. There are two types of such mutations:

  (i) Transition: A purine (adenine or guanine) is replaced by another purine, or a pyrimidine (cytosine or thymine) is substituted by another pyrimidine.

  (ii) Transversion: A purine is replaced by a pyrimidine, or vice versa.

(b) **Insertion and Deletion (Indel):** These mutations involve the addition or removal of one or a few nucleotides within the DNA sequence. Indels can have significant impacts on the stability of an RNA molecule or affect the ribosome's ability to correctly translate the messenger RNA into a protein.

(c) **Duplication:** A segment of DNA is duplicated, resulting in repeated sequences. Duplications in non-coding regions can be either silent (*i.e.* not cause

any changes to the genome) or affect regulatory regions that control gene expression. When coding sequences (i.e. genes) get duplicated, several different fates are possible for the copied genes, which are discussed below.

2.1.3 Fate of Duplicated Genes and Natural Selection

Gene duplication is a significant evolutionary process contributing to genetic novelty and complexity [31]. It involves the creation of an additional copy of a gene within the genome, providing raw material for the evolution of new functions and adaptations. This duplication can arise through various mechanisms, including:

- Tandem duplication: occurs when a gene is copied and inserted adjacent to the original gene, often due to errors in recombination.
- Segmental duplication: involves the duplication of larger chromosomal regions involving one or more genes.
- Whole-genome duplication (WGD): results in the duplication of the entire genome.

These mechanisms introduce redundancy into the genome, creating opportunities for the duplicated genes to follow distinct evolutionary trajectories. Once a gene is duplicated, the two copies may follow different evolutionary paths (fates). Birchler and Yang[3] discussed the primary fates of duplicated genes in their 2022 paper. The most interesting ones are:

- Hypofunctionalization: Both duplicates reduce function, but the combined output meets the functional need.
- Neofunctionalization: A new function appears in one copy after mutations. Both the new and the old functions are essential.
- Subfunctionalization: It is related to neofunctionalization, but the functions between copies are partitioned in this case. Both copies are required for full functionality.

It is worth noting that subfunctionalization is directly related to the evolutionary hypothesis behind `achARNement1.0`and `achARNement2.0`(see in Chapter 4

Natural selection on the other hand is a fundamental mechanism of evolution, describing how certain traits become more or less common within a population over generations. This process is believed to be driven by differential survival and reproduction, where individuals with advantageous traits are more likely to survive and pass their genes to the next generation. Natural selection can operate in different modes depending on the effect of genetic variations on fitness:

1. Positive selection: It occurs when a genetic variant confers an advantage, increasing an organism's fitness. Such advantageous traits are more likely to spread within a population.
2. Negative selection: It is also called "purifying selection". It removes harmful genetic variants from the population and helps maintain the integrity of

essential biological functions.

3. Neutral selection: This refers to genetic variation that occurs without affecting an organism's fitness. Such changes are not influenced by natural selection but are instead governed by genetic drift (changes in the genetic makeup of a population purely by chance). These mutations neither benefit nor harm an organism because they do not alter protein functions. Non-coding regions are more likely to be under neutral selection than coding regions.

Natural selection does not operate in isolation. These different types introduced above often interact, contributing to the evolutionary dynamics of populations. For instance, some genes may initially undergo neutral selection but later become subject to positive selection if environmental conditions change.

2.1.4 Non-coding RNA

Non-coding RNAs (ncRNAs) are a broad class of RNA molecules that are transcribed from DNA but not translated into proteins. These RNAs play various roles in cellular processes rather than serving as templates for protein synthesis. The significance of non-coding RNAs has been increasingly recognized, highlighting their crucial roles in gene regulation, cellular structure, and biochemical pathways. Common examples of ncRNAs are microRNAs (miRNAs), ribosomal RNAs (rRNAs), transfer RNAs (tRNA), long non-coding RNAs (lnoRNAs), small nuclear RNAs (snoRNAs), etc.

Non-coding RNAs are involved in numerous cellular functions; for example, they can participate in gene regulation, epigenetic regulation, structural roles, and defence mechanisms. They are also critical in the study of diseases, including cancer, neurodegenerative diseases, and viral infections[39]. Their potential as diagnostic biomarkers and therapeutic targets is an area of active research. For instance, specific miRNAs are associated with particular types of cancer, and modulating their activity can influence cancer cell proliferation and apoptosis[33].

Understanding the complex roles of non-coding RNAs continues to be a significant focus of molecular biology, with ongoing research uncovering new aspects of how these molecules can control cellular processes and contribute to disease. To understand the function of different ncRNAs, it is important to know how they fold, since their structure is essential to their role in the cell.

2.1.5 RNA Secondary Structure

As mentioned above, ncRNAs accomplish their functions in RNA form, which depends entirely on their molecular structure. The secondary structure, which identifies all the basepairs that are formed between nucleotides of the single-stranded ncRNA molecule, is typically used to represent this molecular structure (see Figure 2.1 for a visual representation). The dot-bracket notation is one of the common formats used for representing secondary structures in text. This notation provides a visual and textual method to denote the pairing relationships between nucleotides in a

single-stranded RNA molecule.

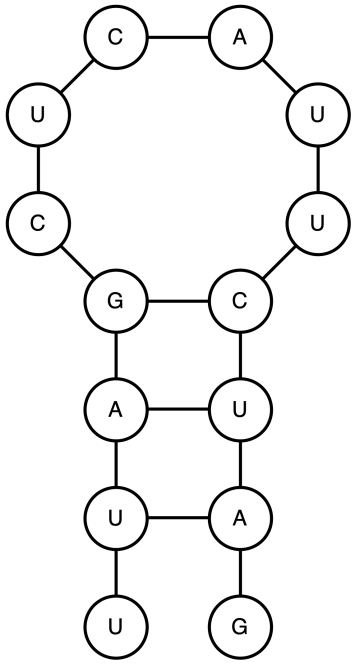

Figure 2.1: Example of a simple RNA secondary structure and its corresponding dot-bracket notation. In this figure, an RNA sequence of length eleven with three base pairs (G-C, A-U and U-A) is shown.

The dot-bracket notation (also known as dot-parenthesis notation) simplifies the depiction of the hydrogen-bonding interactions between nucleotides that form the base pairs typical of RNA secondary structures. In this representation, each nucleotide in the RNA sequence is symbolized by a character:

- A dot '.' represents an unpaired nucleotide, indicating that the nucleotide does not form a hydrogen bond with another nucleotide.
- Brackets '[ ]' or parentheses '( )' are used to denote paired nucleotides. An

opening bracket or parenthesis marks the start of a base-pairing interaction, and a closing bracket or parenthesis marks the end. The pairing of brackets or parentheses must be balanced and properly nested, reflecting the helical nature of double-stranded regions within the molecule.

2.1.6 RNA Families and Clans

Rfam is a comprehensive database of non-coding RNA families, each represented by multiple sequence alignments, consensus secondary structures, and covariance models (CMs)[17]. As the largest general alignment database, it catalogs diverse RNA classes from a wide range of organisms. This database is instrumental in bioinformatics, enabling the annotation of non-coding RNA structures in genetic sequences. Rfam categorizes RNA sequences by matching them with "seeds," which are manually identified exact matches from published data. However, this method struggles to identify relationships between distantly related RNAs. To address this, Rfam introduced "clans," which group families sharing presumed common ancestors (homologous families) based on literature and measures of similarity.

In RNA biology, the classifications of "RNA families" and "RNA clans" are vital for understanding the functional and evolutionary relationships among RNA molecules. RNA families consist of RNA molecules with significant similarities in their primary sequences and secondary structures, often indicating common functional or evolutionary origins. For example, the ribosomal RNA (rRNA) family includes RNA molecules

essential for protein synthesis in all living cells. These families are identified using sequence alignment and secondary structure prediction, revealing conserved sequences and motifs critical for their biological roles.

In contrast, RNA clans represent broader groupings of RNA families that may lack direct sequence similarity but share structural and functional traits[26]. This suggests either a distant common evolutionary origin or convergent evolution toward similar functions. Clans are particularly valuable for organizing RNA families into higher-order categories based on functional mechanisms or structural features, such as ribozymes that perform enzymatic reactions.

2.1.7 Base-stacking

Base-stacking is a fundamental molecular interaction in nucleic acids, where the nitrogenous bases of DNA or RNA align closely on top of one another in a stacked arrangement. This arrangement is stabilized primarily by van der Waals forces and hydrophobic interactions between adjacent bases, creating a stacking effect that contributes significantly to the structural stability of the nucleic acid molecules.

In DNA, base-stacking occurs between pairs of adenine-thymine (A-T) and guanine-cytosine (G-C) bases along the helical axis, promoting the compact and rigid structure of the molecule. The specific stacking interactions are influenced by the chemical properties of the bases and their electronic configurations. This stacking interaction is particularly important in stabilizing the double helix alongside hydrogen bonding

between complementary base pairs.

Base-stacking is also essential for the fidelity and efficiency of biological processes, such as replication and transcription, as it influences the overall structural properties and flexibility of DNA or RNA[23, 21]. This interaction is central to understanding nucleic acid stability, function, and interactions with proteins.

2.1.8 Minimum Free Energy

Minimum Free Energy (or MFE) is a thermodynamic concept that describes the equilibrium state of a system (in terms of energy). In the context of RNA folding, the free energy can be interpreted as the energy released when a completely unfolded RNA molecule folds into its final structure or, equivalently, the energy required to unfold a fully folded RNA molecule back into its unfolded state. MFE is normally reported as a negative value, and according to the definition, a lower free energy value (or a more negative value) corresponds to greater structural stability.

2.2 Bioinformatics and Theory Background

2.2.1 Maximizing Parsimony

Parsimony maximization is one of the main approaches often used for many problems in comparative genomics. In maximum parsimony approaches, the goal is to find the

scenarios or ancestors that minimize the total number of events or a scoring function that represents the costs of all possible events.

2.2.2 Small Parsimony Problem and Ancestral Sequence Reconstruction

The small parsimony problem is the specific problem of finding ancestral sequences on the internal nodes of a given phylogeny in a way that maximizes parsimony. Sequences of extant organisms (on the leaves of the phylogeny) are given as input, and the different approaches traverse the tree in a bottom-up fashion to propagate the information/costs from the leaves to the ancestors. Once the root of the phylogeny is reached, all the information from the whole tree is aggregated, and optimal ancestral sequences can be enumerated at the root level. The choices that are made at the root can then be propagated down the tree to all the other internal nodes.

To solve the problems introduced above, ancestral sequence reconstruction refers to a type of computational and experimental technique used to infer the genetic sequences of ancient or ancestral genes, proteins, or entire chromosomes. Based on a given phylogenetic tree with sequences on the leaves, scientists "reconstruct/infer" the sequences of their common ancestors.

There are different kinds of ancestral character states we can use for ancestral reconstruction. Examples can be:

- DNA or RNA sequences: This involves reconstructing nucleotide sequences of ancestral genomes, chromosomes, genes or non-coding RNA sequences.
- Gene orders: This involves inferring the arrangement (order and content) of genes in ancestral chromosomes.
- Syntenies: This involves inferring the order of conserved blocks of genes (syntenies) that remain on the same chromosome across species.

In general, ancestral sequence reconstruction helps with understanding evolutionary processes, protein functions, and adaptation mechanisms over time. It can be used to study gene content in ancestors and analyze the frequencies of different types of evolutionary events that have transformed genomes over time. Once predicted, ancestral RNA sequences can be synthesized in the laboratory and experimentally tested to investigate their properties, such as stability, activity, and function.

2.2.3 Tree Decomposition

Tree decomposition is a fundamental concept in graph theory that is used to transform a general graph into a tree-like structure, which can simplify the analysis and solution of various computational problems on graphs. This method is particularly significant in the field of algorithm design and complexity analysis, facilitating more efficient algorithms for problems that are otherwise challenging to handle on general graphs.

Definition and Structure: A tree decomposition of a graph $G = (V, E)$ consists of two elements:

1. A tree T, where each vertex v of T is associated with a subset B_x of vertices from V (called a bag).
2. Each bag must satisfy the following conditions:
    - **Union of Bags:** The union of all bags B_x must cover all vertices from V.
    - **Edge Coverage:** For every edge $(u, v) \in E$, there must be at least one bag B_x that contains both u and v.
    - **Coherence:** If a vertex v appears in two bags B_i and B_j, then v must appear in all bags on the unique path between B_i and B_j in the tree decomposition T.

Tree decomposition is critical in the study of graph properties and the development of algorithms, particularly for those graphs that exhibit complex, interconnected structures. By structuring a graph into a tree-like format, many difficult problems become more tractable. For example, problems such as determining the maximum independent set[16], finding the minimum vertex cover[1], and solving other combinatorial challenges can be more efficiently approached within the framework of a tree decomposition[24, 45].

For example, in 2002, Koster et al.[25] illustrated an approach targeting PCSPs

(Partial Constraint Satisfaction Problems) with tree decomposition techniques. A CSP (or Constraint Satisfaction Problem) is to find an assignment of values to all the variables that satisfy all the given constraints or, in other words, an assignment of zero penalty. Unlike CSP, PCSP extends the idea of a CSP by relaxing the requirement that all constraints must be completely satisfied. The authors propose a solution methodology based on tree decomposition and treewidth concepts. They describe the process of breaking down the constraint graph of a PCSP into a tree structure where nodes represent subsets of the graph that can be independently solved with dynamic programming techniques. The efficiency of solving PCSPs using this approach relies on the "treewidth" of the decomposition—smaller treewidth can lead to more manageable subproblems.

Tree decompositions have a key aspect, called the "treewidth". The "width" of a tree decomposition is defined as the size of its largest bag minus one (see Figure 2.2 for an example of a tree decomposition and its treewidth). Furthermore, the "treewidth" of a graph is the minimum width among all possible tree decompositions of the graph. For a given tree decomposition, many combinatorial optimization problems can be solved in polynomial time by dynamic programming. However, the practicality of such an application very much depends on the treewidth of the given tree decomposition. Primarily, smaller treewidth potentially represents better time complexity. Thus, an approach for computing optimal or close to optimal tree decomposition of a graph will be helpful for reducing the time cost of the corresponding algorithm.

In 2005, an article by Bodlaender analyzed several algorithms for solving the problem

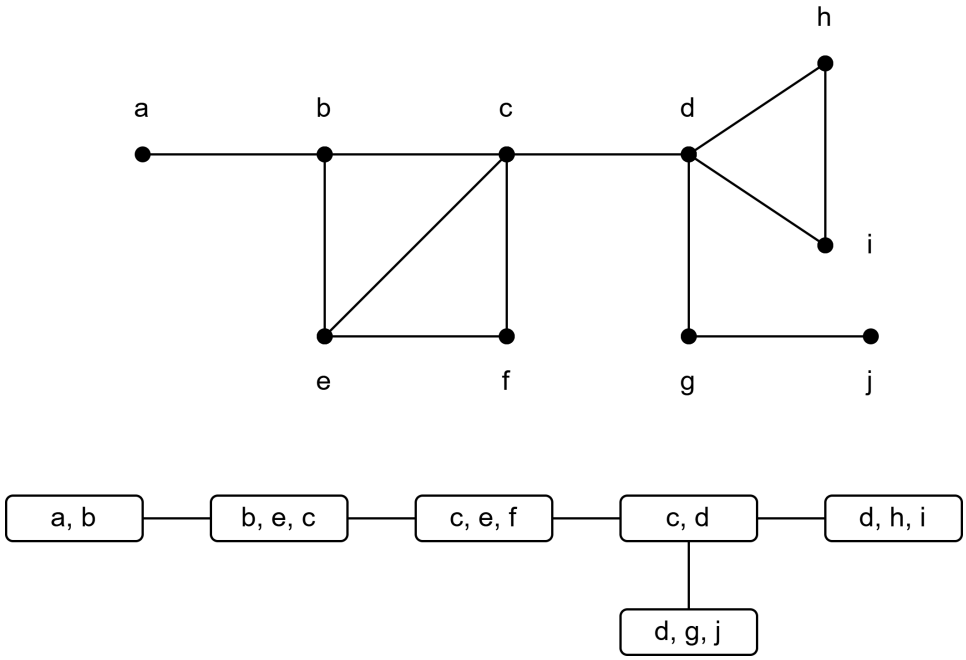

Figure 2.2: Example of a graph and its corresponding tree decomposition. The treewidth of this tree decomposition is 2.

of, given a graph G, finding the treewidth k or finding a tree decomposition of small width[4]. Although the problem, deciding if the treewidth of a graph G is at most k (k is an integer), was proved to be NP-complete, the authors still believe that in many cases, there exist methods that can solve it efficiently. The authors assert that, despite the NP-complete nature of the problem, certain instances can be effectively resolved using exact algorithms[2].

Besides exact algorithms, parameterized complexity approaches (fixed-parameter tractable) exist that could limit specific deterministic algorithms' time and memory complexity to an acceptable range in real-life applications[8]. More precisely, the complexity of such approaches is usually highly dependent on the value of one or

more parameters whose values are bounded in real-life cases.

2.2.4 Tree Decomposition Optimization

Tree optimization for tree decompositions is a strategic approach in graph theory and algorithm design aimed at enhancing the efficiency of solving complex graph-related problems. The primary goal of optimizing a tree decomposition is to minimize the tree's width. The significance of achieving low treewidth lies in its direct impact on reducing the computational complexity of subsequent algorithms, particularly those based on dynamic programming. These algorithms usually start from the leaves of the tree; solutions to the subproblems in each bag will be computed and combined with solutions from child bags to solve the subproblems in parent bags. This bottom-up approach continues until the root bag is processed, yielding a solution to the overall problem. Clearly, algorithms that operate on graphs with smaller treewidths (bags) can execute more efficiently due to the more manageable size of state spaces they must explore.

Research in this area has shown that optimizing tree decompositions can be particularly challenging due to its NP-hard nature[5]. Finding the optimal tree decomposition with the minimum treewidth is computationally expensive and often impractical for large graphs. As a result, much of the work in this field has focused on developing heuristic and approximation algorithms that seek to produce good, though not always optimal, decompositions in a reasonable timeframe[34].

These methods include greedy algorithms, where bags are incrementally constructed and optimized for size; iterative improvement techniques, where an initial decomposition is refined; and metaheuristic approaches like genetic algorithms and simulated annealing, which explore a broader set of possible decompositions in search of a near-optimal solution.

A great example of such methods is the "Tree diet" model proposed by Marchand et al.[30] in 2021. This approach centred on solving the *Tree diet* problem. The problem can be summarized as optimizing a tree decomposition into a target treewidth by removing a minimal set of edges. The authors designed and implemented a dynamic programming FPT (Fixed-Parameter Tractable) algorithm that focuses on reducing the treewidth of graphs derived from RNA structures, although the problem is shown to be Para-NP-hard (NP-hard when the desired treewidth is 1). Naively, the algorithm aimed to remove as few edges as possible to reduce the treewidth; however, depending on the application, there are instances when certain edges are more important than others (e.g. edges that should/prefer not to be removed). The *Tree diet* software provided an option to mark edges as "backbones" to avoid the removal. Overall, this approach is effective in managing the trade-off between computational efficiency and the completeness of the graph structure, enabling more effective handling of complex bioinformatics problems.

Once an optimized tree decomposition is achieved, it can significantly facilitate the resolution of complex problems such as constraint satisfaction, network reliability assessment, and many NP-complete problems, which become more tractable when

decomposed into a tree structure with a minimal width. The theory and applications of tree decompositions and their optimization continue to be a rich field of study in both theoretical and applied computer science[10, 35].

These concepts illustrate the depth and breadth of research on tree decompositions and underscore their practical relevance in optimizing algorithmic performance across various domains in computer science.

2.2.5 Frnakenstein for Multi-target Inverse RNA Folding

In 2012, Lyngsø et al.[29] discussed a genetic algorithm approach to solve the inverse RNA folding problem, specifically targeting the creation of RNA sequences that fold into predefined structures. It is highlighted in the study that although algorithms for RNA folding (predicting the structure from a sequence) are well-established, the inverse problem (designing sequences that achieve target structures) is more challenging, particularly when dealing with multiple target structures. Accordingly, a tool called "Frnakenstein" was developed, which utilizes a genetic algorithm to explore sequence possibilities. The tool adapts the genetic algorithm framework to maintain a population of candidate solutions, evolving these candidates through mutations and recombinations, and selecting the fittest solutions based on their ability to fold into the target structure(s). According to their results, the algorithm is capable of performing the inverse folding without a heavy bias towards CG base pairs, a common issue in other methods that are compared, which is beneficial for avoiding

thermodynamic bias in designed sequences. The algorithm was tested on multiple data sets, including cases where it had to design sequences that could fold into two different structures. It showed about 80 percent success in designs for two-structure targets.

Chapter 3

Literature Review

This literature review chapter summarizes published work that focuses on the reconstruction of ancestral sequences (based on sequence or structure) and the design of RNA sequences. These two types of problems are highly related to our goal of inferring ancestral RNA sequences, and the methodology we present in Chapter 5 combines aspects of both.

3.1 Sequence-based Approaches

3.1.1 The Fitch Method

One of the most well-known approaches for the small parsimony problem was presented by Fitch[14]. In the article, Fitch presented a method that, given the ancestral relationships (phylogeny) among the set of orthologous descendent sequences, finds the sets of optimal ancestral nucleotides at each internal node while considering all substitutions to be of equal cost. Instead of trying to discover the most parsimonious tree, Fitch aimed to identify the most parsimonious sequences at the ancestral (internal) nodes of the phylogenetic tree that reduce the overall number of evolutionary changes ("optimal" ancestral sequences for the internal nodes).

The algorithm is designed to function on a binary tree framework where each leaf node signifies a species, each labelled with character states describing its attributes (e.g. a DNA sequence). The process of assigning states to internal nodes in order to minimize the evolutionary changes along the tree branches encompasses a two-phase operation:

1. **Bottom-up step**: This step starts at the leaf nodes, advancing towards the tree's root. Here, each internal node is assigned a possible set of character states that reconcile the states observed in its descendants, thereby minimizing additional changes. If descendant nodes share common states, their intersection is used; otherwise, their union represents the possible states.

2. **Top-down step**: Upon determining a set of possible states for the root node by the bottom-up pass, the algorithm sequentially assigns states to the remaining internal nodes. This assignment ensures each node aligns with its parent node's state while adhering to the minimal change principle. In other words, when the parent state is present in the set of possible states for the child, that same state is selected for the child. Otherwise, an arbitrary state is selected for the child.

The efficiency of the Fitch algorithm lies in its ability to guarantee the minimal number of changes necessary to explain the observed character states across the tree. It operates with a computational complexity of $O(mnk)$, where m is the number of taxa, n represents the length of each sequence, and k denotes the number of characters (e.g. possible nucleotides).

3.1.2 The Sankoff Method

Another well-known algorithm in the field was presented by David Sankoff[36]. Unlike the Fitch algorithm, the Sankoff algorithm can handle different costs for different types of substitutions, making it suitable for more complex evolutionary models. This algorithm employs dynamic programming to minimize the total cost of evolutionary changes across a phylogenetic tree, considering both character transformations and their associated costs.

The Sankoff algorithm operates as follows:

1. Bottom-up step: Start at the leaves of the tree and work toward the root. At each node, compute the cost of assigning each possible nucleotide to the node based on the assignments of its direct child nodes and the substitution cost between characters (using a cost matrix).

2. Top-down step: Start at the root and assign the optimal character (with the minimum cost). Propagate the character assignments down to the leaves by considering the minimum cost paths computed in the postorder/bottom-up step.

In comparison, the Fitch algorithm is computationally efficient and well-suited for datasets where evolutionary change is uniformly probable across all states, while the Sankoff algorithm provides a more flexible and detailed model for datasets where, for example, transitions have different costs than transversions. The computational complexity of the Sankoff algorithm is proportional to $O(mnk^2)$, where m is the number of taxa/leaves, n is the length of each sequence, and k is the alphabet size (e.g. four for possible nucleotides). It is a bit slower compared to the Fitch method ($O(mnk)$) due to its exhaustive consideration of all possible state transitions and their costs.

3.1.3 Optimizing Sankoff

Traditional Sankoff parsimony is computationally intensive because it involves calculations that scale quadratically with the number of states, making it impractical

for large datasets. This classic method uses a cost matrix to determine the minimum number of changes required in a phylogeny, a calculation that becomes complex as the number of states increases. In 2009, Clemente et al.[9] proposed an improvement on the Sankoff parsimony. The basic idea is that not all transition costs between character states need to be calculated as in the original algorithm. Some of the unnecessary calculations when computing the cells in the scoring tables are avoided, which significantly reduces the number of operations required to compute the parsimony cost of a tree. The results showed significant time efficiency, particularly when dealing with a high number of states.

3.1.4 PAML

PAML is a software package designed for phylogenetic analysis of DNA and protein sequences using maximum likelihood (ML)[47]. The package specializes in estimating parameters of evolutionary models, testing phylogenetic trees, and exploring hypotheses about molecular evolution. It is widely used in evolutionary biology for tasks like ancestral sequence reconstruction, detecting selection, and estimating divergence times. The main approach for the reconstruction of ancestral sequences in the PAML software, as described in the paper, is a maximum likelihood (ML) empirical Bayes (EB) method. This approach utilizes maximum likelihood (ML) estimates to substitute parameters within the model, including branch lengths and substitution rates. In contrast, the hierarchical (full) Bayesian method accounts for sampling errors in parameter estimates by integrating these parameters over a prior

distribution. This distinction between the two methods becomes particularly significant when analyzing small datasets with limited information to reliably estimate parameters.

3.2 Structure-based Approaches

3.2.1 Evolutionary Triplet Models of Structured RNA

The paper "Evolutionary Triplet Models of Structured RNA" outlines a method for reconstructing ancestral RNA sequences and their structures by extending pairwise models of RNA evolution to multiple-sequence models[7]. The approach uses a stochastic context-free grammar (SCFG) to model RNA evolution, incorporating changes such as substitutions, insertions, deletions, and structural transformations. These pairwise models are expanded to phylogenetic trees through a process called transducer composition, which combines branch-specific models into a unified framework. This enables the reconstruction of ancestral sequences by accounting for both evolutionary relationships and RNA structural constraints.

Dynamic programming algorithms are employed to analyze the SCFG models, making it possible to infer ancestral sequences and their structures. The algorithms handle issues like "null cycles" (redundant rules) to ensure computational efficiency. To demonstrate the feasibility of the approach, the paper uses a simplified RNA evolutionary model, the TKF Structure Tree (TKFST), which represents RNA sec-

ondary structures with nested loops and stems. A prototype implementation called Indiegram applies this method to reconstruct ancestral RNAs for small phylogenies, such as those with three taxa.

This work emphasizes the importance of integrating structural constraints into ancestral RNA reconstruction to ensure biologically plausible results. By combining evolutionary modelling with structural information, the method provides a robust framework for studying the origins and evolution of RNA molecules, particularly for small datasets with well-characterized phylogenies.

3.2.2 achARNement1.0

More recently, Tremblay-Savard et al. proposed an approach[43] that targets the problem from a different perspective. The authors argued that most of the existing approaches for inferring ancestral ncRNAs, e.g. the work (the Triplet Model) presented in the previous subsection, have prohibitive time complexity. They also argued that the reconstruction of ancestral RNA sequences of a single ncRNA family with a single secondary structure using a covariation model can introduce bias into the resulting sequences/structures[11, 22, 48]. Indeed, traditional approaches that look at only one family for the reconstruction will have a tendency to produce ancestors near the core of its own evolution network. In other words, the problem with this is that the inferred ancestors tend to be more fit to the structure than the extant sequences, which does not make sense in terms of the evolutionary process

and natural selection. Thus, the authors of [43] proposed the idea of considering simultaneously the structures of two related ncRNA families in the inference process. These two families need to be part of the same clan, meaning that there is a valid assumption that they have evolved from a common ancestral ncRNA that could fold into the two structures. This specific ancestor is what the method aims to infer.

The results proved that, by considering structural constraints from both structure families, their approach effectively reduced the solution space compared to the classical algorithms (*i.e.* `Fitch` & `Sankoff`) and the state-of-the-art[46] (PAML) method while still being able to maintain a level of consistency (that all solutions produced are included in the larger solution spaces created by other methods). Note that this thesis work is based on and inspired by one of their proposed algorithms (*CalculateScores-2structs*, or *achARNement1.0* from now on) that considers both structures during the reconstruction.

3.3 RNA Design Approaches

RNA design approaches are closely related to our work on ancestral RNA reconstruction, as both aim to produce RNA sequences that satisfy specific structural and functional requirements. In ancestral RNA reconstruction, we integrate phylogenetic information and predicted structural features to infer plausible ancestral sequences. Similarly, RNA design focuses on engineering sequences to achieve predefined structures or functions.

3.3.1 Design of Multi-stable RNAs

Our study of ancestral RNA reconstruction with multiple structural constraints shares similarities with the design of multi-stable RNAs. A study by Christoph Flamm and colleagues presented such an approach[15]. It designs RNA sequences that can fold into multiple stable structures. This capability is significant for developing RNA-based switches that can adopt different structures and thereby perform various functions. In the paper, they propose transforming the problem of RNA design into a combinatorial optimization problem solvable using heuristics. The authors first introduced how the number of sequences compatible with the target structures can be predicted based on the paired positions of the given structure(s). These sequences are optimized to favour desired energy landscapes, such as having one or more deep energy minima corresponding to the target structures. To prevent misfolding, the approach is designed in a way that these local minima, calculated based on the target structures, are separated by a high energy barrier. Additionally, their approach is capable of designing RNAs that can change their preferred structures depending on the temperature, with a desired energy barrier.

3.3.2 Sampling for RNA Design with Multiple Target Structures

A more recent example of multi-stable RNA design is the fixed-parameter tractable approach proposed by Hammer et al.[19]. Designing RNA sequences that can fold

into multiple specified target structures is a computational challenge. The problem they solved was to develop an approach to efficiently sample RNA sequences that are capable of adopting multiple predefined structures with specific energy and GC content. The proposed algorithmic framework combined a fixed-parameter tractable (FPT) sampling algorithm with multi-dimensional Boltzmann sampling over distributions controlled by expressive RNA energy models to generate RNA sequences with the desired structural attributes. This approach leverages the combinatorial nature of RNA folding to manage the complexity of the problem, focusing on key parameters such as the number of target structures and sequence length. Results from this approach showed a good performance for multi-stable RNAs. The design of multi-stable (bi-stable) RNA, a secondary focus of our study, aligns closely with the objectives of the problem they address. The primary distinction in their study lies in the focus on paired positions during the sampling process, as their approach is structured around leveraging specific structural information. However, to achieve a comprehensive minimization of information loss, it is crucial to consider all nucleotide positions, both paired and unpaired.

Chapter 4

Problem Statement

4.1 Biological Hypothesis

The hypothesis behind this work is that ncRNA clans represent two homologous structure families that are linked to a common ancestral state. In other words, at some point in the past, there was a unique ncRNA sequence that could fold into both structures. After a duplication of that ancestral ncRNA sequence, the two resulting copies went through a long process of subfunctionalization, which resulted in each copy specializing itself into one of the two structures. This is the evolutionary process that gave rise to the two structure families existing today.

4.2 Input Data

- Two ncRNA families that are in the same clan[12] as well as their secondary structures.
- Multiple species having one copy of both ncRNAs that are from each family and their corresponding sequences.
- A species tree from public databases that represents the speciation history of the organisms considered.

4.3 Goal

In this study, we aimed to develop a model that accounts for these comprehensive factors, integrating both paired and unpaired positions, mutation potential, and base-stacking interactions to optimize sequence accuracy and structural fidelity. More specifically, for the given input data, the goal is to infer the ancestral sequences at each internal node of the species tree (including the root) by considering the sequence and structure conservation of all input ncRNAs (one from each family). Consecutive positions in a sequence (representing base stacking) is another constraint we will consider in addition to base pairs, as they can affect the stability of the structures. With these new factors, the model will become more general and complex, which motivates the use of techniques such as tree decomposition to make the cost calculations more efficient.

Formally, we describe our problem as follows:

- **Variables:** Let X be the set of all sites in an RNA sequence, $x_i \in X$ be the single sites; $D = \{A, C, G, U\}$, the alphabet or domain, denotes the possible nucleotides to assign to a single site; d_i represents the domain of the site at position i.

- **Factor(s)/constraint(s):** In our problem, the factor for each site/variable x_i is to find a/some nucleotide(s) a that minimizes the cost of having $x_i = \{a|a \in d_i\}$. The overall cost consists of two components:

  (i) **Substitution cost**, based on the nucleotide(s) from its children for the same site/position.

  (ii) **Structure cost**, estimated based on the given secondary structures.

  More specifically, the structure cost depends on the nucleotide selections of adjacent sites and the positions they are paired with in the current and the other structure.

Chapter 5

Methodology

5.1 achARNement1.0

To better understand the approach proposed in this thesis, a description of the methodology of `achARNement1.0` by Tremblay-Savard et al.[43] is necessary. Indeed, our new approach is using `achARNement1.0` as a foundation, but completely revamps the cost calculation strategy and the selection of most parsimonious sequences to take into account more structural information.

The previous version (*1.0*) involves three basic steps:

(i) A bottom-up step that does a post-order traversal of the given phylogeny. The most parsimonious cost (minimal costs) of having each possible nucleotide at

every position in the given sequences for each structure family is calculated and stored as cost matrices.

(ii) A middle step that links the minimal cost matrices (that we obtained from the bottom-up step) for both families at the root of the phylogeny. This results in a single cost matrix at the root representing the possible ancestors of all species considered in the tree.

(iii) A top-down/backtracking step that enumerates all the optimal sequences by making selections that follow the maximum parsimony principle (sequences of minimal cost).

The bottom-up step takes into account both structures when calculating the costs. One specific nucleotide at a position could have different costs for the two structures. To combine the costs, a variable G, or *gradient*, is introduced to control the weight that is given to the current structure family (G) versus the other ($1-G$). G is 100% at the leaves and 50% at the root, reflecting the trend of divergence in evolution (*i.e.*, sequences at the leaves are fully specialized into one specific structure, whereas sequences at the root were able to fold into two different configurations equally). We explain below, in more detail, how the minimal costs are calculated.

Cost calculation in detail:

The cost is calculated for each internal node. For each node, let a_i be the chosen nucleotide at position i in the parent sequence, l_i (resp. r_i) be the nucleotide at position i in the left (resp. right) child, and $\bar{l}_i$ (resp. $\bar{r}_i$) be the nucleotide paired

with l_i in the left (resp. right) child. If i does not pair to another position in the current structure, the cost of having a specific nucleotide at the parent is the accumulative substitution cost from its children. If i is a paired position in the current structure, the cost of having a dinucleotide a_i and $\overline{a}_i$ is calculated by:

$$\begin{aligned}
&\min_{l_i,\bar{l}_i\in\{A,C,G,U\}}\{c(l_i)+s(l_i,a_i)+c(\bar{l}_i)+s(\bar{l}_i,\overline{a}_i)+(G*bpc(a_i,\overline{a}_i))\}+\\
&\min_{r_i,\overline{r}_i\in\{A,C,G,U\}}\{c(r_i)+s(r_i,a_i)+c(\overline{r}_i)+s(\overline{r}_i,\overline{a}_i)+(G*bpc(a_i,\overline{a}_i))\}+\\
&(1-G)*\left(\sum_{nc\in\{A,C,G,U\}}bpc(a_i,nc)/4+\sum_{nc\in\{A,C,G,U\}}bpc(\overline{a}_i,nc)/4\right)
\end{aligned} \tag{5.1}$$

Where $c(x_i)$ is the previously calculated optimal cost of having nucleotide x at position i, while $s(x_i, y_i)$ is the substitution cost of having y instead of x at position i. At last, $bpc(x, y)$ is the basepair cost of having (x, y) as the basepair.

Due to limitations of the design, the approach does not consider all levels of dependencies while dealing with the conservation of multiple 2D structures simultaneously, as large cycles of dependent positions can be hard to resolve (see Figure 5.1 for an example). More specifically, when considering multiple structures for calculating the optimal costs, it would be more comprehensive to simultaneously consider all the positions that are interconnected through base pairs in the different structures. However, the existing achARNement method[43] chose to consider only one level of dependencies between two families to reduce the complexity of the calculations. For a position that is paired in both structures, the method considers the position paired

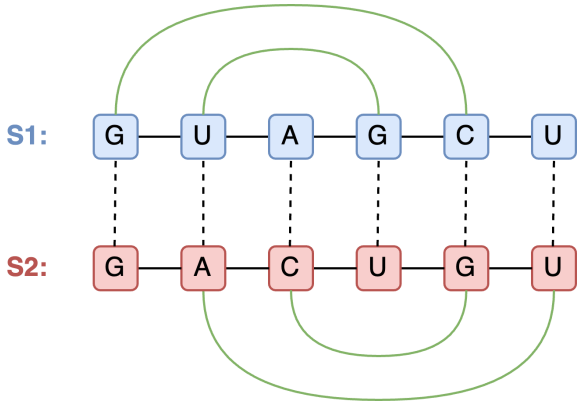

Figure 5.1: Simplified example of structure dependencies. Sequence one is in blue; sequence two is in red. Solid black lines connect the consecutive positions of a sequence, while green curved lines connect the base pairs. The same positions in both sequences are marked with dotted lines.

in the current structure, together with the average cost of any possible nucleotide paired with it in the other structure (see the last line of Equation 5.1).

5.2 achARNement2.0

5.2.1 Using Tree Decomposition

The `achARNement1.0` method follows a dynamic programming design that helps to bound the time-complexity and increases the accuracy of the reconstruction. However, to consider all possible structural dependencies, the running time would not be viable without reducing a considerable amount of calculations. More precisely, the time-complexity would be exponential when, in the worst case, the dependency graph contains all positions of two or more sequences in one path (each position in one

structure is dependent on a different corresponding position in another structure).

Our solution to the problem is to utilize the power of a dynamic program combined with a tree decomposition. However, as mentioned in the background chapter 2, treewidth can have a great impact on the running time of the DP algorithm. For this reason, we applied the *Tree diet* method, introduced in the background, for optimizing (limiting the treewidth) our tree decomposition. It is a method aimed to remove a minimal set of edges such that a given tree decomposition can be slimmed down to a prescribed treewidth[30].

Our approach to ancestral sequence reconstruction consists of two parts: a) preprocessing step that prepares the necessary data for our main algorithm; b) our main algorithm that performs the reconstruction based on the inputs and produces ancestral sequences.

5.2.2 Preprocessing

The preprocessing (see Figure 5.2) works directly on the input data and prepares it for the main algorithm. It involves the following steps:

(i) Build a network based on the two given secondary structures (base pairings).

(ii) Build a tree decomposition that maps the vertices and edges of the network.

(iii) Optimize the tree decomposition to a desired treewidth by removing a minimal set of edges with "Tree diet".

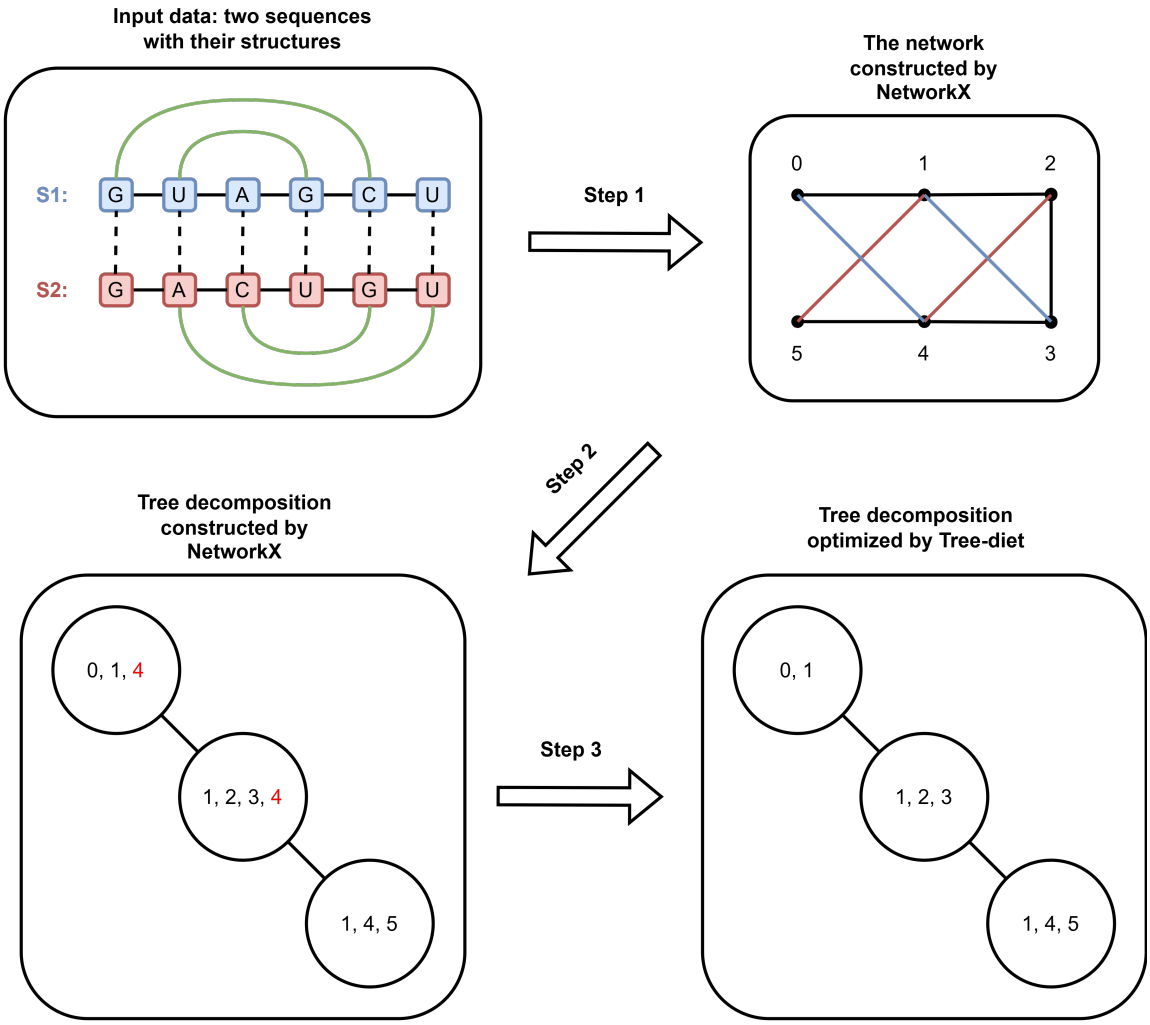

Figure 5.2: The figure presents an example of the three steps for preprocessing. In the last step, the vertices marked in red are picked and removed by Tree diet in order to achieve the desired treewidth (tw = 2 in this example).

The edges of the network in step (i) are based on the stacking relations between adjacent positions in the sequence, and the base pairing relations from two given structures. The vertices are simply representing the positions/sites.

We used the NetworkX package[18] to build the network and construct a tree de-

composition. To apply the Tree diet, the tree decomposition needs to be stored in a compatible format. We built a pipeline (which is included in our tool package) for the conversion. To be able to differentiate the edges for weighing the structure costs, the pipeline colours the edges of basepairs based on the structure they belong to (e.g. edges from structure one can be coloured blue, and edges from structure two can be coloured red, as in Figure 5.2).

Tree diet takes three parameters as input: the network on which the tree decomposition is based, the bags stored as a linked list (the first bag linked to an empty one as the root), and a desired treewidth. There is also an option to keep specified edges (they call them "backbones") in the network during the optimization (edge removal). The resulting tree decomposition from step (iii) is ready for use by our main algorithm.

5.2.3 Main Algorithm

Our proposed algorithm, which we have named `achARNement2.0`, draws inspiration from the method originally proposed by Tremblay-Savard et al.[43], but with a completely new cost calculation approach. Like its predecessor, our proposed algorithm consists of three main steps, as presented below.

Bottom-up Step

A postorder traversal is performed on the species tree as shown in Algorithm 1 down below, which is initially called on the root of the species tree. The goal is to fill up a 3-dimensional cost matrix for each node of the input phylogeny, representing each possible nucleotide assignment (a tuple of nucleotides) in each bag for each structure family. The bags obtained from the tree decomposition are available as a global variable in the form of a list, ordered from the leaves to the root. Each bag contains a tuple of integers, corresponding to positions in the RNA sequence. For example, for a bag containing the tuple (1, 5, 7), a possible nucleotide assignment could be (A, A, A), representing the nucleotides assigned to positions 1, 5 and 7 respectively. When Algorithm 1 reaches the leaves (see lines 1 to 7), it simply assigns a cost of 0 for the nucleotides that are present in the leaf sequences, and ∞ for all the other possible nucleotides. Otherwise, when we are dealing with an internal node (lines 8 to 13), we make a recursive call to the left and right children (postorder) to get the cost matrices for them. Then the cost matrix can be computed for the internal node for each structure family and each bag (one at a time).

The cost matrix calculation is handled by Algorithm 2, which finds the minimal cost for each possible nucleotide assignment in each family for each bag. This cost includes a substitution cost (from the children assignments) and a structure cost (minimum free energy). More specifically, to calculate the cost of each possible nucleotide assignment for a specific bag, we need to consider all possible nucleotide assignments in the left and right child (lines 1-2) to find the ones yielding the minimum cost. The

Algorithm 1: calculateCosts(n)
This method represents the bottom-up step, where we do a post-order traversal of the tree in order to calculate the most parsimonious costs for each possible nucleotide at every site.

input : A node **n** of the species tree with extant sequences at the leaves for the structure families studied.
output: A cost matrix **n.costMatrix** for every internal node of the tree for every structure family

```
1  if n is a leaf then
2      seq = a string that stores the nucleotides of the leaf sequence;
3      All cells in n.costMatrix are first initialized to ∞;
4      for every structure family, fam do
5          for every bag in the tree decomposition do
6              nucAssign = tuple([seq[pos] for pos in bag]);
7              n.costMatrix[fam][bag][nucAssign] = 0;
8  else
9      calculateCosts(n.leftChild);
10     calculateCosts(n.rightChild);
11     for every structure family, fam do
12         for every bag in the tree decomposition do
13             computeBagCost(n, fam, bag);
```

formula used by `achARNement2.0` to calculate the cost of a particular nucleotide assignment a_p at the tuple of positions p, based on a given nucleotide assignment from a child node ch_p for a structure family ss is presented in Equation 5.2 below:

$$c(ch_p) + s(ch_p, a_p) + W * (G * mfe(a_p, ss) + (1 - G) * mfe(a_p, \overline{ss})) \tag{5.2}$$

In that formula, the term $c(ch_p)$ represents the optimal cost, in the child node, that

Algorithm 2: computeBagCost(n, fam, bag)
This method calculates the cost of every possible nucleotide assignments and saves them into the cost matrix.

input : A node **n**, the structure family **fam** and a **bag**.
output: Filled out **n.costMatrix** for the family and bag considered.

```
1 for every possible assignment for the given bag, nucAssign do
2     for every possible assignment for the same bag on its child nodes,
       nucAssignChild do
3         costLeft = Equation 5.2 ;   // with ch_p ← nucAssignChild, a_p
           ← nucAssign and ss ← fam on the left child node
4         costRight = Equation 5.2 ; // with ch_p ← nucAssignChild, a_p
           ← nucAssign and ss ← fam on the right child node
5         Save the minimum values for costLeft and costRight in minLeft
           and minRight;
6     totalCost = minLeft + minRight;
7     n.costMatrix[fam][bag][nucAssign] = totalCost;
```

has been previously determined for assigning that specific set of nucleotides, at the positions designated by p. The function $s(ch_p, ap_p)$ quantifies the substitution cost of substituting every nucleotide from one nucleotide assignment to the other, and it is simply the sum of each individual substitution cost at a specific site. The substitution costs selected for this study are presented in Table 5.1, where the costs for transitions are lower than the transversion costs, which is representative of the fact that transitions occur more frequently during evolution. The term $mfe(a_p, ss)$ denotes the Minimum Free Energy, which is calculated using the ViennaRNA package (more details below). The MFE is calculated for the nucleotide assignment a_p in both the current structure (represented by ss) and the other structure (represented by $\overline{ss}$). The term G represents a percentage which is used to define the contribution of each MFE (from each structure family) in the total structure cost. This percentage

changes following a linear gradient depending on the depth in the phylogeny, where it will be 100% at the leaves and 50% at the root. Finally, a structure weight W is assigned to the whole structure cost before adding it to the substitution cost. This constant is used to normalize the MFE values so that they do not dominate the whole cost formula, because they can be within 3 orders of magnitude of the substitution costs. A weight of 0.001 was found to provide a good balance between substitution and structure costs after initial testing on smaller datasets, so this was selected as the default value. Note that the user is free to adjust this constant, if they want to give more or less weight to the structure cost.

	A	C	G	U
A	0	2	1	2
C	2	0	2	1
G	1	2	0	2
U	2	1	2	0

Table 5.1: Nucleotide substitution matrix.

To be more specific on the MFE calculation, in `achARNement2.0`, the energy of each bag assignment is estimated using the 'energy_of_structure()' method provided by the ViennaRNA package version 2.6 [28]. This method needs to be called on a full target structure and RNA sequence. Since we have access to the two full target structures, but not to the full RNA sequences when working on a specific bag, we needed to find a way to construct the RNA sequences for the MFE calculation. We first filled out the RNA sequence with wildcard symbols (\$), which are ignored by the MFE calculation method. Then, we replaced some of the wildcard symbols with the given nucleotide assignment at the positions of the bag. This allowed us to get

only the energy contribution of the nucleotide assignment for that particular bag.

Middle Step

The minimal cost matrices (one for each family), which were obtained during the bottom-up phase for both families, are linked at the root of the phylogenetic tree. Specifically, under the premise of an ancestral duplication event, it is necessary to establish a combined cost matrix at the root—integrating the cost matrices of both families—prior to determining the nucleotide selections (enumerating optimal sequences). Consequently, a merging procedure is implemented to generate this integrated cost matrix at the root. This is done using the approach of Algorithm 2 but by considering the two cost matrices (one for each structure family) as the left and right child of the root node.

Top-down Step

The goal of the top-down step is to enumerate all the optimal sequences by making selections that follow the principle of parsimony (options that result in the minimal cost). However, since the "options" are stored as assignments of nucleotides for each bag from the tree decomposition with their total costs instead of independent position costs, the way of enumerating and constructing the complete optimal sequences is more involved. By definition of the tree decomposition, for every edge (u, v) in E, there exists at least one bag that contains both u and v. This means that, as long as

the network is a connected graph, any vertex from the network must be included in at least two bags from the corresponding tree decomposition. In other words, every position in the original sequence will be covered by at least two bags from the tree decomposition. Making selections simply following the principle of parsimony for each bag independently will cause a series of nucleotide selections for the bags that might be conflicting on the "overlapped" positions (different selections of nucleotide on the same position). As a result, some, if not most, of the selections would not be transformable into complete sequences.

Instead, to efficiently enumerate the optimal sequences from the potential nucleotide assignments for the bags, we describe a "greedy" method that returns complete sequences by selecting nucleotides for each bag following the reverse order given by the tree decomposition. When multiple assignments (of nucleotides) are estimated to give the same minimal cost, a recursive call will be invoked for each of them. A sequence template (of equal length to the total size of the input sequences) is passed between recursive calls with fixed nucleotides at previously assigned sites and placeholder characters for positions waiting to be assigned. Starting from the root of the tree decomposition, the algorithm progresses by replacing the placeholders on the template with nucleotides that correspond to the minimal cost in the cost matrix for each bag. Consequently, following the assignment in one bag, the sequence template guides the nucleotide selections for subsequent bags. Consistent with the principles of tree decomposition, adjacent bags share at least one vertex (position), ensuring that the search space of possible nucleotide assignments for all unassigned adjacent bags is reduced following each assignment. For instance, in our case, the total number of

possible assignments for a bag is $|D|^k$ where $k \leq tw + 1$ (size of the largest bag), D being the alphabet of an RNA sequence. If a previous bag assigns a nucleotide, the domain of the current bag will be $1 \cdot |D|^{k-1}$.

Algorithm 3: selectNucleos(n)
The method shows how we enumerate the optimal sequences based on the cost matrices.

input : The root of the phylogeny, invoked using **n**.
output: Saves a list of optimal sequences for every structure family, for all internal nodes (including the root) of the tree.

```
1 for for every structure family, fam do
2     seqTemplate = a list of placeholder characters for the sequence length;
3     optimalSeqs = a pair initialized to (∞, [empty list]);
4     n.selectNucleos4Bag(fam, firstBag, seqTemplate, accCost = 0,
        optimalSeqs);
5     sequenceList = optimalSeqs[1];
6     Save sequenceList for fam in the root node;
7 selectNucleosChildren(n);
```

The whole process of the top-down step starts with a call to Algorithm 3 on the root of the phylogeny. At the end of Algorithm 3, once all the optimal sequences have been found and saved for the root node, the 'selectNucleosChildren' method (see Algorithm 4) is called to repeat the same process for each descendant node in the tree. The only difference is that for any internal node that is not the root, the optimal sequences of the parent influence the choice of optimal sequences for its children. Indeed, we need to identify the optimal sequences in the children that made the optimal cost possible for the parent sequences. Both of these algorithms make use of the 'selectNucleos4Bag' method (see Algorithm 5) to assign nucleotides for each bag following the order given by the tree decomposition. With a sequence template

Algorithm 4: selectNucleosChildren(n)
It is part of the top-down step of the algorithms, invoked for each node to select nucleotides.

input : A node **n** from the species tree, for which we have a list of optimal sequences.
output: Saves a list of optimal sequences for every structure family of the child nodes.

```
1  if n.leftChild and n.rightChild are leaves then
2      return ;                    // No more selection needed, task done.
3  for every structure family, fam do
4      for every optimal sequence for fam, parentSeq do
5          assignedBags = all assignments of nucleotides corresponding to
            bags from parentSeq;
6          seqTemplate = a list of placeholder characters for the seq. length;
7          optimalSeqsLeft = a pair initialized to (∞, [empty list]);
8          n.selectNucleos4Bag(fam, bag, seqTemplate, accCost = 0,
            optimalSeqsLeft, assignedBags, n.leftChild);
9          seqTemplate = a list of placeholder characters for the seq. length;
10         optimalSeqsRight = a pair initialized to (∞, [empty list]);
11         n.selectNucleos4Bag(fam, bag, seqTemplate, accCost = 0,
            optimalSeqsRight, assignedBags, n.rightChild);
12         if the left child is not a leaf then
13             Append optimalSeqsLeft[1] to the left child;
14         if the right child is not a leaf then
15             Append optimalSeqsRight[1] to the right child;
16 if the left child is not a leaf then
17     selectNucleosChildren(n.leftChild);
18 if the right child is not a leaf then
19     selectNucleosChildren(n.rightChild);
```

that gets gradually filled throughout the process, after the assignment of the last bag, complete sequences with their costs are returned. These optimal sequences (one set for each structure family) are stored in each node of the phylogeny.

5.2.4 Software Implementation

We implemented a software based on the algorithm described above in Python, which is available in a software package named `achARNement2.0`, freely on GitHub. Besides our new algorithm, `achARNement2.0` also includes the implementation of `Fitch`, `Sankoff`, as well as `achARNement1.0` by Tremblay-Savard *et al.* [43]. To prepare the optimized tree decomposition, we built a pipeline using the popular toolkit *NetworkX* for building the tree decomposition and *Tree diet* for tree optimization. We used tools from the *ViennaRNA* package for the MFE (structure cost) estimation.

Given structures from two homologous ncRNA families and aligned sequences from each of the species (that ensured to possess one copy of each family) included in the given phylogeny, `achARNement2.0` is capable of reconstructing potential sequences of all ancestors (internal nodes including the root) that optimized to our cost/penalty system. For time efficiency, `achARNement2.0` has the option to reconstruct the root node only. The weight given to structures is also adjustable.

Algorithm 5: selectNucleos4Bag(fam, bag, seqTemplate, accCost, optSeqs, assignedBags[Optional]), childNode[Optional]
This method, called from the node **n**, recursively selects nucleotides for each bag.

input : A structure family, **fam**; a **bag**; a **seqTemplate**; **accCost** which accumulates the cost for each bag assignment; **optSeqs**, a tuple (x,y) where x is the cost, y is a list of optimal sequences; **assignedBags**, a list of bag assignments derived from the optimal sequences of the direct ancestor; **childNode**, either left or right child node of **n**.
output: An updated tuple **optSeqs**, representing the optimals sequence(s) constructed and their cost.

```
1  Search positions of bag with fixed nucleotides in the seqTemplate;
2  assignOptions = All assignments of the bag with fixed nucleotide(s) at
     positions of the seqTemplate.;
3  currentDict = empty dictionary ;
4  optAssignList = empty list ;
5  minCost = ∞;
6  for every nucAssign in assignOptions do
7      if assignedBags is None then
8          costNucleos = n.costMatrix[fam][bag][nucAssign];
9      else
10         costNucleos = Equation 5.2 ;  // with ch_p ← nucAssign, a_p ←
             assignedBags[bag] and ss ← fam on childNode
11     if costNucleos < minCost then
12         optAssignList = [nucAssign];
13         minCost = costNucleos;
14     else if costNucleos == minCost then
15         optAssignList += nucAssign;
16     newCost = accCost + minCost;
   // Continued on the next page.
```

```
17 seqTemplateList = a list of new templates created by filling the assignments
     from the optAssignList into the seqTemplate received as a parameter;
18 if not the last bag then
19     for template in seqTemplateList do
20         candidateSeqs = a pair initialized to (∞, [empty list]);
21         if assignedBags is None then
22             n.selectNucleos4Bag(fam, nextBag, template, newCost,
                candidateSeqs);
23         else
24             n.selectNucleos4Bag(fam, nextBag, template, newCost,
                candidateSeqs, assignedBags, childNode);
25         if candidateSeqs[0] < optSeqs[0] then
26             optSeqs = candidateSeqs;
27         else if candidateSeqs[0] == optSeqs[0] then
28             optSeqs[1] += candidateSeqs[1];
29 else
30     optSeqs[0] = newCost;
31     optSeqs[1] = seqTemplates;
```

Chapter 6

Results and Evaluation

6.1 Experiments on Simulated Data

6.1.1 Data Generator

For the simulated dataset, we used the data generator provided with `achARNement1.0`, which is described briefly below. We also reused the three randomly designed structures of length 100 from `achARNement1.0` (with two similar structures and one that differs greatly from the other two). Using these structures, we can make three structure pairs (for family 0, 1, and 2, we have family pairs 01, 02, and 12) for testing.

For each of these pairs of structures, we generated 100 bi-stable sequences with *Frankenstein* (see section 2.2.5), a genetic algorithm approach for solving the inverse

folding problem. These resulting sequences are then used as "seeds" to populate the root ancestor of each of the 100 generated phylogenies for each structure pair. The generated phylogenies were all complete binary trees of depth 6 (with 64 leaves). For each generated phylogeny, the sequences in the internal nodes, as well as the leaves, are generated in a top-down manner. We used five different mutation rates (1%, 5%, 10%, 15%, and 20%) in our tests. The mutation rates are applied to both parent sequences (one from each family) between each level in the tree (e.g. 5% difference between any internal node and its direct children) in a process that is repeated a thousand times to generate a pool of possible descendant sequences. From this pool of sequences, the actual child sequences (one for each family) are sampled according to their MFE for each structure. In other words, mutated sequences from the generated pool that can fold better into the target structure have a higher chance of being selected. In the end, a total of 1500 such trees (100 seeds, three structure pairs, five mutation rates) are generated for the evaluation of `achARNement2.0`.

For comparison, we analyzed the simulated dataset with four algorithms: `Fitch`, `Sankoff`, `achARNement1.0`, and `achARNement2.0`. Using multiple mutation rates as the independent variable in our tests allowed us to evaluate how the performance of these different methods is affected by having an increasing sequence divergence at the leaves, which can severely affect the amount of signal that can be obtained from them.

6.1.2 Results

All the tests were conducted on Ubuntu 22.04.5 using identical hardware: a 20-core Raptor Lake CPU running at 4.5 GHz, with 64 GiB of memory, ensuring a direct comparison of the runtime performance of the four algorithms. For all the test, we bound the treewidth to 2 (*i.e.* largest bag of size 3) using Tree diet. Moreover, all the data points presented below represent an average of 100 replicates.

The results in terms of error percentages for all the reconstructed sequences at all internal nodes of the phylogenies are presented in Figure 6.1. In the figure, `achARNement2.0` shows an improvement in terms of accuracy by exhibiting a lower error percentage over the other three algorithms, especially for family pair 01. We believe that it is due to the high similarity between structure 0 and 1. Also, from the figure, the error percentage of `achARNement2.0` is considerably lower for structured (paired) regions compared to unstructured regions (about 15% max for structured and 22% for unstructured). This meets our expectations based on the fact that our model takes advantage of more structural information compared to the other three algorithms. Another interesting detail is that, in the unstructured positions, `achARNement2.0` outperformed the other algorithms even more. It might be related to the extra base stacking information considered by our approach. `Fitch`, `Sankoff`, and `achARNement1.0` tend to have larger solution spaces for optimal sequences than `achARNement2.0`. As a result, when the mutation rate exceeds 10%, `Fitch` and `Sankoff` could not finish generating the result sequences. `achARNement1.0`, however, finished most of the instances of a 15% mutation rate (except for four). For

`achARNement2.0`, we used the extra structure information as a constraint, which helped eliminate an enormous number of potential sequences (as shown in Figure 6.2 below), and made it possible to significantly reduce the solution space and produce results even for a mutation rate as high as 20%.

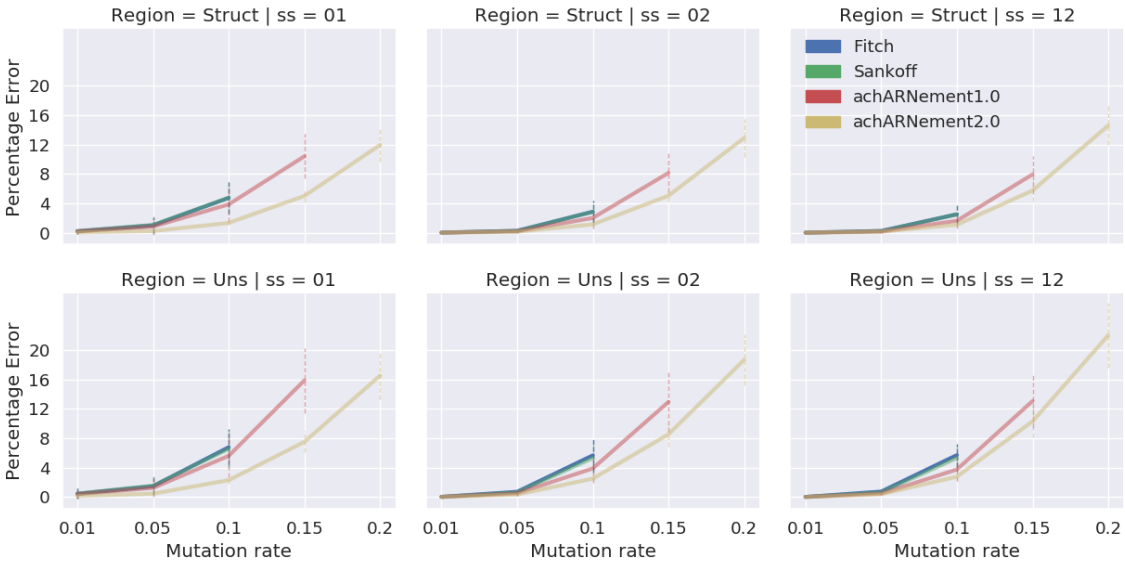

Figure 6.1: The average error percentage of all optimal sequences for both families in a tree. Each column represents a pair of secondary structures. Figures in the top row show error percentages for the structured regions of the sequences. Figures in the bottom row illustrate error percentages for the unstructured regions.

Besides the error percentage, `achARNement2.0` showed a significant improvement with regards to the total number of reconstructed sequences at the root (see Figure 6.2). As mentioned above, by utilizing more structural information, `achARNement2.0` eliminates a massive amount of potential sequences. In fact, the average number of inferred sequences by our new algorithm is several orders of magnitude lower than previous approaches for most mutation rates. It is important to mention that severe non-uniqueness of optimal solutions is a well-documented challenge of ancestral

reconstruction methods [38, 37, 50, 49]. It is hard for biologists who are interested in analyzing the composition of ancestral sequences to deal with software that produce hundreds of thousands or even millions of equally probable sequences. Being able to significantly reduce the number of solutions inferred with `achARNement2.0` is therefore a crucial achievement.

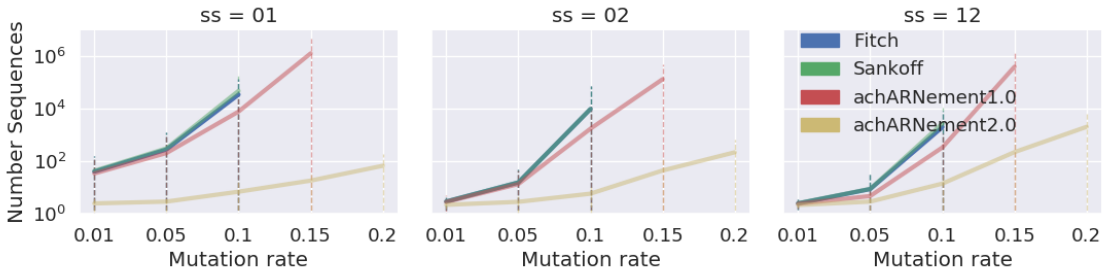

Figure 6.2: Average number of optimal sequences for the root. Each column represents a different pair of secondary structures with five mutation rates: 1% to 20%. Each column shows the number of sequences each algorithm inferred for a given family pair (01, 02, and 12).

Figure 6.3 illustrates the average runtime and standard error across different mutation rates for each algorithm. According to the results, `achARNement2.0` exhibits slower performance compared to the other three algorithms, particularly at lower mutation rates. However, its performance remains highly stable, even at the highest tested mutation rate (20 percent). In contrast, the other three algorithms demonstrate considerable sensitivity to mutation rates, which prevents them from successfully completing instances with high mutation rates in a reasonable amount of time.

Upon conducting an in-depth analysis, we made pairwise comparisons between the output sequences produced by the four distinct algorithms for each mutation rate.

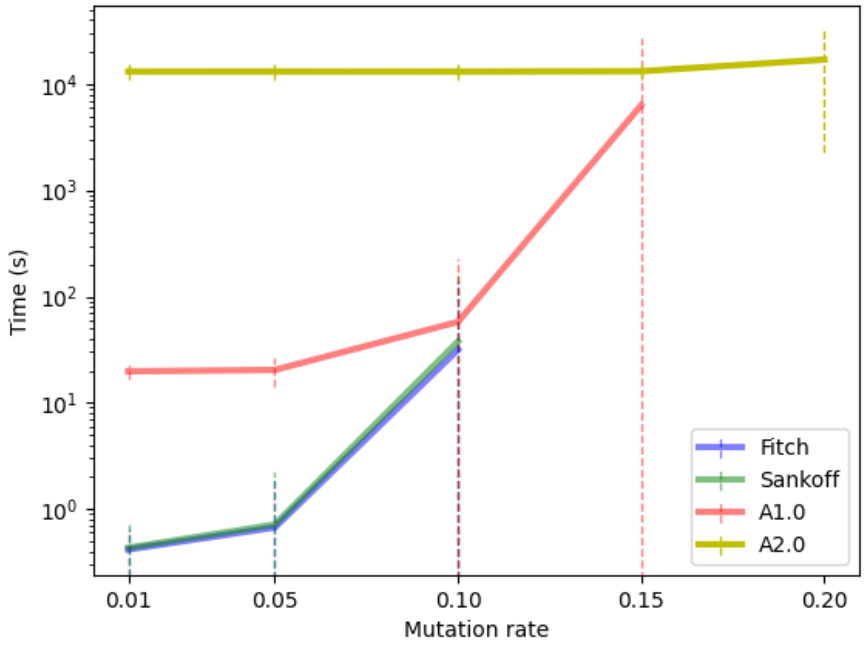

Figure 6.3: The comparison of running times between the four algorithms. The x-axis shows the five mutation rates that we consider (1%, 5%, 10%, 15%, and 20%). The y-axis indicates the running time in seconds (logscale).

It is noteworthy that no method generated the exact same set of output sequences (see Figure 6.4). None of the sets are entirely subsumed by another. In our comparisons, we were interested in verifying how the ancestral sequences inferred at the root of the phylogeny by one method were intersecting with the ones inferred by other methods. The analysis indicates a correlation between lower mutation rates and higher inclusion levels. This is likely because, at lower mutation rates, the extant sequences at the leaves bear greater structural and sequential resemblance to their progenitor, so most methods will have small and similar solution spaces. Observing the inclusion percentage at a mutation rate of 0.1, a significant majority of

the optimal sequences generated by `achARNement2.0` are encompassed within those produced by either `Sankoff` or `Fitch` (over 95 percent). On the other hand, the intersection of the results from `achARNement2.0` and `achARNement1.0`, relative to the total output from `achARNement2.0`, stands at approximately 63 percent for the same mutation rate. Notably, the divergence in output sequences intensifies at a mutation rate of 0.15, reducing the intersection to just under 10 percent. Despite `Fitch` and `Sankoff` generating a comparable number of sequences at a mutation rate of 0.1, their outputs are considerably divergent (approximately 75 percent). Compared to `achARNement1.0`, `achARNement2.0` shows a substantially higher inclusion rate against outputs from `Fitch` and `Sankoff`, even surpassing the inclusion rate observed between `Fitch` and `Sankoff` themselves. It is highly plausible that the majority of optimal sequences reconstructed by `achARNement2.0` are contained within the intersection of sequences from `Fitch` and `Sankoff`.

We also evaluated the MFE, for both structures, of each ancestral sequence inferred by the different methods. To represent the ability of the sequences to fold into both structures, we calculated the harmonic mean of both MFE values. In the last three columns of Figure 6.4, we report the minimum, average and maximum harmonic means of MFE for inferred sequences exclusive to the base algorithm (exc. base), common to both algorithms (common) and exclusive to the other algorithm (exc. other). The table shows that the average harmonic mean of sequences produced by `Fitch`, `Sankoff`, and `achARNement1.0` only (not present in the intersections with `achARNement2.0`) are worse than both the averages of sequences exclusive to `achARNement2.0` and the ones contained in the intersection. The only exception is

mut	algo_base	algo_other	# of seqs (base)	# of seqs (other)	common	Inclu/seq	HM (exc.base)	HM (common)	HM (exc.other)
0.01	A2	A1	333	1942	318	95.50%	-19.15, -13.19, -9.77	-25.25, -16.07, -6.57	-20.97, -11.29, -5.54
0.05	A2	A1	423	11089	363	85.82%	-23.32, -15.25, -8.37	-25.25, -16.4, -6.79	-25.65, -12.91, -4.11
0.1	A2	A1	1315	496510	833	63.35%	-23.39, -16.4, -4.95	-25.42, -16.04, -7.07	-24.92, -11.79, -0.48
0.15	A2	A1	14078	10627280	1354	9.62%	-24.01, -14.57, -3.88	-23.17, -16.86, -8.48	-24.41, -11.69, 60.59
0.01	A2	Sankoff	333	2395	332	99.70%	-9.77, -9.77, -9.77	-25.25, -15.96, -6.57	-21.05, -11.05, -5.54
0.05	A2	Sankoff	423	16530	422	99.76%	-15.96, -15.96, -15.96	-25.25, -16.24, -6.79	-25.65, -12.5, -4.11
0.1	A2	Sankoff	1315	1732020	1272	96.73%	-20.67, -13.37, -8.95	-25.42, -16.26, -4.95	-24.92, -11.35, 0.65
0.01	A2	Fitch	333	2256	333	100.00%	None, None, None	-25.25, -15.94, -6.57	-21.05, -10.99, -5.54
0.05	A2	Fitch	423	15152	423	100.00%	None, None, None	-25.25, -16.24, -6.79	-25.65, -12.72, -4.11
0.1	A2	Fitch	1315	1570976	1262	95.97%	-20.67, -13.88, -10.38	-25.42, -16.26, -4.95	-24.92, -11.34, 0.31
0.01	A1	Sankoff	1942	2395	1942	100.00%	None, None, None	-25.25, -12.07, -5.54	-21.05, -10.25, -5.81
0.05	A1	Sankoff	11089	16530	11089	100.00%	None, None, None	-25.65, -13.02, -4.11	-23.32, -11.73, -4.77
0.1	A1	Sankoff	496510	1732020	359810	72.47%	-21.38, -11.89, -2.61	-25.42, -11.76, -0.48	-24.56, -11.25, 0.65
0.01	A1	Fitch	1942	2256	1852	95.37%	-16.91, -12.61, -6.92	-25.25, -12.05, -5.54	-21.05, -10.2, -6.06
0.05	A1	Fitch	11089	15152	10619	95.76%	-22.36, -12.6, -6.28	-25.65, -13.04, -4.11	-23.32, -12.29, -4.77
0.1	A1	Fitch	496510	1570976	330404	66.55%	-24.12, -11.79, -2.61	-25.42, -11.79, -0.48	-24.56, -11.23, 0.31
0.01	Fitch	Sankoff	2256	2395	2251	99.78%	-10.11, -9.83, -9.5	-25.25, -11.72, -5.54	-16.91, -11.84, -5.81
0.05	Fitch	Sankoff	15152	16530	15059	99.39%	-18.97, -12.11, -4.77	-25.65, -12.82, -4.11	-22.36, -10.32, -4.77
0.1	Fitch	Sankoff	1570976	1732020	1179192	75.06%	-21.85, -10.29, 0.0	-25.42, -11.7, 0.31	-24.12, -10.63, 0.65

Figure 6.4: The figure illustrates the inclusion relationship and harmonic mean of the minimum free energy (MFE) values for sequences generated by the four algorithms using a simulated dataset. The first column shows the mutation rate of the instance. The second column shows the name of the "base" algorithm (A1 refers to `achARNement1.0`, A2 for `achARNement2.0`). The third column shows the name of the other algorithm to which we compare the "base." "Inclu/seq" is the average inclusion percentage for each sequence from `achARNement2.0`. The final trio of columns illustrates the harmonic means of minimum free energy (MFE) in both structures as estimated from sequences exclusive to the base algorithm, sequences common to both algorithms, and sequences exclusive to the other algorithm. Each of these columns contain the minimum, average, and maximum values of harmonic mean.

for the comparison with Sankoff for the 1% mutation rate, for which `achARNement2.0` produced only one sequence that was exclusive and had a lower harmomic mean of MFE. These results show the capability of `achARNement2.0` of reconstructing most of the sequences with the ability to fold well into the two structures while getting rid of the worst sequences inferred (exclusively) by other methods.

6.2 Experiments on Biological Data

6.2.1 Biological Datasets

We also assessed the four methods on the Glm and FinP-traJ clans (which were used in [43]) from Rfam. After we retrieved the sequences and structures of the two families from Rfam, we intersected the genome IDs of these sequences to get the set of common genomes (all bacterial genomes). A bacterial genome tree based on these organisms was taken from BVBRC[32]. To prepare the structures and sequences for the reconstruction approach, we aligned them on their seed alignments using CARNA[41]. Finally, all gapped positions in the resulting alignment were removed since only point mutations are allowed in our model.

6.2.2 Results for the Glm Clan

The Glm clan includes two bacterial small non-coding RNAs, GlmY and GlmZ, which are similar in structure yet perform different functions. These RNAs function hierarchically where GlmY acts upstream and indirectly influences glmS mRNA translation by affecting GlmZ, which directly interacts with glmS mRNA to enhance its translation[44]. For our test, we reused the 74 bacterial genomes, their alignments, as well as the corresponding phylogeny analyzed by Tremblay-Savard et al.[43].

For the root node, `Fitch` and `Sankoff` produced the same 786432 sequences. For

`achARNement1.0`, it produced 196608 ancestral sequences. In contrast, `achARNement2.0` produced only 256 ancestral sequences, which is several magnitudes fewer than the other three algorithms.

In terms of runtime, `achARNement2.0` took roughly 2 hours to complete the analysis of the GLM clan. In comparison, `achARNement1.0` `Fitch` and `Sankoff` took respectively 28 seconds, 12 seconds and 12 seconds to complete.

6.2.3 Results for the FinP-traJ Clan

The FinP-traJ clan in the Rfam database consists of RNA families that include FinP RNA and traJ mRNA, which are involved in a regulatory system affecting bacterial conjugation, specifically in the F-plasmid of *Escherichia coli* and related bacteria. This system plays a crucial role in controlling bacterial fertility through interaction between these two RNA components. Like the Glm clan, we used the same dataset that was used in the study of `achARNement1.0`, consisting of 54 genomes and the corresponding phylogeny. Compared to the two families in the Glm clan, the ones from FinP-traJ are more varied in structure.

For the three algorithms that we included for comparison, the results are as follows: `Fitch` and `Sankoff` inferred the same 12582912 sequences at the root, while the `achARNement1.0` inferred fewer sequences with a total of 3145728. As for `achARNement2.0`, only 128 sequences were inferred, which is significantly lower than previous approaches.

In terms of runtime, `achARNement2.0` took roughly 33 minutes to complete the analysis of the FinP-traj clan. In comparison, `achARNement1.0` `Fitch` and `Sankoff` took respectively 28 seconds, 151 seconds and 167 seconds to complete.

6.2.4 Comparison of Reconstructed Ancestors

Figure 6.5 depicts the results derived from biological datasets regarding the inclusion of ancestral sequences produced by different methods. It is observed that the average harmonic mean of energy across both structures for sequences reconstructed by `achARNement2.0` is significantly lower than that of `achARNement1.0`. This suggests that `achARNement2.0` is more effective in reconstructing sequences that align well with both structures, thereby indicating its superior fitness. Similarly to the analysis performed on simulated data, the intersection of output sequences from the various algorithms is examined for the GLM and FinP-traJ clans. Notably, all sequences reconstructed by `achARNement2.0` at the roots are subsets of the optimal sequences determined by `achARNement1.0`, `Sankoff`, and `Fitch`. This outcome demonstrates that `achARNement2.0` is capable of eliminating a greater number of candidate sequences, benefiting from the newly implemented method for estimating structure costs.

clan	algo_base	algo_other	# of seqs (base)	# of seqs (other)	common	inclusion	HM (exc.base)	HM (common)	HM (exc.other)
GLM	A2	A1	256	196608	256	100.00%	None, None, None	-10.02, -10.02, -10.02	-10.03, -8.67, -7.77
GLM	A2	Sankoff	256	786432	256	100.00%	None, None, None	-10.02, -10.02, -10.02	-10.03, -8.31, -7.06
GLM	A2	Fitch	256	786432	256	100.00%	None, None, None	-10.02, -10.02, -10.02	-10.03, -8.31, -7.06
GLM	A1	Sankoff	196608	786432	196608	100.00%	None, None, None	-10.03, -8.67, -7.77	-9.73, -8.19, -7.06
GLM	A1	Fitch	196608	786432	196608	100.00%	None, None, None	-10.03, -8.67, -7.77	-9.73, -8.19, -7.06
GLM	Sankoff	Fitch	786432	786432	786432	100.00%	None, None, None	-10.03, -8.31, -7.06	None, None, None
FinP	A2	A1	128	3145728	128	100.00%	None, None, None	-12.65, -12.64, -12.62	-13.01, -10.49, -7.45
FinP	A2	Sankoff	128	12582912	128	100.00%	None, None, None	-12.65, -12.64, -12.62	-13.01, -10.06, -6.31
FinP	A2	Fitch	128	12582912	128	100.00%	None, None, None	-12.65, -12.64, -12.62	-13.01, -10.06, -6.31
FinP	A1	Sankoff	3145728	12582912	3145728	100.00%	None, None, None	-13.01, -10.49, -7.45	-12.43, -9.92, -6.31
FinP	A1	Fitch	3145728	12582912	3145728	100.00%	None, None, None	-13.01, -10.49, -7.45	-12.43, -9.92, -6.31
FinP	Sankoff	Fitch	12582912	12582912	12582912	100.00%	None, None, None	-13.01, -10.06, -6.31	None, None, None

Figure 6.5: The figure presents the outcomes derived from `achARNement1.0` and `achARNement2.0` using biological data pertaining to the GLM and FinP-traJ clan. The initial column lists the clan's name. The subsequent two columns denote the algorithms termed "base" and "other." The seventh column indicates the proportion of sequences from the "base" algorithm that are detectable within the outcomes of the "other" algorithm. The last three columns show the harmonic means estimated from base-exclusive sequences, common sequences, and sequences exclusive to the other algorithm (the three values are minimum, average, and maximum, respectively).

Chapter 7

Conclusion

7.1 Summary

In this study, we focused on developing an improved ancestral sequence reconstruction approach based on the `achARNement1.0` methodology proposed by Tremblay-Savard et al.[43]. Accordingly, an improved approach (`achARNement2.0`) is presented, which resulted in a significantly lower error rate on simulated data and a greatly reduced solution space both on simulated and real data. These improvements are achieved by allowing the algorithm to consider more structural information compared to `achARNement1.0`. More specifically, we developed a method inspired by the tree decomposition and dynamic programming that solves the problem of "interdependency" (complexity induced by dependencies among secondary structures) that

achARNement1.0 avoided for better time efficiency. A greedy approach is also presented to enumerate the resulting optimal sequences from potential bag assignments.

The results from the algorithms are also compared and assessed. The application of the new approach to structure cost estimation demonstrated its effectiveness on both simulated and biological datasets. For both simulated and real datasets, it showed a considerable improvement in reducing the solution space of the reconstruction, producing a set that is at times several orders of magnitude smaller than that of the other three algorithms (Fitch, Sankoff, and achARNement1.0).

On simulated data, results in Figure 6.2 show that the proposed algorithm consistently outperformed the other three algorithms in terms of error percentage, especially for higher mutation rates. An even more interesting observation is that a majority of the sequences inferred by achARNement2.0 can be found in the results of the other three algorithms, suggesting that the inferred ancestral sequences are as reliable as those of the other three algorithms.

Results on the biological data can not be analyzed based on error percentage as we do not have the guaranteed ancestral sequences as answer keys (unlike the simulated data). However, there are some noteworthy observations in Figure 6.5. Similarly to the experiments on simulated datasets, achARNement2.0 showed a significant improvement in reducing the set of optimal ancestral sequences. Meanwhile, the harmonic mean of MFE of the output sequences shows that the sequences inferred by achARNement2.0 are more capable of folding into both structures than solutions produced by Fitch, Sankoff, and achARNement1.0.

The runtimes of `achARNement2.0` obtained for all the tests were longer than the ones from the other three methods used for comparison. However, this was compensated by significantly lower error rates and smaller sizes of the reconstructed ancestor sets, which are of great importance for the applicability of the method.

7.2 Limitations

One important limitation of `achARNement2.0`, which was also shared by its predecessor, is that this approach can only consider point mutations/substitutions in the sequences used as input (no insertions and deletions are possible). Accordingly, the input sequences and structures need to be of equal length (gaps obtained after sequence alignment are not allowed in the sequences). On the one hand, this helped us to focus on developing this improved cost calculation approach without the need to consider the complexity introduced by indels. On the other hand, this brought specific challenges in the preparation of the input data. More specifically, when dealing with divergent ncRNA structures and sequences, which can complicate the alignment process, we might need to remove an excessive number of gaps in the end. This not only shortens the sequences but also removes a lot of information that could be otherwise useful for the inference of the ancestors. For example, we attempted to test our approach on a third clan from the Rfam database: the SAM clan (on Rfam: CL00012). However, after removing the gapped positions in their secondary structures after alignment, only a limited (insufficient) number of positions were left

and most of the important structural elements had been cut. Moreover, indels play a crucial role in evolutionary processes, as they can introduce or remove nucleotides, thereby altering the RNA structure and function. Consequently, ignoring them entirely can compromise the accuracy of evolutionary analyses.

Another limitation of the current algorithm is its potential vulnerability to local minima. To be more specific, the nucleotide selection (top-down) step always starts to make selections for the bags from the "root bag" of the tree decomposition. Based on our design, it always selects the optimal assignment(s) for the first bag before moving on to the next. In theory, however, there could be situations where global optimal sequences have a sub-optimal assignment(s) for individual bags, which would not be considered by our approach.

7.3 Future Work

While the results of this study are promising, further development is needed to enhance the robustness and applicability of the approach. One potential direction is to incorporate insertions and deletions (indels) in the sequence reconstruction process. Indels are prevalent in genomic sequences, and their inclusion could provide valuable insights into sequence variation and evolution. Accounting for indels would also eliminate the requirement for input sequences or structures to be of equal length, which would greatly simplify the preprocessing tasks, such as sequence alignment and gap removal. Moreover, this adaptation would enable the model to handle more

structurally diverse families, broadening its applicability to a wider array of genetic data.

Our current analysis has been restricted to two Rfam clans. Incorporating insertions and deletions (indels) could broaden our evaluation to include additional clans such as *SAM* and *ykkc*, which is crucial for assessing the model's generalizability and reliability. Additionally, addressing the issue of local minima, as noted in the limitations section, could be interesting. The tendency for sequences to converge on local minima could potentially lead to a biased outcome. Implementing methods such as beam search could help overcome this problem by exploring a wider range of solutions, thus enhancing the quality of the reconstructed sequences. Furthermore, the ability of `achARNement2.0` to generate a compact set of ancestral sequences could be leveraged by adopting a strategy that retains top-ranked sequences, not just those with the minimal score. This approach would likely improve the model's flexibility and accuracy in reflecting sequence diversity.

As for scalability, it would be very interesting to adapt the model to more than two structure families (e.g. three structures). Potentially, this could lead to an even better solution reduction and a smaller error percentage. The support of indels mentioned earlier could be very helpful for the data preprocessing since the three structures considered might share a smaller portion of identical structural regions compared to two structures. Enabling indels means that the gapped positions can be kept; hence, we could have longer sequences for the input. Also, tree decomposition can be very effective for this setting since we can have more complicated relation

networks for three structures compared to two structures (see Figure 7.1).

Finally, `achARNement2.0` could be helpful in the field of RNA sequence design. Our approach is designed to infer ancestral ncRNA sequences based on the sequences of the predecessors. By considering both sequential and structural constraints, the ancestral sequences are designed to be efficient in folding into multiple functional structures. If we consider this problem from the aspect of RNA design, we can see the predecessors as sequences carrying the desired structures to be conserved. Accordingly, the ancestor at the root will be the sequences that could fold into these target structures. Moreover, using `achARNement2.0`, the preference of the sequence to fold into one of the structures can be controlled by the gradient, which gives it more flexibility in designing the sequences.

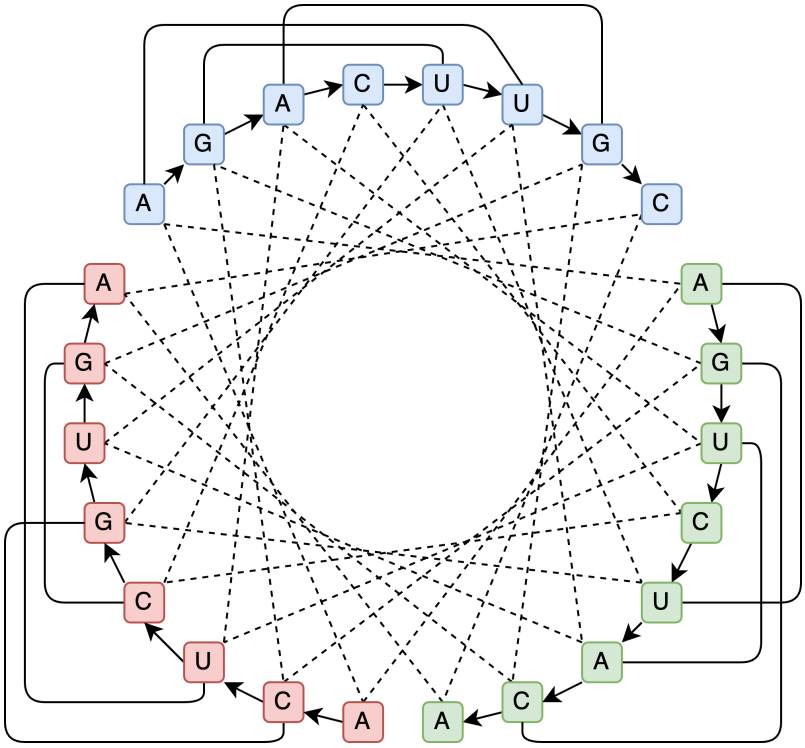

Figure 7.1: Simplified example of 3-structure dependencies. Sequence one is in blue; sequence two is in green; sequence three is in red. Consecutive positions of a sequence are connected by arrows while rounded/curved lines connect the basepairs. The same positions in all three sequences are connected by dotted lines.

Bibliography

[1] Jochen Alber, Frederic Dorn, and Rolf Niedermeier. "Experimental evaluation of a tree decomposition-based algorithm for vertex cover on planar graphs". In: *Discrete Applied Mathematics* 145.2 (2005), pp. 219–231.

[2] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. "Complexity of Finding Embeddings in a k-Tree". In: *SIAM Journal on Algebraic Discrete Methods* 8.2 (Apr. 1987), pp. 277–284. ISSN: 0196-5212. DOI: 10.1137/0608024. URL: https://doi.org/10.1137/0608024.

[3] James A Birchler and Hua Yang. "The multiple fates of gene duplications: deletion, hypofunctionalization, subfunctionalization, neofunctionalization, dosage balance constraints, and neutral variation". In: *The Plant Cell* 34.7 (2022), pp. 2466–2474.

[4] Hans L. Bodlaender. "Discovering Treewidth". In: *SOFSEM 2005: Theory and Practice of Computer Science*. Ed. by Peter Vojtáš et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1–16. ISBN: 978-3-540-30577-4.

[5] Hans L. Bodlaender and Ton Kloks. "Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs". In: *Journal of Algorithms* 21.2 (Sept. 1996), pp. 358–402. ISSN: 0196-6774. URL: https://www.sciencedirect.com/science/article/pii/S0196677496900498.

[6] Guillaume Bourque and Pavel A Pevzner. "Genome-scale evolution: reconstructing gene orders in the ancestral species". In: *Genome research* 12.1 (2002), pp. 26–36.

[7] Robert K. Bradley and Ian Holmes. "Evolutionary Triplet Models of Structured RNA". In: *PLOS Computational Biology* 5.8 (Aug. 2009), e1000483. DOI: 10.1371/journal.pcbi.1000483. URL: https://doi.org/10.1371/journal.pcbi.1000483.

[8] Benjamin A Burton and William Pettersson. "Fixed parameter tractable algorithms in combinatorial topology". In: *International Computing and Combinatorics Conference*. Springer. 2014, pp. 300–311.

[9] José C. Clemente et al. "Optimized ancestral state reconstruction using Sankoff parsimony". In: *BMC Bioinformatics* 10.1 (Feb. 2009), p. 51. ISSN: 1471-2105. DOI: 10.1186/1471-2105-10-51. URL: https://doi.org/10.1186/1471-2105-10-51.

[10] Marek Cygan et al. *Parameterized algorithms*. Vol. 5. 4. Springer, 2015.

[11] Sean R. Eddy and Richard Durbin. "RNA sequence analysis using covariance models". In: *Nucleic Acids Research* 22.11 (June 1994), pp. 2079–2088. ISSN:

0305-1048. DOI: 10.1093/nar/22.11.2079. URL: https://doi.org/10.1093/nar/22.11.2079.

[12] EMBL-EBI. *Glossary: Clan*. URL: https://docs.rfam.org/en/latest/glossary.html. (accessed: 04.24.2021).

[13] Katharine G Field et al. "Molecular phylogeny of the animal kingdom". In: *Science* 239.4841 (1988), pp. 748–753.

[14] Walter M. Fitch. "Toward Defining the Course of Evolution: Minimum Change for a Specific Tree Topology". In: *Systematic Biology* 20.4 (Dec. 1971), pp. 406–416. ISSN: 1063-5157. DOI: 10.1093/sysbio/20.4.406. eprint: https://academic.oup.com/sysbio/article-pdf/20/4/406/4697394/20-4-406.pdf. URL: https://doi.org/10.1093/sysbio/20.4.406.

[15] Christoph Flamm et al. "Design of multistable RNA molecules". In: *Rna* 7.2 (2001), pp. 254–265.

[16] Fedor V Fomin and Dimtirios M Thilikos. "A simple and fast approach for solving problems on planar graphs". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 2004, pp. 56–67.

[17] Sam Griffiths-Jones et al. "Rfam: an RNA family database". In: *Nucleic acids research* 31.1 (2003), pp. 439–441.

[18] Aric Hagberg and Drew Conway. "Networkx: Network analysis with python". In: *URL: https://networkx. github. io* (2020).

[19] Stefan Hammer et al. "Fixed-parameter tractable sampling for RNA design with multiple target structures". In: *BMC bioinformatics* 20.1 (2019), pp. 1–13.

[20] Marc P. Hoeppner, Paul P. Gardner, and Anthony M. Poole. "Comparative Analysis of RNA Families Reveals Distinct Repertoires for Each Domain of Life". In: *PLOS Computational Biology* 8.11 (Nov. 2012), e1002752. DOI: 10.1371/journal.pcbi.1002752. URL: https://doi.org/10.1371/journal.pcbi.1002752.

[21] Christopher A Hunter. "Sequence-dependent DNA structure: the role of base stacking interactions". In: *Journal of molecular biology* 230.3 (1993), pp. 1025–1054.

[22] B. Knudsen and J. Hein. "RNA secondary structure prediction using stochastic context-free grammars and evolutionary history." In: *Bioinformatics* 15.6 (June 1999), pp. 446–454. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/15.6.446. URL: https://doi.org/10.1093/bioinformatics/15.6.446.

[23] Eric T Kool. "Hydrogen bonding, base stacking, and steric effects in DNA replication". In: *Annual review of biophysics and biomolecular structure* 30.1 (2001), pp. 1–22.

[24] Arie Marinus Catharinus Antonius Koster. *Frequency assignment: Models and algorithms*. Arie Koster, 1999.

[25] Arie MCA Koster, Stan PM van Hoesel, and Antoon WJ Kolen. "Solving partial constraint satisfaction problems with tree decomposition". In: *Networks: An International Journal* 40.3 (2002), pp. 170–180.

[26] Felipe A Lessa et al. "Clustering Rfam 10.1: Clans, families, and classes". In: *Genes* 3.3 (2012), pp. 378–390.

[27] David A Liberles. *Ancestral sequence reconstruction*. Oxford University Press on Demand, 2007.

[28] Ronny Lorenz et al. "ViennaRNA Package 2.0". In: *Algorithms for molecular biology* 6 (2011), pp. 1–14.

[29] Rune B Lyngsø et al. "Frnakenstein: multiple target inverse RNA folding". In: *BMC bioinformatics* 13 (2012), pp. 1–12.

[30] Bertrand Marchand, Yann Ponty, and Laurent Bulteau. "Tree Diet: Reducing the Treewidth to Unlock FPT Algorithms in RNA Bioinformatics". In: *WABI 2021-21st Workshop on Algorithms in Bioinformatics*. 2021.

[31] Susumu Ohno. *Evolution by gene duplication*. Springer Science & Business Media, 2013.

[32] Robert D Olson et al. "Introducing the bacterial and viral bioinformatics resource center (BV-BRC): a resource combining PATRIC, IRD and ViPR". In: *Nucleic acids research* 51.D1 (2023), pp. D678–D689.

[33] Yong Peng and Carlo M Croce. "The role of MicroRNAs in human cancer". In: *Signal transduction and targeted therapy* 1.1 (2016), pp. 1–9.

[34] Neil Robertson and Paul D. Seymour. "Graph minors. II. Algorithmic aspects of tree-width". In: *Journal of algorithms* 7.3 (1986), pp. 309–322.

[35] Sigve Hortemo Sæther and Jan Arne Telle. "Between treewidth and clique-width". In: *Algorithmica* 75 (2016), pp. 218–253.

[36] David Sankoff. "Minimal mutation trees of sequences". In: *SIAM Journal on Applied Mathematics* 28.1 (1975), pp. 35–42.

[37] David Sankoff and Mathieu Blanchette. "Multiple genome rearrangement and breakpoint phylogeny". In: *Journal of computational biology* 5.3 (1998), pp. 555–570.

[38] David Sankoff and Mathieu Blanchette. "The median problem for breakpoints in comparative genomics". In: *International Computing and Combinatorics Conference*. Springer. 1997, pp. 251–263.

[39] Baby Santosh, Akhil Varshney, and Pramod Kumar Yadava. "Non-coding RNAs: biological functions and applications". In: *Cell biochemistry and function* 33.1 (2015), pp. 14–22.

[40] Robert W Scotland, Richard G Olmstead, and Jonathan R Bennett. "Phylogeny reconstruction: the role of morphology". In: *Systematic biology* 52.4 (2003), pp. 539–548.

[41] Dragoş Alexandru Sorescu et al. "CARNA—alignment of RNA structure ensembles". In: *Nucleic acids research* 40.W1 (2012), W49–W53.

[42] Glenn Tesler. "Efficient algorithms for multichromosomal genome rearrangements". In: *Journal of Computer and System Sciences* 65.3 (2002), pp. 587–609.

[43] Olivier Tremblay-Savard, Vladimir Reinharz, and Jérôme Waldispühl. *Reconstruction of ancestral RNA sequences under multiple structural constraints*. eng. Oct. 2016.

[44] Johannes H Urban and Jörg Vogel. "Two seemingly homologous noncoding RNAs act hierarchically to activate glmS mRNA translation". In: *PLoS biology* 6.3 (2008), e64.

[45] Jinbo Xu, Feng Jiao, and Bonnie Berger. "A tree-decomposition approach to protein structure prediction". In: *2005 IEEE Computational Systems Bioinformatics Conference (CSB'05)*. IEEE. 2005, pp. 247–256.

[46] Z. Yang. "PAML: a program package for phylogenetic analysis by maximum likelihood". In: *Computer applications in the biosciences : CABIOS* 13 5 (1997), pp. 555–6.

[47] Ziheng Yang. "PAML 4: phylogenetic analysis by maximum likelihood". In: *Molecular biology and evolution* 24.8 (2007), pp. 1586–1591.

[48] Zizhen Yao, Zasha Weinberg, and Walter L. Ruzzo. "CMfinder—a covariance model based RNA motif finding algorithm". In: *Bioinformatics* 22.4 (Feb. 2006), pp. 445–452. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btk008. URL: https://doi.org/10.1093/bioinformatics/btk008.

[49] Chunfang Zheng et al. "Ancient eudicot hexaploidy meets ancestral eurosid gene order". In: *BMC genomics* 14 (2013), pp. 1–13.

[50] Chunfang Zheng et al. "Guided genome halving: hardness, heuristics and the history of the Hemiascomycetes". In: *Bioinformatics* 24.13 (2008), pp. i96–i104.