université PARIS-SACLAY

NNT : 2019SACLX009

Thèse de doctorat

Non-redundant sampling in RNA bioinformatics

Thèse de doctorat de l'Université Paris-Saclay
préparée à Ecole Polytechnique

Ecole doctorale n°573 Interfaces (Approches Interdisciplinaires: Fondements, Applications et Innovation)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 29/03/2019, par

MICHALIK JURAJ

Composition du Jury :

Frédéric Cazals Directeur de Recherche, Inria Sophia Antipolis - Méditerranée	Rapporteur
Ivo Hofacker Professeur, Institute for Theoretical Chemistry - University of Vienna	Rapporteur
Samuela Pasquali Professeur, Univerité Paris Descartes	Examinateur
Adeline Pierrot Maître de Conférences, LRI Université Paris-Saclay	Examinateur
Aïda Ouangraoua Professeur assistant, Université de Sherbrooke	Examinateur
Yann Ponty Chargé de Recherche, CNRS-LIX	Directeur de Thèse
Hélène Touzet Directrice de Recherche, CRIStAL-CNRS (Unité de recherche)	Co-directrice de thèse

Non-redundant sampling in RNA bioinformatics

Juraj Michálik

LIX - Laboratoire d'Informatique d'École Polytechnique, Palaiseau, France

March 21, 2019

PhD supervisors:
Yann Ponty, Chargé de Recherche, LIX CNRS
Hélène Touzet, Directrice de Recherche, CRIStAL CNRS

To my mum, Alena Michálíková

Abstract

Sampling methods are central to many algorithmic methods in structural RNA bioinformatics, where they are routinely used to identify important structural models, provide summarized pictures of the folding landscapes, or approximate quantities of interest at the thermodynamic equilibrium. In all of these examples, redundancy within sampled sets is uninformative and computationally wasteful, limiting the scope of application of existing methods. In this thesis, we introduce the concept of non-redundant sampling, and explore its applications and consequences in RNA bioinformatics.

We begin by formally introducing the concept of non-redundant sampling and demonstrate that any algorithm sampling in Boltzmann distribution can be modified into non-redundant variant. Its implementation relies on a specific data structure and a modification of the stochastic backtrack to return the set of unique structures, with the same complexity.

We then show a practical example by implementing the non-redundant principle into a combinatorial algorithm that samples locally optimal structures. We use this tool to study the RNA kinetics by modeling the folding landscapes generated from sets of locally optimal structures. These structures act as kinetic traps, influencing the outcome of the RNA kinetics, thus making their presence crucial. Empirical results show that the landscapes generated from the non-redundant samples are closer to the reality than those obtained by classic approaches.

We follow by addressing the problem of the efficient computation of the statistical estimates from non-redundant sampling sets. The absence of redundancy means that the naive estimator, obtained by averaging quantities observed in a sample, is erroneous. However we establish a non-trivial unbiased estimator specific to a set of unique Boltzmann distributed secondary structures. We show that the non-redundant sampling estimator performs better than the naive counterpart in most cases, specifically where most of the search space is covered by the sampling.

Finally, we introduce a sampling algorithm, along with its non-redundant counterpart, for secondary structures featuring simple-type pseudoknots. Pseudoknots are typically omitted due to complexity reasons, yet many of them have biological relevance. We begin by proposing a dynamic programming scheme that allows to enumerate all recursive pseudoknots consisting of two crossing helices, possibly containing unpaired bases. This scheme generalizes the one proposed by Reeders and Giegerich, chosen for its low time and space complexities.

We then explain how to adapt this decomposition into a statistical sampling algorithm for simple pseudoknots. We then present preliminary results, and discuss about extensions of the non-redundant principle in this context.

The work presented in this thesis not only opens the door towards kinetics analysis for longer RNA sequences, but also more detailed structural analysis of RNAs in general. Non-redundant sampling can be applied to analyze search spaces for combinatorial problems amenable to statistical sampling, including virtually any problem solved by dynamic programming. Non-redundant sampling principles are robust and typically easy to implement, as demonstrated by the inclusion of non-redundant sampling in recent versions of the popular Vienna package.

Résumé Substantiel

Des Acides RiboNucléiques, ou des ARNs, sont des biopolymères composés des nucléotides A, C, G et U. Ces séquences ont la proprieté intéressante de se replier sur eux-mêmes, leur donnant la possibilité de former des structures différentes qui peuvent exercer des différentes fonctions. Les méthodes empiriques de leur déterminaison étant trop chéres, lentes ou limités, ces structures sont la cible d'étude du domaine de la bioinformatique structurale d'ARN.

Grâce au nombre des structures qu'une séquence moyenne d'ARN peut former, l'échantillonage statistique est devenue une partie instrumentale à des nombreuses méthodes algorithmique pour la bioinformatique structurale d'ARN, servant à identifier des structures secodnaires importantes, des éléments clés d'espace de repliement suivant l'évolution de ces structures et estimation statistique des quantités d'intérêt de ces structures à l'équilibre thermodynamique. Des méthodes qui sont disponibles aujord'hui souffrent d'effect de la redondance qui est en général non-informative et limite l'efficacité générale de ces algorithmes. Pour cette raison, dans cette thèse nous introduisons le principe de la génération non-redondante, permettant à éviter cet effet de manière optimale, et nous analysons ses applications dans le domaine de la bioinformatique structurale d'ARN.

Après introduire le concept général de l'échantillonnage non-redondant, nos montrons que, à l'aide d'un formalisme que nous avons introduit, tout algorithme eéchantillonnant dans la distribution de Boltzmann peut être transformé en une version non-redondante. Ensuite nous expliquons le besoin de mémoriser des structures secondaires échantillonnées ainsi que le processus de leur sélection. Pour le faire, nous avons recherché une structure de données qui constitue une couche autonome à implémenter à l'algorithme qui est modifié en version non-redondante et que cette version est de même complexité que l'échantillonnage classique.

Pour une démonstration pratique, nous avons implémenté ce principe dans un algorithme de l'échantillonnage des structures secondaires localement optimales. Ces structures agissent comme des pièges cinétiques et impactent le résultat final du repliement, ils sont donc instrumentales lors de l'étude cinétique d'ARN. Nous alors utilisons cet outil pour étudier la cinétique d'ARN en modélisant l'espace de repliement des structures secondaires - espace représentant l'ensemble des structures secondaires et des trajectoires de repliement entre eux - et puis réalisant une intégration numérique sur ce modèle. Par la suite, des expériences mon-

trent que des modèles obtenus à partir des échantillons non-redondants présentent une cinétique plus rapide, et sont donc plus proches à la réalité, que ceux à partir des échantillons classiques.

Nous suivons par l'inclusion de l'échantillonnage non-redondant dans la version récente de la bibliothèque Vienna package populaire. Cette implémentation montre la facilité et la robustesse de l'implémentation du principe non-redondant dans des algorithmes existants dans le domaine de la bioinformatique structurale d'ARN. Ensuite nous montrons par des expériences que l'échantillonnage non-redondante de même algorithme est plus efficace que sa version de base et nous discutons sur l'importance d'optimisation de la structure de données sur la performance finale d'échantillonnage non-redondant.

Nous résoudrons par la suite le problème d'estimation statistique efficace à partir des échantillons non-redondants. L'estimation naïve, obtenu en moyennant des valeurs observés pour toutes les échantillons, n'est pas possible dans ce cas car en absence de la fréquence, le résultat est biaisé. Par contre, nous sommes arrivés à établir un estimateur non-trivial et non-biaisé qui est spécifique aux échantillons non-redondants qui suivent la distribution de Boltzmann. Nous montrons sur des expériences concrètes sur des cas modèles que l'estimateur proposé par nous est plus efficace que l'estimateur naïf, et ceci notamment dans le cas ou la majorité d'espace de recherche a été échantillonné.

Finalement, nous introduisons un nouvel algorithme d'échantillonnage des structures secondaires contenant des pseudonoeuds de type H, le type le plus simple consistant de deux hélices s'entrecroisant. Ces éléments, même les plus simples sont typiquement omis pour des raisons d'efficacité, bien que beaucoup d'entre eux possèdent une grande importance biologique. Nous commençons par proposer une schèma de programmation dynamique qui permet à générer n'importe quel pseudonoeud composé de deux hélices comportant des bases non-appariés. Cet algorithme est une extension d'un algorithme existant de Reeders et Giegerich, choisi pour son efficacité temporelle est spatiale ainsi que la facilité relative d'implementation. Par la suite, nous expliquons comment construire le schèma de la programmation dynamique adaptable à un algorithme d'échantillonnage des structures secondaires avec des pseudonoeuds de type H. Après l'introduction des résultats préliminaires, nous expliquons comment il est possible à modifier cet algorithme pour inclure le principe de la génération non-redondante, et on discute sur des améliorations possibles de cet algorithme.

Des travaux présentés dans cette thèse ouvre la porte vers l'analyse des séquences

d'ARN plus longues et les études plus détaillés d'ARN en général. L'échantillonnage non-redondant peut être appliqué pour étudier l'espace de recherche de tous les problèmes combinatoires susceptibles à l'échantillonnage statistique, montrant sa grande versatilité, et il trouve des applications aussi aux des problèmes en dehors du domaine de la bioinformatique structurale d'ARN.

Contents

List of Figures

Acronyms

DNA	Deoxyribonucleic Acid
RNA	Ribonucleic Acid
mRNA	messager Ribonucleic Acid
tRNA	transfer Ribonucleic Acid
rRNA	ribosomal Ribonucleic Acid
ncRNA	non-coding Ribonucleic Acid
NMR	Nuclear Magnetic Resonance
DP	Dynamic Programming
MFE	Minimum Free Energy
CME	Chemical Master Equation

Chapter 1

Introduction

1.1 Preamble

Nucleic Acids are one of the most crucial macromolecule types for every living being. This class encompasses both Deoxyribonucleic Acids (DNA) and Ribonucleic Acids (RNA). The chemical composition of both acid types is quite similar; the main building brick in both cases is nucleotide composed from a phosphate group, a pentose sugar (2-Deoxyribose in DNA, ribose in RNA) and a nitrogenous base [5]. In the case of DNA, these bases are Adenine, Guanine, Cytosine and Thymine, famously abbreviated as A, G, C and T. While the first three can be found only in DNA, Thymine is exclusive to it and in RNA its role is substituted by Uracil (U) instead.

Unlike the chemical composition, the spatial structure structure of DNA and RNA is quite different. DNA is usually found in a double-stranded form arranged in a double helix, where the two strands are 'zipped' together via hydrogen bonds between the nitrogenous bases in an antiparallel manner (Figure 1.1). The orientation of each strand is determined by the sugars it presents - the 5' end of the strand directly branches from 5-th atom of 2-Deoxyribose. Since a given nitrogenous base has only limited number of pairing partners with which it can create sufficiently strong hydrogen bonds, the DNA adheres to base-pairing rules. RNA, on the other hand, is much shorter and can be found mostly in single stranded form, which is similarly oriented by Ribose (Figure 1.2). The base-pairing also applies to the RNA, though the pairs can be created within the same molecule or with other independent RNA, DNA or proteins [65]. Consequently, the combination of intramolecular interactions offers a variety of structures an RNA molecule

can adopt.

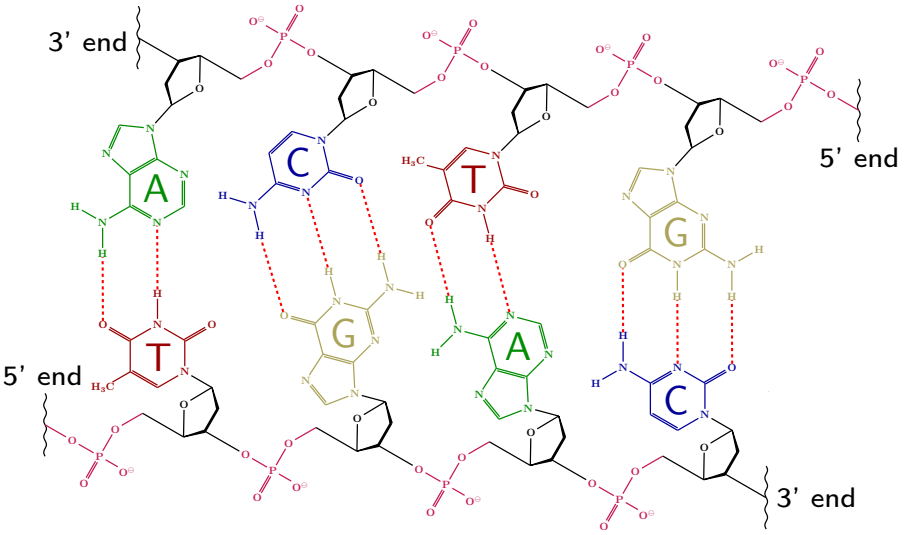

Figure 1.1: **A planar representation of chemical structure of double-stranded DNA.** Each strand is a chain of nucleotides, connected by phosophodiester bond and composed of Phosphate (purple), 2-Deoxyribose (black) and the nitrogenous base. The latest has four main variants - Adenine (A, green), Guanine (G, Yellow), Cytosine (C, Blue) and Thymine (T, brown). The two strands are held together by hydrogen bonds between nitrogenous bases according to base-pairing rules. The 5′ end and 3′ end are determined by the orientation of the 2-Deoxyribose.

The structure of DNA and RNA also projects into the functions they can perform. The stability of a double helix of DNA [90] allows it to serve as a cellular memory - it stores the genetic information within the cellular nucleus and that information is replicated and propagated on the successor during cell division. In the case of RNA, people often think that its main function is the transfer of the information contained within the DNA to the molecular machinery that assembles the proteins - ribosomes [17]. However, besides these RNAs, also known as messenger RNAs (mRNA) there is a wide variety of its other types with equally wide array of functions other than information transfer; since these RNAs do not encode proteins, they are called non-coding RNAs (ncRNA). Their function can vary from structural, including but not limited to ribosomal RNA (rRNA) and transfer RNA (tRNA), to catalytic and regulatory functions such as the ncRNAs involved in CRISPR-Cas9 system [75]; RFAM database currently stores an information about 2687 functional RNA families [49]. The function of an RNA depends mainly on its structure, which is determined by the sequence and base-pairing rules, and consequently the structure of such sequence is often strongly

conserved [102]. This, along with the functional versatility, motivates studies aiming at modeling the structure of RNA.

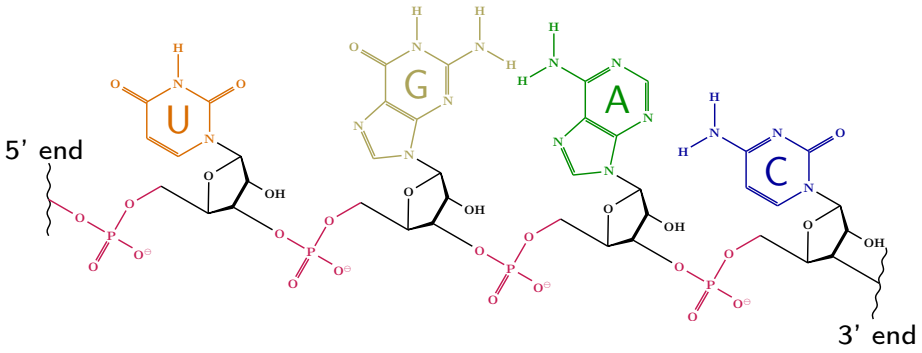

Figure 1.2: **A planar representation of chemical structure of single-stranded RNA.** In this case, the sugar (black) is ribose and Uracil (U, orange) subsitututes Thymine (T, brown).

The first methods of studying of the secondary structures were empirical, which became the standard over the years of their application. Such methods include nuclear magnetic resonance (NMR) [84], X-ray Crystallography [101] and cryo-electron microscopy [33]. Since such methods of finding an RNA structure are rather costly, complex and/or slow, with the advent of computational technologies it became tempting to exploit them to predict these structures. However, the task is not as simple as it may seem. First, in when computing the structures the one with the lowest energy is considered the functional one. The reason for such assumption is that such structure is, under the conditions of thermodynamic equilibrium, where the system does not evelwe anymore, considered the most stable, as it is presents the global optimum. In reality however, this is not always the case. Instead, it might be a locally optimal structure where the RNA remained 'stuck' during the process of the creation of the structure, also known as RNA folding. These structures act as kinetics traps due to them being stabilized by sufficiently high energy barriers meaning sufficiently high energy is needed to escape it [25]. Another possibility is that the half-life some RNAs is too short [86] and the molecule decays before it had enough time to reach the most stable structure. Second, some RNAs present more than one active structure, such as riboswitches, where the second structure is formed only in presence of certain ligand or during co-transcriptional folding when some of bases are not accessible [37]. This implies that merely studying the globally optimal RNA structure may not be sufficient to understand its relation with function. Rather, it is necessary to study also the so-called suboptimal structures - those with close but

higher energy than the possible minimum, and also structural dynamics - simply stated the kinetics of RNA structures. Due to a number of structures that raises exponentially with the length of the RNA sequence [106], exhaustive analysis in unrealistic, and it is necessary to select few representative structures that have high probability of being an active structure. This is the main motivation for the development of methods relying on statistical sampling.

While the decomposition algorithm that would become the basis and standard for the statistical sampling of secondary structures was discovered back in 1984, it took almost twenty years to establish a first stochastic sampling method. That was done by Y Ding and Charles E. Lawrence in 2003 [24]. These methods sample the structures by computing their probability according to a given distribution by decomposing these structures and sampling their local elements [54]. Such sampling can be also done for a specific class of secondary structures, such as locally optimal ones.

Today, many sampling methods are available and can be used to retrieve the most interesting secondary structures according to given criteria. Unfortunately, most of them do not distinguish whether the structure they sample was generated before - they return it whether it was never sampled before, or it repeats tenth time. These repetitions are generally uninformative as shown by methods that partially remove it [50], and the time used to sample repeated structures might be used instead to sample the previously unseen examples that provide new information. The method sampling unique secondary structures would allow to cover the space of secondary structures much quicker. This finds use from faster and easier access to structures that have usually low probability of being sampled in the given distribution but might have biological significance, such as in the case of transitive structures [19], to generate a richer space of secondary structures to propose more accurate models that require their space as an input, namely the energy landscapes. For this reason, we asked whether there was a way to prevent such repeats, a method that is mainly, but not necessarily only, applicable in the domain of the structural RNA bioinformatics.

The main question is whether such approach can be performed in an efficient manner. Since the non-redundant sampling principle necessitates to perform supplementary computations and necessitates more information than the usual, redundant sampling approach [57], it will obligatorily consume more time. Therefore, such method must be as optimal as possible so the gain from the non-redundancy outweighs the added cost, which must be limited to maximum. It must be also easy to implement to all existing secondary structure sampling al-

gorithms, meaning it should constitute a separate layer that is added to the algorithm to which the non redundant principle is implemented.

The work presented within this PhD answers all of the above questions and demonstrates the utility of such approach. Namely, we show how the non-redundant sampling can be used to access more precise prediction of the RNA energy landscapes, and to estimate more precisely the statistics related to their population.

1.2 The Plan of This Work

In this Section, we briefly remind the history of nucleic acids before introducing the basic concepts such as the levels of the structure. This is quickly followed by the Section 2, which elaborates on state-of-the art methods. After introducing notions and concepts necessary to understand this document as well as the concept of the dynamic programming, we detail the algorithms employed in structural RNA bioinformatics that were crucial for our work. In this section, we also introduce a formalization of dynamic programming schemes. Such formalism helps us to generally demonstrate the properties of all algorithms that can be formalized and show the general compatibility with non-redundant sampling principle presented in the next chapter. Finally, we describe the state-of-the-art in RNA kinetics as well as the description of the modeling processes we employed.

We start Section 3 by describing of the combinatorial algorithm that served as our starting point, called Algorithm of Saffarian [80], as well as justifying this choice. We will explain its modification to compute the energies of structures it returns. We then pass on the principle of non-redundant sampling and how it can be implemented into the algorithm. The next part of the chapter will describe the experiment of modeling the RNA kinetic landscape from the generated samples by this and competitor algorithms. We close this chapter by describing how the non-redundant sampling principle can be implemented into other existing algorithms while demonstrating it on the implementation into VIENNARNA library [55], followed by the comparison of its performance with the basic version.

The Section 4 concentrates of statistical analysis of non-redundant sampling algorithm. The loss of frequency due to non-redundancy means that the computation of an average of given property cannot be achieved naively by computing the weighted average. This problem was solved in collaboration with Christelle Rovetta by establishing a non-trivial, unbiased estimator. The details of it, as well

as the experiment where we compare it with the naive variant, are described here.

In Chapter 5, we introduce the sampling algorithm for the secondary structures including H-type pseudoknots. This chapter explains the algorithm used in software `pknotsRG` [76], also extended and employed in `pKiss` [47], to which our algorithm is an extension,as well as the reasons why we decided to extend this algorithm. We then present the preliminary results obtained as well as the future extensions, including using non-redundant sampling principle.

In the Section 6, we summarize the contributions of non-redundant sampling principle and we discuss of their importance within and outside the field of structural RNA bioinformatics.

Finally, the last chapter will feature the conclusion, and include a discussion of the next steps in this line of research.

1.3 Nucleic Acids: A Look to the Past

1.3.1 The Discovery of Nucleic Acids

The discovery of nucleic acids, or more precisely of the DNA, dates back to 1869, when Friedrich Miescher discovered inside human leucocytes a substance that differed from proteins by its properties - he called it 'nuclein' [105, 20]. The name was later changed by Richard Altmann to nucleic acid due to acidic properties the substance exhibited [21]. Between 1885 and 1901, Albrecht Kossel and his students then identified five organic compounds present in nucleic acids; these later turned up to be nitrogenous bases. He also inferred that nucleic acids are most likely implied in synthesis of new tissues.

The differences between RNA and DNA were not apparent at the time of their discovery and at first they were called by their source of origin - DNA was known as the thymus nucleic acid [53] and RNA as the yeast nucleic acid [52], and if it was thought they were exclusive to animals and plants respectively. The key discovery was made in 1933 by Jean Brachet who has shown that the DNA was localized in chromosomes of every cell [11]. Likewise, he has shown that the RNA could be found in cytoplasm of every cell.

Around the same time, Phoebus Levene studied the chemical composition of nucleic acids. His analysis revealed the chemical structure of each nucleotide, the na-

ture of sugars as pentoses, the presence of phosphates and the fact that thymine and uracil are exclusive to DNA and RNA respectively [40]. He also supposed that the DNA is composed from repeated tetranucleotide units interconnected by phosphate ester-sugar bonds. In 1944 Oswald Avery, Colin MacLeod and Maclyn McCarty performed a modified version of Griffith's experiment [39, 4]. In this version the mice were injected by rough (non-virulent) strain of *Pneumococcus* bacteria and the material of heat killed smooth (virulent) strain that was previously purified and that contained DNA. The results have shown that the DNA was the most likely responsible for bacterial transformation, and therefore it was involved in heredity. Consequently, it also disproved the tetranucleotide theory since it was unlikely that a mere chain of tetranucleotide repeats could contain all needed information.

The following important discovery would be made by Erwin Chargaff near the middle of 20^{th} century. He discovered two tendencies within the DNA that are today known as Chargaff's rules [27, 15]. The first discovery was that the content of Thymine and Adenine were equal, as well as that of Guanine and Cytosine. In other words %A = %T and %C = %G. This also definitely invalidated the tetranucleotide theory of Levene since not all nucleotides were found in equal proportions. The second was that these percentages were specific to species. From here, it was only one step towards the famous discovery made in 1953 by James Watson and Francis Crick, who using the X-ray crystallography images of DNA obtained by Rosalind Franklin and Maurice Wilkins created the two-strand double-helix DNA model [12, 98]. This, along with first Chargaff's rule, has also shown that A pairs with T and C with G; consequently these pairs were called Watson-Crick's. The boom around the DNA this discovery has caused then resulted into quickly discovering the genetic code in 1960s [67, 32, 94] and the sequencing methods about dozen years later [81].

1.3.2 mRNA and ncRNA, and the discovery of the secondary structure

The discovery of DNA structure has shown that it is very unlikely that DNA itself would be directly transcribed to proteins. The experiments of Jean Brachet in forties of 20^{th} century suggested that the RNA is related to protein synthesis [11]. This led to Crick to postulate, in 1957, his *Central Dogma of Molecular Biology* [17] that suggested that RNA is the intermediate for passing the information from the DNA in nucleus to the center of protein synthesis in cytoplasm. This was later confirmed by the discovery of lac operon of *E. Coli* and experiments performed on

them by Jacques Monod and Francois Jacob [66]. The unstable RNA they found has become known as messenger RNA (mRNA) since then. This finding however opened a can of new questions like what the intermediate molecule that associates the genetic code to the correct amino acids is and of course the protein synthesis machinery itself. This led to the discovery of the first classes of the non-coding RNAs (ncRNA).

The ribosomes, the center of protein synthesis, are themselves constituted from a special subclass of ncRNAs - ribosomal RNA (rRNA). It was first observed in 1938 by Albert Claude [71] as a cytoplasmic masses he called *microsomes*, eventually identified as ribosomes by George Palade [70]. The transfer RNA (tRNA) - the link ensuring the correspondance between the genetic code and the amino-acids - was theoretically assumed by Crick. It was confirmed by Robert Holley *et al* when he reported a 77 nucleotides long sequence of yeast alanine tRNA along with suggestions of secondary structure for it in 1964 [44]. The last point is, within the context of this thesis, particularly important because it demonstrates how ncRNAs, to which tRNA and rRNA belong, need specific structure to perform their function. It was then no surprise that a number of methods to identify these structures, first empirical like X-ray Crystallography and Nuclear Magnetic Resonance (NMR), later computational predictions, emerged. This thesis concentrates mainly on efficient computational prediction of RNA secondary structures and their kinetics.

1.4 A Structural Intermezzo - Levels of RNA structures

Before moving on towards the state-of-the-art in RNA energy computation and secondary structures prediction, it is necessary to precise what an RNA secondary structure is. That is the main objective of this sub-section.

As already mentioned, RNAs consist of nucleotides (or ribonucleotides) A, G, C and U, which substitutes T from DNA. Each nucleotide consists of a nitrogenous base (from which the nucleotide gets its letter), Ribose and phosphate group. Nitrogenous bases can be also divided into Purines, denoted R, that have two heterocycles (Adenine and Guanine, see Figure 1.3), and Pyrimidines, with symbol Y, that have only one (others). A group of only nitrogenous base and Ribose is called ribonucleoside. Nucleotides are connected by phosphodiester bond between 3^{rd} and 5^{th} atom of Ribose 5-phosphate, from which the 5′ end and 3′ end orientation comes. Unlike DNA, RNA is mostly found in single-stranded form.

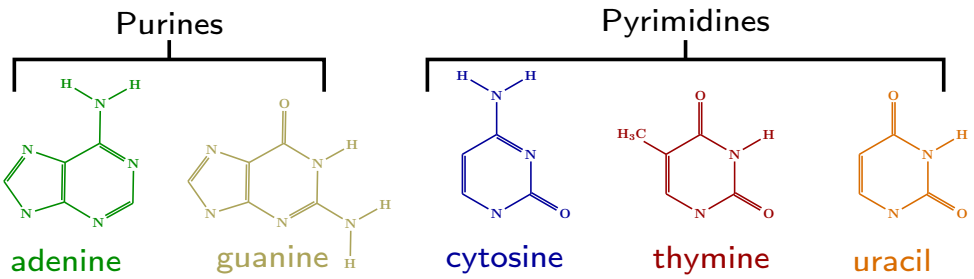

Figure 1.3: **The five nucleotides occuring within nucleic acids.** Thymine does not occur within RNA while Uracil is specific to it.

Depending on the types of non-covalent interactions that is taken into account, we distinguish four levels of structures of RNA. The first three are also exemplified on Figure 1.4

- **RNA Primary Structure**: No non-covalent bonds are considered. In other words, this is the sequence of nucleotides. An example of a primary structure would be a sequence GUCACGUGCAU. This structural level is the most important for the mRNA since they contain the information translated into proteins, but the higher-grade structures depend on it as well.
- **RNA Secondary Structure**: Only hydrogen bonds between nucleotides are taken into an account. Basically, the secondary structure is the set of base pairs that is present for given conformation of specific RNA. An example of such structure is given on Figure 2.2A. **This level of structure is the main point of interest of this thesis.**

  The RNA secondary structure depends on a specific set of base pairs that is considered for given sequence. The principle of base-pairs is hydrogen bond type of interaction between the nucleotides. Not every base pair can create these bonds. The usually considered base-pairs are those discovered by Watson and Crick in DNA - the pairs between A and U and C and G [98]. The G - C pair is stronger [87] due to, if we simplify enough, three hydrogen bonds instead of two that are formed between A and U (see Figure 2.1). Hence the stability of the RNA secondary structure is also considered by the number of G - C pairs, though by all means that is far from the only criterium. There are also other types of base pairs that are called 'wobble'. These pairs were predicted by F. Crick [18] due to the fact that while there are 64 codons, the number of corresponding tRNAs is lower [14], and con-

sequently some of anticodon sites on some tRNA molecules have to pair with more than one nucleotide. Most of wobble base pairs include a special nitrogenous base, hypoxanthine, that is created by post-transcriptional modification of adenine in tRNA [91]. Since that base is omitted within the work, we will consider, in addition to Watson-Crick base pairs, the only wobble base pair not including hypoxanthine, a base pair G - U (Figure 2.1, right).

For the rest of this work, when talking about the secondary structures, we refer to the RNA secondary structures, unless noted otherwise.

- **RNA Tertiary Structure**: All intramolecular bonds are considered. That includes, besides hydrogen bonds between nucleotides, also hydrophobic effects and disulfide bonds. These forces give an RNA molecule a specific tridimensional structure. To have an idea of difference between secondary and tertiary structure, in the case of DNA, the latter is the typical double-helix form, while its secondary structure would be merely a 'ladder'.

  The RNA secondary structure is easier to predict than the tertiary structure due to the fact that the number of interactions to predict is lower. Nevertheless, it already provides a good basis for functional hypotheses, as exemplified by the regulation of tryptophan production on *trp* operon in *E. Coli* [26]. For this reason we decided to concentrate on the kinetics and prediction of RNA secondary structures. The higher level structures will not be developed further here and are only mentioned for the sake of completeness.

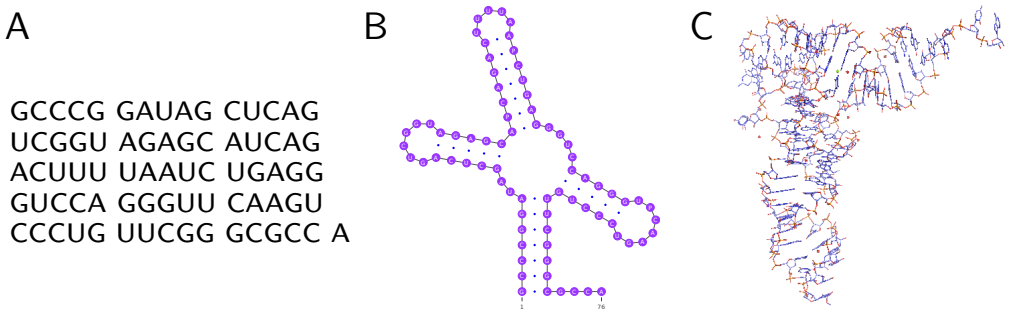

Figure 1.4: **The three levels of RNA structure.** The primary structure **(A)** is essentially a chain of nucleotides. The secondary structure **(B)** also includes the hydrogen bonds between the nucleotides according to valid base pairing rules. The tertiary structure **(C)** takes into account all intramolecular forces. Molecule reference: PDB ID 1FIR [13, 99, 8]. The secondary structure taken from RNA STRAND [2] and visualized by VARNA [23]. The tertiary structure created by PyMOL [82].

- **RNA Quarternary Structure**: In addition to the above, the intermolecular bonds are considered. This level of structure requires also the study of RNA-RNA, RNA-DNA and RNA-protein interaction.

Chapter 2

RNA secondary structure prediction - What has been done

2.1 Notions related to the RNA secondary structures

Before we start dwelling deeper into the realm of RNA sequences and secondary structures, we must establish the basic notions that will be used in this work.

Definition 2.1.1 (Sequence): An RNA sequence $\boldsymbol{w}$ is a chain of nucleotides of length n over the alphabet $\Omega = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{U}\}$.

We do not consider any other symbols outside the grammar Ω.

Definition 2.1.2 (Substring): A **substring of $\boldsymbol{w}$** from position i to position j ($j \leq i \leq j \leq n$) is denoted as $w[i, j]$. A nucleotide at position i, $i \in [1, n]$ is marked as $w[i]$, $w[i] \in \Omega$.

RNAs treated here are strictly linear, with 5′ end being on its left side. The nucleotides can form a different base pairs within the sequence.

Definition 2.1.3 (Base pair): A base pair is a pair of nucleotides at position i and j, $i < j$. It is denoted by (i, j).

For given sequence, many different base pairs can be formed.

Definition 2.1.4 (Base pair set): The set $\boldsymbol{\phi_{pair}(w)}$ is the set containing all possible base pairs (i, j) for a sequence w. Each base pair (i, j) must satisfy the following conditions:

- Each (i, j) must be Watson-Crick or wobble base pair, ie.

$$(i, j) \in \{\{A, U\}, \{C, G\}, \{G, U\}\}.$$

- $w[i]$ and $w[j]$ must be separated by at least θ base pairs:

$$j - i - 1 \geq \theta.$$

The minimum base pair length comes from the steric strain that forbids the base pairing of nucleotides way too close one to another.

Each sequence w can form a certain subset of all possible base pairs $\phi_{pair}(w)$ at any given moment. This applies to its substrings $w[i, j]$ as well.

Definition 2.1.5 (Secondary structure): An **RNA secondary structure**, or simply **secondary structure** $S(i, j)$ is a set of base pairs $\boldsymbol{(a, b)}$ within a substring $w[i, j]$ ($\forall a, b, i \leq a < b \leq j$). We have, $\forall i, j, S(i, j) \subseteq \phi_{pair}(w)$. The secondary structure of entire sequence $\boldsymbol{w}$ is denoted by $S(1, n) = S$.

From this point on, we refer an to RNA secondary structure as a secondary structure or just structure unless noted otherwise.

Throughout this work, we may refer to the secondary structures smaller than n and with undefined limits.

Definition 2.1.6 (Secondary structure with unspecified length): A secondary structure s is a structure with unspecified, resp. implicitly specified $w[i]$, $w[j]$ and n.

Within the same $S(i, j)$, each base pair has to follow, in addition to the requirements from the Definition 2.1, the following conditions:

- One nucleotide can participate at most in one base pair at the same time. This means that if $(i, j) \in S$, $\forall k \neq j, (i, k) \notin S, (j, k) \notin S$.

- At first, we consider for any base pairs $(i, j) \in S$, $(k, l) \in S$ with $i < k$ that either $i < k < l < j$ or $i < j < k < l$, ie one base pair cannot cross another. At later point, in Section 5, we will also investigate some structures that include crossing base pairs.

If $S = \varnothing$ (ie all of bases within the sequence are unpaired), the RNA sequence is called **unfolded**.

Definition 2.1.7 (Space of secondary structures):
Given an RNA sequence w, we define by $\mathcal{S}$, the **space of secondary structures**, a set of all possible secondary structures of any length. The space $\mathcal{S}_n$ is a **space of n-sized secondary structures** observed for $\boldsymbol{w}$. Each is composed of base pairs from $\phi_{\mathrm{pair}}(w)$. Similarly, $\mathcal{S}(i, j)$ the space of all secondary structures $S(i, j)$ for $w[i, j]$.

2.2 Representation of RNA secondary structures

A secondary structure can be shown in various ways. Each of them has its advantage and weak points. The representations used throughout this work are listed here along with an example on Figure 2.2.

- **Radial representation** (2.2A): The paired segments are represented by a rectangular ladder-like structure while the unpaired segments are organized in circles from which the paired regions branch. In the case of a linear RNA, its unpaired base can be represented linearly or in a circle. This visualization clearly depicts different local substructures (see Section 2.4.2), facilitating to understand its function. On the other hand, it badly handles the cases where base pairs cross. It is good to use when one needs to explain the function of a secondary structure.
- **Circular representation** (2.2B): The (linear) RNA is represented in a circular pattern, base pair being represented by a line or an arc. It has no problem to depict crossing base pairs but it is much harder to interpret. This type of secondary structure will not be employed in this work.
- **Linear representation** (2.2C): The RNA is shown as a line, and base pairs are depicted by arcs connecting the concerned nucleotides. This representation can be used to compare two different structures each drawn from other

side of line. It is also an easy way to depict and explain the recursions of secondary structure prediction algorithms, which is main use of this visualization here. However, it might be a bit harder to interpret, especially for long-range interactions.

- **Dot-bracket notation** (2.2D): This is the textual representation of secondary structure with specific symbols. A dot '.' indicates an unpaired base, and a base pair is represented by brackets '(' and ')'. The correspondence of opening and closing brackets works in the same manner as in math; six opened brackets must be closed by six closing brackets and the last bracket opened it the first one that is closed. In the case of crossing base pairs, one of the crossing base pairs can be denoted by square bracket possibly followed by a letter, ie. '['- ']' or '[A'-']A'. This format is hard to interpret by human, though there is a visualization software that can depict the related secondary structure, such as VARNA [23]. Its main purpose is to be machine-readable, which is why it became a standard input and output format for RNA secondary structure prediction software.

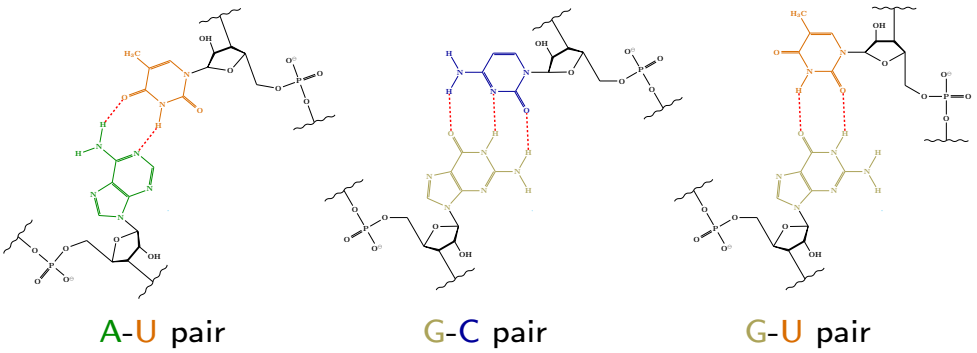

Figure 2.1: **The main base pairs occuring within the RNA.** These base pairs consist of hydrogen bonds (red dashed lines) between the atoms of nucleotides. The Adenosine - Uridine (left) and Guanosine - Cytidine(center) base pairs are also called Watson-Crick. The Guanosine-Uridine (right) base pair is also called wobble and is less frequent due to slightly lower energy.)

2.3 Combinatorics of RNA secondary structures

First methods used to find the RNA secondary structure and structures in general were empiric. Even the discovery of double helix of DNA resulted from an X-ray

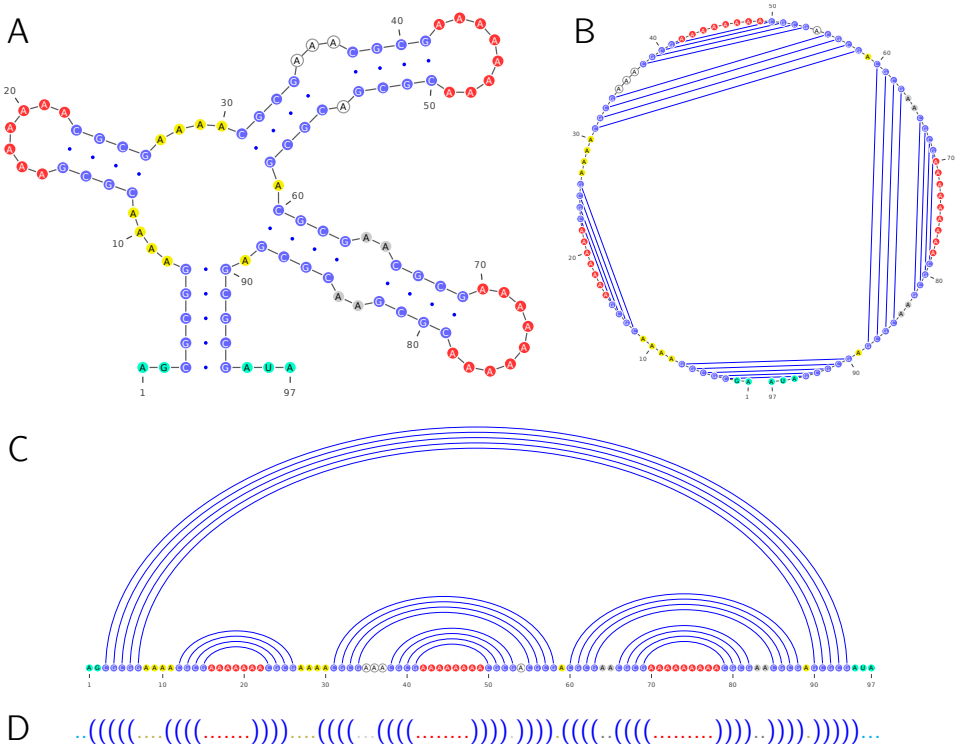

Figure 2.2: **An example of RNA secondary structure and its different elements shown using different representations:** radial (A), circular (B), linear (C) and dot-bracket (D). The different elements are an external loop (cyan), hairpin (red), stem/helix (blue), an internal loop (grey), a bulge (white) and a multibranched loop (yellow). Representations A-C were created using VaRNA [23].

crystallography picture [12]. However the crystallography is impossible if the analyzed substance does not crystallize. The NMR does not have similar problems, and unlike for proteins, the length of the molecule is not a limit either due to the RNA being shorter molecules. However, NMR and other experimental methods to determine the structure of RNA such as the X-ray crystallography and cryogenic electron microscopy have complicated protocols that require some time to prepare and they are expensive even today [10]. Therefore with the advent of computational technology the attention turned to the computational prediction of the molecular structure, where the possibilities to make it cheap and more efficient are much larger.

2.3.1 The number of secondary structures

To predict a secondary structure, it is necessary to have a mathematical or combinatorial description of properties that such a structure has. This property has to be possible to evaluate by a score, computed by appropriate scoring function $\mathfrak{S}$.

Definition 2.3.1 (Scoring function): We call a **scoring function** a function that to each $S \in \mathcal{S}$, associates a certain score:

$$\mathfrak{S} : \mathcal{S} \to \mathbb{R}$$

In the case of RNA, $\mathfrak{S}$ usually computes a free energy. The lower energy of the structure means the better stability of the final secondary structure and hypothetically this one is the most likely structure to be the functional one.

The first approaches of energy computation were simply based on a number of base pairs within the considered secondary structures, without any distinction between stronger G-C, weaker A-U and wobble G-U pairs. The decomposition method - necessary to establish an evaluation function - was introduced by Temple F. Smith and Michael S. Waterman [97, 96] to compute **the number of possible secondary structures**. In principle, each base is, under the restriction defined in Section 2.4.2, either unpaired, or paired to some other base. With this in mind, if we suppose a substring $w[i, j]$, then j is either unpaired, or paired to some $k, i \leq k < j - \theta$, θ implying the minimum base pair length. In the first case, the number of the secondary structures is equal to those of the substring $w[i, j-1]$. In the second case the pair (k, j) generates two new intervals $w[i, k-1]$ and $w[k+1, j-1]$ (see Figure 2.3). The same principle is then reapplied recursively to each new interval until $j - i - 1 \leq \theta$.

Let $N^{d}_{S(i,j)}$ be the number of secondary structures of $w[i, j]$ having exactly d pairs. The recursive relation given by Smith and Waterman algorithm is:

$$N^{d+1}_{S(i,j)} = N^{d+1}_{S(i,j-1)} + \sum_{k=i}^{j-\theta-1} \sum_{u=0}^{d} N^{u}_{S(i,k-1)} \times N^{d-u}_{S(k+1,j-1)} \tag{2.1}$$

Here the first component of the right-hand side part of this equality is the case where i is unpaired, and the second component, with the double sum, is the case where j is paired with some k. Note that for $w[i, j] \mid j - i - 1 \leq \theta$ no base pair is

possible and every nucleotide must stay unpaired. Therefore,

$$\forall\{i,j \mid j-i-1 \leq \theta\}, N^0_{S(i,j)} = 1$$

and

$$\forall\{i,j,d, j-i-1 \leq \theta, d > 0\}, N^d_{S(i,j)} = 0.$$

Let $N_{S(i,j)} = \text{card}(\mathcal{S}_n(i,j))$ be the number of secondary structures of $w[i,j]$, or in other words, $N_{S(i,j)} = \sum_{d=0}^{n} N^d_{S(i,j)}$, n being an upper limit of d. In a similar manner, by generalizing the decomposition from Figure 2.3, we get:

$$N_{S(i,j)} = N_{\mathcal{S}(i,j-1)} + \sum_{k=i}^{j-\theta-1} N_{S(i,k-1)} \times N_{\mathcal{S}(k+1,j-1)} \tag{2.2}$$

Here

$$\forall\{i,j \mid j-i-1 \leq \theta\}, N_{S(i,j)} = 1.$$

The computation of the number of the secondary structures is then simple. First we pre-compute all values of $N_{S(i,j)} 1 \leq i < j \leq n$ in a manner the the shorter sequences are precomputed first and store them in a two-dimensional matrix. The final value is stored within $N_{S(1,n)}$. The time complexity of the algorithm is $\mathcal{O}(n^3)$ since n^2 entries must be treated and for each n options exist at most. The space complexity is $\mathcal{O}(n^2)$ due to storing n^2 entries.

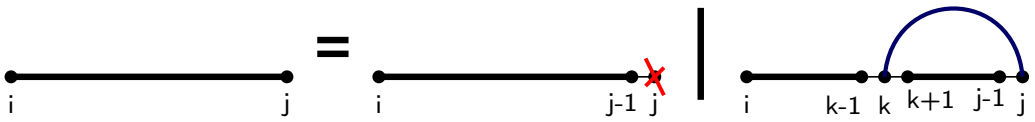

Figure 2.3: **The decomposition of secondary structures used in Smith and Waterman algorithm**. The base j can be either unpaired or paired to some k from $w[i, j-\theta-1]$. Both cases produce shorter substrings for which this process can be recursively applied.

2.3.2 About Dynamic Programming

Enumerating all possible secondary structures by simple recursive function is not the most efficient way to handle this problem. Using the algorithm of Smith and Waterman, Michael Zuker and David Sankoff shown that for the sequence with uniformly distributed nucleotides the asymptotic limit for the number of secondary structures of RNA of length n is around 1.867^n [106] - it raises exponentially. Therefore, trying all possible combinations would also have an exponential

complexity. This is why the approaches using Dynamic Programming (DP) are extremely useful. DP scheme decomposes a central problem, here the secondary structure of w into smaller problems, the structures of $w[i, j]$, computing those and using the results to solve the main problem. Each calculation is done only once and the result stored in DP matrices whose size depends on the number of states that have to be saved, like done by counting function from Equation 2.1. This makes the computation faster due to absence of redundancy.

The DP algorithms have to respect one main principle, called Bellman's Principle of Optimality [7]. To quote Bellman:

> *"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."*

Translated, every intermediate solution, that leads to the main optimal solution, must be also optimal. These are optimal solutions to given subproblems. Note that the DP schemes might be also applied in cases where the algorithm would give an optimal solution from non-optimal ones for given subproblems or inversely, where optimal solution to subproblem does not lead to an optimal global solution, but that reduces them to a mere heuristic method.

All of approaches used for structure prediction and mentioned here them respect the Bellman's Principle of Optimality unless noted otherwise. The core of most of them is to decompose w into $w[i, j] \mid 1 \leq i < j \leq n$ and solve the subproblems for sequences of increasing length $l = j - i + 1$. The result for each subproblem is computed and memorized. Consequently, the results for substrings of length $1, 2...l - 1$ are then used to compute that for sequences of length l. The main problem is then solved for the sequence w. The same principle can be employed to compute suboptimal structures.

Definition 2.3.2 (Suboptimal structure): Suppose each secondary structure has a certain score $\mathfrak{S}(S)$ and the objective is to minimize $\mathfrak{S}$. We call a **suboptimal structure** every S_T that

$$\mathfrak{S}(S) > \min_{S_T \in \mathcal{S}_n} (\mathfrak{S}(S))$$

The equivalent definition can be established when maximizing $\mathfrak{S}$.

2.3.3 Nussinov Algorithm

The first algorithm to make use of the decomposition of Smith and Waterman, as well as the DP, to predict a secondary structure was proposed by Ruth Nussinov *et al* [68]. The function $\mathfrak{S}$ is in this case is the number of the base pairs.

The first version of the algorithm was not unambiguous, meaning that a single structure could be decomposed in a multiple ways. Here we talk about the unambiguous version which was also the base for the secondary structure counting function of Waterman.

Here each base pair has the same weight; G - C, A - U and G - U score as 1. The final score $\mathfrak{N}(1, n)$ gives the maximum number of the base pairs possible for w. Let $\mathfrak{N}_{i,j}$ be this number for any $w[i, j]|1 \leq i < j \leq n$, then we can formulate the algorithm as:

$$\mathfrak{N}_{i,j} = \max \begin{cases} \mathfrak{N}_{i,j-1} & \text{if j is not paired} \\ \max\limits_{\substack{i \leq k \leq j-\theta-1 \\ (i,j) \in \phi_{pair}(w)}} (\mathfrak{N}_{i,k-1} + \mathfrak{N}_{k+1,j-1} + 1) & \text{if j is paired to k} \end{cases} \tag{2.3}$$

The score raises by 1 whenever k and j are paired, otherwise the score remains unchanged. It can be generalized by replacing the bonus 1 in the case of $(k, j) \in S(i, j)$ by a bonus $\mathfrak{B}_{k,j}$ that varies depending on a base pair type:

$$\mathfrak{N}_{i,j} = \max \begin{cases} \mathfrak{N}_{i,j-1} & \text{if j is not paired} \\ \max\limits_{\substack{i \leq k \leq j-\theta-1 \\ (i,j) \in \phi_{pair}(w)}} (\mathfrak{N}_{i,k-1} + \mathfrak{N}_{k+1,j-1} + \mathfrak{B}_{k,j}) & \text{if j is paired to k} \end{cases} \tag{2.4}$$

$\mathfrak{N}_{i,j} = 0$ if $j \leq i + \theta$ since the substring $w[i, j]$ is too short for any base pair to be created there, providing an initialization condition. From there, we can compute $\mathfrak{N}_{i,j}$ for progressively longer substrings. The values are stored in a matrix $\mathfrak{N}$ and reused as needed. The final result is accessible in is $\mathfrak{N}_{1,n}$.

To obtain the structure with the maximum base pairs, we can either directly store the base pairs as the maximum values are selected for different points, or perform a stochastic backtrack after computing $\mathfrak{N}_{1,n}$. The backtracking is done by starting at $\mathfrak{N}_{1,n}$, finding the combination that gives us the stored value - in this case it is either $\mathfrak{N}_{1,n-1}$ or sum of $\mathfrak{N}_{1,k-1}$ and $\mathfrak{N}_{k,n}$ for $1 \leq k \leq n - \theta - 1$. If (k, j) is present, it is added to the structure, and the whole process is repeated until the initial

conditions are reached. The secondary structure generated this way is the one with maximum base pairs.

The complexity of this algorithm is $\mathcal{O}(n^3)$ in time and $\mathcal{O}(n^2)$ in space [68]. The explanation is equivalent to the secondary structure counting function of Waterman.

Further extensions added no-lone-base-pair as a constraint - no base pair can appear alone. This is because the stacks of a base pair have stronger stabilizing effects on the structure than a single base pair due to the stacking effect of the cycles of nitrogenous bases [72]. Despite this, the prediction results of this model were still imprecise, and more complex approaches were necessary.

2.4 Enter the Thermodynamics

The approaches in question targeted to compute the exact free energy of the entire secondary structure. The free energy can be described, in the context of secondary structures, as the amount of the energy you would get/supply to the unfolded RNA sequence for it to reach the given secondary structure. Its unit is kcal.mol^{-1}, the reference being an unfolded state. The most interesting structure S for given w is the one with the lowest possible energy minimum free energy structure (MFE).

To compute MFE, S must be decomposed into a set of elements for which the free energy can be computed without too much problems. Their free energy contribution is then summed up or multiplied, netting the final result. This also implies that the elements must be independent one on another. The decomposition and the resulting elements depend on the energy model which will be applied.

2.4.1 About energy models

The energy model is a model that compute the free energy of a set of specific element. These may vary from simple values such as the energy of G - C bond to much more complex relations that apply only for very specific cases, such as loops that have exactly four unpaired nucleotides, dangling ends - interactions between the base pair and its unpaired neighbors - and so on. Throughout this work, we will mainly talk about and employ the **Turner Energy Model** [92, 93]. This energy model allows to compute the energy of the elements resulting tfrom the decomposition of a secondary structure detailed in Section 2.4.2.

The Turner Energy Model consists of values that are either tabulated and were directly deduced from experimental results for smaller elements, such as the stabilities of terminal wobble G - U pairs in consecutive base pairs [16], or mathematical relations for bigger substructures with more nucleotides that were extrapolated from the empiric results such as melting experiments of the RNA [83]. The energy of each of substructure, when calculated by this model, depends on and only on the properties of the substructure itself and oft those of the adjacent substructures such as neighboring base pair. For this reason, this model is also called Nearest Neighbor Turner Energy Model. This is an important property because it suggests the independence between the distant elements resulting from the decomposition and therefore the energy contribution of each of them can be simply summed up.

The latest iteration of the model, from the year 2014, was also implemented into VIENNARNA [55], which is the library which we will use for the application of this energy model.

2.4.2 A decomposition of RNA secondary structure

There are multiple ways to decompose a secondary structure. The most simple way was presented by Nussinov algorithm, where the only criterium is whether the base is paired or not. For the Turner Energy Model to apply, the structures are decomposed into an elements called loops. These can be sorted into groups depending on their properties, for which the energy is computed in the same manner. This way the DP schemes can be broken down into a number of cases depending on the element that is treated, similarly to distinguish paired and unpaired nucleotides in scheme used in Nussinov Algorithm. Due to the independence condition, the free energy E of a secondary structure S is given by:

$$E = \sum_{l \in S} E_l \tag{2.5}$$

where l stands for loop and E_l is computed by Turner energy model for L that depends on it and potentially on neighboring substructures.

With exceptions, all loops are delimited by two types of base pairs. A single loop can contain different branchings. The base pair of a branching that leads to the 5′ end and 3′ end of the sequence is called a **closing base pair**, here denoted by (i, j), since it encloses the said loop. The pairs enclosed by loop themselves are called

opening base-pairs due to them opening other loops. These are not limited to one and they open the following substructures. Here, they are denoted (k_x, l_x). The opening base pair for a given loop is also a closing base pair for the the following one. The decomposition is illustrated on Figure 2.2A shows an example of each using different colors.

Here we list all possible loops that can appear in secondary structures in $\mathcal{S}_n$. can Each loop has its specific energy to be computed by the Turner Energy Model (See Section 2.1 for definitions). We also mention the type of an element that is not treated here due to the restrictions but may be treated later in this work.

- **An external loop.** It defines the branches and unpaired nucleotides that are not enclosed by any base pair. Consequently, there is no closing base pair in an external loop - they are delimited by $w[1]$ and $w[n]$. There is only one external loop per sequence. On radial representation, it is the linear base (cyan on Figure 2.2). It also contains one or more opening base pairs (k_x, l_x) for following substructures unless w is unfolded.
- **A hairpin**. A stretch of unpaired nucleotides delimited by a single closing base pair (i, j). It does not branch any further and consequently it represents a terminal point for given branching (red on Figure 2.2). Its energy is denoted by $E_H(i, j)$.
- **A stem**. A set of consecutive base pairs (i, j) and $(i + 1, j - 1) = (k, l)$ - a closing base pair is immediately followed by opening base pair. A multiple consecutive stems form a **helix** (blue on Figure 2.2).

  Definition 2.4.1 (Helix): We call by $\mathcal{H}(i, j, lh)$ a **helix** consisting of the base pairs (i, j), $(i + 1, j - 1) \dots (i + lh - 1, j - lh + 1)$ where lh is the length of $\mathcal{H}(i, j, lh)$ and $i + 2lh - 2 + \theta < j$. There cannot be any unpaired base within $w[i, i + lh - 1]$ and $w[j - lh + 1, j]$.

  Stems have stabilizing effect on the secondary structure, therefore a single base pair is a rather rare occurrence. Its energy is denoted by $E_{St}(i, j)$.
- **An internal loop**. A set of two strands of an unpaired nucleotides (i, j) and (k, l). The two unpaired strands must have the same length, or $j - l = k - i$ (grey on Figure 2.2).
- **A bulge**. Similar to the internal loop, except the unpaired stretches are not of the same length α and β (white color on Figure 2.2). The energy by the Turner Energy Model is computed in the same manner for both internal

loop and bulge, so for both cases we note $E_{IL}(i,j,k,l)$. In some cases a stem is also considered as a specific case of an internal loop with $\alpha = \beta = 0$. In that case we note $E_{ILG}(i,j,k,l)$ such that

$$E_{ILG}(i,j,k,l) = \begin{cases} E_{St}(i,j) & \text{if } k = i+1 \text{ and } l = j-1 \\ E_{IL}(i,j,k,l) & \text{otherwise} \end{cases}$$

- **A multibranch loop**. Multiple stretches of unpaired nucleotides delimited by single closing base pair (i,j) and by at least two opening base pairs $(k_x, l_x), x = 1..a, a \geq 2$ (yellow on Figure 2.2).

  The energy computation for the multibranch loop of the secondary structure is the most delicate part that would result to exponential complexity of the associated DP scheme. An energy of a multbranch loop, here denoted $E_M(i,j)$, depends on many factors such as the types of opening and closing base pairs and the number of branchings along with other dependencies. However, it can be reasonably approximated by a simplified Jacobson-Stockmayer expression [45][48]:

$$E_M(i,j) = a + bN_{branch} + cN_{unp} \tag{2.6}$$

  where a is the penalty for the creation of the multibranch loop, b for an unpaired base, c the penalty for the branch within it, and N_{unp} and N_{branch} the number of unpaired bases and branches within the multibranch loop respectively. The values of a, b, c are supposed to be constant. This way it is possible to decompose a multibranch loop into separate helices and compute $E_M(i,j)$ by summing up all contributions/penalties. This is why most of MFE prediction algorithms employing Turner Energy Model use this kind of decomposition.

There is another element that might appear within the secondary structure - a **pseudoknot**.

Definition 2.4.2 (Pseudoknot): A **pseudoknot** is a structure that consisting of couples of base pairs (i,j) and (k,l) for which

$$i < k < j < l.$$

In other words a pseudoknot consists of crossing base pairs. There are different types of pseudoknots of increasing complexity [77]. Due to the fact the intro-

duction of pseudoknots into a secondary structures vastly increases their number and searching for the pseudoknots with the vastest definition is proved to be an NP-complete problem [1, 58]), they will be omitted in the first part of this thesis. Later, we will talk about an algorithm that treats an H-type class of pseudoknots, a problem that can be reduced to a polynomial complexity.

2.4.3 Zuker Algorithm

The first algorithm that made the use of this decomposition alongside the Turner Energy Model was introduced by Michael Zuker *et al* [107, 106, 59]. Unlike Nussinov algorithm, it looks for minimum free energy secondary structure. The decomposition of a secondary structure S in loops necessitates the scoring function$\mathfrak{S}$ being decomposed in three values for every $ww[i,j]$:

- $W_{i,j}$ - the minimum possible energy for $S(i,j)$
- $V_{i,j}$ - the minimum possible energy for $S(i,j) \mid (i,j) \in S(i,j)$
- $WM_{i,j}$ - $w[i,j]$ is contained within a multibranch loop and contains at least one helix

The algorithm itself is given by:

$$
\begin{aligned}
W_{i,j} &= \min \begin{cases} W_{i,j-1} \\ \min\limits_{i \le k \le j-\theta-1} (W_{i,k-1} + V_{k,j}) \end{cases} \\
V_{i,j} &= \min \begin{cases} E_H(i,j) \\ E_S(i,j) + V_{i+1,j-1} \\ \min\limits_{\substack{i<k<l<j \\ k \le l-\theta-1}} (E_{IL}(i,k,l,j) + V_{k,l}) \\ \min\limits_{i<k<j} (WM_{i,k-1} + WM_{k,j} + a) \end{cases} \\
WM_{i,j} &= \min \begin{cases} WM_{i,j-1} + c \\ WM_{i+1,j} + c \\ V_{i,j} + b \\ \min\limits_{i<k<j} (WM_{i,k} + WM_{k+1,j}) \,. \end{cases}
\end{aligned}
\tag{2.7}
$$

where a, b, c are constants and E_H, E_S, E_{IL} are loop energies as defined in Section 2.4.2. For $W_{i,j}$, $w[j]$ is either unpaired or paired to some $w[k] \mid i \le k \le j-\theta-1$. For

$V_{i,j}$, either every $w[k] \mid i < k < j$ is unpaired ad (i, j) forms hairpin, $(i+1, j-1) \in S(i, j)$ forming a stack, $(k, l) \in S(i, j)$ with $i < j < k < l$ creating a bulge or internal loop or multiple pairs $(k_1, l_1), \ldots, (k_u, l_u) \in S(i, j)$ with $i < k_1 < l_1 < \cdots < k_u < l_u < l$. $VM_{i,j}$ might have either $w[i]$ or $w[j]$ unpaired $(i, j) \in S(i, j)$ or it can be split into two another helices. For $VM_{i,j}$, we add the penalty c for each unpaired base found, and sb every new base pair. The entire decomposition is illustrated on Figure 2.4.

The initialization condition is is $\forall\{i, j \mid j \leq i + \theta\}, W_{i,j} = V_{i,j} = WM_{i,j} = 0$ due to $w[i, j]$ being too short. Each step solves and stores the results of $W_{i,j}$, $V_{i,j}$ and $VM_{i,j}$ in three separate DP matrices by increasing length. At the end, the MFE is given by $W_{1,n}$, the precise secondary structure being obtainable either by memorizing the pairs that contributed to the MFE or by traceback from $W_{1,n}$.

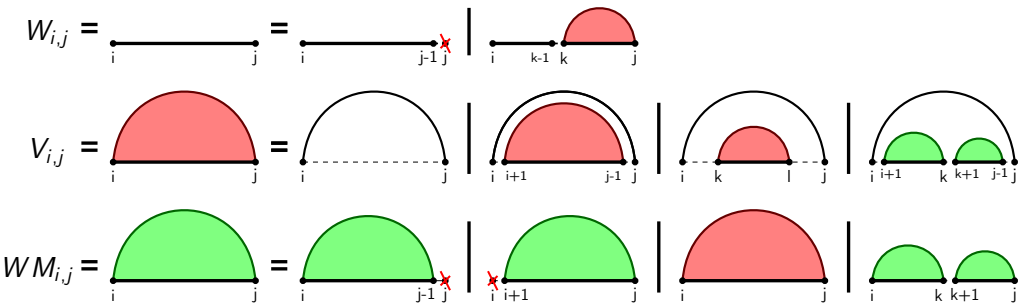

Figure 2.4: **The principle of Zuker Algorithm**. $W_{i,j}$, $V_{i,j}$ and $WM_{i,j}$ represent the minimum energy of a structure S a substring $w[i, j]$ can form, the minimum energy of $S(i, j)$ can form if $(i, j) \in S(i, j)$, and a minimum energy section of a multibranch loop with at least one helix respectively. $W_{i,j}$: for a given substring $w[i, j]$, j can be either unpaired or paired to some k such that $i \leq k \leq j - \theta - 1$. $V_{i,j}$: if (i, j) exists, it can enclose only unpaired bases, a pair $(i + 1, j - 1)$, a single or multiple base pairs. $VM_{i,j}$: The first of last base can be unpaired, $(i, j) \in S(i, j)$ or can contain more helices, in which case the substring is split in two by k.

One of major tools used for secondary structure prediction and structural analysis, ViennaRNA library [55], uses a variant of Zuker algorithm. The modification touches notably multibranch loops, where a new scoring function $M^1_{i,j}$ is introduced and $WM_{i,j}$ is replaced by $M_{i,j}$. While the first $M^1_{i,j}$ contains exactly one helix of multibranch loop within $w[i, j]$, $M_{i,j}$ can contain more of them (but at least one). The algorithm itself, also depicted on Figure 2.5 is follows below.

$$
\begin{aligned}
F_{i,j} &= \min \begin{cases} F_{i,j-1} \\ \min\limits_{i \le k \le j-\theta-1} (F_{i,k-1} + C_{k,j}) \end{cases} \\
C_{i,j} &= \min \begin{cases} E_H(i,j) \\ \min\limits_{\substack{i<k<l<j \\ k \le l-\theta-1}} (E_{ILG}(i,j,k,l) + C_{k,l}) \\ \min\limits_{i<k<j} \left(M_{i+1,k-1} + M^1_{k,j-1} \right) + a + b \end{cases} \\
M_{i,j} &= \min \begin{cases} \min\limits_{i<k<j} \left(c \times (k-i) + M^1_{k,j} \right) \\ \min\limits_{i<k<j} \left(M_{i,k-1} + M^1_{k,j} \right) \end{cases} \\
M^1_{i,j} &= \min \begin{cases} M^1_{i,j-1} + c \\ C_{i,j} + b \end{cases}
\end{aligned}
\tag{2.8}
$$

Here the multibranch loop get split into the last helix $M^1_{k,j}$ and the rest $M_{i,k-1}$. $M_{i,k-1}$ can contain either multiple helices or one helix preceded by a stretch of unpaired nucleotides. $M^1_{k,j}$ might contain unpaired bases at its $w[j]$ end. $F_{i,j}$ and $C_{i,j}$ are equivalent to $W_{i,j}$, respectively $V_{i,j}$ from the base version. The initialization remains unchanged, ie $\forall i, j, j \le i + \theta, F_{i,j} = C_{i,j} = M_{i,j} = M^1_{i,j} = 0$. $F_{1,n}$ gives the free energy of the MFE for w.

The complexity of both algorithms is $\mathcal{O}(n^4)$ in time and $\mathcal{O}(n^2)$ in space. The limiting step is the evaluation done for the internal loops/bulges, which has four different parameters i, j, k, l, hence the $\mathcal{O}(n^4)$ complexity. This complexity can be reduced to $\mathcal{O}(n^3)$ if the maximum length of its unpaired stretches α and β is limited [62], though some modifications succeeded to reduce it with said restriction [60].

In the rest of this work, we will reference this version of the algorithm as VZu (VIENNARNA-Zuker) algorithm.

2.5 From RNA thermodynamics to kinetics

The computations of MFE tell us what structure is the most stable. However, it may not necessarily correspond to the empirically observed functional structure. This may result from the approximations made in Turner Energy Model to simplify the computation, such as approximating the free energy of multibranch

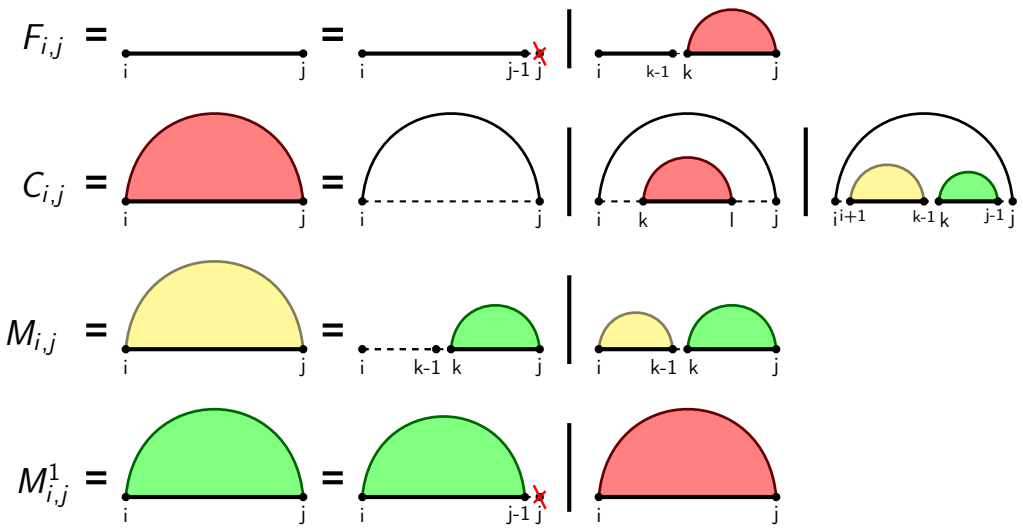

Figure 2.5: **A variant of Zuker (VZu) algorithm used in ViennaRNA library**. $F_{i,j}$ and $C_{i,j}$ are functionally identical to $W_{i,j}$, respectively $V_{i,j}$ from the original version of Turner's Algorithm. $M_{i,j}$ and $M^1_{i,j}$ score the section of a multibranched loop within $w[i,j]$ containing at least one helix and exactly one helix respectively. $M_{i,j}$ is split to a section that contains exactly one helix possibly preceeded by unpaired bases and another that is either completely unpaired or contains at least one another helix. $M^1_{i,j}$ contains one helix potentially followed by unpaired bases.

loops in Equation 2.6. This may cause the MFE to be a different structure in reality, though usually with the similar energy.

Another possibility is that before MFE can form, the RNA sequence stays trapped in secondary structure that has lower energy that the **neighboring structures**.

Definition 2.5.1 (Neighboring structure): We call a **neighboring secondary structure of** S a structure that differs from S one base pair, including base pair shifts or for breaking and recreating a base pair with at least one common nucleotide. the space of neighbors of S is denoted $\mathcal{V}_S$.

The definition of **locally optimal structure** follows.

Definition 2.5.2 (Locally optimal structure): We call a **locally optimal structure** a structure S for which

$$\mathfrak{S}(S) < \min_{S_x \in \mathcal{V}_S} (\mathfrak{S}(S_x)).$$

These structures can act as potential kinetic traps, with significant impact on the folding process.

Finally, one sequence can have more functional structures, as in the example of riboswitches [73], so finding a single MFE will not find the so-called **metastable** structure. Sometimes the metastable structure results from co-transcriptional folding, ie. the structure starts to fold before the transcription process is completed, meaning some bases are not accessible at the beginning of the folding [37].

For these reasons the scientific community became interested in **suboptimal secondary structures**. The main interest of this work is to propose a method that allows us to retrieve the suboptimal structures in an efficient manner. The utility of suboptimal structures is they can be used to create a representation that is very useful for the study of the kinetics of secondary structures, an RNA folding landscape.

2.5.1 RNA folding landscape

Definition 2.5.3 (RNA Folding landscape): An **RNA folding landscape** is the model of the space $\mathcal{S}_n$ of secondary structures $\mathcal{S}_n$ for given sequence w, visualizing their energies and eventually the transitions between them.

It can be seen as a visualization of the system constituted of sequences w folded in every secondary structure $S \in \mathcal{S}_n$. Some of these structures can change one to another, which is represented by a transition within the folding landscape. It can also model only a subset $\mathcal{S}_{s_{sub}} \subset \mathcal{S}_n$. Due to the complexity and multi-dimensionality of RNA folding landscape, it is difficult to depict the entirety of the landscape at once for longer sequences.

For this reason, most representations therefore depict only some of the folding landscape depending on the used reference. For example the Figure 2.6 shows an example of two energy landscapes for Yeast Phenylalanine tRNA (Figure 2.6 left) [100] and Class II Amynoacyl tRNA synthetase (Figure 2.6 right) [79]. The sequences were taken from PDB database [99, 8]. Here, the section of the landscape is defined by two referent structures, which in this case is MFE and some high-energy structure that is distant from the MFE. The energy of secondary structures is indicated either by a color code, such as in the provided example, or by z coordinate in case of tri-dimensional visualization. The tri-dimensional variant then

depicts all structures with higher energy than neighboring states as a peak, while locally optimal structures would be represented by a hole. The lowest point of the entire landscape would be MFE. Alternatively, the RNA folding landscape can be represented as a graph, with nodes being the structures and the edges representing the transitions. By definition, each transition is reversible.

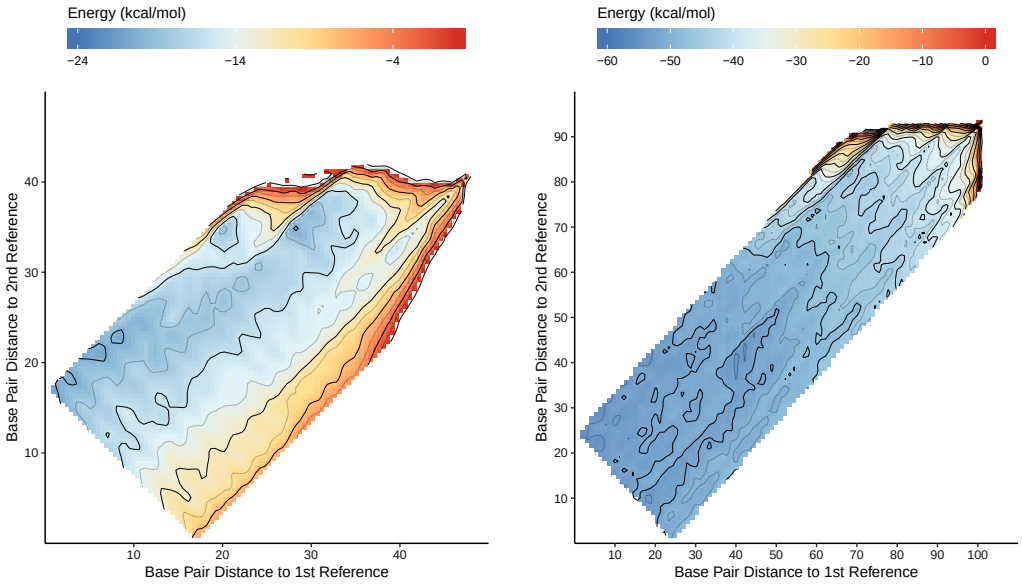

Figure 2.6: **The examples of a folding landscapes**. The folding landscape for Yeast Phenylalanine tRNA (PDB ID 1TRA, left) [100] and Class II Amynoacyl tRNA synthetase (1ASY, right) [79]. The first and second reference are respectively MFE structure and a random high-energy structure distant from MFE structure. The picture was generated by an `Rscript` script that will be possibly available with the future releases of ViennaRNA library [55].

The implication of the RNA folding landscapes in the RNA kinetics stems from the fact that besides the secondary structures themselves, it also may represent their evolution, resp. folding pathways. A **folding** is a process by which a secondary structure forms or changes from one to another. Here, we see the folding as a hierarchical process where an RNA sequence changes its structure to one of the neighboring states. The folding pathway is then a path from RNA folding landscape consisting of secondary structures which RNA adopts when folding from one secondary structure to another. The objective of RNA kinetics is to study these folding pathways and the evolution of an RNA sequence, something for what the folding landscape is a great tool.

The methods to generate and study the RNA folding landscape can be divided into two categories:

- **Step-by-step methods:** The folding process is modeled as a Markov chain and simulated at a certain resolution, usually on base-pair [28] or helix [22] level. However, since the number of the secondary structures raises exponentially with the length of sequence [106] and the number of folding pathways also quickly raises with it [29], this type of analysis becomes quickly infeasible.
- **Coarse-grained methods:** The main idea is to simplify the folding landscape first. This necessitates to identify the key elements - **key landmarks** - of the folding landscape, the most important secondary structures. These are connected together in a manner to accurately represent the kinetics of a given RNA sequence. Such representation is then used to perform an analysis of RNA kinetics. Due to reduced number of structures and folding pathways, this strategy is applicable even to longer RNA sequences. However, its quality greatly depends on that of first step, since the omission of certain key structures might have considerable impact on the analysis of kinetics and probably result in biased observations.

There is a number of currently existing approaches that generate these key landmarks [104][54][50][56] and assemble them into a folding landscape [30][50]. Our work mainly concentrates on establishing a method capable of **selecting the key landmarks** of RNA folding landscapes in a more efficient manner than currently existing methods.

2.5.2 Selecting suboptimal secondary structures

For kinetic analysis to be possible for long RNAs, the folding landscape must be reduced to key suboptimal structures. The question is how to define and select them. This step is instrumental due to having the instrumental impact on the quality of the resulting landscape and adjacent analysis.

There are multiple strategies that we can chose from. The first one is to select only structures within a certain energy range from the MFE. Due to imprecisions of computations by Turner Energy Model, the computed MFE might not be MFE in reality, but it will not be too much far off. Second reason, the energy structures close to MFE are more stable and consequently a better candidates for a functional structure. However, there have been an exceptions to this, such as an artificially designed riboswitch where the energy difference between MFE and its metastable, functionally important, structure is almost 30 kcal.mol^{-1} [42]. Another

thing to consider is that the number of structures selected this way still raises exponentially with n.

Another strategy is to select only suboptimal RNA secondary structures S that are locally stable. The locally stable structures are considered more likely to be a candidates for an important functional structure due to the energy barriers that need to be surpassed for S to be changed, granting it a certain stability and allowing to sequence to exert its function. On the other hand even if the locally optimal secondary structure is not functionally important, it still can be crucial transitive state with the impact on the folding path and by extension on the result. For this reason, in the first part of this thesis we will work with locally optimal structures.

Unfortunately, like for the selection by energy band, the number of locally optimal structures also raises exponentially with n and since the folding landscape presents a high number of them, it can be problematic. Therefore, it is necessary to be able to select the best candidates among them. To achieve this, we resort to the sampling, which has for objective to select the most suitable locally optimal secondary structures according to a given criterium.

2.5.3 Boltzmann Distribution

The selection happens at two levels. The first is the selection of a subspace of $\mathcal{S}_n$ (all secondary structures, only locally optimal ones etc.), while the second is to choose its best candidates. The second selection can be done either in deterministic or stochastic, random manner. Here we will concentrate on the stochastic sampling methods.

To perform a stochastic sampling, it is necessary to have a distribution of the entire space we sample from. The distribution is computed according to the value of $\mathfrak{G}(S)$ by which we want the results to be filtered, such as the lowest free energy or biggest number of base pairs.

The distribution of secondary structures that is frequently used in sampling of RNA secondary structures is the **Boltzmann distribution**, sometimes also called **Gibbs distribution**. This distribution was formulated by Ludwig Boltzmann [85] and later studied by Josias W. Gibbs [34]. The Boltzmann distribution is used mainly in thermodynamics and is a probability distribution of various systems over different possible states.

Definition 2.5.4 (Boltzmann factor): et the system be the sequences w and the states the secondary structures $S \in \mathcal{S}_n$,each with free energy E_S. The Boltzmann factor $\mathcal{Z}_S$ of S is

$$\mathcal{Z}_S = e^{\frac{-E_S}{k_B T}} \tag{2.9}$$

where T is an absolute temperature and k_B is Boltzmann constant or perfect gas constant.

In our work, we consider $k_B \approx 1.987 \times 10^{-3}$ kcal.mol^{-1}.K^{-1}.

Definition 2.5.5 (Partition function): partition function $\mathcal{Z}$ is a description of the statistical properties of the system, here RNA in ensemble of states from $\mathcal{S}_n$, in a thermodynamic equilibrium. It is computed as

$$\mathcal{Z} = \sum_{S \in \mathcal{S}_n} e^{\frac{-E_S}{k_B T}}. \tag{2.10}$$

From this expression and the equation 2.9, we can obtain the probability of observing S.

Definition 2.5.6 (Boltzmann probability): The probability $\mathbf{p(S)}$ of observing w folded in $S \in \mathcal{S}_n$ under the assumption of Boltzmann distribution is:

$$p(S) = \frac{\mathcal{Z}_S}{\mathcal{Z}} = \frac{e^{\frac{-E_S}{k_B T}}}{\sum_{\mathcal{X} \in \mathcal{S}_n} e^{\frac{-E_{\mathcal{X}}}{k_B T}}} \tag{2.11}$$

These probabilities can be used to sample S in a random manner. Since S with low energy have higher $\mathcal{Z}_S$, they are more likely to be chosen, while the high energy structures still have non-zero probability to be selected since $\forall S, \mathcal{Z}_S > 0$. While priority is given to low-energy S, getting S with higher energy is possible.

The first algorithm that computed the partition function for ensemble of secondary structures was John S. McCaskill [62]. He used the VZu algorithm (Equation 2.8) with transforming energy terms to partition function.

2.5.4 Completeness, Unambiguity and Acyclicity

Here we rapidly talk about three main conditions that need to be fulfilled by DP scheme that is used in algorithms computing the partition function. Both will be more detailed later. These two conditions are:

- **Completeness**: The space $\mathcal{S}_n$ or its defined subsection is completely covered by the DP scheme, ie. there is not structure S that cannot be decomposed by said DP scheme while $S \in \mathcal{S}_n$.
- **Unambiguity**: Each secondary structure can be decomposed by the decomposition scheme in a single manner.
- **Acyclicity**: A DP scheme cannot present a decomposition where the element would be decomposed to itself and some other elements.

The DP scheme of Zuker and VZu algorithm presented here (Equation 2.8 and 2.7) completely covers $\mathcal{S}_n$ - each nucleotide can be paired or not (as long as θ is respected), each loop can contain zero, one or more helices and each multibranch loop has to contain at least two helices possibly preceded and followed by unpaired bases. Note that the completeness depends on the definition of $\mathcal{S}_n$ - the Zuker and VZu algorithms are complete when $\mathcal{S}_n$ is defined as in Section 2.1, but not if pseudoknots are allowed.

In regards to unambiguity, the DP scheme of Zuker algorithm does not fulfill it, and is main reason why the second version was established. Suppose we have a segment $WM_{i,j}$ containing two helices $V_{i,k-1}$ and $V_{k+1,j}$ separated by a single unpaired nucleotide k. There are two ways how this local secondary structure can be decomposed:

$$WM_{i,j} \rightarrow WM_{i,k} + V_{k+1,j} \rightarrow V_{i,k-1} + V_{k+1,j}$$
$$WM_{i,j} \rightarrow V_{i,k-1} + WM_{k,j} \rightarrow V_{i,k-1} + V_{k+1,j}$$

This is not problematic when searching for MFE, since the minimum structure will be returned anyway, but when computing Partition function its Boltzmann factor will appear twice, introducing bias. For the VZu algorithm we have:

$$WM_{i,j} = M_{i,k} + M^1_{k+1,j} \rightarrow M^1_{i,k} + C_{k+1,j} \rightarrow M^1_{i,k-1} + C_{K+1,j} \rightarrow C_{i,k-1} + C_{k+1,j}$$

Note that $WM_{i,j}$ is not present in VZu algorithm, it is just to point out the equivalence between the two versions. The second version respects unambiguity and can be used to compute the partition function, which is the base of McCaskill algorithm.

As for acyclicity, the presence of a cycle in DP scheme implies an infinite number of the possible decompositions. This makes their adaptation to scoring algorithms impossible. One possibility of verifying an acyclicity is therefore to see whether each decomposition can be done in finite number of steps. All of scoring algorithms presented in this work are acyclic.

2.5.5 McCaskill Algorithm

The partition function shows an interesting property with the respect to the decomposition of loop and the energies deduced from the Equation 2.5. Since E_S is the sum of the energies of each of its loops E_l, for Boltzmann factor we have:

$$\mathcal{Z}_S = e^{\left(\frac{\sum_{\mathcal{L}\in S} -E_l}{k_B T}\right)} = \prod_{l\in S} e^{\frac{-E_l}{k_B T}} = \prod_{l\in S} \mathcal{Z}_l \tag{2.12}$$

Consequently, the Boltzmann factor of a single structure can be decomposed into the product of contributions of each of the loops l. This can be used to transform the VZu algorithm into McCaskill algorithm. However, since in this case we want to compute the Partition function, we also need to sum the contribution of all secondary structures. We define:

- $\mathcal{Z}^F_{i,j}$: Partition function over all $S(i,j) \in \mathcal{S}(i,j)$
- $\mathcal{Z}^C_{i,j}$: Partition function over $\{S(i,j) | S(i,j) \in \mathcal{S}(i,j), (i,j) \in S(i,j)\}$)
- $\mathcal{Z}^M_{i,j}$: Partition function over all $S(i,j) \in \mathcal{S}(i,j)$ where $S(i,j)$ is a part of multibranch loop containing at least one helix
- $\mathcal{Z}^{M1}_{i,j}$: Partition function over all $S(i,j) \in \mathcal{S}(i,j)$ where $S(i,j)$ is a part of multibranch loop containing exactly one helix

These states are deduced from $F_{i,j}$, respectively $C_{i,j}$, $M_{i,j}$, and $M^1_{i,j}$ of VZu algorithm. The algorithm of McCaskill is then:

$$
\begin{aligned}
\mathcal{Z}^{F}_{i,j} &= \mathcal{Z}^{F}_{i,j-1} + \sum_{i \le k \le j-\theta-1} \mathcal{Z}^{F}_{i,k-1} \times \mathcal{Z}^{C}_{k,j} \\
\mathcal{Z}^{C}_{i,j} &= e^{\frac{-E_H(i,j)}{k_B T}} + \sum_{\substack{i<k<l<j \\ k \le l-\theta-1}} e^{\frac{-E_{ILG}(i,j,k,l)}{k_B T}} \times \mathcal{Z}^{C}_{k,l} + \sum_{i<k<j} e^{\frac{-a-b}{k_B T}} \mathcal{Z}^{M}_{i+1,k-1} \times \mathcal{Z}^{M1}_{k,j-1} \\
\mathcal{Z}^{M}_{i,j} &= \sum_{i<k<j} e^{\frac{-c \times (k-1)}{k_B T}} \mathcal{Z}^{M1}_{k,j} + \sum_{i<k<j} \mathcal{Z}^{M}_{i,k-1} \times \mathcal{Z}^{M1}_{k,j} \\
\mathcal{Z}^{M1}_{i,j} &= e^{\frac{-c}{k_B T}} \times \mathcal{Z}^{M1}_{i,j-1} + e^{\frac{-b}{k_B T}} \times \mathcal{Z}^{C}_{i,j}
\end{aligned}
\tag{2.13}
$$

with $E_H(i,j), E_{ILG}, a, b, c$ having the unchanged significance. The sums are explained by the need of summing up all contributions of each S. Since $\forall w[i,j] \mid j \le i + \theta$ the initial conditions are $E(i,j) = 0$, we have

$$
\forall \{i,j \mid j \le i+\theta\}, \mathcal{Z}_{i,j} = \mathcal{Z}^{B}_{i,j} = \mathcal{Z}^{M}_{i,j} = \mathcal{Z}^{M1}_{i,j} = 1.
$$

The value $\mathcal{Z}_{1,n}$ equals to $\mathcal{Z}$. In the original paper [62], a partition function $\mathcal{Z}$ is instead denoted as $\mathcal{Q}$, which has the same significance. The complexity is the same as for the VZu Algorithm - $\mathcal{O}(n^4)$ in time, which becomes $\mathcal{O}(n^3)$ if the maximum size of the internal loop becomes limited, and $\mathcal{O}(n^2)$ in space.

The algorithms computing $\mathcal{Z}$ of $\mathcal{S}_n$ or its part like that of McCaskill are the first step towards the sampling of secondary structures.

2.5.6 Sampling and Stochastic backtrack

The sampling consists of the random selection of the secondary structures S from the space $\mathcal{S}_n$ according to the probability distribution, in our case the Boltzmann distribution. The sampling algorithm employing McCaskill algorithm was first introduced by the works of Y. Ding and C.E. Lawrence [24]. It consists of two steps:

- **Pre-computation**: The computation of $\mathcal{Z}$. More importantly, the matrices $\mathcal{Z}^F, \mathcal{Z}^C, \mathcal{Z}^M$ and $\mathcal{Z}^{M1}$ are filled during this step, therefore $\forall i,j, 1 \le i < j \le n$, the values $\mathcal{Z}^F, \mathcal{Z}^C, \mathcal{Z}^M$ and $\mathcal{Z}^{M1}$ are calculated.
- **Stochastic backtrack**: Performed once to generate a single S. The principle is similar to the backtrack as presented in Section 2.3.3, however instead of searching for a combination of a loop that gives a value given a matrix, we have a possibility to choose a certain loop l. Suppose we have $w[i,j]$

and a possibility to choose a base pair, an unpaired base, a splitting point or a loop q_1 with a probability pair $p(q_1)$ that occupies some bases and leaves substrings $w[i_1, j_1], w[i_2, j_2] \cdots w[i_x, j_x]$ untouched. The value of $p(q_1)$ is computed from DP matrices. If q_1 is chosen, then we can choose q_2 in the same manner for $w[i_1, j_1]$ with the probability $p(q_2)$ and so on. The process is repeated until the substrings too short to contain a base pair. When no substring is left, the secondary structure is complete.

The algorithm must give a way to access the probability values and these have to be stored in DP matrices. In the case of McCaskill algorithm, they can be obtained from Equation 2.13 where left-hand term represents the weight of all options for $w[i, j]$ and the right-hand terms a list of choices available to be taken, giving access to $p(q)$.

Since the introduction of sampling by Ding and Lawrence, other approaches have appeared, usually employing Turner Energy Model. These include a decomposition of a structure in stacks [54] or by varying a temperature during sampling [50].

2.6 Formalization of Dynamic Programming Schemes

Since we just presented the state-of-the-art DP schemes employed for secondary structure prediction, it is a good moment to propose the formalism that would help us to generalize and unify them.

2.6.1 Definitions

Objective 2.6.1. *Introduce a formalism that allows to describe DP schemes in an unified manner and allowing to explicitly manipulate the different computation steps. This formalism*

- *takes a DP scheme with all possible states and an entry state* q_{root} *as an entry*
- *returns, for each state the associated accessible search space and scoring function expressed in an unified language that can be commonly used for DP schemes evaluation RNA secondary structures.*

To do this, it is first necessary to identify the unitary actions of such DP scheme and define them in a broadest way possible. To avoid a confusion, we will,

throughout this section, demonstrate how to apply it to formalize the Nussinov algorithm. The formalism presented here is inspired by and extends that introduced by Yann Ponty *et al* in relation to the representation of secondary structures as hypergraphs and pseudoknot analysis [74].

A DP scheme relates the (optimal) score for a given subproblem, on the left-hand side of the equation, to the combination of scores achieved over the selected subset of subproblems on its right-hand side.

Definition 2.6.1 (DP scheme): A **DP scheme** is a tuple $(\mathcal{Q}, q_{root}, \rho)$ such that:

- $\mathcal{Q}$ is the set of states q, such as the couple of base pairs (i, j) associated with the right-hand-side terms of the scheme;
- $q_{root} \in \mathcal{Q}$ is the initial state of the scheme, such as $(1, n)$;
- $\rho : \mathcal{Q} \rightarrow (\mathcal{Q}^* \times \mathcal{D} \times \mathbb{R})^*$ is a function associating to each state q a set of derivations relating q to other states, with each derivation having attributed a constructor λ, such as an unpaired base, and a score to it.

Therefore a DP scheme relates the different states by a set of derivations. The notion of the derivation is also connected to that of constructors. We consecutively introduce all of these terms.

Definition 2.6.2 (State): A given subproblem is, in proposed formalism, characterized by a **state** $\mathbf{q} \in \mathcal{Q}$, where $\mathcal{Q}$ denotes the set of all states with associated subproblems.

Therefore a given problem is modeled by a state q and its solution is indicated by a score over this state. The computation of a DP scheme total score requires a memorization of $|\mathcal{Q}|$ scores, pointing to an overall memory complexity of $\mathcal{O}(|\mathcal{Q}|)$.

Example 2.6.1. For Nussinov algorithm, the states are any (i, j) such that $1 \leq i < j \leq n$, where (i, j) is the couple of, not necessarily paired, nucleotides and we have

$$\mathcal{Q} = \{(i, j) \mid 1 \leq i < j \leq n\}.$$

Each subproblem is completely defined by the nucleotide couple (i, j), therefore this definition of Nussinov states is sufficient.

Besides the score, each state q has a set of secondary structures s, that accessible

from this state, associated to it.

Definition 2.6.3 (Search space): A **search space** is set of secondary structures s that are associated to a state $q \in \mathcal{Q}$ and can be **accessed** by the DP scheme from the sets of $q'_1, \ldots, q'_u \in \mathcal{Q}$. We denote it by $\mathcal{L}(q)$.

Note that $\mathcal{L}q$ does not contain only the structures that are merely visited, but all those that can be accessed from q. Also these structures do not have to have the length of n. For this reason we denote $s \in \mathcal{L}(q)$ a secondary structure belonging to a search space of q.

Example 2.6.2. A search space for Nussinov algorithm and sequence w, $\mathcal{L}(q)$ where $q = (1, 5)$ contains all secondary structures that can be generated on $w[1, 5]$.

A structure $S \in \mathcal{L}(q)$ can be generated from some structures

$$(s'_1, \ldots, s'_u) \in (\mathcal{L}(q'_1), \ldots, \mathcal{L}(q'_u))$$

if q and $q'_1, \ldots, q'_u$ are related by the derivation.

Definition 2.6.4 (Constructor): A **constructor** is a function $\lambda : \mathcal{S}_n^k \to \mathcal{S}_n, k \in \mathbb{N}$, that generates a structure $\mathbf{s}$ from a vector of k (sub)structures $\mathbf{s'_1, \ldots, s'_k}$.

By abuse of notation, we might note by $\lambda(\mathcal{X}, \mathcal{Y})$ the construction $\lambda(X, Y)$, with $X \in \mathcal{X}, Y \in \mathcal{Y}$. A **constructor space** is denoted $\mathcal{D}$, the notation used in Definition 2.6.1.

Example 2.6.3. The Nussinov algorithm employs two different constructors: The constructor $\lambda^{unp}_{i,j}$ where the secondary structure s is such that $w[j]$ is unpaired and is applied to $s' \in \mathcal{L}((i, j-1))$, and the constructor $\lambda^{pair}_{i,j,k}$, where s is generated from substructures for states $(i, k-1)$ and $(k+1, j-1)$ and $(k, j) \in \phi_{pair}(w)$. In both cases i, j indicate the interval for which the constructor is employed, while in the second case k denotes the pairing partner to $w[j]$. The operator space is

$$\mathcal{D}_{Nus} = \{\lambda^{unp}_{i,j}, \lambda^{pair}_{i,j,k} \mid 1 \leq i \leq k < j \leq n\}$$

The constructor is attributed to a specific derivation.

Definition 2.6.5 (Derivation): A **derivation** $q \xrightarrow[c_\lambda]{\lambda} q'_1, \ldots, q'_u$ is a function that associates a tuple $(\{q_1, \ldots, q_u\}, \lambda, c)$ to a state q, representing a derivation from q to $q'_1 \cdots q'_u$ by λ and having cost c_λ.

Therefore there are three layers to each derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$:

- Derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ itself, relating the state q to $q'_1, \ldots, q'_u$;
- Construction of accessible search spaces, done by λ on $\{\mathcal{L}(q'_1), \ldots, \mathcal{L}(q'_u)\}$;
- Computing the score of q, from scores of $q'_1, \ldots, q'_u$ and using the function employing c_λ.

Note that some derivations might relate q to $\varnothing$, in which case we call them **terminal derivations**.

Each derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ corresponds to a specific case. In fact, there are two scopes that can be distinguished for a given DP scheme:

- **Cases:** One DP scheme can include multiple cases on its right-hand side. These cases correspond to the derivations. In the case of Nussinov algorithm, there is one case for $w[j]$ being unpaired and a case for each pair (k, j) that can exist.
- **Subproblems:** Single derivation relates a single state q on the left-hand side to states $q'_1, \ldots, q'_u$ or subproblems. For $\lambda^{unp}_{i,j}$ in the case of Nussinov algorithm, it relies one state on the left-hand side to one state on the right-hand side of the DP equation. In the case of $\lambda^{pair}_{i,j,k}$, it is one and two states respectively.

Therefore, $\boldsymbol{\rho(q)}$ from the Defintion 2.6.1 is the set of derivations of form

$$q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$$

that relate q to $q_1, \ldots, q_u$ and generate $s \in \mathcal{L}(q)$ by constructor λ. The score of q is computed from the scores of $q_1, \ldots, q_u$ using value c_λ. Alternatively we can also denote that

$$(\{q'_1, \ldots, q'_u\}, \lambda, c_\lambda) \in \rho(q).$$

Example 2.6.4. We demonstrate that the DP scheme can be completely captured by the proposed formalism $(\mathcal{Q}, q_{root}, \rho)$. Assume that $\mathcal{Q}$, resp. $\mathcal{D}$ contain all q, resp. λ that appear in given DP scheme. Each DP equation relates a single state $q \in \mathcal{Q}$ on left-hand side and a number of different cases each relating a number of states $q'_1, \ldots, q'_u \in \mathcal{Q}^u$ on the right-hand side. Such case is completely defined by the derivation:

$$q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$$

where c_λ is the energy bonus/penalty for using the constructor λ. The DP scheme is expressed by a union of all cases:

$$\rho(q) = \bigcup_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}} (\{q'_1, \ldots, q'_u\}, \lambda, c_\lambda)$$

Therefore as long as necessary λ and q are available, it is possible to define any derivation $q \xrightarrow[c_\lambda]{\lambda} \{q_1, \ldots, q_u\}$ and all of its cases by $\rho(q)$.

Example 2.6.5. In the case of the Nussinov algorithm, $\rho(q)$ two following derivations are possible:

- $(i, j) \xrightarrow[0]{\lambda^{unp}_{i,j}} (i, j-1)$ - $w[j]$ is unpaired;
- $q \xrightarrow[1]{\lambda^{pair}_{i,j,k}} \{(i, k-1), (k+1, j-1)\}$ where $i \leq k < j$ - $w[j]$ is paired to $w[k]$

Consequently, the ρ_{Nus} is:

$$\rho_{Nus}((i,j)) = \bigcup \begin{cases} (\{(i, j-1)\}, \lambda^{unp}_{i,j}, 0) \\ \qquad \text{if } w[j] \text{ is unpaired} \\ \\ \bigcup\limits_{\substack{i \leq k < j \\ (k,j) \in \phi_{pair}(w)}} \{(\{(i, k-1), (k+1, j-1)\}, \lambda^{pair}_{i,j,k}, 1)\} \end{cases}$$

Note that $\rho(q)$ only contains the associations that are possible;

$$q \xrightarrow[c_{\lambda^{pair}_{i,j,k}}]{\lambda^{pair}_{i,j,k}} \{(i, k-1), (k+1, j-1)\}$$

does not exist if $w[k]$ and $w[j]$ cannot be paired.

All possible structures $s \in \mathcal{L}(q)$ are generated by applying the constructor λ from derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ on $(s'_1, \ldots, s'_u) \in (\mathcal{L}(q'_1), \ldots, \mathcal{L}(q'_u))$.

Definition 2.6.6 (Generation of search space): The search space $\mathcal{L}(q)$ can be generated by performing every possible derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$:

$$\mathcal{L}(q) = \bigcup_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}} \lambda(\mathcal{L}(q'_1), \ldots, \mathcal{L}(q'_u))$$

At any moment $\forall q, \mathcal{L}(q) \subseteq \mathcal{S}$.

With the search space $\mathcal{L}(q)$ and its construction defined, we have all elements to formalize the notion of completeness and unambiguity.

Definition 2.6.7 (Completeness): Let q_{root} be an initial state and $\mathcal{S}_n$ the space of all secondary structures of size n. The **completeness** implies:

$$\mathcal{L}(q_{root}) = \mathcal{S}_n \tag{2.14}$$

The completeness implies that every $S \in \mathcal{S}_n$ is generated at one point or another. The space $\mathcal{S}_n$ also might be restricted by specific conditions. In that case the completeness implies that the entirety of the restricted space is generated by DP scheme.

The unambiguity implies that every secondary structure is decomposed in an unique manner.

Definition 2.6.8 (Unambiguity): Let $x = q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ and the $\mathcal{L}(x) = \lambda(\mathcal{L}(q'_1), \ldots, \mathcal{L}(q'_u))$. A DP scheme is unambiguous that if $x = q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ and $y = q \xrightarrow[c_{\lambda'}]{\lambda'} \{q''_1, \ldots, q''_v\}$ are two derivations of the state q such that $x \neq y$ then $\mathcal{L}(x) \cap \mathcal{L}(y) = \varnothing$.

In other words, said structure cannot be decomposed in multiple different ways, or the same s cannot be generated by two different λ_1 and λ_2.

There is also an important condition of acyclicity of DP scheme. If DP scheme presents a cycle, it cannot be adapted for a scoring algorithm since it presents an infinite number of decompositions.

Definition 2.6.9 (Acyclicity): The DP scheme is acyclic if for any given sequence of derivations $\lambda(\mathcal{L}(q'_1), \ldots, \mathcal{L}(q'_u))$, each q is derived at most once.

In other words, no state can appear twice for the same decomposition.

2.6.2 Applications

Assume $c(s)$ is the score of s. From $\rho(q)$, we can define the function that computes the final score for every state $q \in \mathcal{Q}$. As previously seen, there are two scoring strategies that are applied to DP schemes.

Problem 2.6.1:

- ***Input:*** *DP scheme* $(\mathcal{Q}, q_{\text{root}}, \rho)$*;*
- ***Output:*** $\texttt{MinScore}(q)$ *such that:*

$$\text{MinScore}(q) = \min_{s \in \mathcal{L}(q)} (c(s)) .$$

The problem for maximization scoring is similar:

$$\text{MaxScore}(q) = \max_{s \in \mathcal{L}(q)} (c(s)) .$$

Such scoring strategy is applied by Nussinov (maximization) [68] or Zuker (minimization) [107] algorithms. Here we present its global formalization.

With the scoring established, we can proceed to its formulate its computation using DP schemes.

Theorem 2.6.1 (Minimization/maximization scoring):
Let $(\mathcal{Q}, q_{\text{root}}, \rho)$ *be a complete and unambiguous DP scheme.* $\texttt{MinScore}(q)$ *can be computed in* $\mathcal{O}(|\mathcal{Q}|+|\mathcal{D}|)$ *time and* $\mathcal{O}(|Q|)$ *memory by the following recurrence:*

$$\text{MinS}(q) = \min_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1,\ldots,q'_u\}} \left(c_\lambda + \sum_{i=1}^{u} \text{MinS}(q'_i) \right) \tag{2.15}$$

The maximum score $\texttt{MaxS}(q)$ *is computed in* $\mathcal{O}(|\mathcal{Q}|+|\mathcal{D}|)$ *time and* $\mathcal{O}(|Q|)$ *memory by the following recurrence:*

$$\text{MaxS}(q) = \max_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1,\ldots,q'_u\}} \left(c_\lambda + \sum_{i=1}^{u} \text{MaxS}(q'_i) \right) \tag{2.16}$$

Proof 2.6.1. We demonstrate this expression by induction on the maximum number of derivations M that can be done on q.

Base case ($M = 1$): Consider a state q where all derivations take the form

$$q \xrightarrow[c_{\lambda_{\text{term}}}]{\lambda_{\text{term}}} \varnothing.$$

$\mathcal{L}(q)$ contains a set of structures s where each of them is generated uniquely by λ_{term} on q. Consequently, the score $c(s)$ is

$$c(s) = c_{\lambda_{\text{term}}}.$$

We assume completeness, $\mathcal{L}(q)$ is completely generated by λ_{term} from all associated derivations described above. Therefore, the minimum for q is given by

$$\text{MinS}(q) = \min_{q \xrightarrow[c_{\lambda_{\text{term}}}]{\lambda_{\text{term}}} \varnothing} c(s) = \min_{s \in \mathcal{L}(q)} c(s).$$

It follows that $\text{MinS}(q) = \text{MinScore}(q)$.

Induction ($M = m$): Assume that $\forall i \in \mathbb{Z}^+$, the state q'_i, can be derived at most $m-1$ times and that we have

$$\text{MinS}(q'_i) = \text{MinScore}(q'_i).$$

Consider q and the derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$. Consequently, q can be derived at most m times. We have:

$$\mathrm{MinS}(q) = \min_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}} \left(c_\lambda + \sum_{i=1}^{u} \min_{s' \in \mathcal{L}(q'_i)} \big(c(s')\big) \right).$$

The term $\sum_{i=1}^{u} \min_{s' \in \mathcal{L}(q'_i)}(c(s'))$ can be also interpreted as a lowest sum of free energies of the combination $(s'_1, \ldots, s'_u) \in (\mathcal{L}(q'_u), \ldots, \mathcal{L}(q'_u))$, which leads us to

$$\sum_{i=1}^{u} \min_{s' \in \mathcal{L}(q'_i)} \big(c(s')\big) = \min_{\substack{(s'_1, \ldots, s'_u) \in \\ (\mathcal{L}(q'_u), \ldots, \mathcal{L}(q'_u))}} \left(\sum_{i=1}^{u} c(s'_i) \right).$$

Assume now we derive q by some derivation $q \xrightarrow[c_{\lambda'}]{\lambda'} \{q'_1, \ldots, q'_{u'}\}$. We get

$$c_\lambda + \sum_{i=1}^{u'} \min_{s' \in \mathcal{L}(q'_i)} \big(c(s')\big) = c_{\lambda'} + \min_{\substack{(s'_1, \ldots, s'_{u'}) \in \\ (\mathcal{L}(q'_{u'}), \ldots, \mathcal{L}(q'_{u'}))}} \left(\sum_{i=1}^{u'} c(s'_i) \right).$$

This term returns the minimal value corresponding to some combination of $(s'_1, \ldots, s'_{u'}) \in (\mathcal{L}(q'_{u'}), \ldots, \mathcal{L}(q'_{u'}))$ and the contribution of $c_{\lambda'}$ of derivation $q \xrightarrow[c_{\lambda'}]{\lambda'} \{q'_1, \ldots, q'_{u'}\}$. Therefore

$$c_{\lambda'} + \min_{\substack{(s'_1, \ldots, s'_{u'}) \in \\ (\mathcal{L}(q'_{u'}), \ldots, \mathcal{L}(q'_{u'}))}} \left(\sum_{i=1}^{u'} c(s'_i) \right) = c(s_{\lambda'})$$

where $c(s_{\lambda'}) = \lambda'(s'_1, \ldots, s'_{u'}) \in \lambda'(\mathcal{L}(q'_{u'}), \ldots, \mathcal{L}(q'_{u'}))$ that has minimum score along all secondary structures generated by λ', and $s_{\lambda'} \in \mathcal{L}(q)$. This applies to every $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_{u'}\}$. Since $c(s_{\lambda'})$ is minimal for $q \xrightarrow[c_{\lambda'}]{\lambda'} \{q'_1, \ldots, q'_u\}$, we can follow by

$$\mathrm{MinS}(q) = \min_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}} (c(s_\lambda))$$

There is another condition in DP scheme $(\mathcal{Q}, q_{\mathrm{root}}, \rho)$ being complete. In that case the application of every constructor λ generates $\mathcal{L}(q)$. Consequently, we can write

$$\mathrm{MinS}(q) = \min_{\mathcal{L}(q)} (c(s)) .$$

If follows that $\mathrm{MinS}(q) = \mathrm{MinScore}(q)$. The demonstration for $\mathrm{MinS}(q)$ is done in the same way.

Example 2.6.6. Nussinov algorithm is the base pair maximization DP scheme. The maximizing function can be split in cases depending whether $w[j]$ is paired. The maximum number of base pairs for $S(i,j) \in \mathcal{L}((i,j))$ can be computed as

$$\begin{aligned} \mathrm{MaxS}((i,j)) = \max(& \max_{q \xrightarrow[1]{\lambda^{pair}_{i,k,j}} \{(i,k-1),(k+1,j-1)\}} (\mathrm{MaxS}((i,k-1)) + \\ & + \mathrm{MaxS}((k+1,j-1)) + 1), \mathrm{MaxS}((i,j-1))) \end{aligned} \tag{2.17}$$

We can define the similar problem for the computation of the partition function.

Problem 2.6.2:

- ***Input:*** *DP scheme* $(\mathcal{Q}, q_{root}, \rho)$*;*
- ***Output:*** $\mathrm{PfScore}(q)$ *such that:*

$$\mathrm{PfScore}(q) = \sum_{s \in \mathcal{L}(q)} \left(e^{\frac{-c(s)}{k_B T}} \right) .$$

As an example, the partition function is computed by McCaskill algorithm [62]. The formalization of partition function scoring of DP scheme $(\mathcal{Q}, q_{root}, \rho)$ is similar as in the case of the score minimization/maximization.

Theorem 2.6.2 (Partition function scoring): *Let* $(\mathcal{Q}, q_{root}, \rho)$ *be a complete and unambiguous DP scheme.* $\mathrm{PfScore}(q)$ *is computed in* $\mathcal{O}(|\mathcal{Q}| + |\mathcal{D}|)$ *time and* $\mathcal{O}(|Q|)$ *memory by the following recurrence:*

$$\mathrm{PfS}(q) = \sum_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1,\ldots,q'_u\}} e^{\frac{-c_\lambda}{k_B T}} \times \left(\prod_{i=1}^{u} \mathrm{PfS}(q'_i) \right) \tag{2.18}$$

Proof 2.6.2. The demonstration can be made by induction on the maximum number M of successive derivations that can be performed on q.

Base case ($M = 1$): Consider any state q that can be derived maximum once. Therefore every derivation takes a form

$$q \xrightarrow[c_{\lambda_{term}}]{\lambda_{term}} \varnothing.$$

$\mathcal{L}(q)$ contains structures composed of an element generated by λ_{term}. The Boltzmann factor of a single s is then:

$$\mathcal{Z}_s = e^{\frac{-c_{\lambda_{term}}}{k_B T}} = e^{\frac{s}{k_B T}}.$$

If we assume completeness, the set of s generated by all λ_{term} gives $\mathcal{L}(q)$. Therefore

$$\mathrm{PfS}(q) = \sum_{q \xrightarrow[c_{\lambda_{term}}]{\lambda_{term}} \varnothing} e^{\frac{-c_{\lambda_{term}}}{k_B T}} = \sum_{s \in \mathcal{L}(q)} e^{\frac{-c(s)}{k_B T}}.$$

since we assume unambiguity, meaning is no more derivation per the same structure than one. It follows that $\mathrm{PfS}(q) = \mathrm{PfScore}(q)$.

Induction ($M = m$): Consider, $\forall i \in \mathbb{Z}^+$, the states q' that can be derived maximum $m - 1$ times and that they satisfy $\mathrm{PfS}(q') = \mathrm{PfScore}(q')$. Assume now a state q derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$. This implies that q can be derived at most m times. We have

$$\mathrm{PfS}(q) = \sum_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}} \left(e^{\frac{-c_\lambda}{k_B T}} \times \prod_{i=1}^{u} \left(\sum_{s' \in \mathcal{L}(q'_i)} e^{\frac{-c(s')}{k_B T}} \right) \right).$$

The term

$$\prod_{i=1}^{u} \left(\sum_{s' \in \mathcal{L}(q'_i)} e^{\frac{-c(s')}{k_B T}} \right)$$

is also the cartesian product. This product contains the sum of all possible combinations of $(s'_1, \ldots, s'_u) \in (\mathcal{L}(q'_u), \ldots, \mathcal{L}(q'_u))$. Consequently

$$\prod_{i=1}^{u} \left(\sum_{s' \in \mathcal{L}(q'_i)} e^{\frac{-c(s')}{k_B T}} \right) = \sum_{\substack{(s'_1, \ldots, s'_u) \in \\ (\mathcal{L}(q'_1), \ldots, \mathcal{L}(q'_u))}} \left(\prod_{i=1}^{u} e^{\frac{-c(s'_i)}{k_B T}} \right).$$

Therefore, for a specific λ' such that $q \xrightarrow[c_{\lambda'}]{\lambda'} \{q'_1, \ldots, q'_{u'}\}$, we get:

$$e^{\frac{-c'_\lambda}{k_B T}} \times \prod_{i=1}^{u'} \left(\sum_{s' \in \mathcal{L}(q'_i)} e^{\frac{-c(s')}{k_B T}} \right) = e^{\frac{-c'_\lambda}{k_B T}} \times \sum_{\substack{(s'_1,\dots,s'_{u'}) \in \\ (\mathcal{L}(q'_1),\dots,\mathcal{L}(q'_{u'}))}} \left(\prod_{i=1}^{u'} e^{\frac{-c(s'_i)}{k_B T}} \right)$$

$$e^{\frac{-c'_\lambda}{k_B T}} \times \prod_{i=1}^{u'} \left(\sum_{s' \in \mathcal{L}(q'_i)} e^{\frac{-c(s')}{k_B T}} \right) = \sum_{\substack{(s'_1,\dots,s'_{u'}) \in \\ (\mathcal{L}(q'_1),\dots,\mathcal{L}(q'_{u'}))}} e^{\frac{-c_{\lambda'} - \sum_{i=0}^{u'} c(s'_i)}{k_B T}}$$

The energy of $s_{\lambda'}, s_{\lambda'} \in \mathcal{L}(q)$ that was constructed by λ' is equal to the free energy of each of its elements, here $\sum_{i=0}^{u'} c(s'_i)$ and of the new element constructed by λ', $c_{\lambda'}$. Therefore

$$-c_{\lambda'} - \sum_{i=0}^{u'} c(s'_i) = c(s_{\lambda'}).$$

Consequently, we can write, for λ':

$$e^{\frac{-c'_\lambda}{k_B T}} \times \prod_{i=1}^{u'} \left(\sum_{s' \in \mathcal{L}(q'_i)} e^{\frac{-c(s')}{k_B T}} \right) = \sum_{\substack{s_{\lambda'} = \lambda'(s'_1,\dots,s'_{u'}) \in \\ \lambda'(\mathcal{L}(q'_1),\dots,\mathcal{L}(q'_{u'}))}} e^{\frac{-c(s_{\lambda'})}{k_B T}}$$

This property holds true independently of $q \xrightarrow[c_{\lambda'}]{\lambda'} \{q'_1, \dots, q'_{u'}\}$. It remains to sum up the contribution for every $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \dots, q'_u\}$:

$$\mathrm{PfS}(q) = \sum_{q \xrightarrow[c_\lambda]{\lambda} \{q'_1,\dots,q'_u\}} \left(\sum_{\substack{s_\lambda = \lambda(s'_1,\dots,s'_u) \in \\ \lambda(\mathcal{L}(q'_1),\dots,\mathcal{L}(q'_u))}} \left(e^{\frac{-c(s_\lambda)}{k_B T}} \right) \right).$$

If we assume the DP scheme $(\mathcal{Q}, q_{\mathrm{root}}, \rho)$ is complete and unambiguous, we can say that:

- $\forall s \in \mathcal{L}(q)$, the structure must be generated at least by one λ (completeness)
- $\forall s \in \mathcal{L}(q)$, the structure cannot be generated by more than one λ (unambiguity)

This implies that the set off all elements $s_\lambda \in \lambda(\mathcal{L}(q'_1), \dots, \mathcal{L}(q'_u))$ derived all possible derivations $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \dots, q'_u\}$ is equal to $\mathcal{L}(q)$, since each term of s appears

exactly once in the given sum. It follows

$$\mathrm{PfS}(q) = \sum_{s \in \mathcal{L}(q)} \left(e^{\frac{-c(s)}{k_B T}} \right). \tag{2.19}$$

If follows that $\mathrm{PfS}(q) = \mathrm{PfScore}(q)$.

2.6.3 Formalizing VZu Algorithm

Here we work with the unambiguous version of the algorithm employed in the VIENNARNA package. In this case, defining the states solely by the couple (i, j) is insufficient, since the state also depends on whether $(i, j) \in S(i, j)$ we are constructing or whether we are currently processing a multibranch loop branches. For each (i, j), four different states $q_{i,j}$ indexed by the Equations 2.8 must be defined:

$$\begin{aligned}
q^{F}_{i,j} &= \{(i, j)\} \\
q^{C}_{i,j} &= \{(i, j) \mid (i, j) \in \phi_{pair}(w)\} \\
q^{M}_{i,j} &= \{(i, j) \mid w[i, j] \text{ is in multibranch loop containing at least one helix}\} \\
q^{M1}_{i,j} &= \{(i, j) \mid w[i, j] \text{ is in multibranch loop containing exactly one helix}\}
\end{aligned}$$

The set of all states in VZu Algorithm is then given as

$$\mathcal{Q}_{Zuk} = \{q^{F}_{i,j}, q^{C}_{i,j}, q^{M}_{i,j}, q^{M1}_{i,j} \mid 1 \leq i < j \leq n\}.$$

To define $\mathcal{D}_{Zuk}$ we need to create a specific derivation for each relation and case from the Equation 2.8. We employ both $\lambda^{unp}_{i,j}$ and $\lambda^{pair}_{i,j,k}$. We need also to introduce a constructor for each of specific loops. Here we list all types of constructors for $\mathcal{D}_{VZu}$, considering $S(i, j) \in \mathcal{S}(i, j)$ is generated by it:

- $\lambda^{unp}_{i,j}$: $w[j]$ is unpaired;
- $\lambda^{pair}_{i,j,k}$: $(k, j) \in S(i, j)$;
- $\lambda^{H}_{i,j}$: $w[k]$ is unpaired $\forall i < k < j$, $\mathcal{S}(i, j) = \{(i, j)\}$;
- $\lambda^{ILG}_{i,j,k,l}$: , $\{(i, j), (k, l)\} \subseteq S(i, j))$ with $i < k < l < j$;
- $\lambda^{ML}_{i,j,k}$: $(i, j) \in S(i, j)$, at least one helix on $w[i + 1, k - 1]$, exactly one helix on $w[k, j - 1]$.

- $\lambda^{ML2hel}_{i,j,k}$: $w[i,j]$ in multibranch loop, at least one helix on $w[i,k-1]$, exactly one helix on $w[k,j]$.
- $\lambda^{ML1hel}_{i,j,k}$: $w[i,j]$ in multibranch loop, $\{(i',j')\} \subseteq S(i,j)$ with $i \leq< k \leq i' < j' \leq j$, $\{w[l] i \leq l \leq k\}$ is unpaired;
- $\lambda^{p}_{i,j}$: $w[i,j]$ in multibranch loop $(i,j)S(i,j)$

The $\mathcal{D}_{VZu}$ is then defined as an union of these constructors for all possible combinations of corresponding parameters.

Only some of the constructors can be used to generate $\mathcal{L}(q)$. The relations are defined by DP scheme used in VZu algorithm. From it we can define ρ:

$$
\begin{aligned}
\rho_{VZu}(q^{F}_{i,j}) &= \bigcup \left\{ \begin{array}{l} \left\{ (q^{F}_{i,j-1}, \lambda^{unp}_{i,j}, 0) \right\} \\ \bigcup\limits_{\substack{i \leq k < j \\ (k,j) \in S(i,j)}} \left\{ (\{q^{F}_{i,k-1}, q^{C}_{k+1,j-1}\}, \lambda^{pair}_{i,j,k}, 0) \right\} \end{array} \right. \\
\rho_{VZu}(q^{C}_{i,j}) &= \bigcup \left\{ \begin{array}{l} \left\{ (\varnothing, \lambda^{H}_{i,j}, E_{H}(i,j)) \right\} \\ \bigcup\limits_{\substack{i<k<l<j \\ (k,l) \in S(i,j)}} \left\{ (q^{C}_{k,l}, \lambda^{ILG}_{i,j,k,l}, E_{ILG}(i,j,k,l)) \right\} \\ \bigcup\limits_{i<k<j} \left\{ (\{q^{M}_{i+1,k-1}, q^{M1}_{k,j-1}\}, \lambda^{ML}_{i,k,j}, a+b) \right\} \end{array} \right. \\
\rho_{VZu}(q^{M}_{i,j}) &= \bigcup \left\{ \begin{array}{l} \bigcup\limits_{i<k<j} \left\{ (\{q^{M1}_{k,j}\}, \lambda^{ML1hel}_{i,j,k}, c \times (k-i)) \right\} \\ \bigcup\limits_{i<k<j} \left\{ (\{q^{M}_{i,k-1}, q^{M1}_{k,j}\}, \lambda^{ML2hel}_{i,j,k}, 0) \right\} \end{array} \right. \\
\rho_{VZu}(q^{M1}_{i,j}) &= \bigcup \left\{ \begin{array}{l} \left\{ (\{q^{M1}_{i,j-1}\}, \lambda^{unp}_{i,j}, c) \right\} \\ \left\{ (\{q^{C}_{i,j}\}, \lambda^{p}_{i,j}, b) \right\} \end{array} \right.
\end{aligned}
\tag{2.20}
$$

With ρ_{VZu} defined, we complete the formalization of DP scheme used in VZu algorithm by the triplet $(\mathcal{Q}_{Zuk}, q^{F}_{i,j}, \rho_{Zuk})$. Here, $q_{root} = q^{F}_{1,n}$, and if completeness is validated, it stores the free energy of MFE structure. The recursion function to compute MFE can be obtained from this scheme by applying the Theorem 2.6.1 to it.

Such a formalization can be used to compute the overall complexity of the algorithm. The complexity of an energy minimization algorithm is $\mathcal{O}(|\mathcal{Q}| + |\mathcal{D}|)$ in time and $\mathcal{O}(|\mathcal{Q}|)$ in space (see Theorem 2.6.2). In the case of time complexity, the determining factor is the number of constructors for an internal loop, due to con-

taining four parameters with each taking approximately n values, the most of all constructors, and more than $|\mathcal{Q}|$, which is $\mathcal{O}(n^2)$. Therefore $\mathcal{O}(|\mathcal{Q}_{VZu}| + |\mathcal{D}_V Zu|) = \mathcal{O}(n^4)$. This can be reduced to $\mathcal{O}(n^3)$ if the maximum number of unpaired bases in an internal loop/bulge is limited, because in that case one of the parameters can be expressed in function of the limit and the other three parameters, and for other constructors their number is majored by n^3. The memory complexity is $\mathcal{O}(n^2)$, since all states can be stored in a limited number of two-dimensional matrices, therefore $\mathcal{O}(n^2)$.

2.6.4 Formalizing McCaskill Algorithm

The McCaskill Algorithm is similar to the VZu algorithm, the main difference is that the minimizing function MinS is replaced by the function computing the partition function PfS. Besides that, $\mathcal{Q}_{MC} = \mathcal{Q}_{VZu}$ and $\mathcal{D}_{MC} = \mathcal{D}_{VZu}$ and $\rho_{MC} = \mathcal{D}VZu$, therefore VZu and McCaskill algorithm are formalized in the exactly same way. These formalizations can be easily employed to prove the unambiguity of both algorithms, though it is required only in the latter case.

Example 2.6.7. We will demonstrate the unambiguity on VZu algorithm. The unambiguity implies a single $s \in \mathcal{S}$ is not generated by a two different constructors. This must be proven separately for each state q. We simply look what constraints different λ impose on implicated nucleotides.

In the case of $\mathcal{L}(q^F_{i,j})$, the proof is straightforward since $\lambda^{unp}_{i,j}(\mathcal{L}(q^F_{i,j-1}))$ implies that $w[j]$ is unpaired while $\lambda^{pair}_{i,j,k}(\mathcal{L}(q^F_{i,k-1}), \mathcal{L}(q^F_{k+1,j-1}))$ necessitates the presence of pair (k, j), and since base triplet is forbidden by definition, we have, for any i, j, k_1, k_2 such that $1 < i \leq k_1 < k_2 < j \leq n$:

$$\lambda^{unp}_{i,j}(\mathcal{L}(q^F_{i,j-1})) \cup \lambda^{pair}_{i,j,k_1}(\mathcal{L}(q^F_{i,k_1-1}), \mathcal{L}(q^F_{k_1+1,j-1}) = \varnothing$$
$$\lambda^{pair}_{i,j,k_1}(\mathcal{L}(q^F_{i,k_1-1}), \mathcal{L}(q^F_{k_1+1,j-1}) \cup \lambda^{pair}_{i,j,k_2}(\mathcal{L}(q^F_{i,k_2-1}), \mathcal{L}(q^F_{k_2+1,j-1}) = \varnothing$$

In a similar manner, each of the cases computing $\mathcal{L}(q^C_{i,j})$ is mutually exclusive; $\lambda^{HP}_{i,j}, \lambda^{ILG}_{i,j,k,l}, \lambda^{ML}_{i,j,k}$ imply respectively zero, one or multiple base helices for $w[i+1, j-1]$ and each time the values of eventual coefficients k and l are different. Note that in the last case the construction of $q^{M1}_{i,j}$ implies that the last helix in given multibranch loop begins at $w[k]$, implying the mutual exclusivity between different structures generated by $\lambda^{ML}_{i,j,k}$ for different values of k.

In the case of generating $\mathcal{L}(q^M_{i,j})$ the segment $w[i, k-1]$ is either completely unpaired, or hosts another base pair. Like in previous case $w[k]$ indicates the beginning of last helix in multibranch loop, therefore the cases cannot generate the same structure for different k, and $\lambda^{ML1hel}_{i,j,k}$ and $\lambda^{ML2hel}_{i,j,k}$ cannot result in same $S(i,j)$ either due to difference in $w[i, k-1]$.

The explanation for $\mathcal{L}(q^{M1}_{i,j})$ is similar to the one written for $\mathcal{L}(q^F_{i,j})$, the difference being only base pair (i,j) being considered. This implies that for no state q two different constructors λ can generate the same S, therefore VZu algorithm is unambiguous. The same applies to McCaskill algorithm due to it using the same decomposition.

2.6.5 Stochastic backtrack

The general idea behind the stochastic backtrack was explained in Section 2.5.6. With the introduced formalism the central principle can be explained in a much more simplified and compact manner. The stochastic backtrack can be only performed by a DP scheme $(\mathcal{Q}, q_{root}, \rho)$ used to compute the partition function, ie. employs the scoring function PfS. For each state q, we choose what derivation will take place.

Theorem 2.6.3 (Backtrack probability): *By using the probability* $p(\lambda \mid q)$ *of performing a derivation* $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ *when currently being in state* q *and whose value is*

$$p(\lambda \mid q) = \frac{e^{\frac{-c_\lambda}{k_B T}} \times \left(\prod_{i=1}^{u} PfS(q'_i) \right)}{PfS(q)} \tag{2.21}$$

the stochastic proces ulitmately terminates and returns an object $s = \lambda(\mathcal{L}q'_1, \ldots, \mathcal{L}q'_u)$ *with boltzman probability within* $\mathcal{L}(q)$.

Proof 2.6.3. From Proof 2.6.2 we know that contribution of a single derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ towards $PfS(q)$ is

$$e^{\frac{-c_\lambda}{k_B T}} \times \prod_{i=1}^{u} \left(\sum_{s' \in \mathcal{L}(q'_i)} e^{\frac{-c(s')}{k_B T}} \right).$$

It follows directly from Theorem 2.6.5 that $PfS(q)$ gives the sum of contributions of all λ and that the probability $p(\lambda \mid q)$ is computed as shown in Equation 2.21.

The stochastic backtrack of an entire structure is then performed by starting at q_{root} and repeating the process for every state $q'_1, q'_2, \ldots, q'_u$ that gets selected. Note that the order of treatment of states does not matter as long as it is deterministic, but they must be all treated.

2.7 The relation between RNA thermodynamics and RNA kinetics

We mentioned that due to the structural diversity of most larger RNA sequences, the RNA folding landscape is obtained by sampling from a representative set of structures $S \in \mathcal{S}_n$ or some specific subspace as locally optimal structures, free energies computed using Turner Energy Model. This is followed by the study of folding pathways and finally by the analysis of the evolution of the entire system. Therefore, we use a thermodynamic approach to study and simulate the kinetics.

It is worth mentioning the relation between thermodynamics and kinetics. The folding of an RNA can be seen as a type of chemical reaction with the secondary structures instead of compounds. Here we will consider a following transformation of secondary structure $S_{\mathcal{A}}$ of an RNA sequence into the state $S_{\mathcal{B}}$, and we assume its kinetics follows the same stochastic model as a kinetics of chemical reactions [28]:

$$S_{\mathcal{A}} \rightarrow S_{\mathcal{B}}$$

While thermodynamics, here represented by the parameters of Turner Energy Model, will allow us to know about the stability of the different secondary structures and consequently whether the reaction should be possible but nothing about its speed, the kinetics indicates how fast the transformation would happen, but says nothing about the equilibrium state. The thermodynamics therefore helps us to choose the possible transformations, which we can then investigate from the perspective of the kinetics.

However, since we compute the free energy using thermodynamic energy model, the obtained energies, and notably associated Boltzmann distribution are correct while in the state of thermodynamic equilibrium, but not outside of it. If the system still evolves, then the thermodynamic energy model does not apply.

Example 2.7.1. Suppose a system where all RNA molecules are unfolded at some

starting time t_{start}. According to Boltzmann distribution their proportion should be with few exception very small or zero. However, since we are obviously not in equilibrium, their proportion is 1.

On the other hand, the Boltzmann distribution can still provide a good approximation on the kinetics because it indicates towards what state the system evolves.

The main problem with the use of thermodynamic model for the study of kinetics is the lack of so-called activation energies E_a. It can be, however, computed from the energies of the structures participating in transformation. Its values do not depend only on the energy of those structures between which the transformation occurs, but also whether they are neighbors or not. Consider the reaction between two neighboring structures

$$S_{\mathcal{A}} \rightarrow S_{\mathcal{B}},$$

The activation energy of neighboring structures the E_a is the energy that is necessary to supply in order to transform $S_{\mathcal{A}}$ to $S_{\mathcal{B}}$ and its value is:

$$E_a = E_{S_{\mathcal{B}}} - E_{S_{\mathcal{A}}} \tag{2.22}$$

Activation energy is sometimes also denoted as an **energy barrier**. The case of neighboring structures is simple to compute, because there is no intermediate step. If the activation energy is negative, then the transformation is **spontaneous** - happens on its own - and the energy is instead freed.

In the case where $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$ are not neighbors, it depends whether the highest free energy of intermediate product S_{peak} from transformation of $S_{\mathcal{A}}$ to $S_{\mathcal{B}}$ is higher than that of $S_{\mathcal{B}}$ or not.

Definition 2.7.1 (Activation energy): Suppose a transformation between states

$$S_{\mathcal{A}} \rightarrow S_{peak} \rightarrow S_{\mathcal{B}}$$

where S_{peak} is the intermediate structure with the highest free energy and $S_{peak} > max(S_{\mathcal{A}}, S_{\mathcal{B}})$. The activation energy necessary for this transformation to overcome the energy gradient is

$$E_a = E_{S_{peak}} - E_{S_{\mathcal{BA}}}. \tag{2.23}$$

If there are multiple folding pathways between distant $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$, the one considered is that with the lowest $E_{S_{peak}}$, since it is the most optimal. This way we compute an approximation of the E_a.

Such approach has an advantage of being easily computable. The Turner Energy Model is quite complete and contains extensive tabulated values as well as relations to compute the free energy of different substructures. The free energy is relatively easy and fast to compute, an important factor when determining the energies of longer sequences. While the computation of the activation energy for the most optimal folding pathway between $S_{\mathcal{A}}$ and $S_{\mathcal{B}}$ is an NP-hard problem [61], a heuristic methods allowing to compute their approximate values exist, such as `findpath` from ViennaRNA or RNAtabupath [25]. The partition function and associated probabilities can be also computed in a polynomial time using DP-based programming. There are more precise methods such as molecular dynamic simulations [6], but they are difficult to implement and notably very demanding in terms of time computation, consequently being unusable for the analysis of longer RNA sequences. Also, even the approximate analysis using thermodynamic model can give us deeper insight on the RNA kinetics.

The thermodynamic model also describes the ideal state of the system, therefore the state towards which it constantly evolves and which it would achieve if given enough time. This alone can give us indication which folding pathways could be crucial, and complemented by the computation of approximate activating energies we can obtain sufficient estimation of the kinetics of the system. This, along with the relative simpleness of the computations, is the reason why we will use thermodynamic Turner Energy Model to study the kinetics of RNA folding.

2.8 Kinetics of RNA folding landscape

The RNA landscape provides a great tool to study RNA kinetics. Since the size of $\mathcal{S}_n$ makes building and analyzing the RNA folding landscape impossible to perform in reasonable time, the first objective when studying an RNA folding landscape is to simplify it to only the key elements. This applies to the folding pathways as well. Here we explain the general approach to reduce RNA folding landscape which will be followed through this work.

2.8.1 Simplification of a landscape

One way to simplify the folding landscape is to represent only its locally optimal structures Therefore, here we consider the space of only locally optimal secondary structures. Let $\mathcal{S}_{n_L} \in \mathcal{S}_n$ be the space of all locally optimal secondary structures

S_L of length n. Each S_L is connected to a number of secondary structures with higher free energies such that the free energy from said structure to S_L is strictly descending along the path.

Definition 2.8.1 (Basin): The **basin $\mathcal{B}_{S_L}$** of locally optimal structure S_L is the section of RNA folding landscape containing all structures $S_{basins_{S_L}}$ such that there exists a path

$$S_{basins_{S_L}} \to S_1 \to \ldots \to S_u \to S$$

where the energies follow

$$E_{S_{basins_{S_L}}} \geq E_{S_1} \geq \ldots \geq E_{S_u} \geq E_{S_L}$$

In other words, it is every point that is, on folding landscape, in the direction of the gradient towards S_L. Consequently, any secondary structure within $\mathcal{B}_{S_L}$ should spontaneously transform into S_L. Therefore we might consider an approximation of RNA landscape by considering only S_L. The basins are connected by the secondary structures represented by saddle points.

Definition 2.8.2 (Saddle point): A saddle point **S_{peak}** between two neighboring basins $\mathcal{B}_{S_{LA}}$ and $\mathcal{B}_{S_{LB}}$ is a structure S_{peak} such that $S_{peak} \in \mathcal{B}_{S_{LA}} \cap \mathcal{B}_{S_{LB}}$ and

$$E_{S_{peak}} = \min_{S \in \mathcal{B}_{S_{LA}} \cap \mathcal{B}_{S_{LB}}} (E_S).$$

Such saddle point is also called **direct saddle point**. An **indirect saddle point** is a saddle point that can be obtained for two basins by passing through another basins. Note that the energy of indirect saddle of two basins may be lower that that of the direct saddle. This greatly impacts what path will be taken when transformation $\mathcal{S}_{n_{LA}} \to \mathcal{S}_{n_{LB}}$ takes place.

The activation energy associated to transformation $\mathcal{S}_{n_{LA}} \to \mathcal{S}_{n_{LB}}$ is

$$E_a = E_{S_{peak}} - E_{\mathcal{S}_{n_{LA}}}.$$

The energy of indirect saddle is obtained by taking the highest energy of S_{peak} that is met along the path from basin to basin.

Since the saddles are not locally optimal structures, they are not included in $\mathcal{S}_{n_L}$, yet they are crucial to the estimation of the kinetics. However, many methods determine the saddle structures in order to estimate E_a. An example of such software is `findpath` from ViennaRNA or `RNAtabupath` [25]

Since the folding landscape has a ragged nature, it presents many of small holes and therefore many of S_L. This number would still be prohibitively large for the analysis. This is why the sampling methods are employed, with Boltzmann sampling studied in this work prioritizing lower free energy structures that are potentially more interesting.

The reduction of the number of structures used to model a folding landscape also diminishes the number of the folding pathways. Plus, since we consider only one the most optimal path between S_{LA} and S_{LB} where $\mathcal{B}_{S_{LA}}$ and $\mathcal{B}_{S_{LB}}$ are neighbors, the number of folding pathways is reduced even further.

However, it is not always known what basins are neighboring. For most of the existing software, this is not too much of problem since they compute the activation energy between two states without having to know that, but computing energy barrier is computationally rather demanding, therefore we cannot compute the energy barriers between every two existing states. However, the basins $\mathcal{B}_{S_{LA}}$ and $\mathcal{B}_{S_{LB}}$ are more likely to be neighboring the more S_{LA} and S_{LB} are similar. The best course of action is to suppose that the $\mathcal{B}_{S_{LA}}$ and $\mathcal{B}_{S_{LB}}$ are neighboring if they are similar enough.

Definition 2.8.3 (Connected locally optimal structures): The locally optimal structures $\mathcal{B}_{S_{LA}}$ and $\mathcal{B}_{S_{LB}}$ are said to be **connected** if

$$d(S_{LA}, S_{LB}) < c_d$$

where d is the distance function evaluating the similarity of S_{LA} and S_{LB} and c_d is the user defined threshold.
The space of all S_{Ln} with $\mathcal{B}_{S_L}$ connected to $\mathcal{B}_{S_{LA}}$ is denoted

$$\mathcal{C}(S_{LA}) = \{S_L \in \mathcal{S}_{n_L} \mid d(S_{LA}, S_{LB}) < c_d, S_L \neq S_{LA}\}$$

The E_a are then computed for each $\mathcal{C}(S_L)$.

The c_d as well as d function must be defined in a manner that the resulting landscape is connected, otherwise it will be impossible to perform a simulation due to the impossibility to access a part of the landscape from a specific structure.

The final result is the model of the RNA folding landscape reduced to locally optima structures and a key transitive structures with a corresponding set of folding pathways. While this model might be overly simplified, it allows already to study the effect of locally optimal structures on the kinetics of a given RNA sequence, something we are primarily interested in this work. The advantage is that the analysis of such landscape is faster than more complex models. For this reason this was the method we employed during the course of our study.

2.8.2 RNA Kinetics Study

Here we consider $S \in \mathcal{S}_{n_L}$. The study may be performed on structures from $\mathcal{S}_n$ in general, however in our context considering only $\mathcal{S}_{n_L}$ is sufficient. For each S_L, we define $P(S_L, t)$, the proportion of RNA molecules of the system folded in S at the time t. We have $\forall t$,

$$\sum_{S_X \in \mathcal{S}_{n_L}} P(S_X, t) = 1.$$

The folding pathways between the different S_L can be seen as an iterative transformation implicating a certain number of intermediate structures [28]. The entire process is memoryless, since the choice of the next structure does not depend on the previous choices. Therefore, the changes of the proportions within the system can be modeled by a continuous time discrete state Markov Chain process. If we suppose the transitions between S_{LA} and $\mathcal{C}(S_{LA})$ and all transitions are reversible, then each $P(S_{LA}, t)$ depends on all influxes from $\mathcal{C}(S_{LA})$ and all outfluxes towards this space. The evolution of $P(S_{LA}, t)$ can be captured by a chemical master equation (CME) [51].

$$\frac{dP(S_{LA}, t)}{dt} = \sum_{S_{Ln} \in \mathcal{C}(S_{LA})} P(S_{Ln}, t) \times k_{S_{Ln} \to S_{LA}} - P(S_{LA}, t) \times k_{S_{LA} \to S_{Ln}}. \quad (2.24)$$

The values $k_{S_{Ln} \to S_{LA}}$ are called **transition rates**. These describe the speed at which one structure is transformed to another. For the **self-transition rate** of S_{LA} (rate for the S_{LA} not being changed), we have [89]:

$$k_{S_{LA} \to S_{LA}} = - \sum_{S_{Ln} \in \mathcal{C}(S_{LA})} k_{S_{Ln} \to S_{LA}} \quad (2.25)$$

If the system is in equilibrium at $t \to +\infty$, then

$$dP(S_{LA}, t) = 0$$

since system does not evolve, and also $P(S_L, t) = p(S)$, therefore $P(S_L, t)$ can be computed using Equation 2.11. Assuming equilibrium, we can utilize CME to compute the ratio of the transition rates for the same transformation in opposite directions:

$$\begin{aligned}
\frac{e^{\frac{-E_{S_{Ln}}}{k_B T}} \times k_{S_{Ln} \to S_{LA}}}{\mathcal{Z}} &= \frac{e^{\frac{-E_{S_{LA}}}{k_B T}} \times k_{S_{LA} \to S_n}}{\mathcal{Z}} \\
\frac{k_{S_{LA} \to S_{Ln}}}{k_{S_{Ln} \to S_{LA}}} &= \frac{e^{\frac{-E_{S_n}}{k_B T}}}{e^{\frac{-E_{S_{LA}}}{k_B T}}} \\
\frac{k_{S_{LA} \to S_{Ln}}}{k_{S_{Ln} \to S_{LA}}} &= e^{\frac{-\Delta E_{S_{Ln} - S_{LA}}}{k_B T}}
\end{aligned}$$

with $E_{S_{Ln}}, E_{S_{LA}}$ the energies of S_{Ln} and S_{LA} respectively and $\Delta E_{S_n - S_{\mathcal{A}}}$ is their difference. Consequently, the ratio of the transition rates is the Boltzmann factor of their energy difference.

The computation of the transition rates is the necessary step before performing a simulation of a kinetics of specific RNA. The entire folding process is modeled by Markov Chains and the transition rates are used to establish a transition rate matrix that is necessary to perform a numerical integration of the Markov chains.

The simplest way to approximate their values is to use the **Metropolis rule**. In general, this rule assumes that that the transformation is guaranteed if the energy of the final state is lower than the energy of starting state:

$$k_{S_x \to S_y} = \begin{cases} \tau \times e^{\frac{-E_{ax \to y}}{k_B T}} & \text{if } E_{ax \to y} > 0 \\ \tau & \text{otherwise} \end{cases} \tag{2.26}$$

where $E_{ax \to y}$ is the activation energy of transformation $S_x \to S_y$ and $k_{S_x \to S_y}$ the associated transition rate.

If $S_{Lx}, S_{Ly} \in \mathcal{S}^2_{n_L}$, the $E_{ax \to y} \geq 0$, otherwise one of S_L is not locally optimal. Consequently, for the transformation $S_{Lx} \to S_{Ly}$ of locally optimal structures the Metropolis rule can be reduced

$$k_{S_{Lx} \to S_{Ly}} = \tau \times e^{\frac{-E_{ax \to y}}{k_B T}}$$

In the case of locally optimal structures, this expression is in fact similar to **Arrhenius law** [3], another law that can be used to compute the transition rates:

$$k_{S_{Lx} \to S_{Ly}} = A \times e^{\frac{-E_{ax \to y}}{k_B T}} \tag{2.27}$$

The last rule that can be and frequently is used to compute the transition rate is **Kawasaki rule**:

$$k_{S_{Lx} \to S_{Ly}} = \tau \times e^{\frac{-E_{ax \to y}}{2\ boltzT}} \tag{2.28}$$

During this work, the energy barriers were computed using `findpath` for the pairs of locally optimal secondary structures. The details are specified in Section 3.3.4.

2.9 Coarse-grained model of a folding landscape

The different approach to simplify an RNA folding landscape and compute the transition rates is to introduce the **macrostates**. These macrostates represent multiple structures instead of the single one [103].

We might consider each basin of attraction as a singular macrostate, reducing the number of states as well as that of folding pathways. In that case the macrostate associated to the basin $\mathcal{B}_{S_L}$ represents $S \in \mathcal{B}_{S_L}$. The partition function of such macrostate is

$$\mathcal{Z}_{\mathcal{B}_{S_L}} = \sum_{S \in \mathcal{B}_{S_L}} \mathcal{Z}_S.$$

It follows the computing the **free energy of the ensemble** $E_{\mathcal{B}_{S_L}}$ of the basin by:

$$E_{\mathcal{B}_{S_L}} = k_B T \ln(\mathcal{Z}_{\mathcal{B}_{S_L}}).$$

The transition rate between two macrostates $k_{\mathcal{B}_{S_{LA}}} \to k_{\mathcal{B}_{S_{LB}}}$ computes a bit differently than between two structures [103]. If we know the transition rates $k_{S_x \to S_y}$ such that $S_x \in \mathcal{B}_{S_{LA}}$ and $S_y \in \mathcal{B}_{S_{LB}}$ then

$$k_{\mathcal{B}_{S_{LA}}} \rightarrow k_{\mathcal{B}_{S_{LB}}} = \sum_{S_x \in \mathcal{B}_{S_{LA}}} \sum_{S_y \in \mathcal{B}_{S_{LB}}} k_{S_x \rightarrow S_y} \frac{\mathcal{Z}_{S_x}}{\mathcal{Z}_{\mathcal{B}_{S_{LA}}}}$$

There is a number of different software that is capable of generating a model of an RNA landscape from the set of secondary structures along with the computation of the transition rate matrix using here described approach. They naturally implement a heuristic method to estimate the activation energies.

The most notable is `barriers` [30], that creates a barrier tree. This tree is constructed from locally optimal structures by flooding algorithm - the locally optimal structures are the leaves of the barrier tree. Whenever two basins associated to the locally optimal structures merge a saddle point is found and a node connecting those two branches is placed. This is repeated until the root of the tree is found. The barrier tree graphs then represents the activation energies or energy barriers between two structures by the vertical length of the edges, and the free energy of locally optimal structures and saddles is determined by their vertical position.

A more precise analysis is performed by `Basin Hopping Graph` (BHG) [50]. This graph connects all neighboring basins $\mathcal{B}$, represented by nodes and the efges represent the transitions along with the corresponding value of the activation energy E_a. Unlike the barrier tree, which shows only the saddle points between two closest basins on their fusion, the BHG shows all transition paths and therefore does not succumb to a loss of an information. On the other hand its disadvantage is that it may return multiple additional transitive structures generated by heuristics detecting the saddle points, which will make kinetic simulation last longer.

This approach is more precise than just using locally optimal structures but is also much more demanding due to potentially introducing the number of new supplementary structures, which makes the analysis of multiple landscapes or landscapes of longer sequences time consuming.

2.9.1 Kinetic Simulation

The kinetic simulation can be performed once the transition rate matrix was established. This matrix has rows and columns represented by a certain secondary structure that is present within the landscape. The transition rates between the secondary structures that are not directly connected are supposed to be zero. The

numerical integration is usually done by using the software dedicated for this purpose. During this work, we used `treekin` [103].

The method employed in `treekin` is specifically the Metropolis-Hastings (MH) algorithm [63] [41], one of Markov Chain Monte Carlo methods. The stochastic algorithm mostly employed to determine a non-standard distribution, it can be also used, due to its iterative principle, to follow an evolution of the system towards its equilibrium as well. However, to employ the MH algorithm, the transition rates must fulfill the condition of detailed balance. Applying the usual, rejection-based MH algorithm does not always fulfill this condition [29]. However, the algorithm can modified into a rejection-less variant that is similar to Gillespie algorithm employed in chemistry [36].

Assume there is a sequence w which can fold into a certain number of $S_L \in \mathcal{S}_{n_L}$. The simplified general principle is as follows

- **Initialization:** For t_0, have w folded in structure S_{LA}.
- **Iteration:** Perform the following:
  - Find all structures $S_{Ln} \in \mathcal{C}(S_{LA})$. Compute
    $$k_T = \sum_{S_{Ln} \in \mathcal{C}(S_{LA})} k_{S_{LA} \to S_{Ln}}.$$
  - Select one from them, S_{LB}, with the probability
    $$p(S_{LB}) = \frac{k_{S_{LA} \to S_{Ln}}}{k_T}$$
  - Make S_{LB} the new structure of w.
  - Move to a new time $t_1 = t_0 + \Delta t$ such that Δt is randomly generated from an exponential distribution with the rate parameter k_T.
  - Repeat iteration steps until some specified t_{final} is reached.

This is then generalized on all secondary structures in the system in specific initial distribution. The simulation is then performed on the entire system.

The choice of the initial distribution of structures depends on the objective of the kinetic study. We may just want to study the general evolution of the system. For riboswitches, we may want to observe how the concentrations of the MFE and the metastable state. The initial distribution must be adjusted for these specific objectives.

2.10 Sampling Caveats

While the field of RNA kinetics studies rapidly advanced in few last years, there is still much to be done. Kinetic analysis of especially long sequences still represents a problem that is mainly related to the richness of the space of the secondary structures.

The Boltzmann sampling prioritizes the secondary structures with lower free energy. This probability decreases exponentially as the energy increases. Let $-\Delta G$ be a free energy difference between S_2 and S_1. We call by a **frequency ratio** $\eta(S_1, S_2)$ a ratio of probabilities $p(S_1)$ and $p(S_2)$ of observing secondary structures S_1 and S_2 respectively

$$\eta(S_1, S_2) = \frac{e^{\frac{-E_{S_1}}{k_B T}}}{e^{\frac{-E_{S_2}}{k_B T}}} = e^{\frac{E_{S_2} - E_{S_1}}{k_B T}} = e^{\frac{-\Delta G}{k_B T}}. \tag{2.29}$$

While it is desired to sample low energy structures in priority, it may become problematic to generate a higher free energy structures if we want to do so. For $\Delta G = 1$, $\eta(S_1, S_2) = 5.067$, meaning per each 5 sampled S_1, we observe one molecule in S_2. For $\Delta = 10$ this value increases to 11147271. It quickly becomes impossible to access high energy structures without extensive sampling and even then it depends on the random number generator whether such structure will be selected.

A number of sampling strategies that tried to address this problem. As already stated, one method is to ample only locally optimal secondary structures $\mathcal{S}_{n_L}$. This can be checked regularly at each step of decomposition. For example, the software `RNAlocopt` [55] verifies whether the addition of a base pair will not result in the structure not being locally optimal anymore. `RNASLOpt` [54] on the other hand assumes the stability is due to helical regions, which the authors call a stack, and searches for the stacks of certain length. For obtained S_L, their values of E_a are checked against a user-defined threshold whether they are stable enough. However, while $\mathcal{S}_{n_L}$ is substantially smaller than $\mathcal{S}_n$, it still raises exponentially with n [56], therefore restricting $\mathcal{S}_n$ to $\mathcal{S}_{n_L}$ will not completely solve the problem.

Another way to tackle this problem is to reduce the redundancy between of the sampled structures - sample the same structure repeatedly. One such approach is employed in software `RNAlocmin` [50], which uses the **ξ-scheduling**. The core of this approach is to sample multiple times on different temperatures, computed by ξ-scheduling, without scaling the Turner Energy Model-related factors - only the

temperature T of Boltzmann factor is scaled. This results in smaller differences between the probabilities of different S and for temperatures sufficiently high temperature the Boltzmann distribution becomes uniform since the exponential factor asympotically reaches 0. `RNAlocmin` exploits this to sample S at gradually higher temperatures, since smaller differences between probabilities mean higher probability to sample not-yet seen secondary structure. On the other hand the structures estimated at higher temperature are likely to have their energy overestimated and the redunancy is still present to some degree.

The way to prevent the repeats of samples altogether is to introduce the concept of non-redundant sampling. One of main objectives of this work is to research a sampling method that allows to generate each sample at most once without introducing bias in the distribution of not yet sampled structures. In next chapter, we will present the details on the concept of the method itself as well as how to integrate it in most efficient and easiest way in multiple algorithms. The efficient implementation is particularly important in regards to reduce the time consumed by sampling. We will mainly describe its implementation to Saffarian algorithm [80], a state-of-the-art algorithm that will be detailed in next chapter as well.

Chapter 3

RNANR: An Algorithm for Non-Redundant Sampling of RNA Secondary Structures

The main objective of this work is to establish an efficient method to study the kinetics of RNA sequences. We decided to achieve this by researching a non-redundant sampling method for the purpose of RNA secondary structure prediction. Since the quality of samples greatly impacts the quality of the RNA folding landscape, the main tool used to study RNA kinetics, it is reasonable to concentrate on amelioration of this important step. In plus, the non-redundant sampling method should be as optimal as possible, easy to implement, therefore easily separable form DP scheme it is adjacent to, and compatible with most DP schemes that compute partition function.

The algorithm that was used to research the non-redundant sampling is the algorithm based on combinatorics developed by Azadeh Saffarian, Hélène Touzet *et al* [80]. We will reference this algorithm as **Saffarian algorithm**. However, the algorithm itself does not compute the partition function nor includes the Turner Energy Model terms. After thoroughly introducing the mentioned algorithm and its formalization using the notions from Section 2.6, we will explain and validate its necessary modifications to compute these values and most importantly the probabilities $p(S)$. We will then follow with explaining of the concept of non-redundant sampling.

The non-redundant sampling necessitates two things to function correctly. First,

the distribution of not-yet sampled S after forbidding a sampled one must stay unbiased. This means the frequency ratio $\eta(S_1, S2)$ of any unsampled S_1 and S_2 must stay same for each sampling iteration. Second, it must track all samples that were already generated. This also includes the chain of decision, or constructors λ, that were used to generate said structure. For this purpose, a specific data-structure that must be developed and implemented into the DP scheme to enable the non-redundant sampling. We will develop on both points in detail.

We mentioned that this approach should be compatible with most algorithms computing partition function and used for RNA structure prediction. We will support this point by demonstrating how one can implement the non-redundant sampling into `RNAsubopt` from ViennaRNA based on McCaskill algorithm.

3.1 Saffarian Algorithm

The Saffarian Algorithm, developed by Azadeh Saffarian, Hélène Touzet *et al* [80], is a combinatorial algorithm that returns only the locally optimal secondary structures. This algorithm does not compute the energy of any locally optimal secondary structure, instead it makes an assumption on local optimality based on combinatorial principle and then returns all the structures that satisfy that specific condition. The base version of the algorithm performs an exhaustive enumeration of all possible locally optimal secondary structures.

3.1.1 State of the Art and the general principle

The central hypothesis upon which the Saffarian algorithm is based is following: the structure is locally optimal only if it is saturated.

Definition 3.1.1 (Saturated structure): The secondary structure is **saturated** only if $\forall w[i]$ with $1 \leq i \leq n$ that are unpaired, there is no a base pair that can be created without violating the conditions specified in Definitions 2.1.4 and 2.1.5. Such structure is denoted by $\mathbf{S_{SAT}}$.

In other words, there is no possible way to add a new base pair to the saturated structure S_{SAT}. This however does not mean that S_{SAT} contains the maximum base pairs as defined by Nussinov Algorithm, just that there is no base pair that can be added to the base pair configuration of S_{SAT}.

The structures $\mathcal{S}_{n_{SAT}}$ delimit their own space.

Definition 3.1.2 (Saturated structure space): The **saturated structure space** is denoted by $\mathcal{S}_{n_{SAT}} \subseteq \mathcal{S}_n$. The space of the saturated secondary structures $\mathcal{S}_{SAT}(i,j)$ covering $w[i,j]$ is denoted by $\mathcal{S}_{SAT}(i,j)$. The space of all saturated secondary structures is denoted by $\mathcal{S}_{SAT}$ with

$$\mathcal{S}_{SAT} = \bigcup_{1 \leq i < j \leq n} \mathcal{S}_{SAT}(i,j).$$

The reason that the saturated structures can be considered as locally optimal is that the base pairs have stabilizing effect on the structure. However, this is not always true, especially for the lone base pairs - the base pairs that are not stacked with another base pairs. The most of the stability of the structure comes from the helices, or stacked bas pairs [42]. To take this factor into the account, the Saffarian algorithm allows only for helices $\mathcal{H}(i,j,lh)$ with a certain minimum length a. In the rest of this chapter, we suppose $a = 3$.

If the minimum helix length is imposed, the definition of the saturated structure is slightly redefined:

Definition 3.1.3 (Parametrized saturated structure): If all S can contain only helices $\mathcal{H}(i,j,lh), 1 \leq i < j \leq n$ where $lh \geq tr$, with tr an arbitrary threshold, the structure S is **saturated** only if no helix $\mathcal{H}(i,j,lh)$ with $lh \geq tr$ can be added to it.

The structure is parametrized because there is a special parameter a that imposes certain restriction on this structure. The space of all saturated secondary structures also gets redefined accordingly. Specifically, from this point on $\mathcal{S}_{SAT}$, resp. $\mathcal{S}_{n_{SAT}}$ contain only saturated structures, resp. saturated structures of size n, that respect the defined parameters. A number of other parameters will be introduced later.

Suppose two helices $a \geq lh$ and $a \geq lh'$. Between the helices $\mathcal{H}(i,j,lh)$ and $\mathcal{H}(i',j',lh')$, the algorithm assumes two possible relations. First, the helix can be nested in another.

Definition 3.1.4 (Nested helices): The helix $\mathcal{H}(i, j, lh)$ is **nested** in $\mathcal{H}(i', j', lh')$ if $i < i' < j' < j$ and there is at least one base $w[k]$ for $w[k] \in w[i + a, i - 1] \cup w[i + a, i - 1]$ is unpaired meaning $\mathcal{H}(i, j, lh)$ and $\mathcal{H}(i', j', lh')$ are not the one continuous helix.

Second, the helix can be juxtaposed to another while all of them are nested in the exactly same set of helices.

Definition 3.1.5 (Parallel helices): The helices $\mathcal{H}(i, j, lh)$ and $\mathcal{H}(i, j, lh)$ are **parallel** one to another if either $i < j < i' < j'$ or $i' < j' < i' < j'$ and both helices are, at the same time, either nested in the same helix $\mathcal{H}(i'', j'', lh'')$ or not nested in another helix at all.

The principle of the Saffarian algorithm is, first, to enumerate all possible base pairs. These are then used to construct helices $\mathcal{H}(i, j, a)$ that each has the exactly the length of a, the minimum possible helix length. These helices are then assembled to maximum parallel sets.

Definition 3.1.6 (Flat structure): The **flat structure** $\mathfrak{p}$ is the maximum set of parallel helices. Specifically, $\mathfrak{p}(i, j)$ denotes a maximum set of parallel helices $\mathcal{H}(i_1, j_1, a), \ldots, \mathcal{H}(i_u, j_u, a)$ for a sequence $w[i, j], i < i_1 < j_1 < \ldots < i_u < j_u < j$. No another un-nested helix $\mathcal{H}(i', j', a)$ can be added to it.

Note that by definition, $\mathfrak{p}[i, j]$ cannot contain any nested elements. For each $w[i, j]$ a certain number of flat structures can be proposed.

Definition 3.1.7 (Flat structures set): A flat structure set $\mathcal{P}(i, j)$ is a set of all possible flat structures $\mathfrak{p}(i, j)$.

Using the dynamic programming, the flat structures are constructed as follows. We borrow the expression from the related paper [80] and update it including the modifications later made by Hélène Touzet [64].

- If for $i < k \leq j$ there is not any helix $\mathcal{H}(i, k, a)$, then $\mathcal{P}(i, j) = \mathcal{P}(i + 1, j)$.
- If such helices exist, then:

$$\mathcal{P}(i,j) = \bigcup \begin{cases} \bigcup\limits_{i<k\leq j} \{\mathcal{H}(i,k,a)\} \oplus \mathcal{P}(k+1,j) \\ \bigcap\limits_{i<k\leq j} Sat(\mathcal{H}(i,k,a), \mathcal{P}(i+1,j)) \end{cases} \tag{3.1}$$

The $\oplus$ operator denotes the concatenation of a base pair to every element of given set, here of the helix $\mathcal{H}(i,k,a)$ to every flat structure $\mathfrak{p}$ of the set $\mathcal{P}(k+1,j)$. The function $Sat(\mathcal{H}(i,k,a), \mathcal{P}(i+1,j))$ checks whether the structure consisting of a concatenation of the helix $\mathcal{H}(i,k,a)$ and some $\mathfrak{p} \in \mathcal{P}(i+1,j)$ is saturated. In other words, $Sat((i,k), \mathcal{P}(i+1,j))$ returns the concatenation of $\mathcal{H}(i,k,a)$ and some $\mathfrak{p} \in \mathcal{P}(i+1,j)$ only if there is not another helix that can be added to it.

The next step is to assemble these flat structures $\mathfrak{p}$ into the saturated secondary structures S_{SAT}. This creates the nested relations between the helices. The assembling step would be done by searching $\mathfrak{p}(1,n)$ for substring $w[1,n] = w$, then proceeding with the substrings opened by $\mathfrak{p}(1,n)$ until such substrings would become too small to host a helix. However, the algorithm proposed as it is is insufficient, since all helices would have length of a or multiples.

To address this issue, if we check $w[i,j]$ that is nested within a $\mathcal{H}(i-a+1, j+a-1, a)$ from some flat structure, we need to suppose the case where an extension of such helix is possible. In that case we suppose an extension by the pair (i,j), elongating $\mathcal{H}(i-a+1, j+a-1, a)$ by one base pair. This makes possible create any helices with $lh \geq a$.

It is important to notice that if set $\mathcal{P}(i,j)$ of $\mathfrak{p}$ is nested within the helix $\mathcal{H}(i-a+1, j+a-1, a)$ we have to exclude the helix $\mathcal{H}(i,j,a)$ from it. In the opposite case the helices of length of multiples of a could be decomposed in more ways than one, violating the condition of unambiguity.

The algorithm is summed up on Figure 3.1, the substring $w[i,j]$ is specifically treated whether the $w[i-1]$ and $w[j+1]$ are paired or not.

The problem with this algorithm is its time complexity. While its memory complexity is limited to $\mathcal{O}(n^2)$, since we need only two-dimensional matrices to store the results, the time complexity depends on the number of flat structures that may be exponential due to the the upper limit for helices being $n/2$. However, if the number of possible helices for given interval is limited, the complexity becomes polynomial. Therefore, supplementary parametrization can help us to reduce the overall complexity of the Saffarian algorithm.

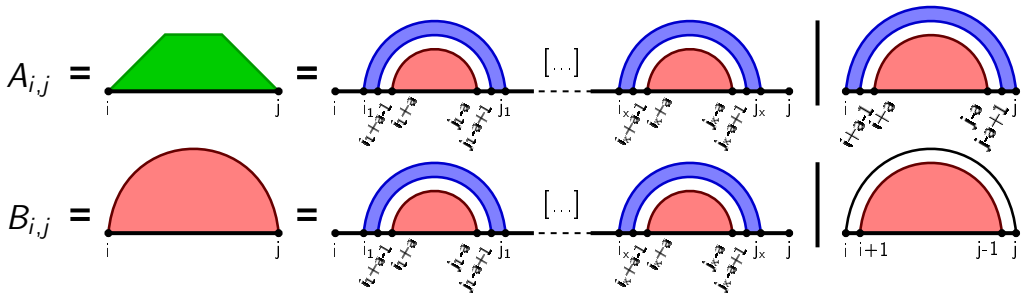

Figure 3.1: **The decomposition employed by Saffarian algorithm.** The substring $w[i,j]$ is treated depending on whether $(i-1, j+1)$ is present or not. If it is absent, all flat structures $\mathfrak{p}(i,j)$ existing for $w[i,j]$ are considered. If it exists, all $\mathfrak{p}(i,j)$ except $\mathcal{H}(i,j,\alpha)$ are considered along with the possibility of extending the helix $\mathcal{H}(i-\alpha+1, j+\alpha-1, \alpha)$ by a single base pair (i,j). The helices are marked by blue.

The Saffarian algorithm is available as the software `RNANR` under open-source public license. The software is coded in `C` and includes the original exhaustive enumeration algorithm created by Azadeh Saffarian *et al* with the update made by Hélène Touzet as well as the results of the research presented within this chapter. It is freely accessible on the site: `https://project.inria.fr/rnalands/rnanr`.

3.1.2 Parametrization of the Saffarian Algorithm

We mentioned the possibility to parametrize the Saffarian algorithm to search only for saturated secondary structures that satisfy the given conditions and its possible impact on the overall complexity of the algorithm. Two of them, θ and α, were already mentioned. The parametrization of Saffarian algorithm were introduced by Hélène Touzet in a previous modification of the original version of the algorithm.

The list of all parameters is as follows:

- **θ - minimum length of a base pair**. This is due to steric strain and angular tension between the bonds which does not allow for a close nucleotides to be paired.
- **α - minimum length of a helix**. The base pair stacks contribute to the stability of the secondary structures, therefore lone base pairs are uncommon [43].
- **π - maximum number of unpaired bases in a hairpin**. The longer unpaired

stretches are less probable due to the condition of saturation.

- $\mathfrak{m}$ is the maximum cumulated size of the unpaired stretches within an internal loop or a bulge. This is the maximum limit of the sum of the lengths α and β of both unpaired stretches. The large bulges/internal loops are considered to be energetically unfavorable [62].
- ν - **maximum length of a base pair**. Specifying this value of this parameter restricts the long-range base pair interactions.
- γ - **maximum number of the branches within the multibranch loop**. The fixed number of the maximum number of the branches decreases the complexity of the Saffarian algorithm from exponential to polynomial.

All parameters are illustrated on Figure 3.2.

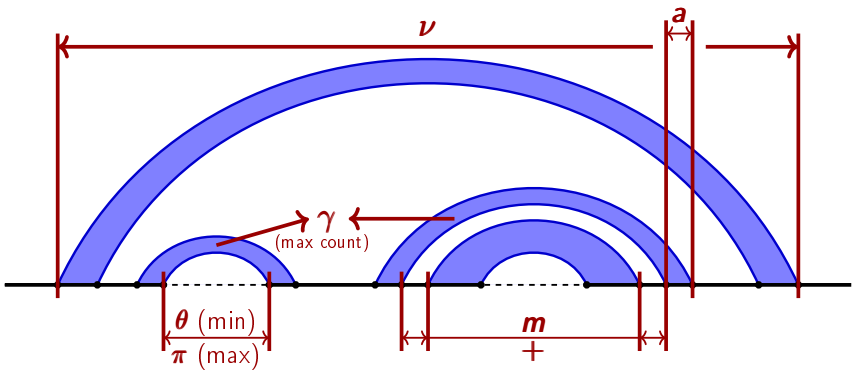

Figure 3.2: **The parameters defined within the Algorithm of Saffarian**. ν denotes the maximum base pair span, which allows to define whether we allow the long-rande interactions or not. θ represents the minimum length of a base pair, while π is the maximum length of a hairpin. γ denotes the maximum number of the branches within the multibranch loop.

The advantage of the parametrization of Saffarian algorithm is it does not induce any supplementary complexity cost. As seen on the example of helix length a and minimum base pair length θ, the parameters are directly implemented into the algorithm with and used with the overall complexity $\mathcal{O}(1)$. Therefore the space $\mathcal{S}_{n_{SAT}}$ can be reduced using the parametrization to restrict the secondary structures without inducing any supplementary complexity cost.

We also redefine $\mathcal{S}_{n_{SAT}}$, resp $\mathcal{S}_{SAT}$ as the set of all saturated secondary structures, resp. those with the length n, that fulfill the conditions imposed by parameters a, π, m, ν and γ.

We specifically discuss the minimum helix length a and the maximum number of the branch of the multibranch loop γ. Limiting both of he values can greatly speed up the algorithm of Saffarian at the cost of a loss of the potentially important part of $\mathcal{S}_{n_{SAT}}$. In the case of the minimum helix length a, the default value is 3. While it has been shown that lone base pairs are a rare occurrence [42], excluding the helices of the length of 2 may have bigger impact. For this reason, we analyze the data of experimentally determined secondary structures and study the helices found within them. The similar analysis will be made to the number of the branches within the multibranch loop.

For this purpose, we used the `RNA STRAND` database [2] - a database of known experimentally determined secondary structures. We performed its statistical analysis in a following manner:

- Create 5 groups of sequences from `RNA STRAND` for which the secondary structures, respectively containing all sequences shorter than 140, 300, 400 and 500 nt, the last group containing all sequences.
- For each group, determine a number of helices having at least a certain length.
- For each group, compute the proportion off all helices having least certain length.

The similar analysis is made for the multibranch loop, except we computed the proportion of all secondary structures which multiloop with most branches has their given number. The results of both analyses is shown on Figure 3.3A and B respectively.

For the number of the branches within the multibranch loop, we can observe that the structures having having multibranch loops with more than 6 branches are rare. Note that for `RNA STRAND`, the number of branches counts in also the pair that closes the multibranch loop, while the paramater of Saffarian algorithm does not, so the number of branches in Figure 3.3A corresponds to $\gamma + 1$. It is safe to parametrize the Saffarian algorithm with $\gamma = 5$, as it leads to a loss of only a negligible proportion of the secondary structures. That would reduce its overall complexity to polynomial, since the maximum number for $|p|$ would be dependent on this parameter instead of the length of the sequence. Such approach seems reasonable considering the number of lost secondary structures is minimal.

The second analysis shows that the helices $\mathcal{H}(i, j, lh)$ with the length $lh = 1$, or

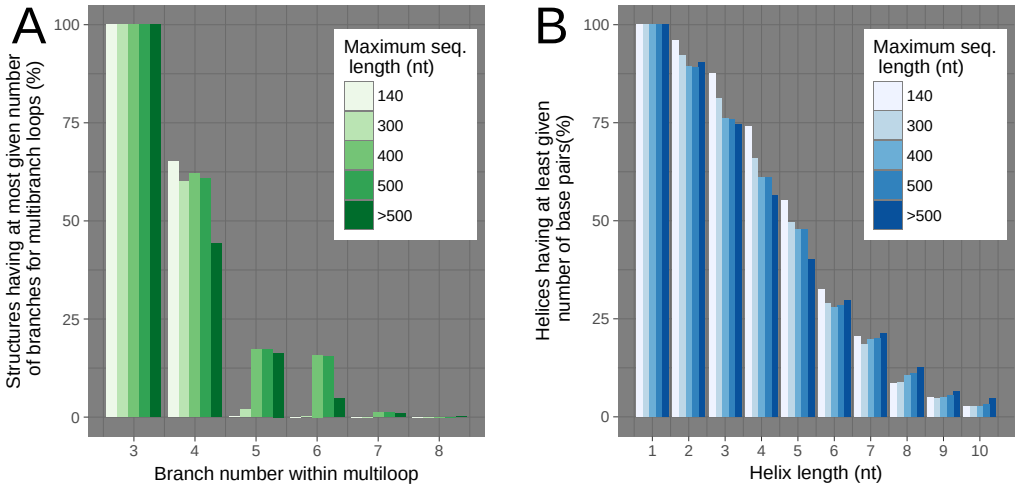

Figure 3.3: **The statistics on the secondary structures from the database RNA STRAND**. The statistics are calculated for the classes of sequences separated by their maximum length. **(A)** The percentage of the structures from RNA STRAND having at maximum the indicated number of branches within each of their multibranch loop. Here this number also include a closing base pair. **(B)** The proportion of helices that have at least the indicated length. The statistics show that both multibranch loops with more than 6 branches and the lone base pairs are uncommon.

lone base pairs (LP), are also rarely observed even among the shorter sequences. In fact, shorter sequences have lower proportions of LP simply because longer sequences have more base pairs, and therefore also LP. However, around 90% of all helices have at least two or more base pairs. On the other hand, only 75% has $\mathcal{H}(i, j, lh)$ with $lh \geq 3$, meaning that setting α will lead to a loss of 1/4 of all helices. However, assuming that the functional structures will not be affected too profoundly by exclusion or eventual elongation of these helices, we can allow to set $\alpha = 3$.

Note that the structures considered here are not all locally optimal or saturated, therefore the statistics for $\mathcal{S}_{n_L}$, resp. $\mathcal{S}_{n_{SAT}}$ might differ a bit.

From this point on, we suppose that $\alpha = 3$. The values of γ are unlimited unless noted otherwise.

3.1.3 Computing the Partition Function

The problem of sampling locally optimal secondary structures was already resolved by A. Lorenz and P. Clote [56] in polynomial time. While the Saffarian al-

gorithm samples in exponential time without any parametrization, this exponential parameter is small and therefore the exponential increase is not as remarkable. Plus, because of the parametrization without complexity overhead, the Saffarian algorithm is a valuable tool for searching the locally optimal structures within the defined parametrized space. Since such conditions can be specified, it is easier to look for the structures whose specifics are known. For this reason we decided to work with this algorithm.

Unfortunately, the algorithm of Saffarian does not compute the energies of the secondary structures, instead it enumerates them exhaustively. Therefore, it is impossible to sample the structures since the partition function cannot be computed without energies. Our task is to analyze how to compute the energies of the flat structures so we can compute the partition function using the dynamic programming.

The decomposition of the secondary structures in flat structures allows is to implement the Turner Energy Model by associating the specific flat structures to specific loops. First, consider all possible cases without an extension of an existing helix of some flat structure. We know each such flat structure $\mathfrak{p}$ contains u helices $\mathcal{H}(i_x, j_x, a)$ such that $1 \leq x \leq u$. Consequently, their energy can be computed by

$$E_{\mathcal{H}(i,j,lh)} = \sum_{y=0}^{a-1} E_{St}(i+y, j-y).$$

Definition 3.1.8 (Flat structure energy): he energy of the flat structure $E_{\mathfrak{p}}$ can be computed as

$$E_{\mathfrak{p}} = C_{\mathfrak{p}} + \sum_{x=1}^{u} \sum_{y=0}^{lh-1} E_{St}(i_x + y, j_x - y) \tag{3.2}$$

where $C_{\mathfrak{p}}$ is an energy term that depends on the properties of $\mathfrak{p}$.

The properties determining $C_{\mathfrak{p}}$ are: - number u of helices $\mathcal{H}(i_x, j_x, a)$ such that $1 \leq x \leq u, lh_x > a$; - whether it is nested in another flat structure.

Depending on these properties, we can identify the type of loop that $\mathfrak{p}$ represents and associate a corresponding value to $C_{\mathfrak{p}}$

- If $u \in \mathbb{N}$ and $\mathfrak{p}$ is not in another helix, it is **an external loop** - $C_f = 1$.
- If $u = 0$ and $\mathfrak{p}$ is in another helix, $\mathfrak{p}$ is an hairpin - $C_{\mathfrak{p}} = E_H(i, j)$.

- If $u = 1$ and p is nested in another helix, p constitutes an internal loop or a bulge depending on i, i_1, j_1, j, therefore $C_p = E_{ILG}(i, i_1, j, j_1)$.
- If $u = 2$ and p is nested in another helix, p constitutes a multibranch loop and $C_p = a + b \times u + c \times N_{unp}$, according to the Equation 2.6.

The extension is a specific case that is equivalent to an elongation of an existing helix, adding an energy term to it.

Definition 3.1.9 (Extension energy): he energy of an extension $p_{\mathcal{E}}$ of helix $\mathcal{H}(i, j, lh)$ such that $\mathcal{H}(i, j, a) \in p, lh \geq a$ is

$$E_{p_{\mathcal{E}}} = E_{St}(i + lh - 1, j - lh + 1).$$

Once each p has been associated to the specific loop case, we can compute the energy of entire secondary structure:

$$E_S = \sum_{p \in S} E_p + \sum_{E_{p_{\mathcal{E}}} \in S} E_{p_{\mathcal{E}}}.$$

With the energies of p, resp. $p_{\mathcal{E}}$ being acquired, we can proceed to computation of the Boltzmann distribution. Such a distribution prioritizes the low-energy structures but does not completely ignore high-level energies.

The partition function is computed by employing a DP scheme that recursively includes one flat structure in another and extends their helices. Due to the presence of an extension, we need to distinguish two cases:

- the interval $w[i, j]$ is not included in some helix $\mathcal{H}(i - lh, j + lh, lh)$ with $lh > a$; the partition function is denoted by $\mathcal{Z}^U_{i,j}$;
- the interval $w[i, j]$ is included in a helix $\mathcal{H}(i - lh, j + lh, lh)$; the partition function is denoted by $\mathcal{Z}^N_{i,j}$.

The values of these two partition functions can be computed as:

$$
\begin{aligned}
\mathcal{Z}^{U}_{i,j} &= \sum_{\mathfrak{p}(i,j)\in\mathcal{P}(i,j)} e^{\frac{-E_{\mathfrak{p}}(i,j)}{k_B T}} \times \prod_{c=1}^{x} \mathcal{Z}^{H}_{i_c+a,j_c-a} \\
\mathcal{Z}^{H}_{i,j} &= \sum_{\mathfrak{p}(i,j)\in\mathcal{P}(i,j)/\mathcal{H}(i,j,a)} e^{\frac{-E_{\mathfrak{p}}(i,j)}{k_B T}} \times \prod_{c=1}^{x} \mathcal{Z}^{H}_{i_c+a,j_c-a} + e^{\frac{-E_{\mathfrak{p}_{\mathcal{E}}}(i,j)}{k_B T}} \times \mathcal{Z}^{H}_{i+1,j-1}
\end{aligned}
\tag{3.3}
$$

In the case of $\mathcal{Z}^{B}_{i,j}$, we need to exclude $\mathcal{H}(i,j,a)$ since it would extend the helix $\mathcal{H}(i-a,j-a,a)$ of the parent flat structure $\mathfrak{p}'$ and would create ambiguous situation since such decomposition is handled by the extensions. The extension is then included in $\mathcal{Z}^{H}_{i,j}$ since we know that it is included in some $\mathcal{H}(i-a,j-a,a)$. The partition function $\mathcal{Z}$ is given by $\mathcal{Z}^{H}_{1,n}$.

The computation of the energies of the loops in `RNANR` employing Saffarian algorithm uses the functions from ViennaRNA which also includes all necessary energy parameters. These functions as well as DP scheme from Equation 3.3 are implemented into the `RNANR` in `C` language. The stochastic backtrack constituting the base of the sampling in the way similar to Section 2.5.6.

The similar approach was used independently in works of Jérôme Waldispühl and Peter Clote [95].

3.1.4 Formalization

Here we express the DP scheme employed in Saffarian algorithm using the formalism we established in Seaction 2.6. Its formalization is quite simple. First, we need to define states $\mathcal{Q}$. As with Zuker and McCaskill algorithm, $w[i,j]$ alone is not sufficient to define a state, but we need to know whether $(i-1,j+1) \in S_{SAT}(i,j)$, with $S_{SAT}(i,j)$. a saturated structure constructed by DP scheme. We therefore define two types of states:

- $q^{U}_{i,j} = w[i,j] | (i,j) \notin S_{SAT}(i,j)$
- $q^{H}_{i,j} = w[i,j] | (i,j) \in S_{SAT}(i,j)$

The set of all states is then defined by

$$\mathcal{Q}_{Saf} = \{q^{U}_{i,j}, q^{H}_{i,j} \mid 1 \leq i < j \leq n\}.$$

There are two types of derivation and associated constructors:

- $\lambda^{flat}_{i,j,\mathfrak{p}(i,j)}$ - associates a flat structure $\mathfrak{p}(i,j) \in \mathcal{P}(i,j)$ to $w[i,j]$
- $\lambda^{ext}_{i,j}$ - an extension of a helix $\mathcal{H}(i-lh, j+lh, lh), lh > a$

We can consequently define ρ_{Saf}:

$$\begin{aligned}
\rho_{Saf}(q^U_{i,j}) &= \bigcup_{\mathfrak{p}(i,j)\in\mathcal{P}(i,j)} \left\{(\{q^H_{i_1+a,i_1-a}, \ldots, q^H_{i_u+a,i_u-a}\}, \lambda^{flat}_{i,j,\mathfrak{p}(i,j)}, E_{\mathfrak{p}}(i,j))\right\} \\
\rho_{Saf}(q^H_{i,j}) &= \bigcup \begin{cases} \bigcup\limits_{\substack{\mathfrak{p}(i,j)\in\mathcal{P}(i,j)/\\ /\mathcal{H}(i,j,a)}} \left\{(\{q^H_{i_1+a,i_1-a}, .., q^H_{i_u+a,i_u-a}\}, \lambda^{flat}_{i,j,\mathfrak{p}(i,j)}, E_{\mathfrak{p}}(i,j))\right\} \\ \\ \left\{(\{q^H_{i,j}\}, \lambda^{ext}_{i,j}, -E_{\mathfrak{p}_{\mathcal{E}}}(i,j))\right\} \end{cases}
\end{aligned} \tag{3.4}$$

Here $q_{root} = q^U_{1,n}$. We have thusly completely defined the DP Scheme used in Saffarian algorithm as $(Q_{Saf}, q^U_{1,n}, \rho_{Saf})$.

To compute the partition function, we proceed by application of the Equation 2.6.2. This gives us:

$$\begin{aligned}
PfS_{Saf}(q^U_{i,j}) &= \sum_{q^U_{i,j} \xrightarrow[E_{\mathfrak{p}}(i,j)]{\lambda^{flat}_{i,j,\mathfrak{p}(i,j)}} \{q'^H_{i_1,j_1}, \ldots, q'^H_{i_u,j_u}\}} e^{\frac{E_{\mathfrak{p}}(i,j)}{k_B T}} \times \left(\prod_{i=1}^{u} PfS_{Saf}(q'^H_{i_x,j_x})\right) \\
PfS_{Saf}(q^H_{i,j}) &= \sum_{q^H_{i,j} \xrightarrow[E_{\mathfrak{p}}(i,j)]{\lambda^{flat}_{i,j,\mathfrak{p}(i,j)}} \{q'^H_{i_1,j_1}, \ldots, q'^H_{i_u,j_u}\}} e^{\frac{-E_{\mathfrak{p}}(i,j)}{k_B T}} \times \left(\prod_{i=1}^{u} PfS_{Saf}(q'^H_{i_x,j_x})\right) + \\
&\quad + e^{\frac{E_{\mathfrak{p}}(i,j)}{k_B T}} \times PfS_{Saf}(q'^H_{i+1,j+1})
\end{aligned}$$

Finally, the search space for each state q can be defined as:

$$
\begin{aligned}
\mathcal{L}_{Saf}(q^{U}_{i,j}) &= \bigcup_{q^{U}_{i,j} \xrightarrow[E_{\mathfrak{p}}(i,j)]{\lambda^{flat}_{i,j,\mathfrak{p}(i,j)}} \{q'^{H}_{i_1,j_1},\ldots,q'^{H}_{i_u,j_u}\}} \lambda_{\mathfrak{p}}(\mathcal{L}(q'^{H}_{i_1+a,i_1-a}),\ldots,\mathcal{L}(q'^{H}_{i_u+a,i_u-a})) \\
\mathcal{L}_{Saf}(q^{H}_{i,j}) &= \bigcup \begin{cases} \bigcup_{q^{U}_{i,j} \xrightarrow[E_{\mathfrak{p}}(i,j)]{\lambda^{flat}_{i,j,\mathfrak{p}(i,j)}} \{q'^{H}_{i_1,j_1},\ldots,q'^{H}_{i_u,j_u}\}} \lambda_{\mathfrak{p}}(\mathcal{L}(q^{H}_{i_1+a,i_1-a}),\ldots,\mathcal{L}(q^{H}_{i_u+a,i_u-a})) \\ \lambda_{ext}(\mathcal{L}_{Saf}(q^{H}_{i+1,j-1})) \end{cases}
\end{aligned}
\tag{3.5}
$$

The stochastic backtrack is then formalized by using the Equation 2.18, where f_λ is either the function computing the partition function for applying $\mathfrak{p}$ or an extension of one of helices of $\mathfrak{p}'$.

3.1.5 Comparison of Saffarian and Turner Local Minima

Saffarian algorithm assumes S_{SAT} is locally optimal if it is saturated. However, nothing so far confirms such assumption, specifically with regards to Turner Energy Model. Since we use this energy energy model to compute the energies of all flat structures and secondary structures determined by the Saffarian algorithm, we would like to study whether the saturated secondary structures are effectively the local minima according to it.

The local optimality implies that all neighbors $S_{neigh} \in \mathcal{V}_{S_{SAT}}$ to S_L satisfy:

$$E_{S_{SAT}} < \min_{S_x \in \mathcal{V}_{S_{SAT}}} E_{S_x}.$$

To validate this condition we apply the method of **gradient walk**. This method searches for all possible neighbors of a saturated secondary structure S_{SAT} and calculates their free energy. Should one of neighbors have lower energy than S_{SAT}, then the neighbor becomes then new minimum and the entire process is repeated until we reach the real local minimum. If S_{SAT} is local minimum then no descent will be possible.

For this purpose, we employ the gradient walk function implemented in ViennaRNA.

It must be pointed out that the space of secondary structures accessible by the

gradient walk is not the same as the one accessible by the Saffarian algorithm. While the gradient walk can access the entirety of $\mathcal{S}_n$, the Saffarian algorithm can access only the parametrized space $\mathcal{S}_{n_{SAT}} \subset \mathcal{S}_n$. This means that after performing the gradient walk on S_{SAT} we might find ourselves with $S \notin \mathcal{S}_{n_{SAT}}$. It is important to know whether the structure is not locally optimal because $\mathcal{S}_{n_{SAT}}$ does not include such structure or it is caused by the conception of the algorithm. For this reason here we will separate the results obtained for the gradient walk to those that are **within search space**, or where the final S is in parametrized $\mathcal{S}_{n_{SAT}}$, and those that are not not, or are **outside search space**.

To compute these values, we used 154 sequences from `RNA STRAND`, or all sequences from the database version from 04/11/2011 having between and including 120 and 170 nt. This ensures the sufficient variability for the secondary structures and avoiding the situations where saturated secondary structures would be also locally optimal by random chance.

The proposed pipeline of validation of saturated structures as locally optimal is:

1. By the Saffarian algorithm, sample 1000 unique saturated secondary structures S_{SAT}. This can be either done by sampling S_{SAT} and removing duplicates until we have 1000 uniques, or more easily performing the non-redundant sampling that will be detailed in the next section.
2. For each S_{SAT}, perform a gradient walk using a dedicated ViennaRNA function.
3. Compare the gradient walk result S_{grad} with corresponding S_{SAT}.
4. For the S_{grad} that are different, check out whether they satisfy the parametrization of Saffarian Algorithm, ie they validate the parameters θ, a, π, m, ν and γ.

The results are listed in Table 3.1. The S_{grad} that are identical to starting S_{SAT} are counted in the category **Within search space**. We found that of 59.57% of S_{grad} within search space, of which about 90% are locally optimal according to the Turner Energy Model. For the $S_{grad} \in \mathcal{S}_{SAT}$ that differ from S_{SAT}, the true local minimum is within two base pairs most of the time (average at 1.550), and within the 2 kcal.mol^{-1}of free energy (average at $\Delta\Delta$1.258). The saturated secondary structures S_{SAT} are therefore identical to S_{grad} in 52%, and in most of others S_{grad} is quite close. However, in the majority of cases where S_{grad} and S_{SAT} differ we have $S_{grad} \notin \mathcal{S}_{SAT}$. This suggests that the local optimum cannot be reached because it is outside $\mathcal{S}_{SAT}$. On the other hand, there are some case where

the saturated structure is not locally optimal. However, since such results are few and far apart, we can indeed assume that $S_{SAT} \in S_L$ and therefore that Saffarian algorithm returns locally optimal structures.

	Samples% avg (std.dev)	ΔΔG avg (std.dev)	Base pair dist. avg (std.dev)
Within search space	59.57% (21.00)	0.071 (0.309)	0.129 (0.289)
Outside search space	40.42% (21.00)	1.248 (0.925)	1.550 (0.619)
Global average	100.00% (–)	**0.547** (0.817)	**0.703** (0.757)

Table 3.1: **The comparison whether the saturated secondary structures are locally optimal**. The category denoted 'Within search space' marks the cases where $S_{grad} \in \mathcal{S}_{SAT}$. These structures are identical to S_{SAT} in about 50% of cases. The category denoted 'Outside search space' specifies locally optimal structures obtained the same way and not fulfilling the parametrization of the Saffarian's algorithm. ΔΔG specifies the average difference of energy between the two structures, while the average base pair distance accounts to the average number of different base pairs between them.

Note that this experiment verifies that S_{SAT} are locally optimal with respect to the Turner Energy Model, but it does not confirm whether the space of saturated secondary structures completely covers the space of locally optimal structures that respect the given parameters. This is not problematic though, since in this case we are interested only in saturated secondary structures.

3.2 The Non-redundant Sampling

Problem 3.2.1:

- ***Input:*** *DP scheme that can be formalized as* $(\mathcal{Q}, q_{root}, \rho)$*;*
- ***Output:*** *set of length* N *of* ***unique*** *secondary structure samples in normalized Boltzmann distribution.*

The advised non-redundant sampling strategy has to fulfill two conditions:

- The method must be **applicable to any DP scheme** computing the partition function for the purpose of predicting suboptimal secondary structures of RNA.

- The method should be **easy to implement**, therefore constituting an interface to the adjacent DP scheme with the lowest number of interactions possible.

The general concept of non-redundant sampling was introduced and discussed for weighted generation from context-free grammars by William Lorenz *et al* [57]. We push said method further, adapt the result for the purpose of RNA secondary structure prediction and put it into practice.

Here, we will, after introducing the necessary notions and explaining the general principle of non-redundant sampling, research the method that fulfills aforementioned criteria. We will discuss on the points that must be addressed to create such method. The two main interdependent issues is the memorization process of the sampled secondary structures and their decomposition and how to ensure that the distribution of non-sampled structures stays unbiased. While the first experiments will be made on the `RNANR`, we will try to generalize the principle as much as possible using the formalism we introduced except for specific examples.

We will follow by the experiments demonstrating the efficiency of this approach when compared to RNA usual sampling methods. We will then describe the implementation of researched non-redundant sampling method in the most optimal way and backing the general applicability of it to the current DP schemes used for sampling of secondary structures from Boltzmann distribution.

3.2.1 Parse tree and Incomplete structures

Before explaining the non-redundant sampling, we must introduce the notion of parse tree and incomplete structures in the context of secondary structure sampling. Most DP schemes only consider what happens for the given q at the given point, but not the events taking place outside of it. The principle of non-redundant sampling necessitates to know this information, and to express it from that generated by DP scheme. The specified concepts are introduced for this reason.

Each DP scheme $(\mathcal{Q}, q_{\mathtt{root}}, \rho)$ can be captured as a context-free grammar [57], where

- The alphabet is the set $\mathcal{D}$ of constructors λ;
- The non terminal symbols are the states $q \in \mathcal{Q}$;

- The production rules are derivations $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$, taking q and returning λ and $q'_1, \ldots, q'_u$ - for now we ignore the score contribution c_λ;
- The axiom is q_{root}.

Therefore each secondary structure can be represented as a parse tree.

Definition 3.2.1 (Parse tree): A **secondary structure parse tree** is a tree $\mathfrak{T}(w, \mathcal{Q}, q_{root}, \rho)$ for which we have:

- Internal nodes $\lambda \in \mathcal{D}$ - each parent of $\mathbf{u}$ subtrees $\mathfrak{T}_1, \ldots, \mathfrak{T}_u$ with derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$;
- Leaves $\lambda_{term} \in \mathcal{D}$ such that $q \xrightarrow[c_{\lambda_{term}}]{\lambda_{term}} \varnothing$.

Such a tree represents a structure where an application of constructor leads to application of λ_{1a} and λ_{1b} etc., each building a specific element.

Now suppose the structure is not completely derived yet on states q. In that case some of the leaves will not be represented by a constructor λ, but instead by some state q which can be derived by a number of derivations $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$.

Definition 3.2.2 (Immature parse tree): Let $\mathcal{Q}_N$ be the list of underived states. An **immature secondary structure parse tree** is a tree $\mathfrak{T}_I(w, \mathcal{Q}, \mathcal{Q}_N, q_{root}, \rho)$ for which we have:

- Internal nodes $\lambda \in \mathcal{D}$ - each parent of $\mathbf{u}$ subtrees $\mathfrak{T}_1, \ldots, \mathfrak{T}_u$ with derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$;
- Leaves $\lambda_{term} \in \mathcal{D}$ such that $q \xrightarrow[c_{\lambda_{term}}]{\lambda_{term}} \varnothing$ or $q \in \mathcal{Q}_N$.

The immature parse tree evolves as the states q get derived as $\lambda_{term} \in \mathcal{D}$ such that $q \xrightarrow[c_{\lambda_{term}}]{\lambda_{term}} \varnothing$. We can represent this evolution by a decomposition tree.

Definition 3.2.3 (Decomposition tree): A **decomposition tree** is a tree

$$\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$$

for which:

- Each internal node V is a **immature** secondary structure parse tree $\mathfrak{T}_I(w, \mathcal{Q}, \mathcal{Q}_N, q_{root}, \rho)$.
- Each leaf V_{term} is **mature** secondary structure parse tree $\mathfrak{T}(w, \mathcal{Q}, q_{root}, \rho)$.
- Each edge $V_1 \rightarrow V_2$ represents a derivation $\lambda_{term} \in \mathcal{D}$ such that $q \xrightarrow[c_{\lambda_{term}}]{\lambda_{term}} \varnothing$, where q is a leaf of a parse tree in some node V.

Example 3.2.1. Suppose a sequence w and a DP scheme $(\mathcal{Q}_{Ex}, q_{root}, \rho_{Ex})$ where

$$\mathcal{Q}_{ex} = \{q_{root}, q_{1A1}, q_{1B1}, q_{1B1}, q_{1B2}\}$$

and the ρ_{Ex} is given by

$$\begin{aligned}
\rho_{ex}(q_{root}) &= \{(\{q_{1A1}, q_{1A2}\}, \lambda_{1A}, c_{1A}), (\{q_{1B1}, q_{1B2}\}, \lambda_{1B}, c_{1B})\} \\
\rho_{ex}(q_{1A1}) &= \{(\varnothing, \lambda_{2A}, c_{2A})\} \\
\rho_{ex}(q_{1A2}) &= \{(\varnothing, \lambda_{2B}, c_{2B})\} \\
\rho_{ex}(q_{1B1}) &= \{(\varnothing, \lambda_{2C}, c_{2C})\} \\
\rho_{ex}(q_{1B2}) &= \{(\varnothing, \lambda_{2D}, c_{2D})\}
\end{aligned}$$

The decomposition tree is given by Figure 3.4. Each internal node is represented by an immature parse tree, and each leaf is a mature parse tree. At the beginning there is only the initial state dproot. This state gets derived by one of two available derivations, $q_{root} \xrightarrow[c_{\lambda_{1A}}]{\lambda_{1A}} \{q_{1A1}, q_{1A2}\}$ and $q_{root} \xrightarrow[c_{\lambda_{1B}}]{\lambda_{1B}} \{q_{1B1}, q_{1B2}\}$, giving two new different states for each case. We derive the states q_{1A1} and q_{1A2} that follow $q_{root} \xrightarrow[c_{\lambda_{1A}}]{\lambda_{1A}} \{q_{1A1}, q_{1A2}\}$, resp. q_{1B1} and q_{1B2} that follow $q_{root} \xrightarrow[c_{\lambda_{1B}}]{\lambda_{1B}} \{q_{1B1}, q_{1B2}\}$ consecutively, leading to the mature parse trees at the end of each branch.

The important point is the order in which the states q of an incomplete structure are derived. This order must be deterministic, otherwise multiple decomposition trees may be created for the same DP scheme and sequence. In our case we assume the state q to be derived is the rightmost one, and that all trees for the same DP scheme $(\mathcal{Q}, q_{root}, \rho)$ and sequence w have always the same layout.

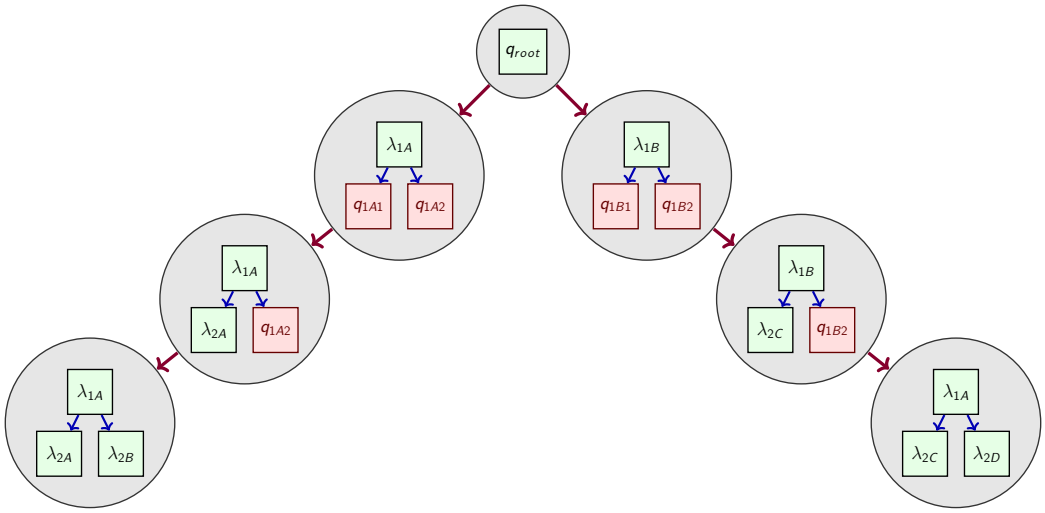

Figure 3.4: **Decomposition tree** $\mathcal{T}(w, \mathcal{Q}_{ex}, q_{root}, \rho_{ex})$**.** Each itnernal node is constituted from an immature parse tree $\mathfrak{T}_I(w, \mathcal{Q}_{ex}, \mathcal{Q}_N, q_{root}, \rho_{ex})$, while the leaves represent a mature parse trees $\mathfrak{T}_I(w, \mathcal{Q}_{ex}, q_{root}, \rho_{ex})$ and $\mathfrak{T}(w, \mathcal{Q}_{ex}, q_{root}, \rho_{ex})$. The nodes are connected by the derivations $q \xrightarrow[c_\lambda]{\lambda} q'_1, \ldots, q'_u$ on the nodes q of immature trees.

We compute the search space $\mathcal{L}(\mathfrak{T})$ associated to a tree $\mathfrak{T}(w, \mathcal{Q}, q_{root}, \rho)$ or $\mathfrak{T}_I(w, \mathcal{Q}, q_{root}, \rho)$ in a recursive manner:

$$\mathcal{L}(\mathfrak{T}) = \begin{cases} \lambda(\mathcal{L}(\mathfrak{T}_1), \cdots, \mathcal{L}(\mathfrak{T}_u))) & \text{if } \lambda \text{ an internal node or mature leaf} \\ \lambda(\mathcal{L}(q)) & \text{if } q \text{ an immature leaf} \end{cases} \tag{3.6}$$

Theorem 3.2.1 (Search space inclusion): *Suppose a decomposition tree $\mathcal{T}$ where V contains two parse trees $\mathfrak{T}_I$ and $\mathfrak{T}'_I$ connected by an edge representing a derivation $q \xrightarrow[c_\lambda]{\lambda} q'_1, \ldots, q'_u$ from $\mathfrak{T}_I$ to $\mathfrak{T}'_I$, $q \in \mathfrak{T}_I$. In that case*

$$\mathcal{L}(\mathfrak{T}'_I) \subseteq \mathcal{L}(\mathfrak{T}_I).$$

Proof 3.2.1. The derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ implies a tree $\mathfrak{T}_I$ contain a state q that can be derived. Since Definition 2.6.1 defines constructor λ as a function with properties of cartesian product and the $\mathfrak{T}_I$, resp. $\mathfrak{T}'_I$ can be defined recursively by the properties of subtrees $\mathfrak{T}_1, \cdots, \mathfrak{T}_u$, we can separate $\mathcal{L}(q)$ from other terms in $\mathcal{L}(\mathfrak{T}_I)$. The derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ introduces λ and the space of structures $\lambda(\mathcal{L}(q'_1, \ldots, q'_u))$. Equation 2.6.1 implies that

$$\lambda(\mathcal{L}(q'_1, \ldots, q'_u)) \subseteq \mathcal{L}(q).$$

Due to the properties of λ as cartesian product, it follows

$$\mathcal{L}(\mathfrak{T}_I') \subseteq \mathcal{L}(\mathfrak{T}_I).$$

We define an incomplete secondary structure, whose notion is directly related to that of an immature secondary structure parse tree.

Definition 3.2.4 (Incomplete structure): We call **an incomplete structure** S_I a secondary structure that contains at least one state q that was not yet derived. For a sequence w, an incomplete structure is always of size n.

Example 3.2.2. The illustrated example for DP scheme

$$(\mathcal{Q}_{Zuk}, q^F_{1,12}, \rho_{Zuk})$$

employed in Zuker algorithm is available on Figure 3.5. The sequence

$$w = \mathsf{ACCCAAAGGAGA}$$

contains an incomplete structure constituted of base pairs $(2, 11)$ and $(3, 9)$ and an underived state $q^C_{4,8}$. We may choose among two possible derivations:

- $q^C_{4,8} \xrightarrow[c_{\lambda^{ILG}_{3,9,4,8}}]{\lambda^{ILG}_{3,9,4,8}} \{q^C_{5,7}\}$ creating a stack of base pairs $(3, 9)$ and $(4, 8)$.
- $q^C_{4,8} \xrightarrow[c_{\lambda^{H}_{4,8}}]{\lambda^{H}_{4,8}} \varnothing$ that creates a hairpin.

An incomplete structure delimits a certain subspace of $\mathcal{S}_n$.

Definition 3.2.5 (Incomplete structure space): The **incomplete structure space** $\mathcal{S}_\mathcal{I}$ of S_I is a space of secondary structures that can be obtained by deriving on underived state q by $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$.

Example 3.2.3. Assume that the state $q^C_{4,8}$ is the only underived state of of an incomplete structure $S_{I\alpha}$ from Figure 3.5. This state can be derived by $q^C_{4,8} \xrightarrow[c_{\lambda^{ILG}_{3,9,4,8}}]{\lambda^{ILG}_{3,9,4,8}} \{q^C_{5,7}\}$ and $q^C_{4,8} \xrightarrow[c_{\lambda^{H}_{4,8}}]{\lambda^{H}_{4,8}} \varnothing$. The former will lead to an incomplete structure $S_{I1} = \{(2, 11), (3, 9), (4, 8)\}$ and a state $q^C_{5,7}$ on which only the derivation by $q^C_{5,7} \xrightarrow[c_{\lambda^{H}_{5,7}}]{\lambda^{H}_{5,7}} \varnothing$

is possible, giving $S_1 = \{(2,11),(3,9),(4,8)\}$. $\lambda^{H}_{4,8}$ leads to a complete structure and $S_2 = \{(2,11),(3,9)\}$. Therefore

$$\mathcal{S}_{\mathcal{I}a} = \{S_1, S_2\}.$$

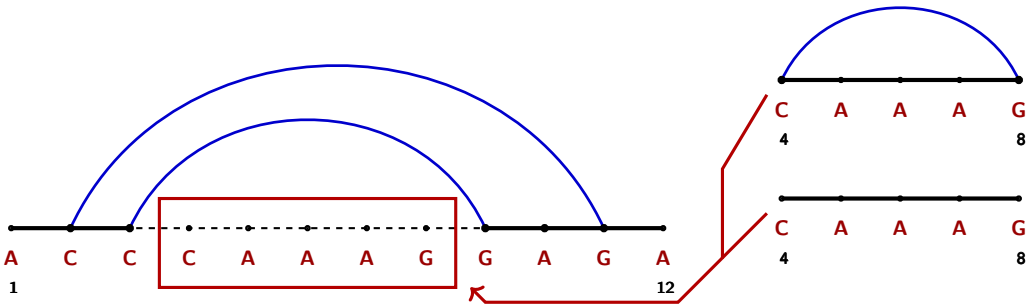

Figure 3.5: **Illustration of the incomplete structure.** Under DP scheme $\{\mathcal{Q}_{Zuk}, q^{F}_{1,12}, \rho_{Zuk}\}$ used in Zuker algorithm, the example of a incomplete structure S_I for the sequence w=ACCCAAAGGAGA contains two base pairs $(2,11)$ and $(3,9)$, and a state $q^{C}_{4,8}$ with no λ. Two constructors, $\lambda^{ILG}_{2,9,4,8}$ and $\lambda^{H}_{3,9,4,8}$, can be attributed to this state to generate two complete secondary structures. $\mathcal{S}_{\mathcal{I}}$ is constituted from two complete structures: one where 4^{th} and 8^{th} nucleotide is paired and one where not.

The partition function of an incomplete structure S_I is

$$\mathcal{Z}_{S_I} = \sum_{S \in \mathcal{S}_{\mathcal{I}}} \mathcal{Z}_S.$$

In the decomposition tree $\mathcal{T}$, each node in V can be seen as an incomplete secondary structure S_I where leaves q are underived states in *secstrinc*. Each node is an immature parse tree $\mathfrak{T}_I$ with known set of underived states q and a chain of performed derivations, as well as a subset of structures $\mathcal{L}(\mathfrak{T}_I)$. Therefore $\mathfrak{T}_I$ represents a secondary structure S_I and $\mathcal{L}(\mathfrak{T}_I) = \mathcal{S}_{\mathcal{I}}$. This also implies that for two incomplete structures $\mathcal{S}_{\mathcal{I}a}$ and $\mathcal{S}_{\mathcal{I}b}$ connected by an edge in $\mathcal{T}$, where $\mathcal{S}_{\mathcal{I}b}$ was obtained by deriving q of $\mathcal{S}_{\mathcal{I}a}$, we have

$$\mathcal{S}_{\mathcal{I}b} \subseteq \mathcal{S}_{\mathcal{I}a}.$$

A mature tree $\mathfrak{T}(w, \mathcal{Q}, q_{root}, \rho)$ represents a completed structure, since it does not have any leaf q. Said structure is constructed by following a path in $\mathfrak{T}$ from root to specified leaf.

General Principle of Non-redundant sampling

Suppose two incomplete structures S_{Ia}, represented by an immature parse tree $\mathfrak{T}_{Ia}$ with a leaf q the next state to be derived, and S_{Ib}, represented by $\mathfrak{T}_{Ib}$ such that S_{Ib} is obtained by derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ of q in S_{Ia} and is consequently connected by an edge in an associated decomposition tree $\mathcal{T}(w, q, q_{root}, \rho)$. In addition to the Equation 2.21, the probability $p(\lambda \mid q)$ can be expressed as

$$p(\lambda \mid q) = \frac{\mathcal{Z}_{S_{Ib}}}{\mathcal{Z}_{S_{Ia}}} \tag{3.7}$$

If we want to perform a non-redundant sampling, we need to make sure that at every moment we choose a derivation, the newly generated incomplete structure S_{Ib} will not lead to $\mathcal{S}_\mathcal{I}$ composed only from the structures that were already sampled. This must be verified every time a new λ is selected. This is simple if one knows what structures $S_\mathcal{F}$ were already sampled and which $S_F \in \mathcal{S}_\mathcal{I}$, with $S_\mathcal{F}$ the corresponding forbidden structure. We denote the space of forbidden structures, structures than must not be generated, by $\mathcal{F}$.

In the context of non-redundant sampling, $\mathcal{F}$ contains all already sampled sampled structures. The set of forbidden structures $\mathcal{F}_{S_I}$ that can be generated by completing S_I is

$$\mathcal{F}_{\mathcal{S}_\mathcal{I}} = \mathcal{F} \cap \mathcal{S}_\mathcal{I}.$$

Its partition function $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}}$ is

$$\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}} = \sum_{S \in \mathcal{F}_{\mathcal{S}_\mathcal{I}}} \mathcal{Z}_S.$$

Definition 3.2.6 (Non-redundant emission probability): Let $\mathcal{F}_{\mathcal{S}_{\mathcal{I}a}}$ and $\mathcal{F}_{\mathcal{S}_{\mathcal{I}b}}$ be the forbidden structures that can be generated by completing S_{Ia}, resp. S_{Ib}. The probability $p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_{\mathcal{I}a}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}b}})$ of choosing a derivation $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ is

$$p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_{\mathcal{I}a}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}b}}) = \frac{\mathcal{Z}_{S_{Ib}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}b}}}}{\mathcal{Z}_{S_{Ia}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}a}}}}. \tag{3.8}$$

Theorem 3.2.2 (Non-redundant generation probability): *Using the non-redundant emission probability for each derivation, we generate a structure* $S \in \mathcal{L}(q_{root}) - \mathcal{F}$ *with global probability*

$$p(S \mid \mathcal{F}) = \frac{\mathcal{Z}_S - \mathbb{1}_{S\in\mathcal{F}} \times \mathcal{Z}_S}{\mathcal{Z} - \mathcal{Z}_{\mathcal{F}}}. \tag{3.9}$$

where $\mathbb{1}_{S\in\mathcal{F}}$ is an indicator function that takes value 1 if $S \in \mathcal{F}$ and 0 otherwise. This is effectively the expression of a probability to sample S; if $S \in \mathcal{F}$ we have $p(S \mid \mathcal{F}) = 0$, otherwise its scale accordingly by the Boltzmann factors of the structures not in $\mathcal{F}$.

Proof 3.2.2. From the decomposition tree $\mathcal{T}(w, q, q_{root}, \rho)$, the generation of a structure S can be represented by a path of $\mathcal{T}$ where leaf $\mathfrak{T}_{I\nu}$ represents S:

$$\mathfrak{T}_{I0} \to \mathfrak{T}_{I1} \to \cdots \to \mathfrak{T}_{I\nu} = \mathfrak{T}.$$

Each $\mathfrak{T}_{I\nu}$ represents an incomplete secondary structure, therefore

$$S_{I0} \to S_{I1} \to \cdots \to S_{I\nu} = S.$$

Assume that $\mathcal{F}_{S_{I\varkappa}}$ is the set of forbidden structures which can be generated from $S_{I\varkappa}$ where $1 < \varkappa < \nu$. The probability $p(S \mid \mathcal{F})$ of choosing S is given by the probability of selecting each λ for which $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ leads to it:

$$\begin{aligned} p(S \mid \mathcal{F}) = p(\lambda_1 \mid S_{I0}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_0}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_1}}) \times p(\lambda_2 \mid S_{I1}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_1}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_2}}) \times \cdots \\ \cdots \times p(\lambda_\nu \mid S_{I\nu-1}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{\nu-1}}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_\nu}}) \end{aligned}$$

From there, we get

$$\begin{aligned} p(S \mid \mathcal{F}) &= \frac{\mathcal{Z}_{S_{I1}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}1}}}}{\mathcal{Z}_{S_{I0}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}0}}}} \times \frac{\mathcal{Z}_{S_{I2}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}2}}}}{\mathcal{Z}_{S_{I1}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}1}}}} \times \cdots \times \frac{\mathcal{Z}_{S_{I\nu}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}\nu}}}}{\mathcal{Z}_{S_{I\nu-1}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}_{\nu-1}}}}} \\ p(S \mid \mathcal{F}) &= \frac{\mathcal{Z}_{S_{I\nu}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}\nu}}}}{\mathcal{Z} - \mathcal{Z}_{\mathcal{F}}} \\ p(S \mid \mathcal{F}) &= \frac{\mathcal{Z}_S - \mathbb{1}_{S\in\mathcal{F}} \times \mathcal{Z}_S}{\mathcal{Z} - \mathcal{Z}_{\mathcal{F}}}. \end{aligned} \tag{3.10}$$

Since this is exactly the expression from Equation 3.9, we validate the Theorem 3.2.1.

The expression is in principle similar to the one employed in Algorithm 1 from the paper presented by Lorenz *et al* [57]. Notice that if $S_{Ib} = \mathcal{F}_{S_{Ib}}$, then correctly $p(\lambda \mid q, \mathcal{F}_{S_{Ia}}, \mathcal{F}_{S_{Ib}}) = 0$.

Unfortunately, this expression cannot be applied directly. As already pointed out, the DP schemes employed for the RNA secondary structure prediction do not consider what is happening outside their specific state. Therefore, they do not compute $\mathcal{Z}_{\mathcal{S}_\mathcal{I}}$ but only $\mathrm{PfS}(q)$ associated to a certain state q. Therefore we need to research a way how to use the values of $\mathrm{PfS}(q)$ to access $\mathcal{Z}_{\mathcal{S}_\mathcal{I}}$ to perform a non-redundant sampling. The second problem is that all these values as well as those of $\mathcal{F}_{\mathcal{S}_\mathcal{I}}$ need to be stored after the computation and must be easily accessible at given moment. Here we will separately address both problems and discuss the solutions.

3.2.2 Computing the probability with forbidden structures

This part discusses how to rely the values from the Equation 3.8 and $\mathrm{PfS}(q)$ that is computed using the DP scheme $(\mathcal{Q}, q_{root}, \rho)$. We assume tree the decomposition tree $\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$ and an incomplete structure S_I, represented by an immature parsing tree $\mathfrak{T}_I(w, \mathcal{Q}, \mathcal{Q}_N, q_{root}, \rho)$ with the following properties:

- Derivations $q_{\lambda_1}, \ldots, q_{\lambda_u}$, attributing free energies $c_{\lambda_1}, \ldots, c_{\lambda_u}$, were applied beforehand to their respective states.
- q is a leaf of $\mathfrak{T}_I$ for which we have a set of derivations

  $$q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_z\}$$

  and according to $\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$ and this state is derived next.
- $\mathfrak{T}_I$ may or may not have other leaves $q_y, 1 \leq y \leq \nu$ for which we have derivations $q \xrightarrow[-c_{\lambda_y}]{\lambda_y} \{q'_{1_y}, \ldots, q'_{\nu_y}\}$ and their derivations follow after that of q.

The partition function of S_I can be separated to the contributions c_λ of different λ and the contributions $\mathrm{PfS}(q)$ of all states to which no λ is attributed.

Theorem 3.2.3 (Incomplete structure partition function): *The partition function $\mathcal{Z}_{S_I}$ of an incomplete secondary structure S_I defined above can be computed by*

$$\mathcal{Z}_{S_I} = \prod_{x=1}^{u} e^{\frac{-c_{\lambda_x}}{k_B T}} \times \mathrm{PfS}(q) \times \prod_{y=1}^{v} \mathrm{PfS}(q_y). \tag{3.11}$$

Proof 3.2.3. We prove the above by the induction on the number of the derivations M necessary to obtain $\mathcal{Z}_{S_I}$ with underived states q.

Base case $(M = 0)$: If no derivation was performed then the only underived state present is the starting state q_{root}. Therefore:

$$\mathcal{Z}_{S_{I0}} = \mathrm{PfS}(q_{root})$$

In this case $\mathcal{Z}_{S_{I0}} = \mathcal{Z}$, since q_{root} is the starting state. We have already shown in Proof 2.6.2 that under the conditions of completeness and unambiguity we have

$$\mathrm{PfS}(q) = \sum_{s \in \mathcal{L}(q)} \left(e^{\frac{-c(s)}{k_B T}} \right)$$

and since completeness implies $\mathcal{L}(q_{root}) = \mathcal{S}_n$, $\mathcal{Z}_{S_{I0}} = \mathcal{Z}_{\mathcal{S}_n} = \mathrm{PfS}(q_{root})$.

Induction $(M = m)$: Suppose S_I was obtained after $m-1$ derivations $q \xrightarrow[\lambda_x]{\lambda_x} \{q'_{1_x}, \ldots, q'_{u_x}\}$ with $1 \leq x \leq m-1$. We suppose by induction that the expression is valid for every S_I on which $m-1$ was performed and that the next incomplete structure S_{Inext}, represented by an immature parse tree $\mathfrak{T}_I(w, \mathcal{Q}, \mathcal{Q}_N{}', q_{root}, \rho)$, is generated by deriving S_I with $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_z\}$. This implies S_{Inext} is accessed after m derivations. From Equation 3.11, we have

$$\mathcal{Z}_{S_{Inext}} = e^{\frac{-c_\lambda}{k_B T}} \times \prod_{x=1}^{u} e^{\frac{-c_{\lambda_x}}{k_B T}} \times \prod_{a=1}^{z} \mathrm{PfS}(q'_a) \times \prod_{y=1}^{v} \mathrm{PfS}(q_y). \tag{3.12}$$

From the Equation 2.21, we can express

$$e^{\frac{-c_\lambda}{k_B T}} \times \left(\prod_{a=1}^{z} \mathrm{PfS}(q'_a) \right) = p(\lambda \mid q) \times \mathrm{PfS}(q).$$

Therefore the Equation 3.12 gives:

$$\mathcal{Z}_{S_{I_{next}}} = p(\lambda \mid q) \times PfS(q) \times \prod_{x=1}^{u} e^{\frac{-c\lambda_x}{k_B T}} \times \prod_{y=1}^{v} PfS(q_y)$$

$$\mathcal{Z}_{S_{I_{next}}} = p(\lambda \mid q) \times \mathcal{Z}_{S_I}$$

which is exactly the same expression given by Equation 3.7, validating the induction step.

Consequently, we can express the Equation 3.8 as

$$p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}}) = \frac{e^{\frac{-c\lambda}{k_B T}} \times \prod_{x=1}^{u} e^{\frac{-c\lambda_x}{k_B T}} \times \prod_{a=1}^{z} PfS(q'_a) \times \prod_{y=1}^{v} PfS(q_y) - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}}}}{\prod_{x=1}^{u} e^{\frac{-c\lambda_x}{k_B T}} \times PfS(q) \times \prod_{y=1}^{v} PfS(q_y) - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}}}.$$

This expression cannot be easily simplified. However, we can reduce it by $\mathcal{Z}_{S_I} / PfS(q)$:

$$p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}}) = \frac{\frac{PfS(q)}{PfS(q)} \times e^{\frac{-c\lambda}{k_B T}} \times \prod_{x=1}^{u} e^{\frac{-c\lambda_x}{k_B T}} \times \prod_{a=1}^{z} PfS(q'_a) \times \prod_{y=1}^{v} PfS(q_y) - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}}}}{\prod_{x=1}^{u} e^{\frac{-c\lambda_x}{k_B T}} \times PfS(q) \times \prod_{y=1}^{v} PfS(q_y) - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}}}$$

$$p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}}) = \frac{\frac{\mathcal{Z}_{S_I}}{PfS(q)} \times e^{\frac{-c\lambda}{k_B T}} \times \prod_{a=1}^{z} PfS(q'_a) - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}}}}{\mathcal{Z}_{\mathcal{S}_\mathcal{I}} - \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}}}$$

$$p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}}) = \frac{e^{\frac{-c\lambda}{k_B T}} \times \prod_{a=1}^{z} PfS(q'_a) - \frac{PfS(q)}{\mathcal{Z}_{S_I}} \times \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}}}}{\mathcal{Z}_{\mathcal{S}_\mathcal{I}} - \frac{PfS(q)}{\mathcal{Z}_{S_I}} \times \mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}}} \tag{3.13}$$

The final equation gives us the value that is similar to the general stochastic expression from Equation 2.21. Since the proportions between the structures outside $\mathcal{F}$ are conserved, the probability $p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}})$ is unbiased. This is the expression that allows by which we compute the probability of choosing λ in a non-redundant manner.

The values of $\mathcal{Z}_{\mathcal{F}_{S_\mathcal{I}}}$ can be accessed from a specific data structure that stores them due to the necessity to know which subset of $\mathcal{F}$ can be generated at given moment. This structure will be detailed later. We still need to compute $\mathcal{Z}_{S_I}$ as well. This can be done recursively, as we know that

$$\mathcal{Z}_{S_{I_0}} = \mathcal{Z} \qquad \text{and} \qquad \mathcal{Z}_{S_{I_{next}}} = \mathcal{Z}_{S_I} \times p(\lambda|q)$$

and $p(\lambda|q)$ without forbidden terms can be computed from Equation 2.21. The exact principle can be,when using the decomposition tree $\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$ with sequence w and the DP scheme $(\mathcal{Q}, q_{root}, \rho)$, summed up in points as follows:

1. Initialize $\mathcal{Z}_{S_I} \rightarrow \mathcal{Z}$ and $q \rightarrow q_{root}$.
2. Compute $p(\lambda \mid q_{root}, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}})$ for every λ such that $q_{root} \xrightarrow[c_\lambda]{\lambda} \{q_1, \ldots, q_u\}$
3. Randomly choose $q_{root} \xrightarrow[c_\lambda]{\lambda} \{q_1, \ldots, q_u\}$ depending on the probabilities.
4. Compute $p(\lambda|S_I)$, then compute $\mathcal{Z}_{S_{I_{next}}}$.
5. Set the value of $\mathcal{Z}_{S_I}$ to $\mathcal{Z}_{S_{I_{next}}}$.
6. Set $q_{root} = q_1$ and store other states in $\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$. Repeat steps 2 - 5 until $S_I = \{S\}$.

Once S is completed, it is added to $\mathcal{F}$. Such approach demands the minimum changes to the original DP scheme, since only updating $\mathcal{Z}_{secstrinc}$ and subtracting the values of $\mathcal{F}_{S_I}$ is necessary. Both requires a specific data-structure which is discussed next.

3.2.3 Adjacent data-structure for non-redundant sampling

Problem 3.2.2:

- ***Input:*** *Decomposition tree* $\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$.
- ***Output:*** *a data-structure* $\mathfrak{d}$ *that:*
  - *Adds nodes that stores for each* $\mathfrak{T}_I(w, \mathcal{Q}, \mathcal{Q}_N, q_{root}, \rho)$ *representing* S_I *a value* $\mathcal{Z}_{forb_{S_I}}$ *if the node does not exist and* S_I *was generated.*
  - *Accesses* $\mathcal{Z}_{forb_{S_I}}$ *if a new structure* $S \in S_I$ *is generated and updates* $\mathcal{Z}_{forb_{S_I}}$ *at the end of given generation cycle by adding* $\mathcal{Z}_S$ *to it.*

The problem implies the adjacent data structure $\mathfrak{d}$ must be based on $\mathcal{T}$. Storing the $\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$ with all nodes would be a memory consuming task. However, each node $\mathfrak{T}_I$ of $\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$ can be reduced only to the derivation that was performed last at the given point (Figure 3.6) or an axiom, dproot. We therefore need to store, along with $\mathcal{Z}_{forb_{S_I}}$, only λ.

Definition 3.2.7 (Decomposition tree reduction): Let $\mathbb{T}$ be the space of all trees $\mathfrak{T}_I(w, \mathcal{Q}, \mathcal{Q}_N, q_{root}, \rho)$. The decomposition tree reduction is a function

$$\mathfrak{R} : \mathbb{T} \to \mathcal{D}$$

that associates to each node $\mathfrak{T}_I$ of $\mathcal{T}$ a λ such that $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ was the last derivation performed in $\mathfrak{T}_I$.

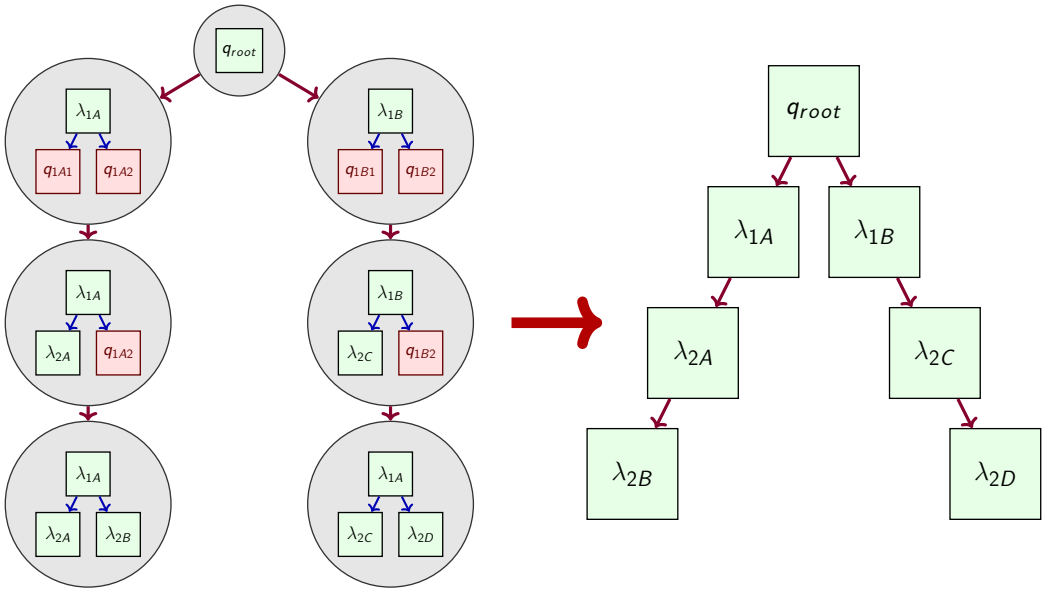

Figure 3.6: **Simplification of a decomposition tree $\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho)$ into a datastructure $\mathfrak{d}(w, \mathcal{Q}, q_{root}, \rho)$.** Each node is reduced to the derivation that was performed last.

Definition 3.2.8 (Non-redundant sampling data structure): The **data-structure for non redundant sampling** is a tree $\mathfrak{d}(w, \mathcal{Q}, q_{root}, \rho)$, based on a decomposition tree

$$\mathcal{T}(w, \mathcal{Q}, q_{root}, \rho) = G(V, E)$$

where

- E is the set of edges from $\mathcal{T}$;
- V is a set of nodes $\mathfrak{R}(\mathfrak{T}_I)$ with $\mathfrak{T}_I \in \mathcal{T}$.

Constructing the entirety of $\mathfrak{d}$ is memory-consuming even with the application of $\mathfrak{R}$. However, $\mathfrak{d}$ can be constructed in parallel with the non-redundant sampling - if a node $\lambda = \mathfrak{R}(\mathfrak{T}_I)$ that would represent $\mathfrak{T}$, resp. S_I is absent, it means no structure $S \in \mathcal{S}_\mathcal{I}$ was sampled yet, therefore $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}} = 0$. If the node λ does exist, the value $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}}$ it stores is used to compute $p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}})$ via the Equation 3.13. The probabilities are then used to sample which λ will be attributed to q, and if the node does not exist yet it is created with $\sum_{y \in \mathcal{F}_{\mathcal{S}_\mathcal{I}}} \mathcal{Z}_y = 0$.

Note that the reduction $\mathfrak{R}$ is accompanied by the loss of the information on the layout of $\mathcal{T}$ and specifically the order of derivation of states q since there may more of them at once. However, the application of recursive function handles this problem as well as using a stack $\mathfrak{W}$ storing the nodes that must be treated and popping them one at a time.

Once we complete a structure and generate S, equivalent to obtaining a mature parse tree $\mathfrak{T}$, we traceback the path from leaf to root while adding $\mathcal{Z}_S$ to the value $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}}$ of each node $\lambda = \mathfrak{R}(\mathfrak{T}_I)$. We have shown that in $\mathcal{T}$ every structure of children node $\mathfrak{T}'_I$ is represented by its parent $\mathfrak{T}'_I$, therefore S is contained by all nodes from its respective mature parse tree $\mathfrak{T}$ to root.

A special mention goes to the linking of the children and parent node. This is important and has the impact on the efficiency of the non-redundant sampling implementation. The obvious method would be to create a hash table where the values of λ identifying each node serve to compute the hashing function. However, the simple linked list seems to be more efficient and is simpler to implement. In that case the nodes are liked in the same order as they are treated in DP scheme $(\mathcal{Q}, q_{root}, \rho)$. In that case we move to the next node of linked list it corresponds to the case in DP scheme, and if it does not, it means that no structure from $\mathcal{S}_\mathcal{I}$ represented by the node labeled λ was treated beforehand, implying $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_\mathcal{I}}} = 0$. If

such a structure is sampled, the node is inserted in linked list at the corresponding position, preserving the order.

Example 3.2.4. The concrete example is given at Figure 3.7 for DP scheme

$$(\mathcal{Q}_{\mathrm{Zuk}}, q^{\mathrm{F}}_{1,9}, \rho_{\mathrm{Zuk}})$$

and sequence w. Here no structure containing a pair $(2, 8)$ was sampled beforehand, meaning node $\lambda^{\mathrm{pair}}_{1,8,2}$ was not created before. If such structure is to be sampled, respectively $\lambda^{\mathrm{pair}}_{1,8,2}$ is selected with $p(\lambda^{\mathrm{pair}}_{1,8,2} \mid q^{\mathrm{F}}_{1,8}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}1}}, \varnothing)$ with $\mathcal{S}_{\mathcal{I}1}$ being the incomplete structures represented by node $\lambda^{\mathrm{pair}}_{1,9}$, the node is inserted at the position copying the order in DP scheme.

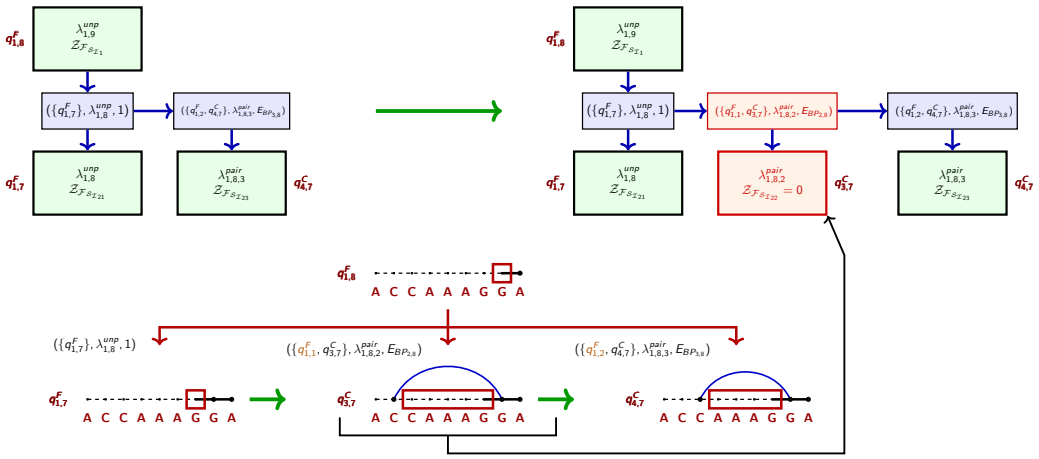

Figure 3.7: **Linking the parent and children nodes in the data-structure for non-redundant sampling in tree** $\mathfrak{d}(w, \mathcal{Q}_{\mathrm{Zuk}}, q^{\mathrm{F}}_{1,9}, \rho_{\mathrm{Zuk}})$. Here no secondary structure with pair $(2, 8)$ was created beforehand, therefore node $\lambda^{\mathrm{pair}}_{1,8,2}$ is not yet created. If such structure is sampled the not gets inserted to its respective position, copying the order within DP scheme $(\mathcal{Q}_{\mathrm{Zuk}}, q^{\mathrm{F}}_{1,9}, \rho_{\mathrm{Zuk}})$.

3.2.4 Non-redundant sampling algorithm

The algorithm of non-redundant sampling algorithm for DP scheme $(\mathcal{Q}, q_{\mathrm{root}}, \rho)$ consists of the data structure $\mathfrak{d}(w, \mathcal{Q}, q_{\mathrm{root}}, \rho)$ storing the informations to compute the probabilities of choosing non-redundantly a certain case for the state q according to the relations introduced in Section 3.2.2. Here we resume the entire algorithm. These probabilities evolve with each sampling iteration implying the necessity of update of informations. The $\mathfrak{d}$ itself is constructed as different derivations are performed. Its pseudocode can be consulted on Algorithm 31.

Assume the number of the unique samples we want to generate is N. The principle of the non-redundant sampling can be resumed as follows:

1. Pre-compute all DP matrices of DP scheme $(\mathcal{Q}, q_{root}, \rho)$.
2. Initialize the adjacent structure for non-redundant sampling

$$\mathfrak{d}(w, \mathcal{Q}, q_{root}, \rho)$$

   by placing q_{root} at its root.
3. Set $\mathtt{Node} = q_{root}$, $\mathcal{Z}_{S_I} = \mathcal{Z}$ and prepare a empty stack $\mathfrak{W}$.
4. For each sample:
   (a) Get the value of $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}_q}}}$ stored in the $\mathtt{Node}$.
   (b) List all derivation $q_n \xrightarrow[c_{\lambda'}]{\lambda'} \{q''_1, \ldots, q''_w\}$ where q_n is:
   - q_{root} if $\mathtt{Node} = q_{root}$;
   - first node from stack $\mathfrak{W}$ if $w = 0$;
   - q'_1, given by $q \xrightarrow[c_\lambda]{\lambda} \{q'_1, \ldots, q'_u\}$ otherwise.

   (c) Initialize a sum $\mathcal{Z}_{temp} = 0$.
   (d) Check if node λ' is present. Extract $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}'}}}$ from it if is, otherwise $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}'}}} = 0$. Compute $p(\lambda_a \mid q_n, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}'}})$ (Equation 3.13).
   (e) Update $\mathcal{Z}_{temp} = \mathcal{Z}_{temp} + p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_q}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}'}})$. If $\mathcal{Z}_{temp} > r$, choose derivation $q \xrightarrow[c_{\lambda'}]{\lambda'} \{q''_1, \ldots, q''_w\}$ and update $\mathcal{Z}_{S_I} = \mathcal{Z}_{S_I} \times p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}'}})$.
   (f) If for $q \xrightarrow[c_{\lambda'}]{\lambda'} \{q''_1, \ldots, q''_w\}$ we have $w > 1$, put all states other than q''_1 in $\mathfrak{W}$.
   (g) Create node λ' linked to $\mathtt{Node}$ with $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}_{q'_1}}}} = 0$ if it does not exist. Set $\mathtt{Node} = \lambda'$.
   (h) Repeat steps (a) - (g) until no derivation is possible.
   (i) Backtrack from q to q_{root} while updating $\mathcal{F}_{\mathcal{S}_{\mathcal{I}_q}}$ for all nodes λ on path. Repeat substeps of step 4 for each sample.

Notice that the main interaction between the DP scheme $(\mathcal{Q}, q_{root}, \rho)$ and the layer assuring the non-redundancy modeled by $\mathfrak{d}(w, \mathcal{Q}, q_{root}, \rho)$ is only when extracting $\mathcal{F}_{\mathcal{S}_\mathcal{I}}$, creating new node λ if absent and backtracking once the complete structure is generated. This makes it very easy to implement it into DP scheme data-structure. However, the non-redundant layer must be adapted for the DP scheme in question.

```
Function Main():
    Data: A DP scheme (𝒬, q_root, ρ), number of structures N
    Result: S_set a set of unique secondary structures
    M_DP ← Precompute_DP() # Pre-computes and fills all DP-matrices
    S_set←∅
    𝔚 ← ∅
    𝔡 ← N(q, 0, ∅) # constructs node N(id, 0, N_parent) in 𝔗(𝒬, q_root, ρ)
    ptr ← q_root
    𝒵_{S_I} ← 𝒵
    while len(S_set) < N do
        S ← unfolded
        while q do
            λ_list ← list_deriv(ptr)
            𝒵_temp = 0
            r ← random(0,1) # random float between 0 and 1
            Node_next ←null # stores the child of ptr
            while λ in λ_list and 𝒵_temp < r do
                Node_next = get_child(λ, ptr, 𝔚) # gets child for case λ of ptr, returns null if
                  node does not exist
                𝒵_temp = 𝒵_temp+get_probability(M_DP, ptr, λ, Node_next)
            𝒵_{S_I} ← 𝒵_{S_I} ×get_probability(M_DP, ptr, λ, ℱ_{𝒮_{ℐ_q}}, Node_next)
            S ←add_object(λ)# adds element associated to λ
            List_deriv ← get_all_states(ptr, λ) # lists q'_1, ··· , q'_u for
              q --λ'/(e^{-c_{λ'}/k_B T})--> {q'_1, ..., q'_u}
            if len(List_deriv) > 1 then
                Push List_deriv[2 :] to 𝔚
            if not Node_next and List_deriv ≠ ∅ then
                Node_next ←N(λ, 0, ptr)
                ptr = Node_next
        Push S to 𝒮_set
        count = count + 1
        ptr ← traceback(ptr)# traceback to root, update nodes
```

3.3 RNANR: Implementation and results

The non-redundant sampling was implemented into the the program using the Saffarian algorithm and sampling the locally optimal secondary structures - `RNANR`, standing for "RNA non-redundantly". The general principle stays the same, the only specificity is the flat structures $\mathfrak{p}$. Besides the non-redundant sampling and the computation of probabilities using Turner Energy Model researched and implemented by us, it also includes the exhaustive enumeration and parametrization developped by A. Saffarian *et al* [80] and later modified by H. Touzet [64]. The extension by non-redundant sampling was coded like the original program in `C`. The most energy-computing functions related to the Turner Energy Model are computed by using VIENNARNA library.

The `RNANR` software is available at the site: `https://project.inria.fr/rnalands/software/rnanr/`.

This section discusses the various aspects of the software, the problems that were encountered, and the results of the different tests that were realized.

3.3.1 Numerical instability issues

The problem related specifically to the non-redundant sampling is the numerical instability. This is not the problem for the regular sampling since the contribution of the sampled structures are not cumulated, but in the case of the non-redundant sampling, these values are summed up and if the sampling performed is of the sufficient size, the Boltzmann factors $\mathcal{Z}_S$ of the secondary structures S become eventually too small for the small precision deviations to become an issue. This leads to selecting wrong or malformed secondary structures once $\mathcal{Z}_S$ low enough are reached. This is more probable to happen for bigger number of unique samples, and is especially noticeable for the sequences having secondary structures important free energy differences - specifically for the the arficially designed riboswitch **SV11** generated by Qβ replicase [9] the numerical instability appears as soon as after 5000 samples.

To counter the described problem, we employ the GNU MPFR library [31], a multiple-precision floating-point computation library based on GMP [38] that allows to store floating point values with arbitrary precision. This includes the values $\mathcal{Z}_S$ that are too small when compared to $\mathcal{Z}$, for which higher precision is used. In the case of `RNANR`, the values of partition functions and Boltzmann

factors are stored this way. Since using arbitrary precision arithmetics slows the software down, the version using standard double-precision floating-point format it is also available.

3.3.2 Sorting flat structures

The Saffarian algorithm is specific by the fact that the size of $\mathcal{P}(i,j)$ for a state $q^U_{i,j}$ or $q^P_{i,j}$ is usually higher than for other DP schemes $(\mathcal{Q}, q_{root}, \rho)$. The execution time of non-redundant sampling using Saffarian algorithm therefore greatly depends on the ordering of the flat structures within the list. Since a given $\mathfrak{p}$ is chosen, for $q^U_{i,j}$ or $q^P_{i,j}$ from $\mathcal{P}(i,j)$ by generating the random number r and then summing up the probabilities $p(\lambda \mid q, \mathcal{F}_{\mathcal{S}_\mathcal{I}}, \mathcal{F}_{\mathcal{S}_{\mathcal{I}_{next}}})$ until the sum surpasses r the program will execute faster with lower number of structures. This can be achieved by ordering $\mathcal{P}(i,j), \forall, 1 \leq i < j \leq n$ by $e^{\frac{-c_\mathfrak{p}}{k_B T}}$ by descending order. The effect of this sorting is shown on Figure 3.7; while in unsorted case we have to traverse seven different $\mathfrak{p}$, in second case, where they are sorted, it is only two. This becomes less effective for later stages of sampling since the low energy structures are picked first, however it considerably speeds up the initial stage of the algorithm.

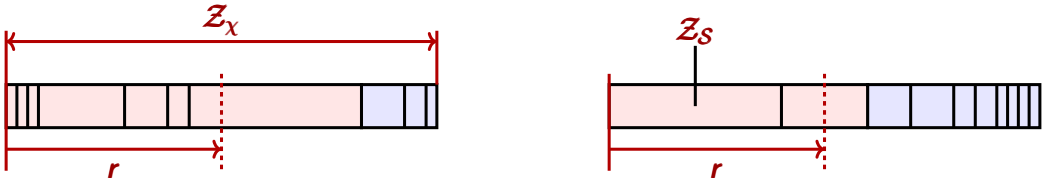

Figure 3.8: **The effect of the sorting of list of available flat structures**. During the random selection of the flat structure $\mathfrak{p}$ from their set $\mathcal{P}(i,j)$, their sorting allows their faster selection. For the unsorted example (left), we need to traverse seven $\mathfrak{p}$ before performing the selection, while we need to check only two in the case of sorted list (right).

The sorting of lists $\mathcal{P}(i,j), \forall, 1 \leq i < j \leq n$ was implemented into `RNANR` alongside the the non-redundant sampling algorithms.

3.3.3 Non-redundant sampling and the speed gain

The main motivation behind researching the non-redundant sampling is the faster generation of unique samples than for usual sampling approaches. We denote by $\mathcal{U}$ a set where any element appears at most once. A set of samples generated by non-redundant sampling has usually bigger coverage than the set of the same size with obtained using classic sampling.

Definition 3.3.1 (Coverage): The coverage Cov($\mathcal{U}$) is is given by:

$$\mathrm{Cov}(\mathcal{U}) = \frac{\sum\limits_{S \in \mathcal{U}} \mathcal{Z}_S}{\mathcal{Z}} \tag{3.14}$$

The coverage is therefore a proportion of partition function $\mathcal{Z}$ that is covered by the Boltzmann factors of $\mathcal{U}$.

The bigger coverage implies the bigger probability that the next sample will be already in $\mathcal{U}$ unless sampled non-redundantly. In the case of non-redundant sampling, this is not a problem, since each sample is guaranteed to be unique. Consequently, non-redundant sampling saves time that would be otherwise spent by generation of duplicate secondary structures.

Definition 3.3.2 (Theoretical speed-up): We call a **theoretical speed-up** a value $\mathbb{E}(|\mathcal{U}|)$ gained for a sampling of a set of unique structures of size $|\mathcal{U}|$ by a non-redundant sampling:

$$\mathbb{E}(|\mathcal{U}|) = 1 + \frac{\sum\limits_{i=0}^{|\mathcal{U}|} \left(1 - \sum\limits_{S \in \mathcal{U}} \frac{e^{\frac{-E_S}{k_B T}}}{\mathcal{Z}}\right)^{-1}}{|\mathcal{U}|} \tag{3.15}$$

The theoretical speed-up is therefore an average number of times a single structure is redundantly sampled.

We computed theoretical speed-up on two sequences:

- 5S ribosomal RNA of Thermoplasma acidophilum (123 nt);
- Telomerase of RNA of Tetrahymena silvana (154 nt).

Results 3.3.1. The speed-up was computed using the Equation 3.15, with the se-

quences being sampled and the necessary values being determined from these samples. The results indicate the exponential increase of the speedup as $|\mathcal{U}|$ increases (Figure 3.9). The length of the sequence has impact on the speed of increase and asymptotic limit - it is slower for longer sequences.

The important point of this example is that redundancy is an important factor when sampling the secondary structures and might noticeably slow down the sampling algorithms, justifying the research of non-redundant sampling methods.

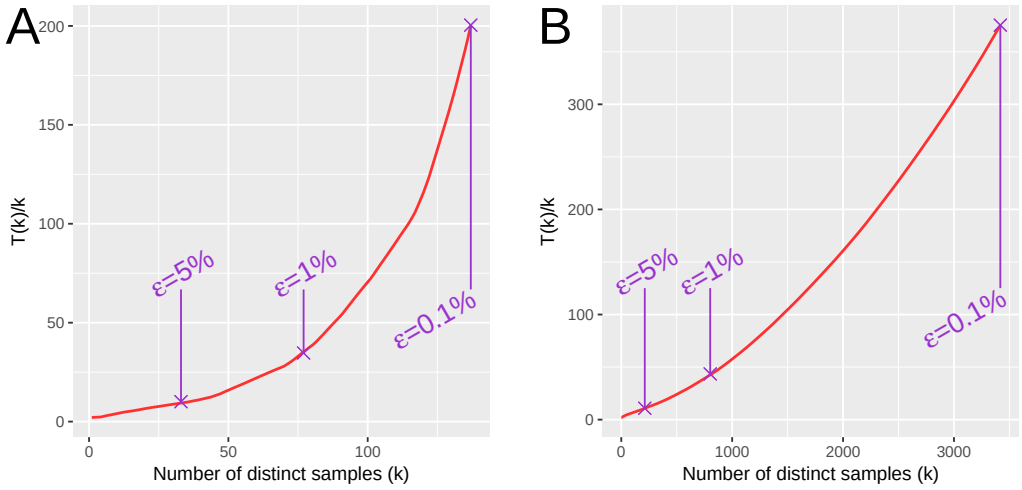

Figure 3.9: **The theoretical speed-up for the sampling of two sequences**. The theoretical speed-up computed for the sequence of 5S ribosomal RNA of Thermoplasma acidophilum 123 nt long **(A)** and for Telomerase of RNA of Tetrahymena silvana (154 nt, **B**). Both sequences were taken from **RNA STRAND** database. Note the exponential increase with the size of $\mathcal{U}$

To compare the practical speed-up or gain of time, we reproduced the experiment from the paper introducing the locally optimal secondary structures sampling algorithm `RNAlocmin` based on scaling the sampling temperature [50] and using `RNAsubopt` from ViennaRNA as its basis. This experience compares `RNAlocmin` to another locally optimal secondary structure sampling algorithm - `RNAlocopt` [56]. All of the above programs use McCaskill's algorithm with the DP scheme $(\mathcal{L}_{MC}, q^{F}_{1,n}, \rho_{MC})$. The experiment consisted of tracing the number of unique secondary structures for certain time intervals. The sequence used is SV11, an artificially designed riboswitch generated by Qβ replicase [9]. We also compared the newer version of `RNAlocmin`. The results are shown on Figure 3.10.

For the classical sampling algorithms - both versions `RNAlocmin` and `RNAlocopt`, we notice the exponential slow-down. This is related to the notion of theoretical speed-up - as the number of unique structures $|\mathcal{U}|$ increases, so does the probability of sampling an already existing secondary structure from $\mathcal{U}$. The effect is specifically noticeable for`RNAlocopt` which does not handle the redundancy in any way, meaning for SV11 after certain time the new structures are basically not sampled. While `RNAlocmin` attenuates the effect of the redundancy to a certain degree by swithcing the sampling temperature, generating the new distribution of secondary structures where $\mathcal{U}$ has smaller proportion, the redundancy is still present.

On the other hand, `RNANR` does not sample the same structure twice and therefore the effect of the redundancy is not present, therefore the graph is linear and the sampling process by `RNANR` is, after short period of pre-computation notably of flat structures, much faster than by software employing classical approaches. Due to the sorting of flat structures it is expected that there should be slow attenuation of the speed once the considerable amount of secondary structures is sampled, however this does not pose a problem for usual sizes $|\mathcal{U}|$.

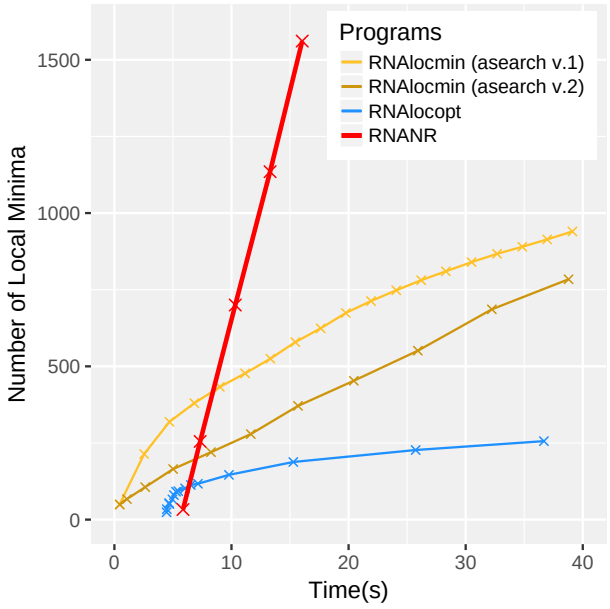

Figure 3.10: **The comparison of the speed gain of `RNANR` with classical sampling approaches.**

The practical result underline those obtained by analysis of theoretical speed-up - the redundancy is an important slow-down factor and its removal might speed up the sampling process considerably.

3.3.4 Landscape quality

While the previous analyses have shown that the non-redundant sampling is an important optimization factor for sampling DP based algorithms, it says nothing about the quality of the RNA landscapes that can be modeled from $\mathcal{U}$. However, due to considerable lack of kinetic data regarding secondary structures of RNAs it is impossible to find a gold standard to which we could compare our results.

The second option is to pose a hypothesis on the properties of correct model of RNA folding landscape and perform the comparison between different software using a score based on these properties.

The good representation of a landscape should feature the most important key landmarks. This means not only potentially functional structures, but also the transitive states that represent the fastest folding pathways between said states. This implies that the best folding landscape should present the fastest folding kinetics, since slower kinetics implies missing some important transitive states. From this we can establish a measure of a quality of a folding landscape.

Definition 3.3.3 (Switching time): Let $P(S_1, t)$, resp. $P(S_2, t)$ be the proportions of sequences w formed a structure S_1, resp. S_2, and $P(S_1, 0) < P(S_2, 0)$. The switching time $\mathbf{t_{\mathcal{SW}}}$ is the lowest time such that $P(S_1, t_{\mathcal{SW}}) \geq P(S_2, t_{\mathcal{SW}})$.

Lower $t_{\mathcal{SW}}$ implies faster kinetics and based on what precedes also the better landscape. Note that however if $\mathcal{Z}_{S_1} < \mathcal{Z}_{S_2}$, then it is possible that $t_{\mathcal{SW}} \to +\infty$ since the equilibrium implies higher $\mathcal{Z}_{S_1}$. It is therefore important to choose correct states that will have their proportions studied.

The ideal candidates for the study would be riboswitches. Here the the first structure S_1 would be MFE (usually first functional structure) while the second would be the metastable structure - the second active state. This ensures $t_{\mathcal{SW}}$ of finite time. The problem is the number of sequences of known riboswitches is limited.

For this reason, we created a set of 250 bistable sequences in a following manner:

- Create a sequence of length 100 nt with uniform distribution of nucleotides.
- Using `RNANR`, sample 1000 unique locally optimal secondary structures, taking parametrization into an account.

- Keep sequences where at least one secondary structure differing at least by 20 base pairs from MFE with them having less that 5kcal.mol^{-1}free energy difference. These structures are saved as well.

From these structures we generate a landscape using the different sampling software and the kinetic analysis described below using different software, which are then compared by their switching times $t_{\mathcal{SW}}$. Besides `RNANR`, `RNAlocopt` and `RNAlocmin`, we also compare `RNASLOpt` that detects stable locally optimal secondary structures [54]. We also compare `RNAsubopt` from VIENNARNA library as a reference.

The pipeline for building an RNA landscape model from the set is based on the state-of-the art approaches detailed in Section 2.8. Since using `barriers` or `BHG` introduces too many transitive states and masks the results of the sampling by ameliorating them, we instead compute the activation energies and associated transition rates by using `findpath` and Metropolis rule respectively. In this case the landscapes are not represented by macrostates (see Section 2.9), but a network of locally optimal structures. The entire process of modeling an RNA folding landscape is as follows:

- By each software, sample a set of secondary structures U_{set} of size 50, 75 and 100 samples.
- For each set remove duplicates and perform a gradient descent for each sample, generating $\mathcal{U}$.
- From each $\mathcal{U}$, create a graph $G(V, E)$ where:
  - $V = \mathcal{U}$;
  - E connects two structures S_1 and S_2 from $\mathcal{U}$ such that the number of different base pairs between them is less than K, where K is the lowest number for which $G(V, E)$ is **connected**. The value of K is shared between all $\mathcal{U}$ made from U_{set}.
- For each edge of G, compute the energy barrier using `findpath`, then associated transition rate by Metropolis rule (Section 2.8.2).
- Identify the two initial structures. Set the proportion of metastable state $P(S_{meta}, 0) = 1$ and perform a kinetic simulation via numerical integration using `treekin` [103]. Determine $t_{\mathcal{SW}}$.

Since the results are biased in the favor of RNANR that was used to establish the initial bistable sequences, we compensate this by giving a tolerance of difference of 10 base pairs for identifying the two initial states by the other software. The lowest $t_{\mathcal{SW}}$ implies the best kinetics.

Results 3.3.2. The results are shown on Figure 3.11. As expected, RNANR most frequently identifies both structures due to being used in their generation (Figure 3.11A). The cases where not even RNANR retrieves both states implies that the state was not in first 50, 75 or 100 samples. However, the low retrieval percentage for other software also results from the redundancy, since after the removal of duplicates the number of samples in $\mathcal{U}$ was usually lower for them than for RNANR.

On the other hand, the switching time values $t_{\mathcal{SW}}$ are consistently lowest for RNANR and with better proportion than successful identification of both metastable structures (Figure 3.11B). This implies that statistically RNANR returns better model of the RNA folding landscape under assumption that the faster kinetics implies better model. In the case that none of the software was successful at retrieval of both metastable structures, the sequence was marked as 'None', which is possible even for RNANR due to the sampling set used for bistable sequence generation having size of 1000. Any software failing to retrieve both structures was also disqualified.

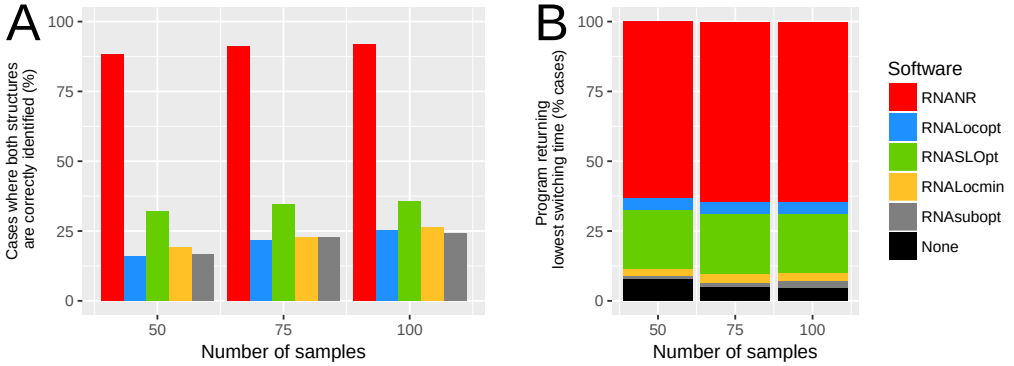

Figure 3.11: **The results of the quality analysis by the different software**. The successfull identification shows RNANR was the most successful with retrieval of two metastable structures **(A)**. This is expected due to RNANR being used to create the bistable sequences. However the lowest switching time is returned with better proportion than the retrieval of the structures **(B)**. This hints at statistically better performance of RNANR when it comes to the analysis of RNA landscape. For **(B)**, the 'None' category indicates the case where no software was able to retrieve both structures.

The concrete example of RNA folding landscape comparison is featured on Fig-

ure 3.12 for he software RNANR and RNASLOpt, which returned the second best results after our software. For the given sequence w (Figure 3.12A) the bistable sequence presents an MFE structure (Figure 3.12D) and a metastable structure (Figure 3.12B). These structures are indicated on the landscapes generated from both software (Figure 3.12C). While in the case of RNANR the two structures are connected to two different halves of the landscape that are clearly weakly connected, in the case of the RNA landscape presented by RNASLOpt they are connected to the same half. This creates an auxiliary half in which w might get stuck due to limited transitions, making difficult to escape to state outside this part and slowing down the entire kinetics. Likely for this reason the switching time between MFE and metastable structure is lower for RNANR.

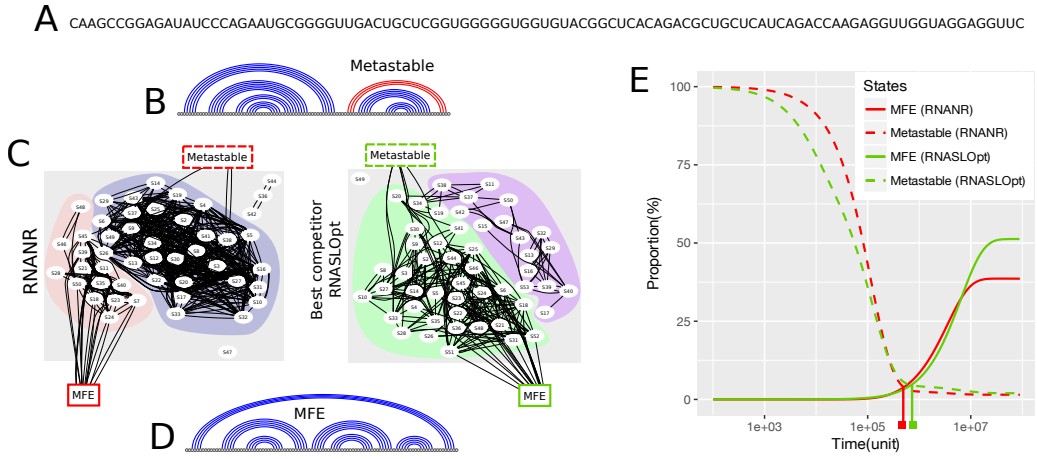

Figure 3.12: **An example of folding landscapes between RNANR and RNASLOpt.** For the RNA sequence **(A)** the metastable **(B)** and MFE **(D)** locally optimal secondary structures are generated, with helix in red being presend only for SRNANR. The landscapes generated by RNANR and RNASLOpt **(C)** are similar and consist of two parts, but for RNANR both structures are connected to different parts while the landscape generated by RNASLOpt presents an auxiliary part. Likely for this reason the switching time is slightly lower for the landscape created by RNANR **(E)**.

The presented results suggest that RNANR and specifically non-redundant principle allow for better modeling of RNA folding landscapes, very probably due to an absence of redundancy and therefore presenting higher number of potentially interesting structures when constituting the landscape.

3.4 Non-redundancy for other DP schemes

While the first example employed Saffarian algorithm, using the introduced formalism we have shown that the non-redundant principle can be employed indeed for any DP scheme $(\mathcal{Q}, q_{root}, \rho)$ as long as that scheme computes PfS. In theory it is enough to create an interface building a data structure $\mathfrak{d}(w, \mathcal{Q}, q_{root}, \rho)$ for specific w that is adapted to said DP scheme and then connect the layer to the data structure of the DP scheme, an easy task due to number of interventions necessary for the DP scheme being rather limited.

To prove this point, we implemented the non-redundant sampling principle into `RNAsubopt` from ViennaRNA library.

3.4.1 RNAsubopt

The `RNAsubopt` employs the DP scheme used in McCaskill algorithm

$$(\mathcal{Q}_{MC}, q^{F}_{1,n}, \rho_{MC}).$$

The interface data-structure $\mathfrak{d}(w, \mathcal{Q}, q_{root}, \rho)$ is built in a similar way than for the Saffarian algorithm, however the options for a given state q are not sorted like the flat structures and in plus the implementation also includes some options:

- Linking parent and children nodes:
  - **Hash table:** the has function uses the values from $q \xrightarrow[c_{\lambda'}]{\lambda'} \{q'_1, \ldots, q'_u\}$;
  - **Linked list:** the same implementation as in `RNANR` (see Section 3.2.3);
- Numerical precision - this is the same as for `RNANR`:
  - **Double floating-point**;
  - **Arbitrary precision arithmetics:** stores the partition functions of forbidden incomplete structures $\mathcal{Z}_{\mathcal{FS}_{\mathcal{I}}}$ in an arbitrary precision arithmetic format using MPFR; much slower but numerically stable when sampling a lot of structures or sequences with structures with important differences of free energy between them;

In this case the arbitrary precision-arithmetics was only included in the layer ensuring the non-redundant sampling, which was shown to be sufficient. That is convenient since it allows to limit the modifications of the DP scheme even more.

The performance profiling revealed that the memory allocation constitutes the major bottleneck of the non-redundant sampling program. For this reason the nodes of $\mathfrak{d}(w, \mathcal{Q}, q_{root}, \rho)$ are not allocated alone, but in blocks that are preallocated, since this allows us to reduce the overhead associated to the single allocation. The size of the blocks is chosen sufficiently large to not allocate new blocks too frequently, but also sufficiently small to be easy to fit to available memory.

To establish a performance of the given implementation options, we extracted 365 sequences from the database RFAM [49], specifically from the families `RF00001`, `RF00005`, `RF00061`, `RF00174` `RF01071` and `RF01731` such that the %GC content of extracted sequences should be the most representative. For each sequence, we sampled 100, 200, 500, 1000, 2000, 5000, 10000, 20000, 35000 and 50000 samples by both the classical sampling and non-redundant sampling employing hash, linked lists and MPFR. We then analyzed the performance of the usual, redundant, and non-redundant versions of the sampling, both done by `RNAsubopt` from ViennaRNA.

The performance tests were executed on a laptop equipped with a CPU Intel® Core™ i7-5600U CPU with 2.60GHz $\times$ 4 on Linux Ubuntu 16.04 LTS and with 16 GB RAM.

Results 3.4.1. Here we compare the obtained results for four representative selected sequences w of diverse lengths n:

- `AE013598.1/1368227-1368452`, from `RF00174`, length 226 nt;
- `M84754.1/1-364`, from `RF00061`, length 364 nt;
- `Z50061.1/3-119` from `RF00001`, length 117 nt;
- `AE015927.1/1705719-1705109` from `RF01071`, length 611 nt;

To shorten the notation, each w will be referred by the name of their family.

At the first time, we consider the total number of unique structures sampled. We started by comparing of the version of the non-redundant sampling, implemented using hash table organization of the non-redundant layer constituted from the data-structure $\mathfrak{d}$, to the redundant sampling method. The results are shown on Figure 3.13. In general we notice the overhead of %50 independently of the length of the sequence, which is not satisfying. As stated, this is mostly due to the memory allocation, since nodes of the hash table are allocated separately. For this reason we developed a second approach that implements the linked lists.

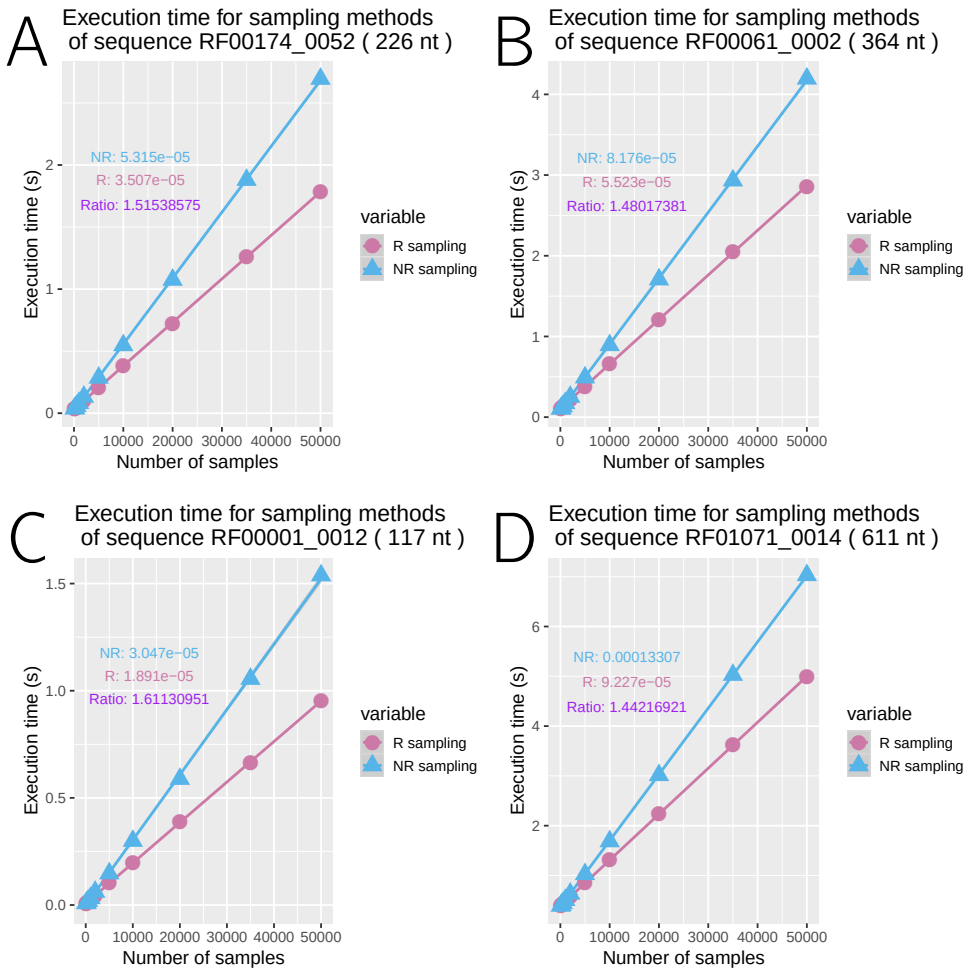

Figure 3.13: **The performance of the non-redundant sampling implementing the hash table organization of the non-redundant layer compared to the usual sampling approach by `RNAsubopt`**. The total number of the samples is considered. The four sequences from respective families `RF00174` **(A)**, `RF00061` **(B)**, `RF00001` **(C)** and `RF01071` **(D)** were sampled for secondary structures by a redundant (blue) and non-redundant (light purple) method. The lines represent the linear approximation of the graphs. The dark purple number indicates the speed ratio between redundant and non-redundant sampling, here indicating the overhead of 50%.

The results for the equivalent analysis, but the non-redundant sampling being implemented by using the linked lists instead, is depicted on Figure 3.14. In this case, the time complexity overhead is reduced to much more manageable 10%-20%. This is caused by two factors. First, the linked list necessitates to allocate less objects than the hash table, reducing the overhead on object allocations. Second, the allocation was optimized by allocating a memory blocks instead of single

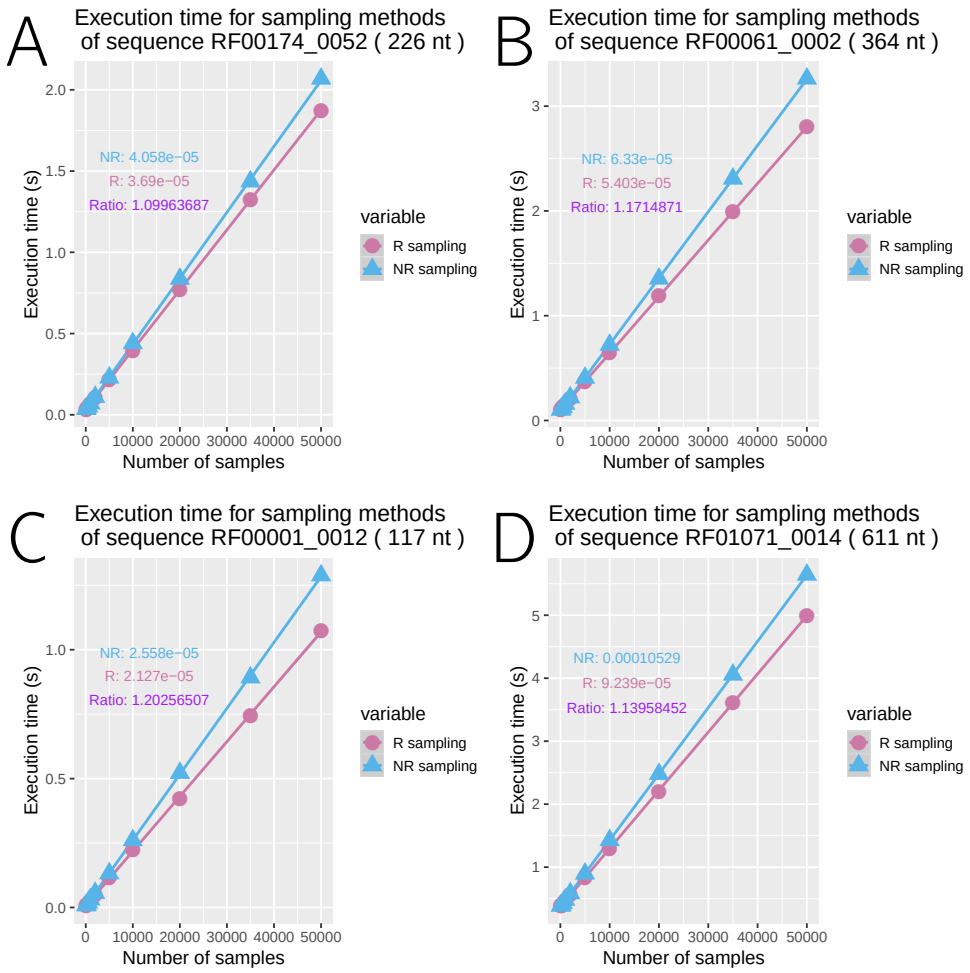

Figure 3.14: **The performance of the non-redundant sampling implementing the linked list organization of the non-redundant layer compared to the usual sampling approach by `RNAsubopt`.** The total number of the samples is considered. The four sequences from respective families `RF00174` **(A)**, `RF00061` **(B)**, `RF00001` **(C)** and `RF01071` **(D)** were sampled for secondary structures by a redundant (blue) and non-redundant (light purple) method. The lines represent the linear approximation of the graphs. The time complexity overhead is in this case reduced to 10%-20%.

objects. As a positive side-effect, the non-redundant sampling algorithm implementing the linked lists consumes less memory, allowing sample even longer sequences.

As a side note, we also provide a reduced example comparing the efficiency of the non-redundant sampling implemented via linked lists where the non-redundant sampling layer stores the values using the MPFR library. The results for the two

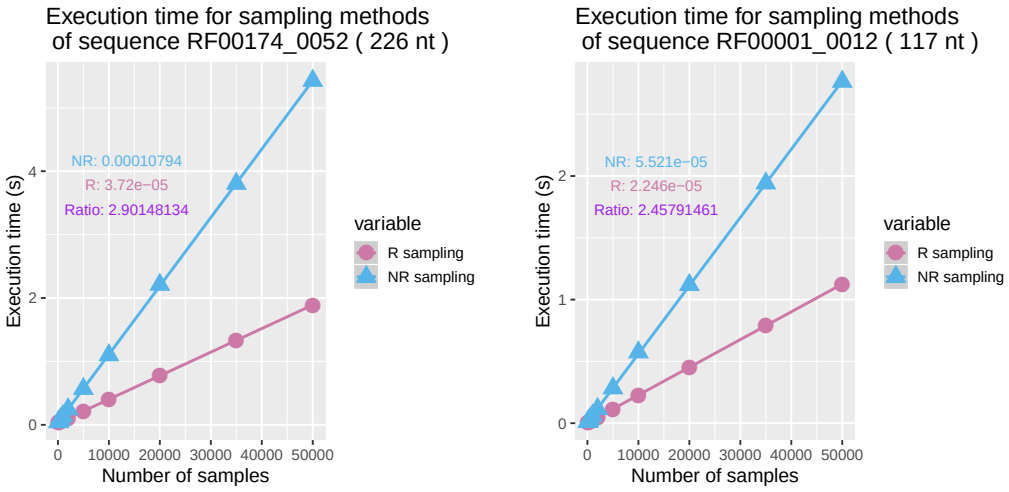

Figure 3.15: **The performance of the non-redundant sampling implementing the linked list organization of the non-redundant layer and storing the values using MPFR compared to the usual sampling approach by `RNAsubopt`**. The total number of the samples is considered. The two sequences from respective families `RF00174` (left) and `RF00061` (right) were sampled for secondary structures by a redundant (blue) and non-redundant (light purple) method. Using the arbitrary precision arithmetic considerably raises the time complexity overhead more than doubling the total time complexity.

sequences from the families `RF00174` and `RF00061` can be consulted on Figure 3.15. Using the arbitrary precision arithmetic is accompanied by the considerable augmentation of the time complexity, in the case of the sequence from the family `RF00174` by rising from 10% to 190%. Therefore, while MPFR may be necessary to use when sampling a more delicate sequences, such as the arficially designed riboswitch **SV11** generated by Qβ replicase [9] which present considerable free energy differences between the secondary structures of interest, it is best to avoid it.

We now concentrate on the performance of both sampling methods in regards to the number of unique secondary structures sampled. For the classical, redundant sampling, we remove the duplicates, and then we compare the number of the secondary structures sampled at the same time. The results are show on Figure 3.16.

For the shorter sequences from families `RF00174` and `RF00001` the non-redundant sampling gives higher number of unique structures for the same execution time than the usual sampling approach. This is because for the shorter sequences it is probable to generate a duplicate quite early on, therefore the usual sampling methods tend to generate lot of them. On the other hand, this is unlikely to happen for long sequences early on during the sampling, since there is much more

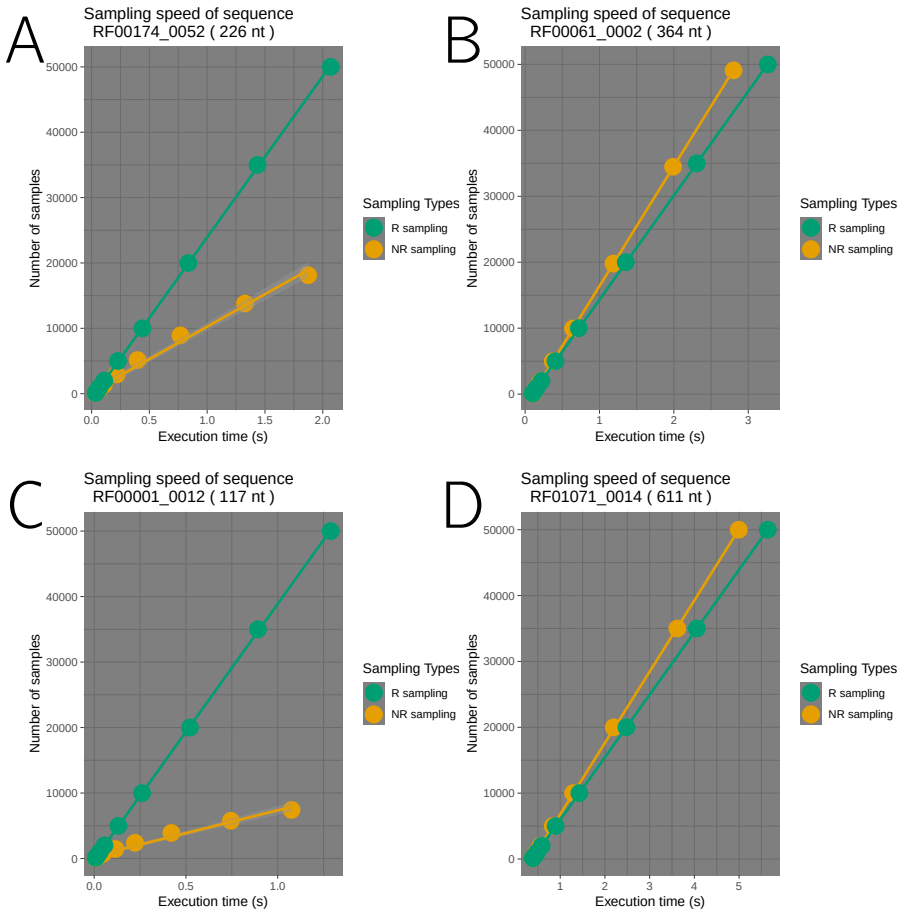

Figure 3.16: **The performance of the non-redundant sampling implementing the linked list organization compared to the usual sampling approach by `RNAsubopt`, counted for the number of unique structures**. Only the unique samples are considered. The four sequences from respective families `RF00174` **(A)**, `RF00061` **(B)**, `RF00001` **(C)** and `RF01071` **(D)** were sampled for secondary structures by a redundant (yellow) and non-redundant (green) method. The lines represent the linear approximation of the graphs. For the shorter sequences A and C, the non-redundant sampling performs better due to high number of the duplicates generated by the usual sampling approach. For the sequences B and D, it performs worse due to the complexity overhead and little probability to generate duplicates for such long sequences at the beginning.

options to generate a secondary structure, and consequently the probability of generating an already sampled structure is low. In this case, it is the complexity overhead that slows the non-redundant sampling estimator when compared to its classic counterpart, consequently performing worse.

We have demonstrated in Section 3.3.3 that the speed gain becomes more impor-

tant the more structures we sample. For the longer sequences the slow down from the redundancy eventually causes the performance of the usual sampling method to fall enough so that of the non-redundant sampling overcomes it, but it may take a considerable number of secondary structures to sample. The extensive, long performance test with high number of samples is one of the analyses that may be interesting to perform in the future.

3.5 Conclusion - Non-redundant sampling

In this section, we presented the concept of non-redundant sampling. The formalism that we introduced in the previous chapter allowed us to demonstrate how its principle can be implemented into all DP schemes that compute the partition function. The non-redundant sampling necessitates a creation of a layer with specific data-structure to function, notably it presents a way to store the values associated to the sets of structures that still can be generated. However, we have demonstrated that the layer has limited number of interactions with the DP scheme itself, making it easy to implement. Moreover, the numerical instability issues can be solved by implementing the arbitrary precision arithmetics uniquely to the data-structure related to the non-redundant sampling, limiting the modification of the DP scheme layer even more.

The non-redundant sampling was tested for the Saffarian algorithm [80], implemented in `RNANR` jointly with H. Touzet and the McCaskill algorithm used by `RNAsubopt` from VIENNARNA library [55]. Both software show the some advantages over the competing software. Notably, `RNANR` samples unique secondary structures more efficiently than its competitors and it seems to return better predictions of RNA folding landscapes. The non-redundant sampling implemented in `RNAsubopt` shows a better performance regarding sampling unique secondary structures for shorter sequences.

The optimized implementation is an important part of the efficiency of the non-redundant sampling. While the conception of the non-redundant sampling layer itself is easy enough, to function efficiently it must be as optimized as possible. This section has shown the impact of the correct implementation of the layer, where the replacement of the hash tables by linked lists reduced the complexity overhead by 30%. It is also important to note that using the arbitrary precision arithmetics considerably slows the algorithm down, so it is best to use it only when really needed.

One caveat is that for sampling a limited set of secondary structures for long sequences by non-redundant sampling is inefficient. This is mainly due to complexity overhead, which may be limited but not completely removed. However, this effect is expected to disappear once a sampling set of sufficient size was sampled, since the effect of the redundancy amplifies with it.

Chapter 4

Estimation of non-redundant sampling sets

The non-redundant sampling samples the unique secondary structures. While this means faster coverage of the associated folding landscape, it is also accompanied by the introduction of a dependency between the samples as well as lacking the information about their frequency. This proves problematic when one wants to estimate a certain quantity of such a sample.

The absence of the frequency means that the usual estimator cannot be employed. However, we have researched, the estimator specific to non-redundant samples. This estimator proves better convergence than the usual approach employed for classic sampling sets, and exhibits smaller variance in most cases.

In this section, after introducing the necessary notions and the classic way of estimating a certain quantity for redundant sampling sets, we explain how to obtain the equivalent for the non-redundant sampling sets. This will be followed by experiments using the implementation of non-redundant sampling for ViennaRNA library on the sequences from the RFAM database [49] cases to estimate a number of representative quantities and the the analysis of the influence of different properties of investigates sequences on the quality of the estimation.

The work described in this whole section was done in collaboration with Christelle Rovetta, a post-doctorant of the AMIBio team.

4.1 The non-redundant estimator

The estimators in general have for objective to give an estimation of a certain quantity, represented by a specific function.

Definition 4.1.1 (Feature function): A **feature function** $F : \mathcal{S}_n \to \mathbb{R}$ that associates to each secondary structure S a certain numerical value.

Example 4.1.1. F returning whether a base pair (i, j) is present in the secondary structure S:

$$F(S) = \begin{cases} 1 & \text{if } (i,j) \in S \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

We can therefore specify the central problematics of this section.

Problem 4.1.1:

- ***Input:*** *A sample set $\mathcal{U}$ drawn from non-redundant distributions with probability of $S \in \mathcal{U}$:*
$$p(S \mid \mathcal{F}) = \frac{\mathcal{Z}_S - \mathbb{1}_{S \in \mathcal{F}} \times \mathcal{Z}_S}{\mathcal{Z} - \mathcal{Z}_{\mathcal{F}}}. \tag{4.2}$$
- ***Output:*** *An unbiased estimation $\widehat{F}(\mathcal{U})$ of F for $\mathcal{U}$ equal to the expectation $\mathbb{E}(F(\mathcal{U}))$.*

There are many usual approaches to estimate the probability of the classic, redundant samples, however, for the non-redundant generation the situation is a bit more complicated.

4.1.1 Empirical mean

Let $\mathcal{X}$ be the classical, redundant sampling set of ordered samples $S_1, \ldots, S_{|\mathcal{X}|}$ and $\mathcal{X}_i = (S_1, \ldots, S_i)$ for $1 \leq i \leq |\mathcal{X}|$. The empirical mean is given by

$$\widehat{F}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{S \in \mathcal{X}} F(S).$$

This estimator uses the frequency of the samples as a way to approximate its probability. This means that the estimator is unbiased, because

$$\mathbb{E}\left(F(\mathcal{X})\right) = \frac{1}{|\mathcal{X}|}\sum_{S\in\mathcal{X}}\mathbb{E}\left(F(S)\right) = \mathbb{E}\left(F(\mathcal{S}_n)\right).$$

As a reminder, $\mathcal{S}_n$ is the set of all secondary structures for w of length n, constituting a variable with $|\mathcal{S}_n|$ states.

In the case of non-redundant sampling set $\mathcal{U}$ the above no longer applies, as the absence of frequencies means the probability cannot be expressed by it. We need a way to express the probability for $\mathcal{U}$ before the non-redundant sampling set specific estimator can be established.

Example 4.1.2. Let have 100 identical RNA secondary structures with a single base pair and 50 identical molecules with two base pairs. The average number N_{BP} of the base pairs in such set is:

$$\mathbb{E}(N_{BP}) = \frac{1}{150}\sum_{i=1}^{100} 2 + \sum_{i=1}^{50} 1 = \frac{4}{3} \tag{4.3}$$

But if we try to use such approach on exhaustive non-redundant sampling set, of size 2, we get:

$$N_{BP} = \frac{1}{2}\times(1+2) = \frac{3}{2} \tag{4.4}$$

which is not equal to the expected value $\mathbb{E}(N_{BP}) = \frac{4}{3}$.

4.1.2 Removing the dependency from the estimator

We remind that the probability $p(S \mid \mathcal{U})$ of non-redundantly choosing a sample S as:

$$p(S \mid \mathcal{U}) = \frac{\mathcal{Z}_S - \mathbb{1}_{S\in\mathcal{U}}\times\mathcal{Z}_S}{\mathcal{Z} - \mathcal{Z}_{\mathcal{U}}}. \tag{4.5}$$

In this section, we suppose $\mathcal{U}\subseteq\mathcal{S}_n$ is ordered, $\widetilde{S}_1,\cdots,\widetilde{S}_{|\mathcal{U}|}\in\mathcal{U}$ being generated in this order. We can define $\mathcal{U}_i = (\widetilde{S}_1,\cdots,\widetilde{S}_i)$ with $1\leq i\leq|\mathcal{U}|$ as a subset of $\mathcal{U}$.

First we establish a k^{th} moment of $\mathcal{U}$.

Definition 4.1.2 (k^{th} moment of set): he k^{th} moment of a set $\mathcal{U}$ is the value:

$$\mu_{\mathcal{U}}^{(k)}(F) = \sum_{\widetilde{S} \in \mathcal{U}} p(\widetilde{S} \mid \mathcal{U}) \times F(\widetilde{S})^k$$

With this, we can define the non-redundant estimator.

Definition 4.1.3 (Non-redundant estimator): We define by **non-redundant sampling set estimator** $\widehat{F}(\mathcal{U})$ the estimator

$$\widetilde{F}(\mathcal{U}) = \frac{1}{|\mathcal{U}|} \left(\sum_{i=0}^{|\mathcal{U}|} F(\widetilde{S}_i) \left(1 - \mu_{\mathcal{U}_{i-1}}^{(0)}(F)\right) + \mu_{\mathcal{U}_{i-1}}^{(1)}(F) \right).$$

Example 4.1.3. We return to the example of 100 identical RNA secondary structures with a single base pair and 50 identical molecules with two base pairs, where $|\mathcal{U}| = 2$. The Equation 4.1.2 becomes

$$\widetilde{N_{BP}}(\mathcal{U}) = \frac{1}{2}(N_{BP}(\widetilde{S_1}) + N_{BP}(\widetilde{S_2}) - N_{BP}(\widetilde{S_1}) \times p(\widetilde{S_1} \mid \mathcal{U})) + p(\widetilde{S_1} \mid \mathcal{U}) \times N_{BP}(\widetilde{S_2})).$$

Since $\mathbb{E}\left(\widetilde{N_{BP}}(\widetilde{S_1})\right) = \mathbb{E}\left(\widetilde{N_{BP}}(\widetilde{S_2})\right) = \mathbb{E}(N_{BP}) = \frac{4}{3}$, this becomes

$$\begin{aligned}
\mathbb{E}\left(\widetilde{N_{BP}}(\mathcal{U})\right) &= \frac{1}{2}\left(\frac{4}{3} + \frac{4}{3} - \frac{4}{3} \times p(\widetilde{S_1} \mid \mathcal{U})) + p(\widetilde{S_1} \mid \mathcal{U}) \times \frac{4}{3}\right) \\
\mathbb{E}\left(\widetilde{N_{BP}}(\mathcal{U})\right) &= \frac{1}{2}\left(\frac{8}{3}\right) \\
\mathbb{E}\left(\widetilde{N_{BP}}(\mathcal{U})\right) &= \frac{4}{3} = \mathbb{E}(N_{BP})
\end{aligned}$$

Theorem 4.1.1 (Unbiasedness): *The non-redundant estimator is unbiased.*

Proof 4.1.1. Consider the conditional expectation $\mathbb{E}_{\mathcal{U}}(F(\widetilde{S}))$ that

$$\mathbb{E}_{\mathcal{U}}(\widehat{F}(\widetilde{S})) = \sum_{\widetilde{S} \in \mathcal{S}_n} p(\widetilde{S} \mid \mathcal{U}) \times F(\widetilde{S}).$$

Replacing the probability by its expression from Equation 4.5 gives:

$$\mathbb{E}_{\mathcal{U}}(F(\widetilde{S})) = \sum_{\widetilde{S}\in\mathcal{S}_n} \frac{\mathcal{Z}_{\widetilde{S}} - \mathbb{1}_{\widetilde{S}\in\mathcal{U}} \times \mathcal{Z}_{\widetilde{S}}}{\mathcal{Z} - \mathcal{Z}_{\mathcal{U}}} \times F(\widetilde{S})$$
$$\mathbb{E}_{\mathcal{U}}(F(\widetilde{S})) = \sum_{\widetilde{S}\in\mathcal{S}_n/\mathcal{U}} \frac{\mathcal{Z}_{\widetilde{S}}}{\mathcal{Z} - \mathcal{Z}_{\mathcal{U}}} \times F(\widetilde{S})$$

since if $\widetilde{S} \in \mathcal{U}$ then $p(\widetilde{S} \mid \mathcal{U}) = 0$. It follows

$$\mathbb{E}_{\mathcal{U}}(F(\widetilde{S})) = \sum_{\widetilde{S}\in\mathcal{S}_n/\mathcal{U}} \frac{p(S)}{1 - \mu_{\mathcal{U}}^{(0)}(F)} \times F(\widetilde{S})$$
$$\mathbb{E}_{\mathcal{U}}(F(\widetilde{S})) = \frac{1}{1 - \mu_{\mathcal{U}}^{(0)}(F)} \left(\sum_{\widetilde{S}\in\mathcal{S}_n/\mathcal{U}} p(S) \times F(\widetilde{S}) \right)$$
$$\mathbb{E}_{\mathcal{U}}(F(\widetilde{S})) = \frac{1}{1 - \mu_{\mathcal{U}}^{(0)}(F)} \left(\sum_{\widetilde{S}\in\mathcal{S}_n} p(\widetilde{S}) \times F(\widetilde{S}) - \sum_{\widetilde{S}\in\mathcal{U}} p(\widetilde{S}) \times F(\widetilde{S}) \right).$$

since $\mathcal{U} \in \mathcal{S}_n$. The term $\sum_{\widetilde{S}\in\mathcal{S}_n} p(\widetilde{S}) \times F(\widetilde{S}) = \mathbb{E}(F(\mathcal{S}_n))$ since it is estimated on $\mathcal{S}_n$. This gives us

$$\mathbb{E}_{\mathcal{U}}(F(\widetilde{S})) = \frac{1}{1 - \mu_{\mathcal{U}}^{(0)}(F)} \left(\mathbb{E}(F(\widetilde{S})) - \mu_{\mathcal{U}}^{(1)}(F) \right)$$
$$\mathbb{E}(F(\widetilde{S})) = \mathbb{E}_{\mathcal{U}}(F(\widetilde{S})) \times (1 - \mu_{\mathcal{U}}^{(0)}(F)) + \mu_{\mathcal{U}}^{(1)}(F).$$

The right-hand side of the expression can be used to establish an estimator $\widetilde{F}(\mathcal{U})$

$$\widetilde{F}(\mathcal{U}) = \frac{1}{|\mathcal{U}|} \left(\sum_{i=0}^{|\mathcal{U}|} F(\widetilde{S}_i) \left(1 - \mu_{\mathcal{U}_{i-1}}^{(0)}(F) \right) + \mu_{\mathcal{U}_{i-1}}^{(1)}(F) \right). \tag{4.6}$$

It remains to prove that the estimator $\widetilde{F}(\mathcal{U})$ is unbiased, meaning we have to show that

$$\mathbb{E}\left(\widetilde{F}(\mathcal{U})\right) = \mathbb{E}\left(F(\mathcal{S}_n)\right).$$

This can be done by induction on the size of $\mathcal{U}$. **Base case ($|\mathcal{U}| = 1$):** In this case $\mathcal{U}_0 = \varnothing$ and $\mu^{(0)}_{\mathcal{U}_0} = \mu^{(1)}_{\mathcal{U}_0} = 0$. From that we have:

$$\mathbb{E}\left(\widetilde{F}(\mathcal{U})\right) = \mathbb{E}\left(F(\widetilde{S}_1)\right) = \mathbb{E}\left(F(\mathcal{S}_n)\right)$$

Induction ($|\mathcal{U}| = m + 1$): Suppose that the property is valid for $|\mathcal{U}| = m$ and we have to verify it for $|\mathcal{U}| = m + 1$. We have

$$\begin{aligned}
\mathbb{E}\left(\widetilde{F}(\mathcal{U})\right) &= \mathbb{E}\left(F(\mathcal{S}_n)\right) \\
\mathbb{E}\left(\widetilde{F}(\mathcal{U})\right) &= \mathbb{E}\left(\frac{1}{m+1}\left(\sum_{i=0}^{m+1} F(\widetilde{S}_i)\left(1 - \mu^{(0)}_{\mathcal{U}_{i-1}}(F)\right) + \mu^{(1)}_{\mathcal{U}_{i-1}}(F)\right)\right) \\
\mathbb{E}\left(\widetilde{F}(\mathcal{U})\right) &= \mathbb{E}\left(\frac{m}{m+1}\widetilde{F}(\mathcal{U}_m)\right) + \frac{1}{m+1}\mathbb{E}\left(F(\widetilde{S}_{m+1})\left(1 - \mu^{(0)}_{\mathcal{U}_m}(F)\right) + \mu^{(1)}_{\mathcal{U}_m}(F)\right) \\
\mathbb{E}\left(\widetilde{F}(\mathcal{U})\right) &= \frac{m}{m+1}\mathbb{E}\left(F(\mathcal{S}_n)\right) + \frac{1}{m+1}\mathbb{E}\left(F(\widetilde{S}_{m+1})\left(1 - \mu^{(0)}_{\mathcal{U}_m}(F)\right) + \mu^{(1)}_{\mathcal{U}_m}(F)\right).
\end{aligned}$$

The only thing left is to prove that $\mathbb{E}\left(F(\widetilde{S}_{m+1})\left(1 - \mu^{(0)}_{\mathcal{U}_m}(F)\right) + \mu^{(1)}_{\mathcal{U}_m}(F)\right) = \mathbb{E}\left(F(\mathcal{S}_n)\right)$. We pose

$$B_{m+1} = F(\widetilde{S}_{m+1})\left(1 - \mu^{(0)}_{\mathcal{U}_m}(F)\right) + \mu^{(1)}_{\mathcal{U}_m}(F).$$

We define $p(\mathcal{U}_m)$ as a probability of having a set of unique structures $\mathcal{U}_m$. We then develop $\mathbb{E}(B_m)$:

$$\begin{aligned}
\mathbb{E}(B_m) &= \sum_{\mathcal{U}_m \| |\mathcal{U}_m| = m} p(\mathcal{U}_m)\mathbb{E}_{\mathcal{U}_m}\left(F(\widetilde{S}_{m+1})(1 - \mu^{(0)}_{\mathcal{U}_m}(F)) + \mu^{(1)}_{\mathcal{U}_m}(F)\right) \\
\mathbb{E}(B_m) &= \sum_{\mathcal{U}_m \| |\mathcal{U}_m| = m} p(\mathcal{U}_m)\mathbb{E}_{\mathcal{U}_m}\left(\mu^{(1)}_{\mathcal{U}_m}(F) + \sum_{\widetilde{S} \in \mathcal{S}_n / \mathcal{U}_m} p(\widetilde{S} \mid \mathcal{U}_m)F(\widetilde{S})(1 - \mu^{(0)}_{\mathcal{U}_m}(F))\right) \\
\mathbb{E}(B_m) &= \sum_{\mathcal{U}_m \| |\mathcal{U}_m| = m} p(\mathcal{U}_m)\left(\mu^{(1)}_{\mathcal{U}_m}(F) + \sum_{\widetilde{S} \in \mathcal{S}_n / \mathcal{U}_m} \frac{p(\widetilde{S})}{1 - \mu^{(0)}_{\mathcal{U}_m}(F)}F(\widetilde{S})(1 - \mu^{(0)}_{\mathcal{U}_m}(F))\right) \\
\mathbb{E}(B_m) &= \sum_{\mathcal{U}_m \| |\mathcal{U}_m| = m} p(\mathcal{U}_m)\left(\mu^{(1)}_{\mathcal{U}_m}(F) + \sum_{\widetilde{S} \in \mathcal{S}_n / \mathcal{U}_m} p(\widetilde{S})F(\widetilde{S})\right).
\end{aligned}$$

Since $\mathcal{U}_m \in \mathcal{S}_n$, we can write

$$\begin{aligned}
\mathbb{E}(B_m) &= \sum_{\mathcal{U}_m \| |\mathcal{U}_m|=m} p(\mathcal{U}_m) \left(\mu^{(1)}_{\mathcal{U}_m}(F) + \sum_{\widetilde{S} \in \mathcal{S}_n} p(\widetilde{S}) F(\widetilde{S}) - \sum_{\widetilde{S} \in \mathcal{U}_m} p(\widetilde{S}) F(\widetilde{S}) \right) \\
\mathbb{E}(B_m) &= \sum_{\mathcal{U}_m \| |\mathcal{U}_m|=m} p(\mathcal{U}_m) \left(\mu^{(1)}_{\mathcal{U}_m}(F) + \mathbb{E}(F(\mathcal{S}_n)) - \mu^{(1)}_{\mathcal{U}_m}(F) \right) \\
\mathbb{E}(B_m) &= \sum_{\mathcal{U}_m \| |\mathcal{U}_m|=m} p(\mathcal{U}_m) \mathbb{E}(F(\mathcal{S}_n)) \\
\mathbb{E}(B_m) &= \mathbb{E}(F(\mathcal{S}_n))
\end{aligned}$$

from which it follows directly that

$$\mathbb{E}\left(\widetilde{F}(\mathcal{U})\right) = \mathbb{E}(F(\mathcal{S}_n)).$$

The proposed estimator however also presents one important disadvantage of it being inconsistent. When the number of the employed sampling set raises indefinitely, it does not converge towards the true value. While the estimator is unbiased, meaning its expected value $F(\mathcal{U})$ gives the true value of $F(\mathcal{S}_n)$, it does not converge towards it for the entire sampling run. While it would be easy to modify the estimator in that direction, for instant by switching to a weighted average as one approaches the full collection, this case is quite unrealistic and will be addressed in future extensions.

When it comes to variance, the numerical analysis does not allow us to conclude decisively on its value when comparing $\widetilde{F}(\mathcal{U})$ and $\widehat{F}(\mathcal{X})$. In practice it also depends on the complexity overhead of the implementation, as the same-size non-redundant set $\mathcal{U}$ cannot be produced in the same time as the classic set $\mathcal{X}$ due to the additional modifications to be made by the non-redundant sampling layer. The experiments presented in the next section compare the performance of both estimators.

4.2 Applications of non-redundant estimator

Here we present a number of applications of non-redundant estimator to specific cases and the comparison of the performance of the classic estimator $\widehat{F}(\mathcal{X})$ applied on sampling set $\mathcal{X}$ and that of the classic, redundant estimator $\widetilde{F}(\mathcal{U})$. For every experiment, unless noted otherwise, we used the dataset extracted from

RFAM database [49], specifically 365 sequences of RFAM families RF 00001, RF 00005, RF 00061, RF 00174, RF 01071 and RF 01731. The sequences were chosen in a manner so that their GC content is as homogeneous as possible. Both the redundant, from this point on referred as classical, and non-redundant sampling is performed by RNAsubopt of ViennaRNA library, the latter of which was implemented by us as discussed in Section 3.4.1.

4.2.1 Estimating base pair probabilities

Definition 4.2.1 (Dotplot matrix): A **dotplot matrix** is a matrix D associated to a sequence w storing the values:

$$D_{i,j} = \sum_{S \in \mathcal{S}_n} \begin{cases} p(S) & \text{if } (i,j) \in S \\ 0 & \text{otherwise} \end{cases}$$

In other words, $D_{i,j}$ is the average probability of the given structure having a base pair (i,j). This matrix can be computed in an exact manner by RNAfold from ViennaRNA library. Therefore, we compute the reference value D using RNAfold and the estimations by using the classic mean average estimator $\widehat{D}(\mathcal{X})$ and the non-redundant sampling specific estimator $\widetilde{\mathcal{U}}$ introduced by us. For both $\mathcal{U}$ and $\mathcal{X}$, we sampled 10^3 structures. We then computed their respective errors:

$$e_R = \frac{\|D - \widehat{D}(\mathcal{X})\|}{n(n-1)} \qquad \text{and} \qquad e_{NR} = \frac{\|D - \widetilde{D}(\mathcal{U})\|}{n(n-1)}$$

where e_R and e_{NR} are respectively the redundant and non-redundant error and $\|a\| = \sqrt{a^2}$. The results were classed according to coverage (reminder: $Cov(\mathcal{U}) = \frac{\sum_{S \in \mathcal{U}} z_S}{z}$; for $\mathcal{X}$ the unique set was generated from them), length n of w and the GC-content.

Results 4.2.1. The results are depicted on Figure 4.1. For all Figures, the line marks the difference between e_R and e_{NR}, and the color the estimator that performs better (blue if $e_R > e_{NR}$, orange otherwise). There does not seem to be an influence of the GC-content, though $\widetilde{D}(\mathcal{U})$ seems to perform better for the sequences where GC-content is biased (Figure 4.1A). This is a little surprising, considering that higher GC-content would imply stronger structures due to higher number of G-C bonds, bigger free energy differences and consequently more redundancy.

On the other hand, there is an influence of the n as well as Cov (Figure 4.1B and C resp.). The estimator $\widetilde{D}(\mathcal{U})$ performs better than $\widehat{D}(\mathcal{X})$ for shorter sequences and higher Cov. This is logical since the complexity overhead is not as noticeable for shorter sequences. Similarly, the effect is more remarkable the bigger percentage of the search space is covered.

We also include the Figure 4.2 that indicates the total number of cases where $\widetilde{\mathcal{U}}$ performs better. While the difference between the errors is in all cases close to 0, the non-redundant estimator $\widetilde{D}(\mathcal{U})$ performs better in around 300 cases out of 365. Coherently the majority of cases where $\widetilde{D}(\mathcal{U})$ performs better is those with high coverage Cov, the intervals of which are marked by the color code. Due to inconsistency, the number of sampled structures must stay relatively small.

The example of dotplot matrix D indicate that in the majority of the cases the estimator specific to non-redundant samples performs better. On the other hand, the difference of performances is smaller for longer sequences, which, despite bing expected and coherent with the observations from Section 3.4.1 where the performance of the non-redundant sampling was worse for long sequences and small $|\mathcal{U}|$, is still a little disappointing since the main reason behind the establishment of the non-redundant sampling is the analysis of the longer sequences.

For two specific sequences w, `M30199.1/68 - 167` from the family `RF00001` and `CP000679.1/1996671 - 1997302` from `RF01071`, which will be referred by their specific families, we also compared the overall performance of both estimators $\widehat{D}(\mathcal{X})$ and $\widetilde{D}(\mathcal{U})$. with the respect to the number of performed samples and the execution time. The maximum number of sampled structures for both w and sampling sets $\mathcal{U}$ and $\mathcal{X}$ was 10^4. For the time performance, we sampled the sampling sets $\mathcal{U}$ and $\mathcal{X}$ for the approximately same execution time. The sampling was executed on a laptop equipped with a CPU Intel® Core™ i7-5600U CPU with 2.60GHz $\times$ 4 on Linux Ubuntu 16.04 LTS and with 16 GB RAM. We then analyzed the evolution of e_R and e_{NR} in function of the number of samples and execution time.

Results 4.2.2. The results for both results are shown on Figure 4.3. For both sequences w we notice the better overall performance of $\widetilde{D}(\mathcal{U})$ than that of $\widehat{D}(\mathcal{X})$ for comparable number of samples and execution time, since $\widetilde{D}(\mathcal{U})$ show lower error for the same execution time/number of samples. The results for the number of samples is expected, since the non-redundant sample $\mathcal{U}$ holds more information and the associated estimator $\widetilde{D}(\mathcal{U})$ shows smaller variance. The better performance of $\widetilde{D}(\mathcal{U})$ demonstrates that the non-redundant sampling can be imple-

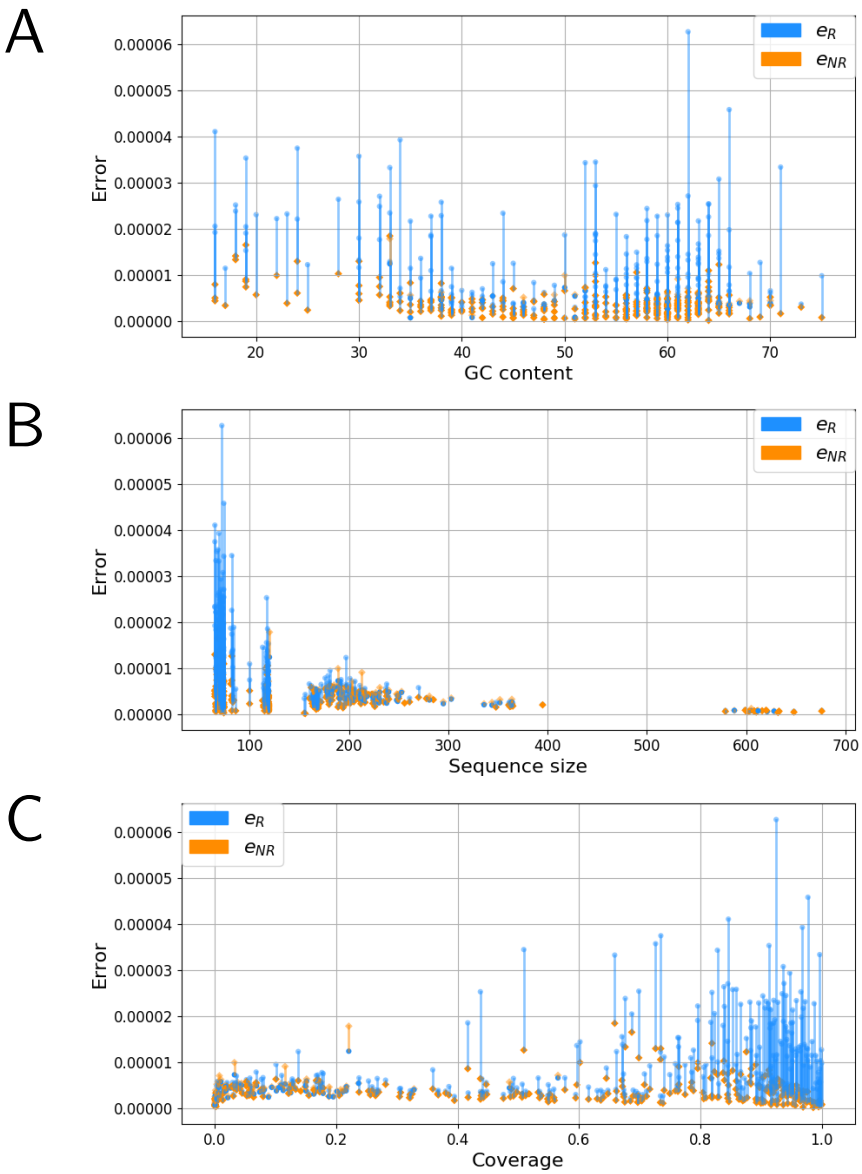

Figure 4.1: **The comparison of efficiency e_R and e_{NR} for the estimation of a dotplot matrix D for 10^3 samples**. The lines represent the difference between e_R and e_{NR} - blue marks $e_R \leq e_{NR}$, while orange the opposite. The results are sorted by GC-content **(A)**, size n of the sequence w **(B)** and the coverage **(C)**. Our estimator performs better notably for shorter sequences and higher coverage.

mented efficiently into the associated DP schemes with low complexity overhead.

Keep in mind however that the above examples are not generalizable and as Fig-

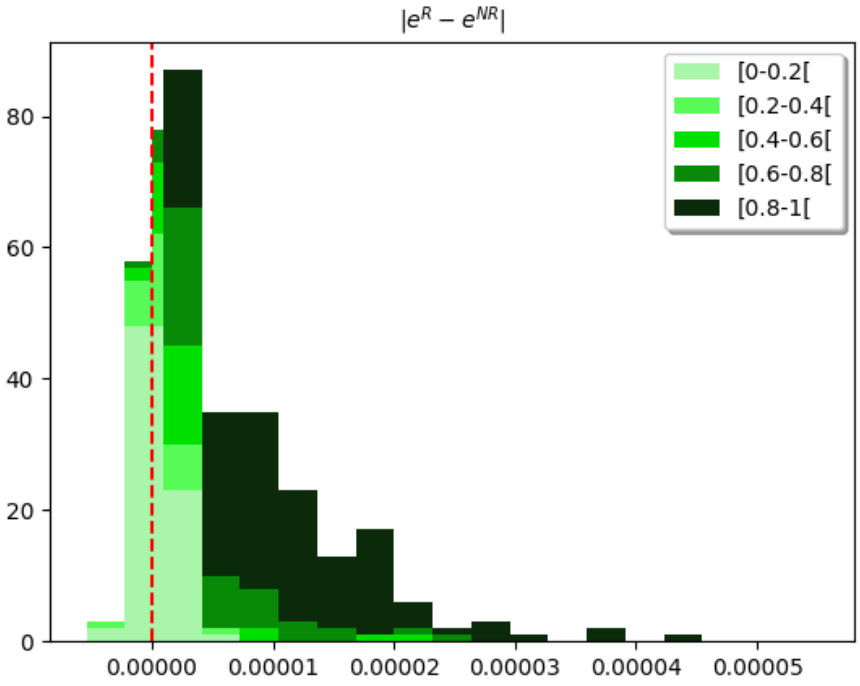

Figure 4.2: **The histogram of $e_R - e_{NR}$ for the estimations of a dotplot matrix** D. The red dotted line denotes $e_R - e_{NR} = 0$. The positive values indicate the cases for which $\widetilde{D}(\mathcal{U})$ performs better than $\widehat{D}(\mathcal{X})$. The coloring indicates the coverage intervals Cov of the sampling set of w for which the estimators were calculated.

ure 4.1 shows, the estimator $\widetilde{D}(\mathcal{U})$ can perform worse than its classical counterpart in some cases.

Also, notice that in the case of the sequence from the family `RF01071`, the error starts to raise after some time, resp. number of samples (Figure 4.3 left bottom). This might point to an inconsistency of our estimator. While our estimator is unbiased, meaning its estimated value is equal to the expected value of given feature function, here D, it does not converge towards the predicted value for an infinitely large sample. This is error we unfortunately did not have the time to rectify, but it is planned to be addressed in future.

4.2.2 Graph distance distribution

Here we suppose the RNA secondary structure S as a graph $G_S = (V, E)$ where:

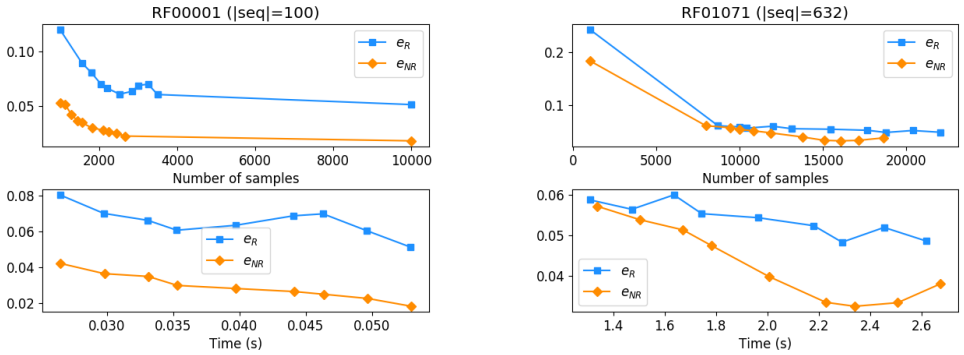

Figure 4.3: **The performance of the estimators $\widehat{D}(\mathcal{X})$ and $\widetilde{D}(\mathcal{U})$ for the two sequences w from the family `RF00001` (left) and `RF01071` (right)**. The graphs show the evolution the errors e_R (blue) and e_{NR} (orange) in function of the number of samples (top) and execution time (bottom).

- V are nucleotides of the sequence;
- E connect two nucleotides connected by the backbone or a base pair.

Definition 4.2.2 (Graph distance): A **minimum graph distance** $d_{i,j}(S)$ is the minimal distances between two nucleotides $w[i]$ and $w[j]$ observed for some $S \in \mathcal{S}_n$, with S represented by $G_S(V, E)$.

Example 4.2.1. For a secondary structure $S = \{(1, 10)\}$ of a sequence w with length $n = 10$, the minimum graph distance is $d(2, 9)(S) = 3$, following the path (2)-(1)-(11)-(9).

By definition, all nucleotides $w[i]$, $w[j]$ neighboring in backbone or in a base pair have a $d_{i,j}(S)$.

Here we first test a distribution of $d_{i,j}(S)$ $\forall S \in \mathcal{S}_n$, denoted $\mathfrak{D}_{i,j}(S)$ generated by the sampling methods themselves. Unlike for the dotplot matrix application, there is not an implemented and efficient way to compute the minimum graph distance distribution for secondary structures in an exact manner. However, we may exploit the consequences of the Chebyshev's inequality.

Theorem 4.2.1 (Chebyshev's inequality): *The **Chebyshev's inequality** is, for a random variable* X *with finite expected value* μ *and a finite non-zero standard deviation* σ *and a real positive number* k*, given by:*

$$p((X - \mu) \geq k\sigma) \leq \frac{1}{k^2}$$

In the other words, only a part of the values X will be more than a certain distance of the mean μ. The consequence of this inequality is the fact that for sufficient number of the samples a significant proportion of them will be distributed around the value μ. We can therefore generate the reference structure by sampling redundantly sufficiently extensive number of the secondary structures and computing $\mathfrak{D}_{i,j}(\mathcal{X})$.

In our case the population is all secondary structures $S \in \mathcal{S}_n$. By exploiting the Chebyshev's inequality we compute the reference distribution $\mathfrak{D}_{i,j}(\mathcal{S}_n)$, then compare it to the results of both classical sampling $d_{i,j}(\mathcal{X})$ and the non-redundant one $\mathfrak{D}_{i,j}(\mathcal{U})$. Due to the extensive computations necessary to generate a reference, this experiment is performed only for the 100 nt long sequence `M30199.1/68-167` from the `RF00001` family. The reference is obtained by sampling $|\mathcal{X}_{ref}| = 10^6$ structures. For testing the non-redundant and classical sampling, we generate $|\mathcal{X}| = |\mathcal{U}| = 10^3$ samples. We look specifically for the distribution of $\mathfrak{D}_{16,82}$ of nucleotides $w[16]$ and $w[82]$.

Results 4.2.3. We plot the histograms of $\mathcal{S}_n$, $\mathcal{X}$ and $\mathcal{U}$ in function of distance classes, one class constituting a specific distance returned for given sampling set. These plots are shown on Figure 4.4. The plot shows the superposition of the distribution of the distance classes for $\mathcal{X}$ and $\mathcal{S}_n$ or reference (Figure 4.4 left) and $\mathcal{U}$ and $\mathcal{S}_n$ (Figure 4.4 right). Both distributions are very close to the reference.

The precise results are shown in Table 4.1. The errors (3^{rd} column) were computed by:

$$e_R = \|\mathfrak{D}_{16,82}(\mathcal{S}_n) - \mathfrak{D}_{16,82}(\mathcal{X})\| \quad \text{and} \quad e_{NR} = \|\mathfrak{D}_{16,82}(\mathcal{S}_n) - \mathfrak{D}_{16,82}(\mathcal{U})\|.$$

The non-redundant sampling distribution $\mathfrak{D}_{16,82}(\mathcal{U})$ is closer to the reference, proving the non-redundant sampling to be more efficient in this case. Also note that $\mathcal{U}$ shows more distance classes that $\mathcal{X}$ and is closer to the expected value - this is not surprising since non-redundant sampling finds more unique structure and consequently has better chance to find more of them.

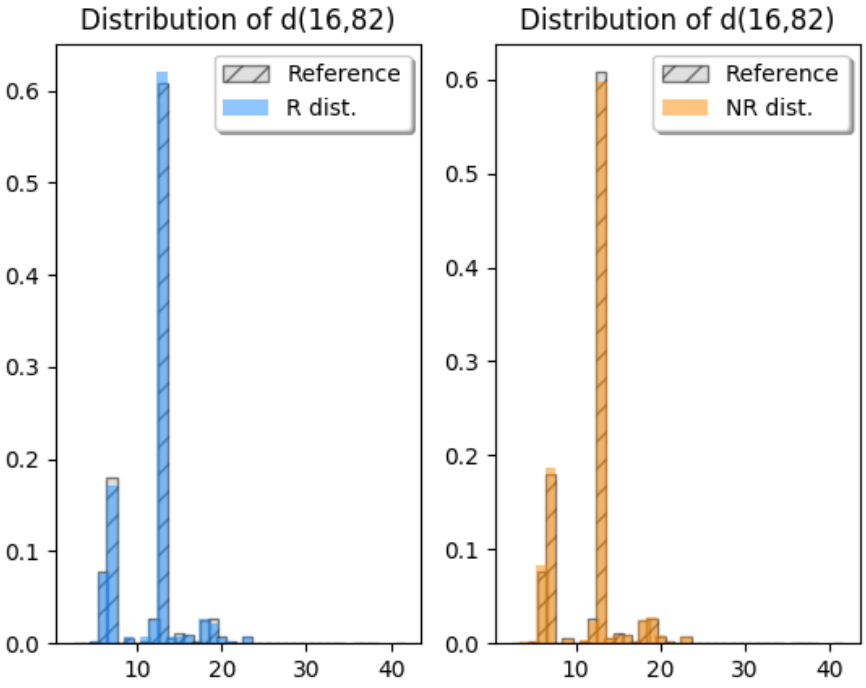

Figure 4.4: **The comparison of distributions of distances $\mathfrak{D}_{i,j}$ for sampling sets $\mathcal{X}$ and $\mathcal{U}$ generated by classical and non-redundant sampling respectively**. The expected distribution is superposed to that for sampling sets $\mathcal{X}$ **(left)** and $\mathcal{U}$ **(right)**. Both distributions are close to the expected one.

To test the estimators of the both sampling approaches, we use a feature function of the weighted graph distance of all possible couples of nucleotides.

Definition 4.2.3 (Weighted graph distance matrix): A weighted **graph distance matrix** $\mathbb{M}$ for a sequence w of length n is a matrix $n \times n$ containing the values:

$$\mathbb{M}_{i,j} = \sum_{S \in \mathcal{S}_n} d_{i,j}(S) \times \frac{\mathcal{Z}_S}{\mathcal{Z}}.$$

Upon estimating $\widehat{\mathbb{M}}(\mathcal{X})$ and $\widetilde{\mathbb{M}}(\mathcal{U})$, we get their respective errors as

$$e_R = \|\mathbb{M} - \widehat{\mathbb{M}}(\mathcal{X})\| \qquad \text{and} \qquad e_{NR} = \|\mathbb{M} - \widetilde{\mathbb{M}}(\mathcal{U})\|.$$

Again, $\mathbb{M}$ is computed from 10^6 samples produced by classical sampling by exploiting the consequences of Chebyshev's inequality and is taken as a reference.

Experiment type	Distance classes	Distance from $\mathfrak{D}_{16,82}(\mathcal{S}_n)$
$\mathfrak{D}_{16,82}(\mathcal{S}_n)$	36	0
$\mathfrak{D}_{16,82}(\mathcal{X})$	18	0.01813
$\mathfrak{D}_{16,82}(\mathcal{U})$	22	0.01496

Table 4.1: **The comparison of distributions of minimum graph distance $d_{16,82}$ for 10^3 samples**. Value classes denote the number of different distances returned for given set of samples.

Both $e_R and e_{NR}$ were computed for up to 10^4 samples.

For the previous experiment, we also determined two values B_1 and B_2:

- B_1 is the number of samples necessary to have $e_R > e_{NR}$ for the first time;
- B_2 is the number of samples necessary to always have $e_R > e_{NR}$ for any subsequent samples.

The above experiment was repeated 50 times, each producing 10^4 samples. For each iteration that gave us specific $\mathcal{X}$ and $\mathcal{U}$, the values B_1 and B_2 were calculated. We then plotted their distribution.

Results 4.2.4. The results for e_R and e_{NR}, respectively B_1 and B_2, are shown on Figures 4.5, resp. Figure 4.6. The comparison of errors e_R and e_{NR} indicates that $\widetilde{M}(\mathcal{U})$ approaches the reference, $\mathbb{M}$, more quickly than $\widehat{M}(\mathcal{U})$. More importantly, the variation of $\widetilde{M}(\mathcal{U})$ is smaller than for $\widehat{M}(\mathcal{U})$.

For the analysis of the values of B_1 and $always best$, we notice that the non-redundant estimator shows, for the sequence `M30199.1/68-167`, the better value for the first time after sampling about 2500 samples at average. The value becomes consistently better after sampling close to 3200 structures. Interestingly, the non-redundant estimator shows at average better values that the classical estimator despite being inconsistent, though this can be explained by not sampling sufficiently big number of the secondary structures.

The experiments on minimum graph distance and graph distance matrices notably confirm that the estimator $\widetilde{F}$ for non-redundant sampling sets $\mathcal{U}$ has smaller variance that $\widehat{F}$ for $\mathcal{X}$, as well as giving the estimation that for the reasonable number of secondary structures are closer to the true value.

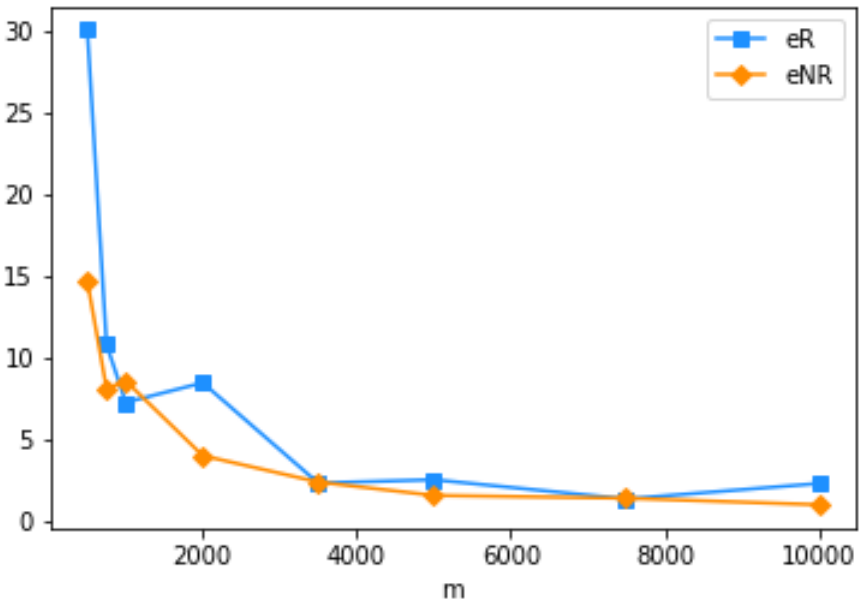

Figure 4.5: **The evolution of e_R and e_{NR} for the number of the samples of $\mathcal{X}$ and $\mathcal{U}$ of sequence `M30199.1/68-167`**. The non-redundant estimator $\widetilde{F}(\mathcal{U})$, marked in orange, shows smaller variations than its classic variant $\widehat{F}(\mathcal{X})$, blue, and in general seems to show being close to the true value.

4.2.3 Shape probabilities

In this experiment, by RNA shape we denote a simplification of dot-bracket format for RNA secondary structures [35]. There are many ways how one can simplify a secondary structure, so in our case we will define it in a following way.

Definition 4.2.4 (Shape): The shape abstraction mapping π [35] is a transformation $f_{SHA}(S)$ of a dot-bracket format of a secondary structure S where:

- Dot `'.'` is removed.
- Helix is replaced by a single pair of square brackets;

Example 4.2.2. The structure `'(((...(...)))'` becomes `'[[]]'`.

The SHAPE is the last example on which the estimators were tested. We define a feature function F_{SHA} such that

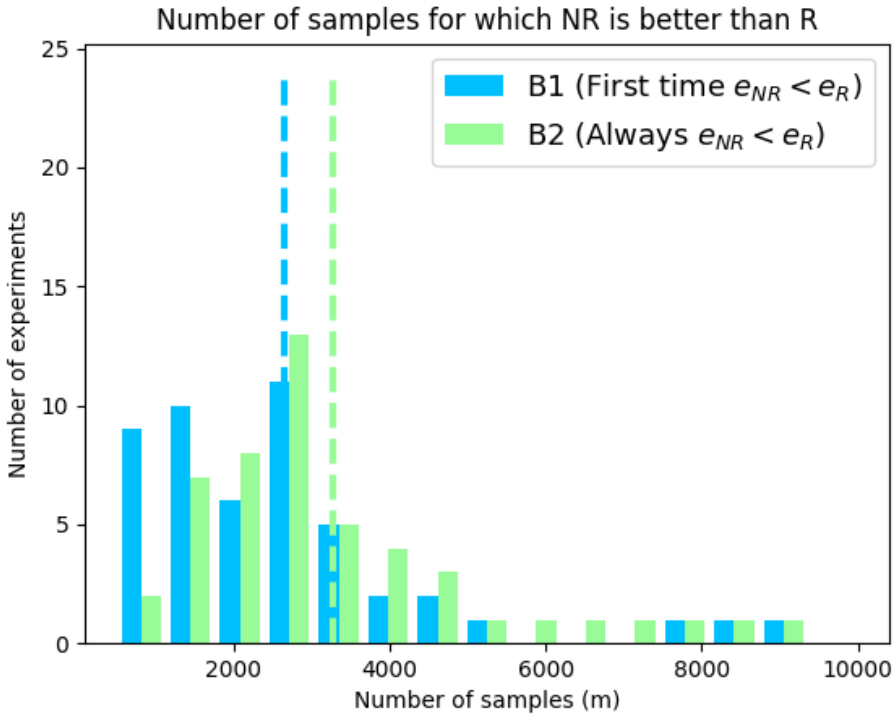

Figure 4.6: **The distribution of the values B_1 and B_2 obtained for the sequence `M30199.1/68-167`**. Blue histogram shows the distribution of number of samples for B_1 ($e_N R < e_R$ for the first time for given samples $\mathcal{X}$ and $\mathcal{U}$), the green B_2 ($e_N R < e_R$ from this point on). The blue and green dashed lines how both the respective average for B_1 and B_2.

$$F_{SHA}(\mathcal{S}_n) = \sum_{S \in \mathcal{S}_n} \begin{cases} 1 & \text{if } f_{SHA}(S) = f_{SHA}(S_{MFE}) \\ 0 & \text{otherwise} \end{cases}$$

where S_{MFE} is the MFE structure. In other words, $F_{SHA}(\mathcal{S}_n)$ is the probability that the shape of $S \in \mathcal{S}_n$ is identical to MFE.

The value of $F_{SHA}(\mathcal{S}_n)$ is estimated by the non-redundant estimator $\widetilde{F}(\mathcal{U})$ for a set of unique samples $\mathcal{U}$ and by its classical equivalent $\widehat{F}(\mathcal{X})$ for redundant sampling set $\mathcal{X}$. This value was evaluated for the sequences w and respective families:

- `M30199.1/68 - 167` from `RF00001`;
- `AY344021.1/1 - 348` from `RF00061`;
- `CP000283.1/2593935 - 2594143` from `RF00174`;

- CP000679.1/1996671 - 1997302 from RF01071.

To simplify the notation, we will refer to these sequences by the name of their respective families.

Like for the previous application, the computation of the exact value of $F_{SHA}(\mathcal{S}_n)$ is difficult, therefore we resort to estimating the value from sufficiently large sampling set of 10^6. For each w, we sampled $|\mathcal{X}| = |\mathcal{U}| = 10^4$ secondary structures. We then followed the evolution of the values of both estimator with regards to the expected value $F_{SHA}(\mathcal{S}_n)$.

Results 4.2.5. The results of this analysis are depicted on Figure 4.7. For all four sequences, the both estimators approach the expected value after 10^4 samples. It is interesting to observe however that in case of the sequence from RF00001 (Figure 4.7A), $\widehat{F}(\mathcal{X})$ approaches the value quicker than the non-redundant estimator $\widetilde{F}(\mathcal{U})$. In all other cases our estimator approaches to the expected value $F_{SHA}(\mathcal{S}_n)$ quicker than $\widehat{F}(\mathcal{X})$.

Another interesting case is the sequence from the family RF00061 (Figure 4.7C). In this case the value $\widetilde{F}(\mathcal{U})$ starts to diverge from the expected value. This can be explained either by the rounding errors, or by the inconsistency of the proposed non-redundant estimator. Since $\widetilde{F}(\mathcal{U})$ is not consistent it may diverge once $\mathcal{U}$ starts to get too large.

4.3 Conclusion - Non-redundant estimator

In this section we discussed the problem of the estimating a specific quantities or properties, represented by a quantifiable feature function, of non-redundant sampling set. The lack of the frequencies forbids us from simply using the weighted average as non-redundant estimator. On the other hand, using the conditional probability leads us to an unbiased estimator that allows us to estimate these quantities from non-redundant sampling.

The advantage of this estimator seems to be its smaller variance, as shown by the practical applications on different quantities such as base pair probability or minimum graph distance. In the former case, our estimator performed worse in about 65 cases, having an important edge over the classical sampling approach and the quantity estimated by the weighted average. The better performance was noted specifically for shorter sequences and bigger coverage, however the

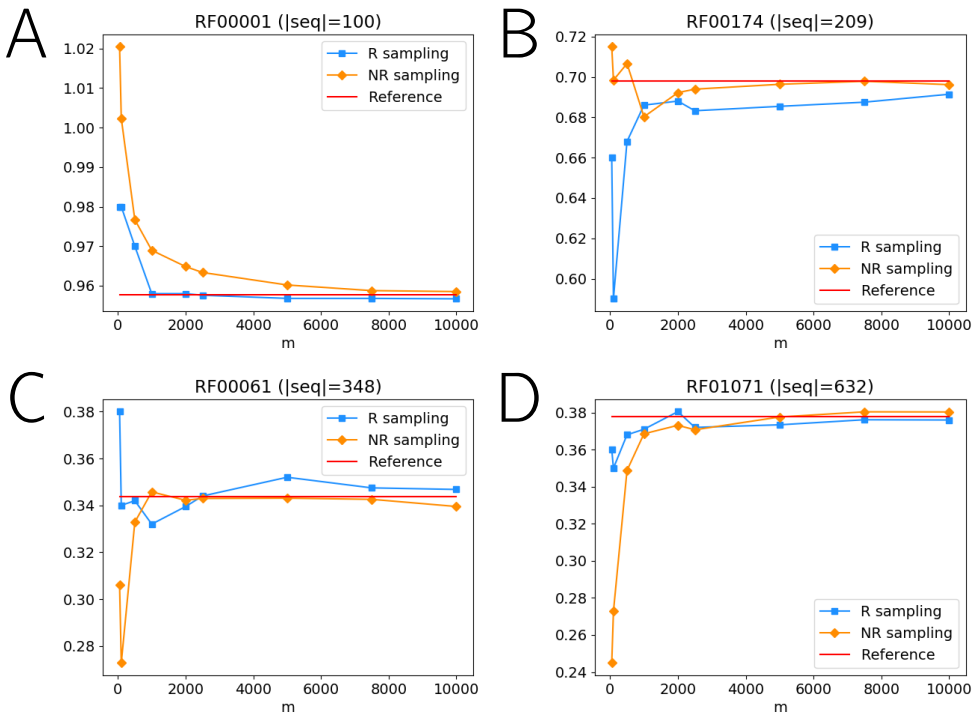

Figure 4.7: **The evolution of the estimators $\widehat{\mathrm{F}}(\mathcal{X})$ and $\widetilde{\mathrm{F}}(\mathcal{U})$ for the number of the samples**. For four different sequences of families `RF00001`**(A)**, `RF00174`**(B)**, `RF00061`**(C)** and `RF01071`**(D)**, the graphs show the evolution of $\widehat{\mathrm{F}}(\mathtt{sampled})$ (blue) and $\widetilde{\mathrm{F}}(\mathcal{U})$ (orange) and their comparison to the expected value (red).

non-redundant estimator still seems to perform better for long sequences than the classic counterpart, even if with a smaller difference. It is also important to notice that our estimator can perform worse in some specific cases, such as small sampling sets for long sequences where the redundancy does not really matter.

The fact that the non-redundant estimator performs better than the classical version not only with the respect to the number of samples, but also the execution time demonstrates that the non-redundant sampling can be implemented efficiently into DP sampling schemes, with little complexity overhead. However, it may need some considerable optimization, like was the use of the linked lists for the implementation used in `RNAsubopt` in ViennaRNA.

The major problem related to the non-redundant sampling set estimator is the fact that the estimator is inconsistent, or it does not converge towards its expected value even when it is unbiased once the number of samples becomes sufficiently big. This problem will be addressed in the future. One of the propositions cur-

rently on the table is to weight the estimator by appropriate values, but the concrete values must be analyzed first.

Chapter 5

Extending towards pseudoknots

Until now, we assumed, as stated in Section 2.1, that the secondary structures contain only the base pairs that cannot cross each another. This condition is necessary to make the decomposition into loops, the basis of all the DP schemes detailed in this work possible since we can assume the independence between what happens inside a given base pair and what happens outside of it. This hypothesis also allows the computation of the free energies of the secondary structures by the Turner Energy Model.

However, such simplification removes an entire class of substructures from the equation - the pseudoknots. We already stated in Definition 2.4.2 that the pseudoknot consists of crossing base pairs. There are many different types of pseudoknots from a simple crossing of two helices to complicated structures [77]. While complex pseudoknots are unlikely to appear in biologically important structures, many simpler pseudoknots have important role in biology [88, 46].

The reason for such restriction is simple - reduce the time complexity. It was demonstrated that accounting for a pseudoknot in its most general definition is an NP-complete problem [1, 58]. Even the most efficient algorithms for the simplest classes of pseudoknots have $\mathcal{O}(n^4)$ time complexity compared to $\mathcal{O}(n^3)$ of the usual DP schemes. However, due to their biological significance, it may be interesting to analyze the simpler pseudoknots and to propose an efficient sampling method on which a non-redundant sampling principle can be eventually extended.

The work in this section is preliminary and still in progress. For this reason, this section is more the presentation of the concept and the main ideas than the re-

sults and the completed software. We begin by the introduction of the algorithm behind the software `pknotsRG` [76] that was also included in `pKiss` [47]. Both software predict the structures with the most optimal pseudoknots for a given interval. The one returns pseudoknots consisting of two perfect helices crossing, while the second returns also kissing hairpins, where one helix crosses once with two others. We will then discuss how to generalize this algorithm while keeping the complexity reasonable, and how to extend it on suboptimal structures. Finally, we will discuss how to extend the non-redundant sampling principle on proposed algorithm and on the future work that needs to be done in this direction.

5.1 State of the Art - `pknotsRG` algorithm

Here we explain the algorithm used in `pknotsRG` [76]. This program predicts the so-called **canonical simple recursive pseudoknots**.

Definition 5.1.1 (Canonical simple recursive pseudoknot): A **canonical simple recursive pseudoknot** consists of exactly two helices $\mathcal{H}_1(i, j, lh)$ and $\mathcal{H}_2(i', j', lh')$ with $i < i'$, of maximum possible length lh and lh' and such that

$$i + lh < i' < i' + lh < j - lh < j < j' - lh.$$

The helices, as defined in Definition 2.4.2, must not contain an unpaired base pair. Such helices are also called **perfect helices**. The simple recursive pseudoknot is then a crossing of two such helices. The maximality of lh, lh' implies that the helix is extended until it is not possible to create a base pair anymore. If the helices would overlap, $\mathcal{H}_1(i, j, lh)$ with lower i takes the priority and is extended to its maximum and $\mathcal{H}_2(i', j', lh)$ takes the remaining base pairs. This also completely defines the internal pseudoknot segments $w[i + lh, i' - 1]$, $w[i' + lh, j - lh]$ and $w[j + 1, j' - lh]$ which we respectively denote $\mathcal{I}_1, \mathcal{I}_2$ and $\mathcal{I}_3$. The example of a canonical simple recursive pseudoknot is given on Figure 5.1.

The algorithm to search for said pseudoknots is designed in a following manner:

- For each substring $w[i, j]$ with $i < j$, precompute and store the maximum length a of a perfect helix $\mathcal{H}(i, j, a)$ in a look-up table.

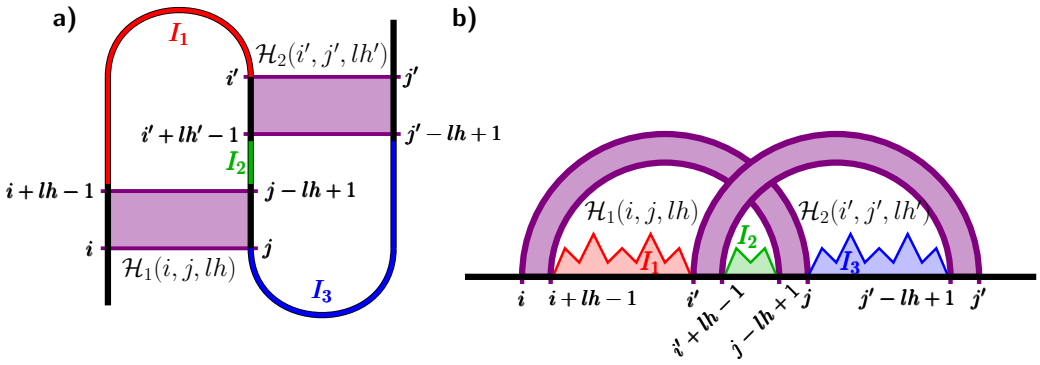

Figure 5.1: **An example of a canonical simple recursive pseudoknots represented radially (a) and linearly (b).** Two perfect helices $\mathcal{H}_1(i, j, lh)$ and $\mathcal{H}_2(i', j', lh')$ cross each other, creating three internal segments $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$.

- For each quadruplet (i, j, i', j'), if $\mathcal{H}_1(i, j, a)$ and $\mathcal{H}_2(i', j', a')$ do not overlap, meaning $i' + a' - 1 < j - a + 1$, create the pseudoknot directly. If they do, set $a' = j - i' - a + 1$ and create the pseudoknot. Compute its energy and for each $w[i, j]$, keep only the pseudoknot with the lowest one.

The complexity of this algorithm is $\mathcal{O}(n^4)$ in time and $\mathcal{O}(n^2)$ in space. The limiting factor is to analyze all quadruplets (i, j, i', j'), giving the complexity of $\mathcal{O}(n^4)$ since once all indexes are known the the pseudoknot is assembled from the values in the look-up table which are computed in $\mathcal{O}(n^2)$ complexity. In the base version, only the most optimal pseudoknot is stored for given substring $w[i, j]$, and the precomputation step stores the maximum length of helix starting with (i, j), giving the memory complexity of $\mathcal{O}(n^2)$.

The free energy of a pseudoknot is computed as the sum of the energies of $\mathcal{H}_1(i, j, a)$ and $\mathcal{H}_2(i', j', a')$ and adding a penalty for creating a pseudoknot. The pseudoknot penalty was not determined empirically and therefore may be inaccurate.

This algorithm is employed by both `pknotsRG` and `pKiss`. Additionally, `pKiss` allows to search for canonical simple recursive kissing hairpins by four different methods, which are not discussed here. The webservice `pKiss` can be consulted at https://bibiserv2.cebitec.uni-bielefeld.de/pkiss.

There are many other algorithms that compute the pseudoknots of varying complexity and specificity. However, many of them are too stringent or too general for our application, while the algorithm discussed here has low complexity and needs only to be generalized to encompass a wider class of pseudoknotted structures.

5.2 Generalizing the pknotsRG algorithm

The algorithm employed in pknotsRG as well as pKiss is useful for discovering the simplest of pseudoknots in an efficient manner. However the number of returned pseudoknots is restricted and due to choosing only the best pseudoknot for given $w[i, j]$, it does not propose suboptimal solutions for said structure. Therefore, we want to propose an extension of the previous algorithm.

Problem 5.2.1:

- ***Input:*** *A sequence w;*
- ***Output:*** *Set of secondary structures S that contain pseudoknots produced by a combing two, not necessarily perfect, helices in a Boltzmann distribution.*

As said, there are many existing algorithms that can generate pseudoknotted structures; the algorithm of pknotsRG is only one example that was chosen as a starting point due to its complexity and implementation advantages. Other examples would be Akutsu algorithm [1], which likewise treats only pseudoknots consisting from two helices, or that established by Elena Rivas and Sean R. Eddy [78], capable to predict even more complicated pseudoknots but at higher time and memory costs. Our algorithm, while not as fast as that used in pknotsRG, is able to predict multiple pseudknotted secondary structures, including those with suboptimal psuedoknots, that pknotsRG is unable to find, justifying its research.

We first introduce the concept of imperfect helices. We then continue how to build a list of helices, which the may be used to assemble the pseudoknots.

5.2.1 Imperfect helices

Our first objective is to establish a table of all helices that can be used to specifically build the pseudoknots we are interested in. In our case these helices may be imperfect.

Definition 5.2.1 (Imperfect helix): An **imperfect helix** $h(i, j, k, l)$ is a set of base pairs constituting the helix and being delimited by (i, j) and (k, l). $w[i, k]$ and $w[l, j]$ may contain unpaired bases.

Note that in this case $j - l = k - i$ is not necessarily true. The imperfect helices $h(i, j, k, l)$ having identical (i, j) and (k, l) may be brought together in common sets.

Definition 5.2.2 (Imperfect branch): An **imperfect branch** $\widehat{\mathcal{H}}(i, j, k, l)$ with $i < k < l < j$ is a set of all helices $h(i, j, k, l)$.

The partition function of $\widehat{\mathcal{H}}(i, j, k, l)$ is the sum of Boltzmann factors of $h(i, j, k, l)$:

$$\mathcal{Z}_{\widehat{\mathcal{H}}(i,j,k,l)} = \sum_{h(i,j,k,l) \in \widehat{\mathcal{H}}(i,j,k,l)} h(i, j, k, l)$$

From $\widehat{\mathcal{H}}(i, j, k, l)$ we can establish the definition of H-type pseudoknots.

Definition 5.2.3 (H-type pseudoknot): A **H-type pseudoknot** $\mathfrak{K}_H(h_1, h_2)$ is a union of two imperfect helices $h_1(i, j, k, l) \in \widehat{\mathcal{H}}_1(i, j, k, l)$ and $h_2(i', j', k', l') \in \widehat{\mathcal{H}}_2(i', j', k', l')$ that cross each other.

By abuse of notation we can write

$$\mathfrak{K}_H(h_1, h_2), \forall h_1 \in \widehat{\mathcal{H}}_1, h_2 \in \widehat{\mathcal{H}}_2$$

as $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$.

This definition is very general and encompasses all possible H-type pseudoknots. The canonical simple recursive pseudoknots are their specific subcategory. However, only searching all imperfect helices $h(i, j, k, l)$ would induce at least $\mathcal{O}(n^4)$ memory complexity to store them all.

To restrict the number of h, we introduce the parameter Δ that implies:

- $|j - l - (k - i) \leq < \Delta$;
- For any two **consecutive** base pairs (i_1, j_1) and (i_2, j_2) such that

  $$\{(i_1, j_1), (i_2, j_2)\} \in h(i, j, k, l) \quad \text{and} \quad i_1 < i_2,$$

  we have $|j_1 - j_2 - (i_2 - i_1)| \leq \Delta$;
- $w[i_u, j_u]$ a substring of $w[i, k]$ or $w[l, j]$ that is completely unpaired may be at most of length Δ.

Only the helices that fulfill the above criteria are considered for pseudoknots $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$.

Example 5.2.1. Here we suppose an imperfect helix $h(i,j,k,l) \in \widehat{\mathcal{H}}(i,j,k,l)$ with $k = i+8$ and $l = j-8$, shown on Figure 5.2. This helix is a subset of six base pairs

$$h(i,j,k,l) = \{(i,j,(i+1,j-1),(i+3,j-5),(i+4,j-6),(i+7,j-7),(k,l)\}.$$

If we consider $\Delta = 3$, then helix in $h(i,j,k,l)$ is considered since there at most 3 consecutive unpaired nucleotides and the maximum skew is between pairs $(i+1, j-1)$ and $(i+3, j-5)$, where

$$j-1-(j-5)-(i+3-(i+1)) = 4-2 = 2.$$

On the other hand, if $\Delta = 2$ then helix in $h(i,j,k,l)$ will be excluded due to an unpaired stretch of length 3 it contains, and $h(i,j,k,l)$ will not be in $\widehat{\mathcal{H}}(i,j,k,l)$.

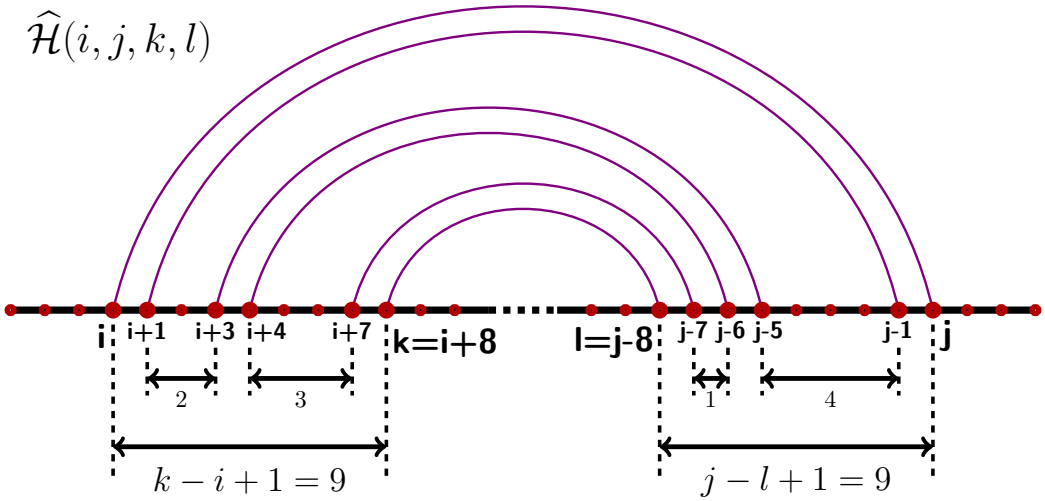

Figure 5.2: **An example of imperfect helix contained by** $h(i,j,i+8,j-8)$. We can verify that this helix fulfills $\Delta = 2$, and therefore in that case is considered as an element of $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$.

Such a restriction is justified since it means that one base pair of $h(i,j,k,l)$ cannot be too distant from the others and prevents its general asymmetry as well. This is important since we are not looking what happens for the unpaired bases between base pairs, meaning too much unpaired base pairs in $h(i,j,k,l)$ would leave too many of them in the final structure.

In the terms of complexity, such a restriction means that instead of having at maximum n^2 possible imperfect helices for each couple (i,j), we have at maximum

$n\Delta$ of them due to the position of l being determined by Δ and that of k. This effectively reduces the time complexity of searching for helices to $\mathcal{O}(n^3\Delta)$. The memory complexity is likewise reduced.

The introduction of the H-type pseudoknots $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$ forces us to redefine the space of all secondary structures $\mathcal{S}_n$, since the condition of no-crossing base pairs is no longer valid. From this point on, we suppose $\mathcal{S}_n$ contains all secondary structures $S \in \mathcal{S}_n$ that may contain elements $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$ with Δ in effect. All other non-conflicting restrictions still apply as defined in Section 2.1.

We are now interested to enumerate all imperfect helices starting with a base pair (i, j). For this reason, we introduces a notation of a **set of imperfect branches** $\boldsymbol{\mathscr{H}^{\Delta}_{i,j}}$ a set of all imperfect branches $\widehat{\mathcal{H}}(i, j, k, l)$ with $i \leq k < l \leq j$ containing only helices $h(i, j, k, l)$ that validate Δ.

To keep the complexity of the algorithm building the helices reasonable, we propose the following dynamic programming scheme. Each base pair (i, j) can be either itself alone a helix $\widehat{\mathcal{H}}(i, j, i, j)$ or it may extend some $\widehat{\mathcal{H}}(i', j', k, l)$ such that $i < i' < j' < i'$. We also need to know the partition function $\mathcal{Z}_{\widehat{\mathcal{H}}}$ of each helix. To simplify a notations, let $\mathcal{Z}_{\widehat{\mathcal{H}}'} = \mathcal{Z}_{\widehat{\mathcal{H}}(i',j',k,l)}$. Therefore:

$$\mathscr{H}^{\Delta}_{i,j} = \bigcup \begin{cases} (i, j, 1) \\ \bigcup\limits_{\substack{|j-j'-(i'-i)|<\Delta \\ |j-l-(k-i)|<\Delta \\ j-j'<\Delta \\ i'-i<\Delta}} \{(k, l, \mathcal{Z}_{\widehat{\mathcal{H}}'} \times e^{\frac{E_{ILG}(i,j,i'j')}{k_B T}}) \mid (k, l, \mathcal{Z}_{\widehat{\mathcal{H}}'}) \in \mathscr{H}^{\Delta}_{i',j'}\} \end{cases}. \quad (5.1)$$

We can then enumerate all possible helices in a recursive manner.

The conditions $j - j' < \Delta$ and $i' - i < \Delta$ come from the restriction of the length of unpaired nucleotides in $h(i, j, k, l) \in \widehat{\mathcal{H}}(i, j, k, l)$. This way we can enumerate all possible imperfect helices starting with (i, j), including suboptimal helices.

Originally, $\mathscr{H}^{\Delta}_{i,j}$ was intended to be stored in form of posets, see Section 5.2.3. Since the performance of such a solution was worse than expected, we use a simple table of $n \times (\Delta + 1)$ dimensions that stores the sum of $\mathcal{Z}_{\widehat{\mathcal{H}}}$ of all helices $\widehat{\mathcal{H}}(i, j, k, j + i - k - \delta)$, where $\delta = j - l - (k - i)$ and therefore $|\delta| < \Delta$. This way we can look directly whether any imperfect helix $h(i, j, k, l)$ exists or not. The imperfect exact helix can be obtained by performing the stochastic backtrack on the value $\mathcal{Z}$ from the position $(i, j, k, j + i - k - \delta)$ in $\mathcal{O}(n \times \Delta)$ time if i, j are known. While this makes the values of $\mathscr{H}^{\Delta}_{i,j}$ not necessary, the relation stays very useful when precomputing all helices for $w[i, j]$.

5.2.2 Assembling the pseudoknots

The generation of all possible pseudoknots $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$ for a given substring $w[i, j]$ is rather simple once we pre-compute the table of imperfect helices. We need to simply adapt the previous DP scheme.

Definition 5.2.4 (Pseudoknot space): We call a **pseudoknot space** $\mathscr{K}(i, j)$, or an H-type pseudoknot space, a space of all pseudoknots **spanning over** substring $w[i, j]$ defined as

$$\mathscr{K}(i, j) = \bigcup_{\substack{k<i'<k'<l \\ j'<l'}} \{\widehat{\mathcal{H}}_1(i, j', k, l) \cup \widehat{\mathcal{H}}_2(i', j, k', l')\}.$$

The internal segments $\mathcal{I}_1, \mathcal{I}_2$ and $\mathcal{I}_3$ are not considered.

The space $\mathscr{K}(i, j)$ can be generated as follows:

- Choose all possible $j' \in [i, j]$.
- For each j', choose all possible helices $\widehat{\mathcal{H}}_1(i, j', k, l)$.
- For each $\widehat{\mathcal{H}}_1(i, j', k, l)$, choose all possible $\widehat{\mathcal{H}}_2(i', j, k', l')$ such that $k < i' < k' < l$ and $l' > j'$.

The compatibility of $\widehat{\mathcal{H}}_1(i, j', k, l)$ and $\widehat{\mathcal{H}}_2(i', j, k', l')$, ie. whether $k < i' < k' < l$ and $l' > j'$, can be checked directly in the table $n \times (2\Delta+1)$. The partition function $\mathcal{Z}_{\mathfrak{K}}$ of a single $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$ is

$$\mathcal{Z}_{\mathfrak{K}} = e^{\frac{-E_{\mathfrak{K}_H}}{k_B T}} \times \mathcal{Z}_{\widehat{\mathcal{H}}_1} \times \mathcal{Z}_{\widehat{\mathcal{H}}_2}$$

where $E_{\mathfrak{K}}$ is the energy penalty for constructing a pseudoknot $\mathfrak{K}_H$. Notice that this equation does not count in the contributions inside the pseudoknot. The partition function $\mathcal{Z}_{\mathscr{K}}$ of $\mathscr{K}(i, j)$ is then

$$\mathcal{Z}_{\mathscr{K}}(i, j) = \sum_{\mathfrak{K}(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2) \in \mathscr{K}(i,j)} \mathcal{Z}_{\mathfrak{K}}.$$

Again, this partition function counts the pseudoknots and pseudoknots only.

The time complexity of this algorithm is $\mathcal{O}(n^6 \times \Delta^2)$, given by the selection of two helices. The memory complexity is $\mathcal{O}(n^3 \times \Delta)$ since for each $\mathscr{K}(i, j)$ we need only to remember its partition function which is then traced back.

5.2.3 Pseudoknots and partially ordered sets

Here we shortly discuss the usability of the partially ordered sets, shortly posets, with respect to the pseudoknots. The idea is to organize the imperfect branches from $\mathscr{H}^{\Delta}_{i,j}$ in posets.

Definition 5.2.5 (Poset of helices): The **poset of helices** is a graph $G(V, E)$ where:

- V are the imperfect branches $\widehat{\mathcal{H}}(i, j, k, l)$;
- E are directed edges from $\widehat{\mathcal{H}}_P(i, j, k, l)$ to $\widehat{\mathcal{H}}_C(i, j, k', l')$ where

$$k \leq k' < l' \leq l$$

  and the k', resp. l' is minimal, resp. maximal.

The minimality, resp. maximality of k', resp l' implies that $\widehat{\mathcal{H}}_C(i, j, k', l')$ should be as similar to $\widehat{\mathcal{H}}_P(i, j, k, l)$ as possible. Other helices that include $\widehat{\mathcal{H}}_P(i, j, k, l)$ can be accessed by transitivity.

Example 5.2.2. The example is provided on Figure 5.3. As shown, it does not have to be a tree, as a single child branch $\widehat{\mathcal{H}}_C(i, j, k', l')$ may have multiple parents $\widehat{\mathcal{H}}_P(i, j, k, l)$.

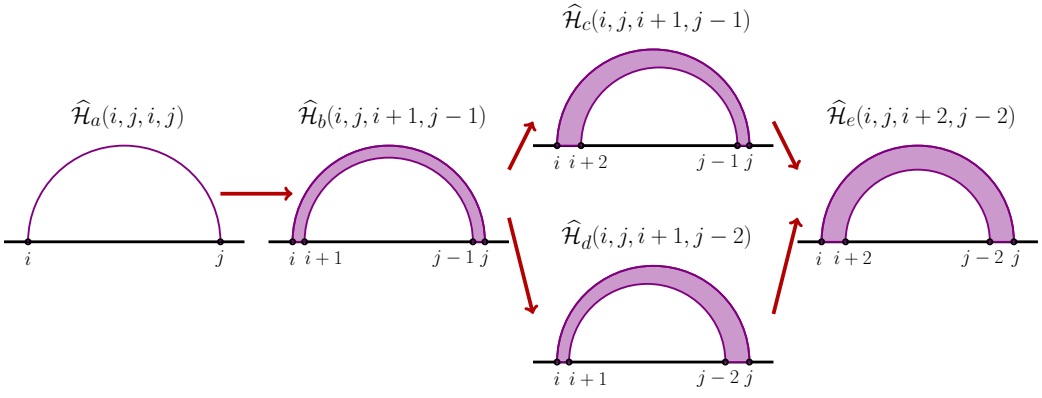

Figure 5.3: **The organization of helices in poset**. The imperfect branches where the helices it contains can be encompassed in another are connected by an edge.

The reasoning behind organizing imperfect helices into posets is the automatic incompatibility of $\widehat{\mathcal{H}}_C(i, j, k', l')$ with some $\widehat{\mathcal{H}}_1(i_1, j_1, k_1, l_1)$ if its parent $\widehat{\mathcal{H}}_P(i, j, k, l)$

is incompatible when constructing a pseudoknot. In other words, any of children of $\widehat{\mathcal{H}}_P(i,j,k,l)$ are not compatible with $\widehat{\mathcal{H}}_1(i_1,j_1,k_1,l_1)$ if $\widehat{\mathcal{H}}_P(i,j,k,l)$ itself is incompatible with it. Since $\widehat{\mathcal{H}}_C(i,j,k',l')$ consumes more nucleotides than $\widehat{\mathcal{H}}_P(i,j,k,l)$, it means that it would produce bigger overlap with $\widehat{\mathcal{H}}_1(i_1,j_1,k_1,l_1)$ when constructing a pseudoknot. The opposite is not true, it is still necessary to check the specific branch $\widehat{\mathcal{H}}_C(i,j,k',l')$ even if all its parents are compatible with $\widehat{\mathcal{H}}_1(i_1,j_1,k_1,l_1)$. However, an incompatible parent $\widehat{\mathcal{H}}_P(i,j,k,l)$ allows to automatically exclude all $\widehat{\mathcal{H}}_C(i,j,k',l')$ from constructing a pseudoknot $\mathfrak{K}_H(\widehat{\mathcal{H}}_1,\widehat{\mathcal{H}}_C)$, potentially reducing a time complexity.

Unfortunately, in practice it is simpler to just allocate a $n \times (\Delta+1)$ matrix, since this allows us to directly search only for compatible helices once $\widehat{\mathcal{H}}_1(i_1,j_1,k_1,l_1)$ is chosen. The preliminary performed on short sequences (order of 30 nt) with $\Delta = 4$ have shown three times longer execution times for poset implementation than for the simple matrix. This is mainly due to memory allocation overhead related to the posets aggravated by the fact that a separate poset is needed for each (i,j). Therefore, we decided to use the simple table implementation, which is also discussed in the rest of this Section.

5.2.4 DP scheme for secondary structures with pseudoknots

Once we designed and algorithm to enumerate all pseudoknots $\mathfrak{K}_H(\widehat{\mathcal{H}}_1,\widehat{\mathcal{H}}_2)$ with restriction Δ and to compute the associated partition function, we must establish a DP scheme that allows for sampling of all secondary structures having such pseudoknots. The easiest way to do this is to extend some existing DP scheme $(\mathcal{Q}, q_{root}, \rho)$ so it includes the specified pseudoknots. This can be achieved by adding a dedicated table that stores the partition function, and whose values are precomputed by the algorithm that enumerates all pseudoknots.

Since we need to sample the secondary structures in a Boltzmann distribution, we choose the DP scheme employed in McCaskill algorithm as our starting point. We consider a new matrix K, that stores the partition functions $\mathcal{Z}^K_{i,j}$ of the all pseudoknots spanning over $w[i,j]$ and includes the internal sequences it delimits. That means that for $\mathfrak{K}_H(\widehat{\mathcal{H}}_1,\widehat{\mathcal{H}}_2)$, with $\widehat{\mathcal{H}}_1(i,j',k,l)$ and $\widehat{\mathcal{H}}_1(i',j,k',l')$, we we need to take into an account what happens for the substrings $w[k,i']$, $w[k',l]$ and $w[j',l']$. The values of these sequences internal to pseudoknot depend on how we want the pseudoknot to be scored.

Since each pseudoknot $\mathfrak{K}_H(h_1,h_2)$ can be seen as a special case of a multibranch

loop, we might want to evaluate the internal segments as a special case of an internal loop, evaluating for example the structures of a segment $w[k, i']$ by $\mathcal{Z}^{M}_{k,i'}$. However, in pseudoknot the entire $w[k, i']$ can be unpaired, while $\mathcal{Z}^{M}_{k,i'}$ does not account for such a case.

To solve this problem, we introduce the new matrix I such that

$$\mathcal{Z}^{I}_{i,j} = \mathcal{Z}^{M}_{i,j} + c \times (j - i + 1)$$

where $c \times (j-i+1)$ is the penalty for leaving $w[i, j]$ unpaired in multibranch loop and $\mathcal{Z}^{M}_{i,j}$ accounts for all other possibilities. The `pknotsRG` and `pKiss` use the default value of 9kcal.mol^{-1}as a penalty for creating a H-type pseudoknots; we used the same value.

Since $\mathfrak{K}_H(h_1, h_2)$ is composed from $h_1(i, j', k, l)$ and $h_1(i', j, k', l')$, we also need to include the matrix that allows us to determine this helices from $\widehat{\mathcal{H}}_1(i, j', k, l)$ and $\widehat{\mathcal{H}}_1(i', j, k', l')$ respectively. Since from $K_{i,j}$ we know we target $\widehat{\mathcal{H}}(i, j, k, l)$ we have $\mathcal{Z}^{H}_{i,j,k,l}$ such that

$$(k, l, \mathcal{Z}^{H}_{i,j,k,l}) \in \mathscr{H}^{\delta}_{i,j}$$

and the recursive formula to construct all helices $\widehat{\mathcal{H}}(i, j, k, l)$ from $\widehat{\mathcal{H}}(i', j', k, l)$ is directly given by Equation 5.1:

$$\mathcal{Z}^{H}_{i,j,k,l} = \sum_{(k,l,\mathcal{Z}^{H}_{i',j',k,l}) \in \mathscr{H}^{\delta}_{i',j'}} e^{\frac{E_{ILG}(i,j,i',j')}{k_B T}} \times \mathcal{Z}^{H}_{i',j',k,l}.$$

We can now define $\mathcal{Z}^{K}_{i,j}$. Since $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$, with $\widehat{\mathcal{H}}_1(i, j', k, l) = \widehat{\mathcal{H}}_1$ and $\widehat{\mathcal{H}}_2(i, j', k, l) = \widehat{\mathcal{H}}_2$, must belong to $\mathscr{K}(i, j)$, using Definition 5.2.2 we may specify $\mathcal{Z}^{K}_{i,j}$ as:

$$\mathcal{Z}^{K}_{i,j} = \sum_{\substack{k<i'<k'<l<j'<l' \\ i<k \\ l'<j}} e^{\frac{-E_{\mathfrak{K}}}{k_B T}} \times \mathcal{Z}^{H}_{i',j,k',l'} \times \mathcal{Z}^{H}_{i',j,k',l'} \times \mathcal{Z}^{I}_{k,i'} \times \mathcal{Z}^{I}_{k',l} \times \mathcal{Z}^{I}_{j',l'}$$

where $E_{\mathfrak{K}}$ is the energy penalty for creating $\mathfrak{K}_H(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$ and $\mathcal{Z}_{\widehat{\mathcal{H}}}$ is the partition function of $\widehat{\mathcal{H}}(i, j, k, l)$.

The last step is to establish a cases where a pseudoknot may appear. In the case of McCaskill algorithm this is rather simple, as we consider that pseudoknot appears in all cases where a regular base pair does. This means that we can simply duplicate all the rules where $\mathcal{Z}^{C}_{i,j}$ appears, then replace it with $\mathcal{Z}^{K}_{i,j}$.

However, it is necessary to proceed with caution in the case of the pseudoknot appearing inside a base pair alone. Since we decided to count its energy as for multibranch loop, we must have to count in the contributions for unpaired stretches $w[i+1, k-1]$ and $w[l+1, j-1]$ as in the case of multibranch loop, as well as the penalties a and $2b$ for opening the multibranch loop itself and for creating two branches within it respectively. If the pseudoknot appears in a multibranch loop with other loops, only the contribution $2b$ for creating two branches is added.

The complete DP scheme, the modified McCaskill algorithm that includes the pseudoknots, is given below:

$$
\begin{aligned}
\mathcal{Z}^{F}_{i,j} &= \mathcal{Z}^{F}_{i,j-1} + \sum_{i \le k \le j-\theta-1} \mathcal{Z}^{F}_{i,k-1} \times \mathcal{Z}^{C}_{k,j} + \sum_{i \le k \le j-\theta-1} \mathcal{Z}^{F}_{i,k-1} \times \mathcal{Z}^{K}_{k,j} \\
\mathcal{Z}^{C}_{i,j} &= e^{\frac{-E_H(i,j)}{k_B T}} + \sum_{\substack{i<k<l<j \\ k \le l-\theta-1}} e^{\frac{-E_{ILG}(i,j,k,l)}{k_B T}} \times \mathcal{Z}^{C}_{k,l} + \\
&\quad + \sum_{i<k<j} e^{\frac{-a-b}{k_B T}} \mathcal{Z}^{M}_{i+1,k-1} \times \mathcal{Z}^{M1}_{k,j-1} + \sum_{\substack{i<k<l<j \\ k \le l-\theta-1}} e^{\frac{-a-2b-c\times(j-l+k-i-2)}{k_B T}} \mathcal{Z}^{K}_{k,l} \\
\mathcal{Z}^{M}_{i,j} &= \sum_{i<k<j} e^{\frac{-c\times(k-1)}{k_B T}} \mathcal{Z}^{M1}_{k,j} + \sum_{i<k<j} \mathcal{Z}^{M}_{i,k-1} \times \mathcal{Z}^{M1}_{k,j} \\
\mathcal{Z}^{M1}_{i,j} &= e^{\frac{-c}{k_B T}} \times \mathcal{Z}^{M1}_{i,j-1} + e^{\frac{-b}{k_B T}} \times \mathcal{Z}^{C}_{i,j} + e^{\frac{-2b}{k_B T}} \times \mathcal{Z}^{K}_{i,j} \\
\mathcal{Z}^{K}_{i,j} &= \sum_{\substack{k<i'<k'<l<j'<l' \\ i<k \\ l'<j}} e^{\frac{-E_{\mathfrak{K}}}{k_B T}} \times \mathcal{Z}^{H}_{i',j,k',l'} \times \mathcal{Z}^{H}_{i',j,k',l'} \times \mathcal{Z}^{I}_{k,i'} \times \mathcal{Z}^{I}_{k',l} \times \mathcal{Z}^{I}_{j',l'} \\
\mathcal{Z}^{I}_{i,j} &= c \times (j-i+1) + \mathcal{Z}^{M}_{i,j} \\
\mathcal{Z}^{H}_{i,j,k,l} &= \sum_{(k,l,\mathcal{Z}^{H}_{i',j',k,l}) \in \mathscr{H}^{\delta}_{i',j'}} e^{\frac{E_{ILG}(i,j,i',j')}{k_B T}} \times \mathcal{Z}^{H}_{i',j',k,l}
\end{aligned}
\tag{5.2}
$$

This DP scheme is depicted on Figure 5.4. As seen, the appearance of the pseudoknot $K_{i,j}$ copies that of a base pair $C_{i,j}$. To treat pseudoknots in an unambiguous manner we need three supplementary matrices, bringing their total number to 7.

The complexity of this algorithm is largely determined by the complexity of of pseudoknot search, since its complexity is $\mathcal{O}(n^6 \times \Delta^2)$, when compared to the complexity $\mathcal{O}(n^3)$ of the rest of the algorithm if the length of internal loop unpaired stretches is limited. Likewise, the memory complexity is $\mathcal{O}(n^3 \times \Delta)$ due to storage of partition functions for helices.

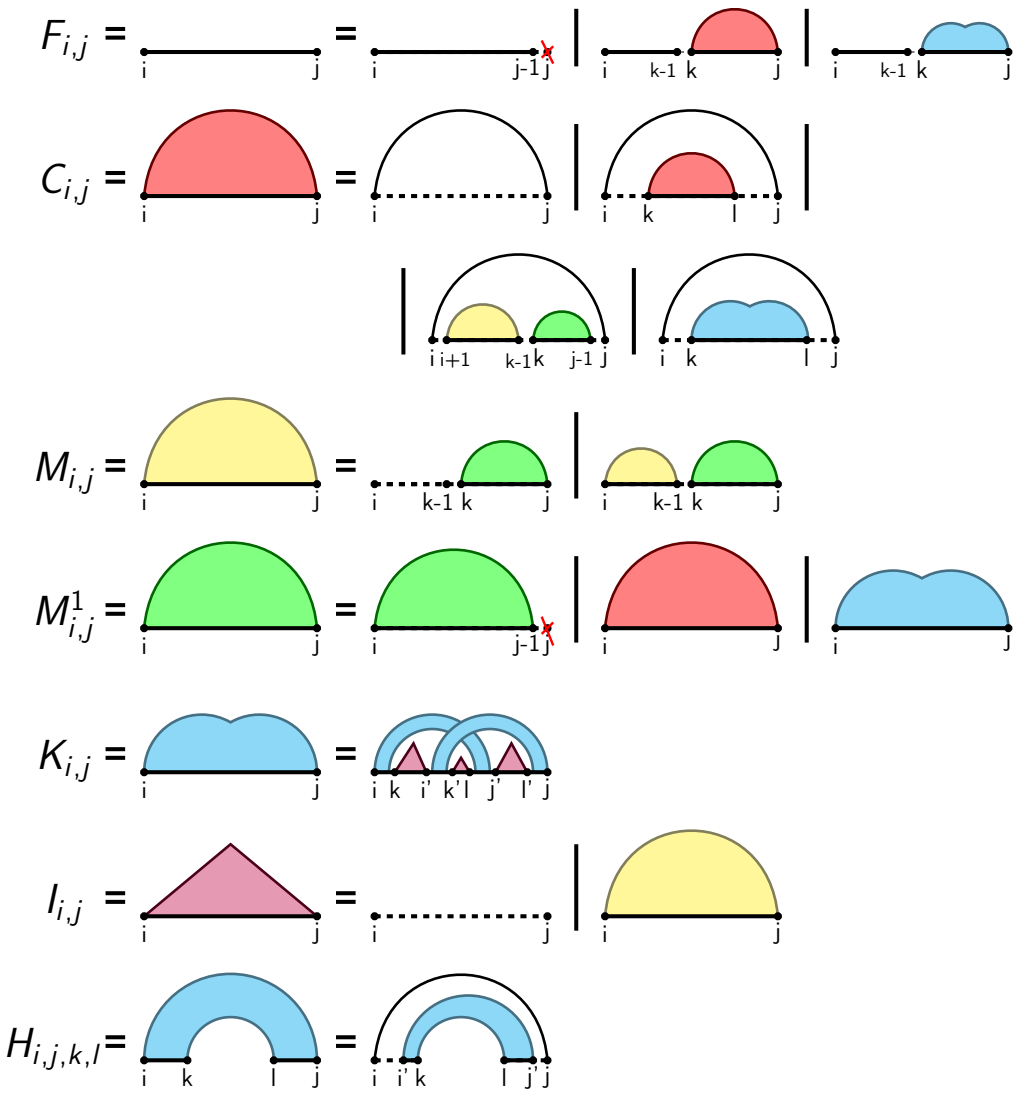

Figure 5.4: **The DP scheme for the sampling of the secondary structures including pseudoknots**. $K_{i,j}$ and $I_{i,j}$ indicate a substring $w[i,j]$ holding a pseudoknot and a substring internal to a pseudoknot respectively. $H_{i,j,k,l}$ stores impefect helices $h(i,j,k,l)$. See Section 5.2.4 for further details.

5.2.5 Formalizing the extension

We can now formalize the DP scheme for pseudoknotted structures described in this Section. We denote it by $(\mathcal{Q}_{MCv}, q^C_{1,n}, \rho_{MCv})$, with MCv standing for McCaskill Variant. The parting point for establishing ρ_{MCv} is ρ_{MCv} is described by Equation 2.20 describing VZualgorithm, which has the same formalization as McCaskill algorithm.

Taking into account how the energy of the proposed pseudoknots is computed, the DP scheme $(\mathcal{Q}_{MCb}, q^{C}_{i,n}, \rho_{MCv})$ introduces six new elements:

- A type-H pseudoknot appearing at the level of external loop;
- A type-H pseudoknot appearing inside basepair alone;
- A type-H pseudoknot appearing within a multibranched loop;
- A decomposition of a pseudoknot into its elements, namely imperfect branches $\widehat{\mathcal{H}}_1$ and $\widehat{\mathcal{H}}_2$ and the intermediate segments $w[k, i']$, $w[k', l]$ and $w[j', l']$.
- An unpaired segment appearing inside a pseudoknot, not counting imperfect helices;
- A loops appearing inside a pseudoknot, not counting imperfect helices;

This translates into establishing a new constructors for each element:

- $\lambda^{pk_F}_{i,j,k}$ a pseudoknot in an external loop;
- $\lambda^{pk_C}_{i,j,k,l}$ a pseudoknot in a base pair;
- $\lambda^{pk_M}_{i,j}$ a pseudoknot as a branch in multibranch loop;
- $\lambda^{pk_H}_{i,j,k,l,i',j',k',l'}$ a decompostion of a pseudoknot to imperfect helices

  $$h_1(i, j', k, l) \in h_1(i, j', k, l), h_2(i', j, k', l') \in h_2(i', j, k', l').$$

  and internal segments $w[k, i']$, $w[k', l]$ and $w[j', l']$.
- $\lambda^{I_p k}_{i,j}$ an unpaired segment in pseudoknot;
- $\lambda^{M_p k}_{i,j}$ a loop in pseudoknot.

Creating a helix is equivalent to creating a stack/bulge/internal loop - in that case we can reuse the constructor $\lambda^{ILG}_{i,j,k,l}$.

Therefore, ρ_{MCv} may be given as

$$\rho_{MCv}(q^F_{i,j}) = \bigcup \begin{Bmatrix} \left\{(\{q^F_{i,j-1}\}, \lambda^{unp}_{i,j}, 1)\right\} \\ \bigcup\limits_{\substack{i\le k<j \\ (k,j)\in S(i,j)}} \left\{(\{q^F_{i,k-1}, q^C_{k+1,j-1}\}, \lambda^{pair}_{i,j,k}, 1)\right\} \\ \bigcup\limits_{\substack{i\le k<j \\ (k,j)\in S(i,j)}} \left\{(\{q^F_{i,k-1}, q^K_{k+1,j-1}\}, \lambda^{pk_F}_{i,j,k}, 1)\right\} \end{Bmatrix}$$

$$\rho_{MCv}(q^C_{i,j}) = \bigcup \begin{Bmatrix} \left\{(\varnothing, \lambda^H_{i,j}, E_H(i,j))\right\} \\ \bigcup\limits_{\substack{i<k<l<j \\ (k,l)\in S(i,j)}} \left\{(\{q^C_{k,l}\}, \lambda^{ILG}_{i,j,k,l}, E_{ILG}(i,j,k,l))\right\} \\ \bigcup\limits_{\substack{i<k<l<j \\ (k,l)\in S(i,j)}} \left\{(\{q^K_{k,l}, \lambda^{pk_C}_{i,j,k,l}\}, a+2b+c\times(j-l+k-i-2))\right\} \\ \bigcup\limits_{i<k<j} \left\{(\{q^M_{i+1,k-1}, q^{M1}_{k,j-1}\}, \lambda^{ML}_{i,k,j}, a+b)\right\} \end{Bmatrix}$$

$$\rho_{MCv}(q^M_{i,j}) = \bigcup \begin{Bmatrix} \bigcup\limits_{i<k<j} \left\{(\{q^{M1}_{k,j}\}, \lambda^{ML1hel}_{i,j,k}, c\times(k-i))\right\} \\ \bigcup\limits_{i<k<j} \left\{(\{q^M_{i,k-1}, q^{M1}_{k,j}\}, \lambda^{ML2hel}_{i,j,k}, 1)\right\} \end{Bmatrix}$$

$$\rho_{MCv}(q^{M1}_{i,j}) = \bigcup \begin{Bmatrix} \left\{(\{q^{M1}_{i,j-1}\}, \lambda^{unp}_{i,j}, c)\right\} \\ \left\{(\{q^C_{i,j}\}, \lambda^{p}_{i,j}, b)\right\} \\ \left\{(\{q^K_{i,j}\}, \lambda^{pk_M}_{i,j}, -2b)\right\} \end{Bmatrix}$$

$$\rho_{MCv}(q^K_{i,j}) = \bigcup_{\substack{i<k \\ l'<j \\ k<i'<k'<l<j'<l'}} \left\{(\{q^H_{i,j',k,l}, q^H_{i',j,k',l'}, q^I_{k,i'}, q^I_{k',l}, q^I_{j',l'}\}, \lambda^{pkH}_{i,j,k,l,i',j',k',l'}, E_{\mathfrak{K}})\right\}$$

$$\rho_{MCv}(q^I_{i,j}) = \bigcup \begin{Bmatrix} \left\{(\varnothing, \lambda^{I_p k}_{i,j}, c\times(j-i+1))\right\} \\ \left\{(\{q^M_{i,j}\}, \lambda^{M_p k}_{i,j}, 1)\right\} \end{Bmatrix}$$

$$\rho_{MCv}(q^H_{i,j,k,l}) = \bigcup_{\substack{i<i'<k \\ l<j'<j}} \left\{(\{q^H_{i',j',k,l}\}, E_{ILG}(i,j,i',j'))\right\}$$

The search space and the partition function can be formalized for each state q by exploiting the Definition 2.6.1 and Theorem 2.6.2 respectively. The formalism also validates the Equation 5.2 since these relations stem from the Equation 3.11 that was already proven.

The example of formalizing a DP scheme $(\mathcal{Q}_{MCv}, q^C_{1,n}, \rho_{MCv})$ shows a real strength

of the introduced formalism, since it allows to formalize even a states represented by a disconnected substrings, such as represented by the state $q^{H}_{i,j,k,l}$. This is possible mainly due to the general definitions of $q \in \mathcal{Q}$ and $\lambda \in \mathcal{D}$ that allows them to take as many parameters as necessary, meaning that it is possible to define internal limits of a helix. Consequently, the introduced formalism can be used to express the DP schemes used in algorithms that sample pseudoknotted structures.

5.2.6 Implementation

The above DP scheme was adapted to an algorithm for searching a suboptimal pseudoknots in a program `PKnotFind`. The program was written in `C`. For the base of the McCaskill algorithm, we used the implementation employed in ViennaRNA library [55]. The newest iteration of this library allows for a very handy inclusion of an auxiliary grammar, which is the function we used to implement the rules handling the H-type pseudoknots.

While the current iteration of the library allows to add only one additional rule, this is not a problem as both computation of values $I_{i,j}$ and $H_{i,j}$ can be done before computing $K_{i,j}$. Likewise for traceback of the sampled solution, these values are accessed only from pseudoknots, meaning their values need to be accessed only its traceback. However, since the auxiliary grammar was not included for the stochastic backtrack at the time, we had to adapt its code for the purpose of pseudoknot sampling.

The preliminary version of the program `PKnotFind` is available at:
`https://github.com/JurMich/PknotFind`

5.3 Results

This section provide a few preliminary results testing notably validating the concept of the algorithm. Unfortunately, due to a lack of time we were not able to provide more substantial results such as benchmarks, which is the task that is left for the future.

5.3.1 Toy Examples

The first test that we performed were destined to verify whether the algorithm covers the H-type pseudoknots $\mathfrak{K}(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$ we are interested in. Since the entirety of sequences w of the search space is hard to enumerate for long w, we performed these tests on artificial sequences designed specifically for them. For this reason we assume $\Delta = 2$ as well.

The easy way to test the completeness of the DP scheme $(\mathcal{Q}_{\text{MCb}}, q^{C}_{\text{I},n}, \rho_{\text{MCv}})$ is to assume a temperature $T \to +\infty$ without it affecting the energy values, in which case we have, $\forall S$:

$$\mathcal{Z}_S^{+\infty} = e^{\frac{-E_S}{k_B T}}_{T \to +\infty} = 1$$

which implies uniform distribution and $\mathcal{Z}^{+\infty} = |\mathcal{S}_n|$. For simpler sequences, it is simple to count manually the number of possible secondary structures, including those with $\mathfrak{K}(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2)$, then validate the results by computing the partition function when $T \to +\infty$.

The first sequence for which the test was performed is $w = \mathsf{CACAAGAG}$. The reason we provide the results for this short sequence is it is explicit to enumerate all possible secondary structures that satisfy the given conditions, notably the minimum distance between base pairing nucleotides $\theta = 3$ and $\Delta = 2$:

```
........
(......)
(....)..
..(....)
[.{..].}
```

Here, '{}' and '[]' represent the base pairs of the pseudoknots. In this case $\mathcal{Z}^{+\infty} = |\mathcal{S}_n| = 5$.

Results 5.3.1. With `PKnotFind`, all five solutions are found, including the pseudoknotted structures. More importantly when generating $N = 10^5$ samples with the occurrence of each structure being counted, we found the following numbers:

```
20109 ........
19953 ..(....)
20025 (......)
19920 (....)..
19993 [.{..].}
```

These counts point to an uniform distribution of $\mathcal{Z}_S^{+\infty}$ as stated. Therefore for the simplest examples, the `PKnotFind` returns expected solutions.

The second interesting sequence we tested is

$$w = \mathsf{CACACAAGAGCAAGAG}.$$

While the enumeration of search space structures is not as explicit as in previous example, it is still easy to enumerate all secondary structures satisfying the given conditions. In this case $|\mathcal{S}_n| = 33$. In this case we sampled $N = 10^6$ samples.

Results 5.3.2. We found all 33 structures with `PKnotFind` including the pseudoknotted structures. Notably, we also found the secondary structure

$$\texttt{[.\{..\{....]..\}}\},$$

which would not be found if the we searched only for the perfect helices. This structure also satisfies Δ, with the maximum number of an unpaired bases in helix being 2 as well as the difference between the length of both sides of helices $|(j-l)-(k-i)| = |15-12-6-3| = |-2| \leq \Delta$.

The results also show that the number of secondary structures raises very quickly if even the simplest pseudoknots are considered and Δ is small. For example for the random sequence

$$w = \mathsf{GCUACGGUCAUCAUCGUAUAGC}$$

we obtain $\mathcal{Z}^{+\infty} = |\mathcal{S}_n| = 233032$ structures. This makes validation for longer sequences and especially the cases where pseudoknots are combined with multi-branched loops difficult, notably to check for uniform distribution since it takes too many samples to have enough samples for each structure to verify. However, we managed to find a few sequences w with $|\mathcal{S}_n| < 2.5 \times 10^4$ and for $N = 10^7$ to verify the uniformity of generated distribution as well as that the number of the structures equals to $\mathcal{Z}^{+\infty} = |\mathcal{S}_n|$, which supports the validity of `PKnotFind`.

We also benchmarked the preliminary version of `PKnotFind` used for tests with one based on posets (see Section 5.2.3). The test for sequences of around 30nt and $\Delta = \delta$ have shown thrice as high execution time for the latter version. For this reason we abandoned the development of the version of `PKnotFind` with posets and stayed with the current one.

5.3.2 Comparison with `pKiss`

We performed the comparison of the test on random sequence

$$w = \text{GCUACGGUCAUCAUCGUAUAGC}$$

of 22 nt with `pKiss`. For `pKiss` we used the suboptimal mode with energy band (marked as an absolute deviation) of 140kcal.mol^{-1}. The strategy employed was the strategy P, not returning kissing hairpins. The minimum hairpin length was set to $\theta = 3$ and we also allowed lone base pairs.

For `PKnotFind`, we defined $\theta = 3$ and $\Delta = 2$. We used uniform energy model, since our main objective is to compare the search space of both software. We then generated 10^7 samples.

Results 5.3.3. The `pKiss` has returned 11415 structures, 39 of which present H-type pseudoknots with perfect helices. Even with lone base pairs allowed the minimum length of all helices in returned pseudoknots was 2. The pseudoknotted structure having the lowest energy was `[[.{{{]].....}}}.....`, with -1.5kcal.mol^{-1}.

The `PKnotFind` predicts 233032 structures. More importantly, we found all 39 pseudoknotted structures that were predicted by `pKiss`. This points to a complete coverage of the algorithm and DP scheme it is based on, as we do not omit any important pseudonknotted structures. It also predicts a considerable number of psuedknotted structures, including those with helix length of 1. It also found the structures that `pKiss` was not able to find due to being restricted only to perfect helices, such as `[{{{{{{{]...}}}}}}..}..`

We notice that the number of secondary structures, notably of those with pseudoknots, raises exponentially if i) the lone base pairs are allowed and ii) if conditions on helices are relaxed. This is the case even for low Δ. While `PKnotFind` is able to cover bigger number of secondary structures than `pKiss` when it comes to the H-type pseudoknots, this does not always have a positive impact. Notably, the lone base pairs have no stabilizing effect on the secondary structure and the construction of the pseudoknot is accompanied by a heavy free energy penalty, meaning that the construction of the pseudoknot of the helices consisting of lone base pairs is inefficient. For this reason, it would be a good idea to always consider the imperfect helices consisting of stacks of length at least 2 base pairs in the case of `PKnotFind`.

However, the main advantage of our algorithm is the sampling mode, which allows to sample secondary structures with H-type pseudoknots, since the partition function is computed for all possible pseudoknots spanning over $w[i, j]$. The energy model of the pseudoknot is based on multibranch loop, but the penalty for creating it still remains to be determined.

5.4 Pseudoknots and non-redundant sampling

The implementation of non-redundant sampling principle could not be done due to time constraints. In this section we will briefly discuss on how to implement it into the algorithm using the DP scheme $(\mathcal{Q}_{MCv}, q^{C}_{1,n}, \rho_{MCv})$.

The implementation can be done in an equivalen way to the one described in Sections 3.3 and 3.4.1. In fact, the latter describes the implementation in the algorithm used by VIENNARNA library, the same algorithm that was used as a starting point for the algorithm described in this Section. The only difference is the complexity of the implementation, since we need to ensure for pseudoknots that they are unique as well.

To ensure this, we proceed in the same manner as for other cases. The nodes of the data-structure $\mathfrak{d}(w, \mathcal{Q}_{MCv}, q^{C}_{1,n}, \rho_{MCv})$ store the partition function of all structures accessible from incomplete structure $\mathcal{Z}_{\mathcal{F}_{\mathcal{S}_{\mathcal{I}}}}$ represented by a parse tree

$$\mathfrak{T}_{I}(w, \mathcal{Q}_{MCv}, \mathcal{Q}_{NMCv}, q^{C}_{1,n}, \rho_{MCv}).$$

The main difference is the list of new constructors λ that was added along with the associated derivations, which means the extra cases to be handled and consequently increased complexity.

The formalism is useful here since it shows us exactly what new cases were added and must be treated. Likewise any scheme that can be formalized that way is also susceptible to non-redundant sampling, which was globally proven in Section 3.2.1. Therefore the implementation of non-redudant sampling to an algorithm based on DP scheme $(\mathcal{Q}_{MCb}, q^{C}_{1,n}, \rho_{MCv})$ is definitely possible and the only constraint is to handle all necessary derivations, which are more numerous due to a presence of extra elements.

5.5 Conclusion - Extension on Pseudoknots

In this section we presented the algorithm of the `PKnotFind`, an extension of an algorithm used in `pknotsRG`, resp. `pKiss`. This algorithm enumerates all imperfect helices that do not present longer unpaired stretches and bigger differences between the length of both sides of a helix than Δ and no bigger deviation between consecutive base pairs is present. These helices are then combined into H-type pseudoknots if they do not overlap.

The complete algorithm to sample stochastically the pseudoknotted structures is generated by incorporating the said algorithm into McCaskill algorithm based on DP scheme $(\mathcal{Q}_{MC}, q^C_{1,n}, \rho_{MC})$. This is done by enumerating all cases where a pseudoknot can appear, which are identical to the cases of appearance of a base pair. However, the energy constants are different, since in this algorithm, we treat a pseudoknot as a multibranched loop. We therefore needed to duplicate the rules for base pair construction while adjusting the mentioned values.

The practical tests are only preliminary, but they confirmed the expected behavior of the algorithm for $T \to +\infty$ with free energy values unaffected, namely the fact $\mathcal{Z}^{+\infty} = |\mathcal{S}_n|$ and the samples being uniformly distributed. The former could be verified for few sequences specifically designed for this purpose, while the latter was also tested for sequences with $|\mathcal{S}_n| < 2.5 \times 10^4$ by extensive sampling. We also confirm that `PKnotFind` is able to find the pseudoknotted secondary structures predicted by `pKiss`, pointing to the correct conceptual scheme behind the software. In addition it is able more pseudoknots than `PKnotFind` due to the relaxed conditions on the helix. However, not all of these structures seem to be useful, and it is advised for the future to include only the imperfect helices consisting of stacks at least 2 base pairs long. However, more tests are necessary to completely validate the `PKnotFind` software, and the benchmark tests are also necessary.

The following task that must be completed is the implementation of non-redundant principle into `PKnotFind`. This is simplified by the formalization of DP-scheme $(\mathcal{Q}_{MCv}, q^C_{1,n}, \rho_{MCv})$, since it clearly shows newly added derivations and corresponding constructors. This allows to adapt simply the non-redundant version of McCaskill algorithm in ViennaRNA to pseudoknots and of course to adapt the pseudoknot constructing algorithm itself by expanding the non-redundant layer to data-structure $\mathfrak{d}(w, \mathcal{Q}_{MCv}, q^C_{1,n}, \rho_{MCv})$. This can be done, seeing that the given DP scheme can be formalized, and therefore it constitutes a perspective into the

future.

Chapter 6

Conclusion and Perspectives

Our main objective concentrated on the analysis and application of the non-redundant sampling in the domain of the structural RNA bioinformatics. Since in the most of the cases, the redundancy is non-informative, therefore it only hinders the sampling process by slowing it down. We studied the methods of the removal of such redundancy without introducing bias and its application in the domain of structural bioinformatics. We notably studied the efficiency of such approach by the folding landscpae modeling and and the comparison with other, classic methods.

We began by establishing the concept of non-redundant sampling and the elements that are related to it. It was demonstrated that for the non-redundant sampling without bias to be possible, it is necessary to remember not only the secondary structures that were sampled, but also the process of their generation. For this reason, we proposed a specific data-structure that stores and handles these informations. This data-structure must be as efficient as possible, since its efficiency greatly determines the overall performance of the non-redundant sampling. We also designed this structure in a manner that it constitutes a separate layer with a limited number of interactions with the algorithm that is extended by a non-redundant sampling. This allows an easy implementation of the principle to any existing algorithms, and when necessary its variables can be adapted for arbitrary precision arithmetics via MPFR library [31].

As the first test, we implemented the non-redundant principle in the algorithm of Saffarian [80], a combinatorial algorithm that samples saturated secondary structures that are assumed to be locally optimal. Since such structures have a

considerable impact on the RNA kinetics due to acting as the kinetic traps, such algorithm is a prime candidate for RNA folding landscape analysis. The generated results demonstrate that the Saffarian algorithm extended by non-redundant sampling samples unique secondary structure more efficiently than the competitors. In addition, the returned models of RNA landscapes were, according to our criteria, closer to the reality than the results obtained by classic sampling methods.

To give users an easy access to the non-redundant sampling principle, we implemented it into the popular `ViennaRNA` library [55]. This library employs the DP scheme of McCaskill algorithm. We compared the efficiency of both non-redundant and classical versions of the sampling implemented in the library. With the efficient implementation the non-redundant sampling takes only 10-20% more time than the classic version to generate the same amount of samples. When it comes to unique samples, our algorithm clearly performs better for shorter sequences. For longer sequences, it performs worse at first due to small probability of generating a duplicates and the proportional overhead. However, since the redundant and non-redundant samplings are of the same complexity, it is expected to become more efficient after a sufficient number of the samples is generated.

One may ask whether the redundancy is really devoid of any information, and may point to the problem of the estimation of specific quantities of secondary structures. It is impossible to apply the naive estimator - computing wighted average of all samples - on non-redundant sampling sets. However, we researched an unbiased non-trivial estimator that allows perform such task. We tested the performance of such estimator on some model cases, with demonstrating that our estimator performs better on average that the simple case. However, the problem is the non-trivial estimator does not converge for the entire sampling run. While it may be easily modified by replacing it for a weighted average when the size of the sample approaches to the maximum number of unique structures, the case is quite unrealistic and must be resolved in the future.

Finally, we tackled the problem of the sampling of pseudoknotted secondary structures. The pseudoknots are usually omitted despite their biological relevance due to the impact on the complexity if one searches for pseudoknots [1]. The algorithm used in `pknotsRG` [76] and `pKiss` [47] is an efficient algorithm that allows to find the secondary structures with H-type pseudoknots composed of perfect helices. We extended this principle on imperfect helices presenting unpaired bases. The McCaskill algorithm serves as the starting point for our approach to sample secondary structures presenting such pseudoknots. While the

algorithm shows it can sample the same pseudoknotted structures as `pknotsRG` and many more possessing imperfect helices, their number is too high and some of them may be uninformative. This is the specifically the case of the pseudoknots composed from single base pair helices. In the future we plan to restrict the algorithm to exclude such cases. Another step that remains to be done is the implementation of the non-redundant sampling principle to the algorithm described above.

To demonstrate the different properties, we also introduced the general language to formalize the DP schemes employed algorithms used in structural RNA bioinformatics. This language is powerful due to its relative simplicity and flexibility that allows to formalize numerous DP schemes with it and prove their properties all at once. As shown, it can be used even to express even the DP schemes employed in algorithms for sampling pseudoknotted structures. It can also efficiently describe the most frequently employed scoring strategies, such as minimization of the energy and partition function limitation.

The final implementations include the software `RNANR`, written in `C` with total of 4000 lines, done in collaboration with H. Touzet, that includes non-redundant sampling as well as exhaustive enumeration of the saturated secondary structures, as well as the choice between double and arbitrary numerical precision arithmetic.

To make the non-redundant sampling principle easily accessible to the biggest number of users, we implemented it it the ViennaRNA [55] library where it is usable notably in `RNAsubopt` software. This implementation was doable in 1700 lines written in `C` code along with the version including arbitrary precision arithmetics using MPFR library [31]. This demonstrates the easiness of the implementation of non-redundant principle particularly well.

The final program, `PKnotFind`, that is used for sampling of the secondary structures with H-type pseudoknots, was also written in `C`, uses the algorithm and part of code of ViennaRNA as its base, and has 2000 lines of codes (more for the version containing posets that was ultimately abandoned). Besides these software, the scripts of total length around 22000 lines of code were created, mostly in `Python` an `R` to perform the tests and visualize the results.

Perspective-wise, the non-redundant sampling has waste possibilities of application in the field of the structural RNA bioinformatics. In general, there are many problems where it is necessary to predict a sequence, structure or a composition. Such prediction problems frequently require an access to the suboptimal struc-

tures. As long as such problem can be solved using dynamic programming,these problems might be subjected to the the non-redundant sampling principle. It is also useful in the cases where it is necessary to establish a model of the search space, such as we demonstrated in the case of the folding landscapes. This is not restricted to the domain of the secondary structures; for example, the non-redundant sampling might be useful to sample the space of sequences for the purpose of RNA design.

To study the search spaces and similar problems, another class of approaches called parametric inference models [69] exists. It represents each biological problem as a graphical model defined by a certain number of parameters. These graphical models are then solved to access the value of these parameters or corresponding space delimited by the range of their values. At first look it may seem that such methods are in competition with non-redundant sampling by basically specifying it by a set of parameters. However, even parametric inference models can benefit from non-redundant sampling. More often than not the parameters feature a certain degree of liberty. Sometimes the number of free parameters can be high, equating to important size of combinations. In this case the sampling can used to extract interesting combinations of parameters, and non-redundant sampling would allow access faster to higher number of them.

But the principle applied for the non-redundant sampling has much richer application than the non-redundant sampling itself. Since the non-redundant sampling principle simply sees the sampled structures as a forbidden structures, it can be easily adapted to not sample certain structures from the beginning, all without introducing the bias between all other structures. This means we can easily sample only certain secondary structures that do not present a certain local substructure, such as specific forbidden loop, or just a specific structure that was blacklisted beforehand, making it potentially a very powerful tool.

The non-redundant sampling has also the potential application outside the field of RNA bioinformatics due to its general concept [57]. One possible application would concern the simulations of the networks where each branch has maximum capacity. The non redundant sampling would restrict the number of times that a specific path can be taken depending on these capacities, and would perform the simulation of the behavior of such network. This might potentially have a significant use in the domain of the medicine, where different circulatory systems could be simulated this way. This could be helpful to study the circulatory system diseases via the simulations, reducing or removing the necessity of study in humans. It could potentially have the application in other domains as well, such

as the urban engineering and traffic networks.

All in all, the non redundant sampling provides a method for optimization of algorithm performance, within the domain of structural RNA bioinformatics or outside of it, and might be the step towards the analysis of the longer RNA sequences.

Index

Bibliography

[1] Tatsuya Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Appl Math*, 104(1-3):45–62, 2000.

[2] Mirela Andronescu, Vera Bereg, Holger H Hoos, and Anne Condon. RNA STRAND: the RNA secondary structure and statistical analysis database. *BMC bioinformatics*, 9:340, August 2008.

[3] Svante Arrhenius. Über die dissociationswärme und den einfluss der temperatur auf den dissociationsgrad der elektrolyte. *Zeitschrift für physikalische Chemie*, 4(1):96–116, 1889.

[4] O T Avery, C M Macleod, and M McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types : Induction of transformation by a Desoxyribonucleic Acid fraction isolated from Pneumococcus Type III. *The Journal of experimental medicine*, 79:137–158, February 1944.

[5] Tania A Baker, James D Watson, Stephen P Bell, A Gann, MA Losick, and R Levine. *Molecular biology of the gene*. Benjamin-Cummings Publishing Company, 2003.

[6] David R Bell, Sara Y Cheng, Heber Salazar, and Pengyu Ren. Capturing RNA folding free energy with coarse-grained molecular dynamics simulations. *Scientific reports*, 7:45812, April 2017.

[7] Richard Bellman. The theory of dynamic programming. Technical report, RAND Corp Santa Monica CA, 1954.

[8] Helen Berman, Kim Henrick, and Haruki Nakamura. Announcing the worldwide protein data bank. *Nature Structural and Molecular Biology*, 10(12):980, 2003.

[9] Christof K Biebricher, Stephan Diekmann, and Rüdiger Luce. Structural analysis of self-replicating RNA synthesized by Qβ replicase. *Journal of molecular biology*, 154(4):629–648, 1982.

[10] Sandro Bottaro, Francesco Di Palma, and Giovanni Bussi. Towards de novo RNA 3D structure prediction. *arXiv preprint arXiv:1502.05667*, 2015.

[11] J Brachet. Nucleic acids in the cell and the embryo. *Symposia of the Society for Experimental Biology*, pages 207–224, 1947.

[12] Gregory Braun, Dennis Tierney, and Heidrun Schmitzer. How Rosalind Franklin discovered the helical structure of DNA: Experiments in diffraction. 49:140–143, 2011.

[13] P Bénas, G Bec, G Keith, R Marquet, C Ehresmann, B Ehresmann, and P Dumas. The crystal structure of HIV reverse-transcription primer tRNA(Lys,3) shows a canonical anticodon loop. *RNA (New York, N.Y.)*, 6:1347–1355, October 2000.

[14] Patricia P. Chan and Todd M. Lowe. GtRNAdb 2.0: an expanded database of transfer RNA genes identified in complete and draft genomes. *Nucl. Acid. Res.*, 44:D184–D189, 2016.

[15] E Chargraff, R Lipshitz, and C Green. Composition of the desoxypentose nucleic acids of four genera of sea-urchin. *The Journal of biological chemistry*, 195:155–160, March 1952.

[16] Jonathan L Chen, Abigael L Dishler, Scott D Kennedy, Ilyas Yildirim, Biao Liu, Douglas H Turner, and Martin J Serra. Testing the nearest neighbor model for canonical RNA base pairs: revision of GU parameters. *Biochemistry*, 51:3508–3522, April 2012.

[17] F Crick. Central dogma of molecular biology. *Nature*, 227:561–563, August 1970.

[18] F. H. C. Crick. Codon—anticodon pairing: The wobble hypothesis. *J. Mol. Biol*, 19:548–555, 1966.

[19] Jan Cupal, Christoph Flamm, Alexander Renner, and Peter F Stadler. Density of states, metastable states, and saddle points: Exploring the energy landscape of an RNA molecule. In *ISMB*, pages 88–91, 1997.

[20] Ralf Dahm. Friedrich Miescher and the discovery of DNA. *Developmental biology*, 278:274–288, February 2005.

[21] Ralf Dahm. Discovering DNA: Friedrich Miescher and the early years of nucleic acid research. *Human genetics*, 122:565–581, January 2008.

[22] Ludmila V Danilova, Dmitri D Pervouchine, Alexander V Favorov, and Andrei A Mironov. RNAKinetics: a web server that models secondary structure kinetics of an elongating RNA. *Journal of bioinformatics and computational biology*, 4:589–596, April 2006.

[23] Kévin Darty, Alain Denise, and Yann Ponty. VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics (Oxford, England)*, 25:1974–1975, August 2009.

[24] Y. Ding and E. Lawrence. A statistical sampling algorithm for RNA secondary structure prediction. *Nucleic Acids Res*, 31(24):7280–7301, 2003.

[25] Ivan Dotu, William A Lorenz, Pascal Van Hentenryck, and Peter Clote. Computing folding pathways between RNA secondary structures. *Nucleic acids research*, 38(5):1711–1722, 2009.

[26] H Du, A V Yakhnin, S Dharmaraj, and P Babitzke. trp RNA-binding attenuation protein-5′ stem-loop RNA interaction is required for proper transcription attenuation control of the Bacillus subtilis trpEDCFBA operon. *Journal of bacteriology*, 182:1819–1827, April 2000.

[27] D Elson and E Chargraff. On the desoxyribonucleic acid content of sea urchin gametes. *Experientia*, 8:143–145, April 1952.

[28] C Flamm, W Fontana, I L Hofacker, and P Schuster. RNA folding at elementary step resolution. *RNA (New York, N.Y.)*, 6:325–338, Mar 2000.

[29] Christoph Flamm and Ivo L. Hofacker. Beyond energy minimization: approaches to the kinetic folding of RNA. *Monatshefte für Chemie - Chemical Monthly*, 139(4):447–457, 2008.

[30] Christoph Flamm, Ivo L. Hofacker, Peter F. Stadler, and Michael T. Wolfinger. Barrier trees of degenerate landscapes. *Zeitschrift für Physikalische Chemie*, 216(2):155, 2002.

[31] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. MPFR: a multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software (TOMS)*, 33(2):13, 2007.

[32] R S Gardner, A J Wahba, C Basilio, R S Miller, P Lengyel, and J F Speyer. Synthetic polynucleotides and the amino acid code. VII. *Proceedings of the National Academy of Sciences of the United States of America*, 48:2087–2094, December 1962.

[33] Rees F Garmann, Ajaykumar Gopal, Shreyas S Athavale, Charles M Knobler, William M Gelbart, and Stephen C Harvey. Visualizing the global secondary structure of a viral RNA genome with cryo-electron microscopy. *Rna*, 2015.

[34] J Willard Gibbs. *Elementary principles in statistical mechanics*. Courier Corporation, 2014.

[35] Robert Giegerich, Björn Voss, and Marc Rehmsmeier. Abstract shapes of rna. *Nucleic acids research*, 32:4843–4851, 2004.

[36] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. 22:403–434, 1976.

[37] Sha Gong, Yanli Wang, Zhen Wang, and Wenbing Zhang. Co-transcriptional folding and regulation mechanisms of riboswitches. *Molecules (Basel, Switzerland)*, 22, July 2017.

[38] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.5 edition, 2012. `http://gmplib.org/`.

[39] F Griffith. The Significance of Pneumococcal Types. *The Journal of hygiene*, 27:113–159, January 1928.

[40] István Hargittai. The tetranucleotide hypothesis: a centennial. *Structural Chemistry*, 20(5):753, October 2009.

[41] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. 57:97–109, 1970.

[42] Paul G Higgs. RNA secondary structure: physical and computational aspects. *Quarterly reviews of Biophysics*, 33(3):199–253, 2000.

[43] Ivo L Hofacker, Peter F Stadler, and Peter F Stadler. RNA secondary structures. *Reviews in Cell Biology and Molecular Medicine*, 2006.

[44] R W Holley, J Apgar, G A Everett, J T Madison, M Marquisee, S H Merrill, J R Penswick, and A Zamir. Structure of a Ribonucleic acid. *Science (New York, N.Y.)*, 147:1462–1465, March 1965.

[45] Valerie Hower and Christine E Heitsch. Parametric analysis of RNA branching configurations. *Bulletin of mathematical biology*, 73(4):754–776, 2011.

[46] Jonathan L Jacobs, Ashton T Belew, Rasa Rakauskaite, and Jonathan D Dinman. Identification of functional, endogenous programmed- 1 ribosomal frameshift signals in the genome of Saccharomyces cerevisiae. *Nucleic acids research*, 35(1):165–174, 2006.

[47] Stefan Janssen and Robert Giegerich. The RNA shapes studio. *Bioinformatics*, 31(3):423–425, 2014.

[48] Daniel Jost and Ralf Everaers. Prediction of RNA multiloop and pseudoknot conformations from a lattice-based, coarse-grain tertiary structure model. *The Journal of chemical physics*, 132(9):03B601, 2010.

[49] Ioanna Kalvari, Joanna Argasinska, Natalia Quinones-Olvera, Eric P Nawrocki, Elena Rivas, Sean R Eddy, Alex Bateman, Robert D Finn, and Anton I Petrov. Rfam 13.0: shifting to a genome-centric resource for non-coding RNA families. *Nucleic acids research*, 46:D335–D342, January 2018.

[50] Marcel Kucharik, Ivo L Hofacker, Peter F Stadler, and Jing Qin. Basin Hopping Graph: a computational framework to characterize RNA folding landscapes. *Bioinformatics (Oxford, England)*, 30:2009–2017, Jul 2014.

[51] Paola Lecca. Stochastic chemical kinetics. *Biophysical reviews*, 5(4):323–345, 2013.

[52] P.A. Levene. The structure of Yeast Nucleic Acid. *J. Biol. Chem*, (40):415–424, 1919.

[53] P.A. Levene. On the structure of Thymus Nucleic Acid and on its possible bearing on the structure of plant nucleic acid. *J. Biol. Chem*, (48):119–125, 1921.

[54] Yuan Li and Shaojie Zhang. Finding stable local optimal RNA secondary structures. *Bioinformatics (Oxford, England)*, 27:2994–3001, Nov 2011.

[55] Ronny Lorenz, Stephan H Bernhart, Christian Höner Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. ViennaRNA Package 2.0. *Algorithms for molecular biology : AMB*, 6:26, Nov 2011.

[56] William A Lorenz and Peter Clote. Computing the partition function for kinetically trapped RNA secondary structures. *PLoS One*, 6:e16178, Jan 2011.

[57] William Andrew Lorenz and Yann Ponty. Non-redundant random generation algorithms for weighted context-free grammars. *Theoretical Computer Science*, 502:177–194, 2013.

[58] Rune B Lyngsø and Christian NS Pedersen. Pseudoknots in RNA secondary structures. In *Proceedings of the fourth annual international conference on Computational molecular biology*, pages 201–209. ACM, 2000.

[59] Rune B Lyngsø, Michael Zuker, and Christian NS Pedersen. An improved algorithm for RNA secondary structure prediction. *BRICS Report Series*, 6(15), 1999.

[60] Rune B Lyngsø, Michael Zuker, and Christian NS Pedersen. Internal loops in RNA secondary structure prediction. In *Proceedings of the third annual international conference on Computational molecular biology*, pages 260–267. ACM, 1999.

[61] Ján Maňuch, Chris Thachuk, Ladislav Stacho, and Anne Condon. NP-completeness of the energy barrier problem without pseudoknots and temporary arcs. *Natural Computing*, 10(1):391–405, 2011.

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

[63] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. 21:1087–1092, 1953.

[64] Juraj Michálik, Hélène Touzet, and Yann Ponty. Efficient approximations of RNA kinetics landscape using non-redundant sampling. *Bioinformatics (Oxford, England)*, 33:i283–i292, July 2017.

[65] G Milman, R Langridge, and M J Chamberlin. The structure of a DNA-RNA hybrid. *Proceedings of the National Academy of Sciences of the United States of America*, 57:1804–1810, June 1967.

[66] J Monod and F Jacob. Teleonomic mechanisms in cellular metabolism, growth, and differentiation. *Cold Spring Harbor symposia on quantitative biology*, 26:389–401, 1961.

[67] M W Nirenberg and J H Matthaei. The dependence of cell-free protein synthesis in e. coli upon naturally occurring or synthetic polyribonucleotides. *Proceedings of the National Academy of Sciences of the United States of America*, 47:1588–1602, October 1961.

[68] Ruth Nussinov, George Pieczenik, Jerrold R Griggs, and Daniel J Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied mathematics*, 35(1):68–82, 1978.

[69] Lior Pachter and Bernd Sturmfels. Parametric inference for biological sequence analysis. *Proceedings of the National Academy of Sciences*, 101(46):16138–16143, 2004.

[70] G E Palade. Studies on the endoplasmic reticulum. II. Simple dispositions in cells in situ. *The Journal of biophysical and biochemical cytology*, 1:567–582, November 1955.

[71] G E Palade. Albert Claude and the beginnings of biological electron microscopy. *The Journal of cell biology*, 50:5d–19d, July 1971.

[72] M Petersheim and D H Turner. Base-stacking and base-pairing contributions to helix stability: thermodynamics of double-helix formation with CCGG, CCGGp, CCGGAp, ACCGGp, CCGGUp, and ACCGGUp. *Biochemistry*, 22:256–263, January 1983.

[73] Jacob T Polaski, Samantha M Webster, James E Johnson, and Robert T Batey. Cobalamin riboswitches exhibit a broad range of ability to discriminate between methylcobalamin and adenosylcobalamin. *Journal of Biological Chemistry*, pages jbc–M117, 2017.

[74] Yann Ponty and Cédric Saule. A combinatorial framework for designing (pseudoknotted) RNA algorithms. In *International Workshop on Algorithms in Bioinformatics*, pages 250–269. Springer, 2011.

[75] Devashish Rath, Lina Amlinger, Archana Rath, and Magnus Lundgren. The CRISPR-Cas immune system: biology, mechanisms and applications. *Biochimie*, 117:119–128, October 2015.

[76] Jens Reeder, Peter Steffen, and Robert Giegerich. pknotsRG: RNA pseudoknot folding including near-optimal structures and sliding windows. *Nucleic acids research*, 35(suppl_2):W320–W324, 2007.

[77] Christian M Reidys, Fenix WD Huang, Jørgen E Andersen, Robert C Penner, Peter F Stadler, and Markus E Nebel. Topology and prediction of RNA pseudoknots. *Bioinformatics*, 27(8):1076–1085, 2011.

[78] E Rivas and S R Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of molecular biology*, 285:2053–2068, February 1999.

[79] Metal Ruff, S Krishnaswamy, M Boeglin, A Poterszman, A Mitschler, A Podjarny, B Rees, JC Thierry, and D Moras. Class II aminoacyl transfer RNA synthetases: crystal structure of yeast aspartyl-tRNA synthetase complexed with tRNA (asp). *Science*, 252(5013):1682–1689, 1991.

[80] Azadeh Saffarian, Mathieu Giraud, Antoine de Monte, and Hélène Touzet. RNA locally optimal secondary structures. *Journal of computational biology : a journal of computational molecular cell biology*, 19:1120–1133, Oct 2012.

[81] F Sanger and A R Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of molecular biology*, 94:441–448, May 1975.

[82] Schrödinger, LLC. The PyMOL Molecular Graphics System, Version 1.8. November 2015.

[83] Susan J Schroeder and Douglas H Turner. Optical melting measurements of nucleic acid thermodynamics. *Methods in enzymology*, 468:371–387, 2009.

[84] Lincoln G Scott and Mirko Hennig. RNA structure determination by NMR. In *Bioinformatics*, pages 29–61. Springer, 2008.

[85] Kim Sharp and Franz Matschinsky. Translation of Ludwig Boltzmann's paper "On the relationship between the second fundamental theorem of the mechanical theory of heat and probability calculations regarding the conditions for thermal equilibrium" sitzungberichte der kaiserlichen akademie der wissenschaften. mathematisch-naturwissen classe. abt. ii, lxxvi 1877, pp 373-435 (wien. ber. 1877, 76: 373-435). reprinted in wiss. abhandlungen, vol. ii, reprint 42, p. 164-223, barth, leipzig, 1909. *Entropy*, 17(4):1971–2009, 2015.

[86] Claudia Steglich, Debbie Lindell, Matthias Futschik, Trent Rector, Robert Steen, and Sallie W Chisholm. Short RNA half-lives in the slow-growing marine cyanobacterium prochlorococcus. *Genome biology*, 11(5):R54, 2010.

[87] E. Stofer, C. Chipot, and R. Lavery. Free Energy calculations of Watson-Crick base pairing in aqueous solution. *J. Am. Chem. Soc.*, 121:9503–9508, 1999.

[88] Carla A Theimer, Craig A Blois, and Juli Feigon. Structure of the human telomerase RNA pseudoknot reveals conserved tertiary interactions essential for function. *Molecular cell*, 17(5):671–682, 2005.

[89] Carsten Timm. Random transition-rate matrices for the master equation. *Physical Review E*, 80(2):021140, 2009.

[90] Ignacio Tinoco. Nucleic acid structures, energetics, and dynamics. *The Journal of Physical Chemistry*, 100(31):13311–13322, 1996.

[91] Francesca Tuorto and Frank Lyko. Genome recoding by tRNA modifications. *Open biology*, 6, December 2016.

[92] D H Turner, N Sugimoto, and S M Freier. RNA structure prediction. *Annu Rev Biophys Biophys Chem*, 17:167–192, 1988.

[93] Douglas H Turner and David H Mathews. NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure. *Nucleic acids research*, 38:D280–D282, January 2010.

[94] A J Wahba, R S Gardner, C Basilio, R S Miller, J F Speyer, and P LL. Synthetic polynucleotides and the amino acid code. VIII. *Proceedings of the National Academy of Sciences of the United States of America*, 49:116–122, January 1963.

[95] J Waldispühl and P Clote. Computing the partition function and sampling for saturated secondary structures of RNA, with respect to the Turner energy model. *Journal of computational biology : a journal of computational molecular cell biology*, 14:190–215, Mar 2007.

[96] Michael S Waterman. Secondary structure of single-stranded nucleic acids. *Adv. math. suppl. studies*, 1:167–212, 1978.

[97] Michael S Waterman and Temple F Smith. RNA secondary structure: A complete mathematical analysis. *Mathematical Biosciences*, 42(3-4):257–266, 1978.

[98] J D Watson and F H C Crick. Molecular structure of nucleic acids. A structure for deoxyribose nucleic acid. 1953. *Annals of internal medicine*, 138:581–582, April 2003.

[99] John Westbrook, Zukang Feng, Li Chen, Huanwang Yang, and Helen M Berman. The Protein Data Bank and structural genomics. *Nucleic Acids Res*, 31:489–491, Jan 2003.

[100] E Westhof and M Sundaralingam. Restrained refinement of the monoclinic form of yeast phenylalanine transfer RNA. Temperature factors and dynamics, coordinated waters, and base-pair propeller twist angles. *Biochemistry*, 25(17):4868–4878, 1986.

[101] Eric Westhof. Twenty years of RNA crystallography. *RNA*, 21(4):486–487, 2015.

[102] K P Williams and D P Bartel. Phylogenetic analysis of tmRNA secondary structure. *RNA (New York, N.Y.)*, 2:1306–1310, December 1996.

[103] Michael T. Wolfinger, Andreas Svrcek-Seiler, Christoph Flamm, Ivo L. Hofacker, and Peter F. Stadler. Efficient computation of RNA folding dynamics. *J. Phys. A: Math*, 37, 2004.

[104] S. Wuchty, W. Fontana, I.L. Hofacker, and P. Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopol.*, 49:145–164, 1999.

[105] C Xie and J P O'Leary. DNA and the brains behind its discovery. *The American surgeon*, 62:979–980, November 1996.

[106] Michael Zuker and David Sankoff. RNA secondary structures and their prediction. *Bulletin of mathematical biology*, 46(4):591–621, 1984.

[107] Michael Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9(1):133–148, 1981.

université PARIS-SACLAY
ÉCOLE DOCTORALE
INTERFACES
Approches Interdisciplinaires : Fondements, Applications et Innovation

Titre : Echantillonage sans remise en Bioinformatique des Acides RiboNucléiques

Mots clés : ARN, programmation dynamique, échantillonage non-redonddante, algorithme combinatoire, modèle thermodynamique, cinétique

Résumé : Un échantillonnage statistique est central à de nombreuses méthodes algorithmiques pour la bioinformatique structurale des ARNs, où ils sont couramment utilisés pour identifier des modèles structuraux importants, fournir des résumés des espaces de repliement ou approcher des quantités d'intérêt dans l'équilibre thermodynamique. Dans tous ces exemples, la redondance dans l'ensemble échantillonné est non-informative et inefficace, limitant la portée des applications des méthodes existantes. Dans cette thèse, nous introduisons le concept de l'échantillonnage non-redondante et nous explorons ses applications et conséquences en bioinformatique des ARN.
Nous commençons par introduire formellement le concept d'échantillonnage non-redondante et nous démontrons que tout algorithme échantillonnant dans la distribution de Boltzmann peut être modifié en une version non-redondante. Son implémentation repose sur une structure de données spécifique et la modification d'une remontée stochastique pour fournir l'ensemble des structures uniques, avec la même complexité.
Nous montrons alors une exemple pratique en implémentant le principe d'échan- tillonnage non-redondant au sein d'un algorithme combinatoire qui échantillonne des structures localement optimales. Nous exploitons cet outil pour étudier la cinétique des ARN, modélisant des espaces de repliement générés à partir des structures localement optimales. Ces structures agissent comme des pièges cinétiques, rendant leur prise en compte essentielle pour analyser la dynamique des ARN. Des résultats empirique montrent que des espaces de repliement générés à partir des échantillons non-redondants sont plus proches de la réalité que ceux obtenus par un échantillonnage classique.
Nous considérons ensuite le problème du calcul efficace d'estimateurs statistiques à partir d'échantillons non redondants. L'absence de la redondance signifie que l'estimateur naïf, obtenu en moyennant des quantités observés dans l'échantillon, est eronné. Par contre, nous établissons un estimateur non-trivial non-biaisé spécifique aux échantillons non-redondants suivant la distribution de Boltzmann. Nous montrons que l'estimateur des échantillons non-redondants est plus efficace que l'estimateur naïf, notamment dans les cas où la majorité des l'espace de recherche est échantillonné.
Finalement, nous introduisons l'algorithme d'échantillonnage, avec sa contre-partie non-redondante, pour des structures secondaires présentant des pseudonoeuds de type simple. Des pseudonoeuds sont typiquement omis pour des raisons d'efficacité, bien que beaucoup d'entre eux possèdent une grande importance biologique. Nos commençons par proposer une schèma de programmation dynamique qui permet d'énumérer tous les pseudonoeuds composés de deux hélices pouvant contenir des bases non-appariés qui s'entrecroisent. Ce schèma généralise la proposition de Reeders et Giegerich, choisi pour sa base complexité temporelle et spatiale. Par la suite, nous expliquons comment adapter cette décomposition à un algorithme d'échantillonnage statistique pour des pseudonoeuds simples. Finalement, nous présentons des résultats préliminaires et nous discutons sur l'extension de principe non-redondant dnas ce contexte.
Le travail présenté dans cette thèse ouvre non seulement la porte à l'analyse cinétique des séquences d'ARN plus longues, mais aussi l'analyse structurale plus détaillée des séquences d'ARN en général. L'échantillonnage non-redondant peut être employé pour analyser des espaces de recherche pour des problèmes combinatoires susceptibles à l'échantillonnage statistique, y inclus virtuellement tous problèmes solvables par la programmation dynamique. Les principes d'échantillonnage non-redondant sont robustes et typiquement faciles à implémenter, comme démontré par l'inclusion d'échantillonnage non-redondant dans les versions récentes de Vienna package populaire.

Title : Non-redundant sampling in RNA bioiformatics

Keywords : RNA, dynamic programming, non-redundant sampling, combinatorial algorithms,thermodynamic model, kinetics

Abstract : Sampling methods are central to many algorithmic methods in structural RNA bioinformatics, where they are routinely used to identify important structural models, provide summarized pictures of the folding landscapes, or approximate quantities of interest at the thermodynamic equilibrium. In all of these examples, redundancy within sampled sets is uninformative and computationally wasteful, limiting the scope of application of existing methods. In this thesis, we introduce the concept of non-redundant sampling, and explore its applications and consequences in RNA bioinformatics.
We begin by formally introducing the concept of non-redundant sampling and demonstrate that any algorithm sampling in Boltzmann distribution can be modified into non-redundant variant. Its implementation relies on a specific data structure and a modification of the stochastic backtrack to return the set of unique structures, with the same complexity.
We then show a practical example by implementing the non-redundant principle into a combinatorial algorithm that samples locally optimal structures. We use this tool to study the RNA kinetics by modeling the folding landscapes generated from sets of locally optimal structures. These structures act as kinetic traps, influencing the outcome of the RNA kinetics, thus making their presence crucial. Empirical results show that the landscapes generated from the non-redundant samples are closer to the reality than those obtained by classic approaches.
We follow by addressing the problem of the efficient computation of the statistical estimates from non-redundant sampling sets. The absence of redundancy means that the naive estimator, obtained by averaging quantities observed in a sample, is erroneous. However we establish a non-trivial unbiased estimator specific to a set of unique Boltzmann distributed secondary structures. We show that the non-redundant sampling estimator performs better than the naive counterpart in most cases, specifically where most of the search space is covered by the sampling.
Finally, we introduce a sampling algorithm, along with its non-redundant counterpart, for secondary structures featuring simple-type pseudoknots. Pseudoknots are typically omitted due to complexity reasons, yet many of them have biological relevance. We begin by proposing a dynamic programming scheme that allows to enumerate all recursive pseudoknots consisting of two crossing helices, possibly containing unpaired bases. This scheme generalizes the one proposed by Reeders and Giegerich, chosen for its low time and space complexities. We then explain how to adapt this decomposition into a statistical sampling algorithm for simple pseudoknots. We then present preliminary results, and discuss about extensions of the non-redundant principle in this context.
The work presented in this thesis not only opens the door towards kinetics analysis for longer RNA sequences, but also more detailed structural analysis of RNAs in general. Non-redundant sampling can be applied to analyze search spaces for combinatorial problems amenable to statistical sampling, including virtually any problem solved by dynamic programming. Non-redundant sampling principles are robust and typically easy to implement, as demonstrated by the inclusion of non-redundant sampling in recent versions of the popular Vienna package.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France