# RNAVLAB 2.0: COMBINING WEB APPLICATIONS, GRID COMPUTING, AND DYNAMIC PROGRAMMING TO OVERCOME RESOURCE LIMITATIONS IN RNA SECONDARY STRUCTURE ANALYSIS

by

Abel Licon

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Spring 2010

# RNAVLAB 2.0: COMBINING WEB APPLICATIONS, GRID COMPUTING, AND DYNAMIC PROGRAMMING TO OVERCOME RESOURCE LIMITATIONS IN RNA SECONDARY STRUCTURE ANALYSIS

by

Abel Licon

Approved: _____

Michela Taufer, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____

B. David Saunders, Ph.D.
Professor and Chair Department of Computer and Information Sciences

Approved: _____

George H. Watson, Ph.D.
Dean of College of Arts and Sciences

Approved: _____

Debra Hess Norris, M.S.
Vice Provost for Graduate and Professional Education

# ACKNOWLEDGEMENTS

I want to thank my advisor Dr. Michela Taufer for seeing the potential in me so early in my academic career and for the continuing guidance and support that has been imperative to my success. I would like to thank Dr. Ming-Ying Leung and Dr. Kyle Johnson for all their support and collaboration through the RNAVLab project and all the GCLab members past and present for all their help. I would like to thank all my committee members for taking the time out of their busy schedule to read my thesis and provide me with their valuable feedback.

I would also like to thank my Dad, Salvador Licon Jr., for showing me what it means to be a good father and husband, my Mother, Rosa Licon, for keeping me humble and kind-hearted, and my brother, Salvador Licon III, for being my best friend.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

As ribonucleic acid (RNA) molecules play important roles in many biological processes including gene expression and regulation, their secondary structures have been the focus of many recent studies. Despite the computing power of supercomputers, computationally predicting secondary structures with thermodynamic methods is still not feasible when the RNA molecules have long nucleotide sequences and include complex motifs such as pseudoknots. Furthermore, there is no consolidated environment for access to the several available prediction and analysis tools.

In this thesis we address this problem by extending a virtual laboratory for studying RNA secondary structures, called RNA Virtual Laboratory (RNAVLab 2.0), with a Web application that allows scientists to easily and effectively access a set of heterogeneous tools for the study of secondary structures supported by heterogeneous computational resources. We design a dynamic programming algorithm for finding the optimal, non-overlapping segmentation of a long RNA sequence into segments (chunks) given a scoring function based on energy values. We integrate our algorithm into RNAVLab 2.0 to enable the prediction of the chunks independently and the generation of a complete secondary structure prediction from the combined local energy minima. We measure the prediction accuracy for the 14 longest sequences in Group A in CONTRAfold using RNAVLab 2.0 and show that 12 times out of 14 our virtual environment outperforms other methods based on global energy minima, while in the other two cases it has similar accuracy results.

# Chapter 1

# INTRODUCTION

## 1.1 Problem Overview and Proposed Solution

The secondary structure of an RNA molecule is the collection of hydrogen bonds between the nucleotide bases. RNA secondary structures can be classified into two basic categories called stem-loops and pseudoknots. Both kinds of secondary structures on overlapping RNA viral genes have been implicated in important viral gene expression processes[17, 23, 33]. Today there is an active research community that brings together scientists with diverse training and expertise ranging from physics to computer science and biology, to develop, test, and apply computational methods to directly link RNA secondary structures in general and pseudoknots in particular to the RNA functionalities.

Despite the computing power of supercomputers, computationally predicting secondary structures with thermodynamic methods is still not feasible when the RNA molecules have long nucleotide sequences (i.e., longer than 1000 nucleotides) and include complex motifs such as pseudoknots. Furthermore, there is no consolidated environment for accessing the several available prediction and analysis codes. When dealing with the prediction and analysis of secondary structures, scientists have very few sources from which to retrieve information on the structures. To retrieve, sort, and elaborate pieces of information from these sources, scientists have to do significant handwork sorting, merging, and comparing results as well as extrapolating conclusions. For example, when studying pseudoknots secondary

structures, scientists need to access databases such as PseudoBase [31] that are not always provided with advanced search engines. Information from the database has to be copied and pasted into files with different format (e.g., FASTA) and this task is left to the scientists. Ultimately, the scientists have to submit the retrieved data to portals such as PseudoViewer [2] for the pseudoknots' visualization. Ideally, scientists should be led through the different steps by a unified, user-friendly portal that screens them from database issues as well as provides them with powerful tools for search, formatting, and visualization.

In a first attempt to address this problem, we built RNAVLAb 1.0 (RNA Virtual Laboratory) [26]. RNAVLab 1.0 was a Java desktop application that scientists can download and run on their local machine. RNAVLab 1.0 was a first prototype of a virtual laboratory to consolidate many powerful RNA secondary structure prediction and analysis tools. In this thesis we propose RNAVLab 2.0, an extension of the first prototype that aims to improve the type and quality of services provided by RNAVLab as well as the quality of secondary structures predictions.

## 1.2 Thesis's Contributions

The contribution of this thesis are threefold:

- We extend RNAVLab to include Web applications and Web services that allow scientists to easily access heterogeneous computational resources from a Condor pool. The management of the resources is done dynamically at runtime.

- We design a dynamic programming algorithm that runs in polynomial time and allows scientists to achieve the optimal, non-overlapping segmentation of a long RNA sequence into segments (chunks). The secondary structure of each chunk can be predicted independently, then combined with the structures predicted for the other chunks to generate a complete secondary structure prediction that is thus a combination of local energy minima.

- We integrate the algorithm into RNAVLab and use the extended virtual environment to predict secondary structures that could not otherwise be predicted due to lack of computing resources by limiting the size of the chunks while using energy as a scoring function. We show that these predictions are comparable in accuracy to those using a single whole sequence prediction.

RNAVLab 2.0 bridges the gap between existing tools and the scientists: a Web front-end provides a consolidated environment and a Java back-end supplies the scientists with the resources to use the available tools. The Web front-end is developed using the Ruby on Rails framework which provides a scaffold for quickly building modular and maintainable Web applications and Web services. The multi-threaded Java back-end handles the scheduling of prediction and analysis jobs submitted via the front-end. The two parts communicate over a TCP control connection and shared MySQL database. Our dynamic programming algorithm extends the tools in RNAVLab by allowing the prediction of the secondary structure of very long RNA sequences when computing or storage resources are limited (either because of hardware or prediction code capabilities). Our algorithm searches all possible non-overlapping primary sequence segmentations and maximizes a given scoring function.

### 1.3 Outline

Chapter 2 introduces the concept of RNA secondary structure predictions as well as describes the main algorithms that attempt to computationally predict them. Chapter 3 extensively describes RNAVLab 2.0, including an overview of the Ruby on Rails (RoR) Web application front-end and the Java back-end multi-threaded daemons which handle the scheduling of predictions, alignment, and comparisons as well as the communication with the RoR front-end. Chapter 4 describes the dynamic programming algorithm for optimally segmenting a long RNA primary sequence into

chunks. Chapter 5 evaluates the proposed dynamic programming algorithm once integrated in RNAVLab and compares the accuracy (in terms of sensitivity and selectivity) to more traditional prediction methods. Chapter 6 concludes this thesis with a summary and future work.

# Chapter 2

# BACKGROUND AND RELATED WORK

## 2.1 RNA Secondary Structure Prediction

Ribonucleic Acid (RNA) is made up of four types of nucleotide bases: adenine (A), cytosine (C), guanine (G), and uracil (U). A sequence of these bases is strung together to form a long, single-stranded RNA molecule. RNA molecules vary greatly in size, ranging from about twenty nucleotide bases in microRNAs to a long polymer of over 30,000 bases in complete viral genomes [29]. Among the four nucleotide bases, C and G form a complementary base pair by hydrogen bonding, as do A and U. Although an RNA molecule is a linear polymer, it tends to fold back on itself to form a three dimensional (3D) functional structure mostly by pairing between complementary bases. The 3D structure of an RNA molecule is often the key to its function. Because of the instability of RNA molecules, experimental determination of their precise 3D structures is a time consuming and rather costly process. However, useful information about the molecule can be gained from knowing its secondary structure that refers to the collection of hydrogen bonded base pairs in the molecule. Essentially, all RNA secondary structures are made up of elements that can be classified into two basic categories: stem-loops and pseudoknots (see Figure 2.1). Both kinds of secondary structure elements have been implicated in important biological processes like gene expression and regulation [17, 23, 33].

Since finding the structures of RNA in the wet lab is prohibitively expensive and time consuming, much research has been done to predict these structures computationally. Different computational methods have been integrated in prediction

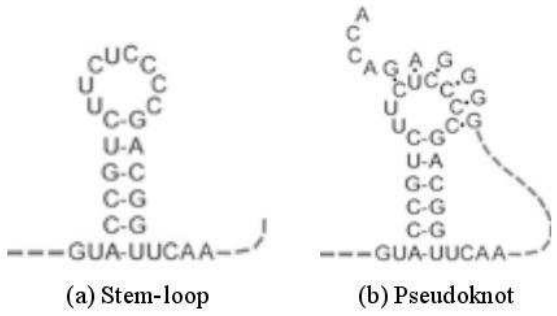



**Figure 2.1:** *Stem-loops and Pseudoknots*

codes from methods based on free energy minimization (MFE) to methods based on consensus structures and machine learning techniques. Existing prediction codes are normally based on one or more of these methods and can support additional features such as the prediction of pseudoknots, parallel prediction, and calculating the partition function. Despite the significant community effort in the past three decades and the increasing computing power available to scientists for their simulations, we cannot accurately predict the secondary structure of large RNA sequences ($>$ 300 nucleotides) efficiently. Figure 2.2 summarizes the main prediction codes and classifies them in terms of their prediction strategies (i.e., Minimization Free Energy, Consensus or Machine Learning Techniques) and additional features (i.e., including pseudoknots, parallelization, and calculating the partition function). The codes are sorted chronologically.

### 2.1.1 Predictions based on Minimization Free Energy

The development of mathematical models and computational prediction algorithms based on free energy minimizations (also called MFE) for stem-loop structures started in the late '70s to early '80s [15, 35]. One of the first attempts to predict RNA secondary structures computationally was done by **Nussinov** and used a dynamic programming algorithm to maximize the amount of base pairs in a fold [15]. The algorithm searches the space of possible bindings and gives back the

structure with the maximum amount of pairs. Nussinov's algorithm was extended in **Mfold** [35] to include the energy contributions found in wet lab experiments for a variety of short secondary structures [30]. Given a particular structure, a bond was weighted according to the energy that it can take to break that bond.

Despite the computing power of supercomputers and emerging advanced technologies, e.g., multi-core architectures, the prediction of secondary structures of long RNA sequences (on the order of thousands of nucleotides) based on thermodynamic methods is still not feasible, especially if the structures include complex secondary structures such as pseudoknots. The time and space required for accurate predictions of pseudoknots based on energy minimizations grow very rapidly with the sequence length. Figure 2.3 shows the time and memory (in logarithmic scale) allocated for the prediction of RNA pseudoknots with various lengths using one of the most accurate prediction programs, **PknotsRE**[20]. **PknotsRE**'s algorithm has a runtime and memory demand in the order of $N^6$ and $N^4$, respectively, where N is the length of the input sequence. The program conducts an exhaustive search for the optimal structure with the lowest free energy and has the capability to predict rather complex structures, even some non-planar structures, for short RNA segments up to 200 nucleotides in length on a commodity desktop PC. To overcome the tremendous demand in computing resources needed for pseudoknot prediction, alternative algorithms have been proposed (e.g., **PknotsRG** [19] and **NuPack** [18]) that restrict the types of pseudoknots to keep runtime and memory size under control. For instance, **PknotsRG** [19] limits the types of pseudoknots to simpler structures for longer segments, up to 800 nucleotides. However, a large variety of pseudoknots have been shown to occur in nature [31]. Their omission from computational methods may significantly affect the prediction accuracy.

Parallel implementations of MFE algorithms aims to address the computational complexity and reduce it by using advanced parallelization techniques and

parallel libraries. In **GTFold**, the energy calculations of loops are parallelized using OpenMP [12]. GTFold uses the same MFE algorithm as MFold but at the same time obtains two orders of magnitude speedup over these two codes with comparable accuracy thanks to the parallelization. Like many other methods, GTFold approach does not consider pseudoknots because they introduce even more complex data dependencies than non-pseudoknotted approaches.

MFE dynamic programming algorithms have been extended beyond the single predictions to find the partition function of all the possible structures of a given sequence in **McCaskill** [13] and **NuPack** [18]. The partition function gives the probability that any pair of bases will bind and thus can be used to find the most probable structure. In particular, **NuPack** extends this model to include pseudoknots but has a high running time of $O(N^4)$ [18].

### 2.2 Predictions based on Consensus Structures

Software packages for RNA secondary structure predictions based on homology search benefit from RNA sequence and secondary structure alignments. In the **Sankoff** algorithm, two or more sequences are folded and aligned simultaneously [21]. This can be thought of as combining the Waterman and Smith alignment algorithm [22] and Nussinovs maximal base pair matching algorithm [15]. Although mathematically optimal, the running time of this approach is prohibitively expensive, i.e., $O(N^3M)$ with N being the length of the sequence and M being the number of sequences. Heuristic approaches such as in the **DynAlign** code reduce the search space to tradeoff thoroughness for performance [11]. Again, none of these codes includes pseudoknots. This approach is more interesting when trying to find a consensus structure in two functionally related sequences.

### 2.2.1 Predictions based on Machine Learning Techniques

Machine learning techniques have been extensively used in bioinformatics for problems ranging from gene finding [9] using Hidden Markov Models to protein structure prediction [16] using Genetic Algorithms (GAs). Thus, it is no surprise that these techniques have been also applied to the problem of RNA secondary structure prediction. Since the prediction problem can be seen as an optimization problem, GAs can be used to search the space of possible structures using energy functions as fitness functions as in e.g., **STAR** [6]. Using a genetic algorithm approach, **STAR** searches the space of secondary structures for the minimum free energy structure. Instead of exhaustively searching the space of secondary structures like **MFold** does, STAR uses an evolutionary algorithm to obtain a structure with the minimum energy. Although the structure is not guaranteed to be the absolute minimum since GAs only search a sub-set of the space, it was found that the accuracy was comparable to that of exhaustive searches. The main advantage of using the GA technique is that it can be easily parallelized, and although that does not help dramatically with non-pseudoknotted structures predictions, which have a relatively low degree polynomial running time, it can be extremely useful when applied to higher order polynomial algorithms that include pseudoknots.

As an alternative to thermodynamic methods for RNA secondary structure prediction, Stochastic Context Free Grammars (**SCFG**) have been proposed for secondary structure prediction [8]. These approaches rely on estimating probability distributions over a set of transformation rules that define how the fold is formed. **SCFGs** have the ability to learn the parameters of a generative model by observing a set of sequences with their corresponding secondary structures. In general, **SCFG**s are outperformed by physics based approaches, although recently **ContraFold** proposed a generalization of **SCFG**s where a flexible and richer feature set allows the inclision of free energy parameters more akin to thermodynamic models

[3]. However, the complexity of the RNA secondary structures predicted by these methods is restricted by the expressibility of their grammars, thus highly complex structures, such as pseudoknots, cannot be predicted by **SCFG**s.

### 2.2.2 Limitations of Current Implementations

There are several limiting factors common to many of the codes in Figure 2.2. First of all, most of the research does not include pseudoknotted structures. Predicting pseudoknots increases the complexity of the computational problem. Moreover, although these programs are accurate for sequences a few hundred bases in length, their accuracies diminish for longer sequences due to a lack of experimental energy results for long RNA sequences that can be used for tuning the computational algorithm [5]. In general, the accuracy of predicting pseudoknotted structures is even lower than that of non-pseudoknotted structures. Finally, most of the algorithms are difficult to parallelize and thus to run faster on available parallel and distributed computer systems. For instance, MFE approaches use a dynamic programming algorithm that has many different recurrence relations and are not as easily parallelizable as other dynamic programming algorithms, especially when considering pseudoknotted structures [24]. Although there are approaches to parallelize these algorithms [12], they do not scale well and do not include pseudoknots. It is clear that there is still much work to do in RNA secondary structure predictions and in coming up with methods to parallelize the existing algorithms and improve their accuracies.

## 2.3 Computational Environments for Prediction and Analysis

When dealing with RNA secondary structures, scientists have several sources of data (databases) and tools available. To retrieve pieces of information from these sources as well as to sort and elaborate the data with these tools, scientists have to do significant hand-work by sorting, computing, merging, and comparing results

as well as extrapolating conclusions. For instance, when dealing with pseudoknots, scientists need to access databases such as PseudoBase [31], which are not always provided with advanced search engines. The data from the database has to be copied into files of different format (e.g., FASTA and EMBL).

When it is time to predict secondary structures, the scientists have to download and compile prediction codes on their platforms; the platforms do not always include the required libraries and do not always fully support the code execution. Even if portable, some of these codes require significant amount of computing power that is not always available on the scientists PC. An alternative is to submit the data to portals that provide prediction and visualization functions. However, most portals provide only prediction functions using a single code. If the scientists want to compare predictions using different codes, they are forced to navigate multiple portals with different interfaces and submission policies. Ideally, scientists should have a consolidated, easy-to-use environment that provides them with powerful tools for prediction, analysis, and visualization.

Figure 2.2 provides an overview of the existing environments for prediction and analysis; the figure classifies the environments based on features such as source availability, visualization capability, batch processing integration, storage capability, inclusion of multiple prediction codes, integration of prediction tools, Web service capability, and alignment capability.

### 2.3.1 Classification Features

The following is a description of the features used for classifying existing computational environments in Figure 2.4.

- **Source Availability-** The access to the source code allows scientists to further extend and change the methods in existing prediction codes. Not all the prediction codes are open-source.

- **Visualization-** Computational methods for prediction treat the RNA primary sequence and secondary structure as a sequence of characters. Although this type of representation makes it easy for a program to manipulate the data, it is not very easy to understand and interpret for the scientists. Tool that provides predictions of secondary structures should also provide some kind of easy to understand and interpret visualization capability. A popular representation is to portray the sequence in an image that shows the sequence and all the base pairs in a 2-D space.

- **Batch Processing-** Scientists frequently are not just studying a single sequence but a family of sequences or a set of families. The ability to submit more than one sequence prediction at a time with the same parameters is essential in improving the workflow and overall usability of the tools.

- **User Database-** The analysis of a sequence is not over when the structure is predicted. Many scientists study the same sequences for several months and from different prospectives while they wait for wet-lab results. If an online user database is not provided, scientists are forced to either save predictions locally in their own databases (most likely their file system that often is not backed up) or waste resources by re-predicting the structure over and over. With an online user database, scientists can come back at a later time and add additional sequences and make additional analysis on the saved predictions, again increasing productivity and keeping unnecessary re-predictions to a minimum.

- **Multiple Prediction Codes-** It is useful to have several predictions codes that incorporate different prediction strategies available in the same Web application. Each technique has its own strengths and weaknesses. When multiple options are available, scientists are able to exploit the strengths with a

given data set, as well as benchmark the codes given a set of sequences with experimentally found structures.

- **Comparison Tools-** It is very useful to quantify the similarity of secondary structures across sequences. This capability can give more insights into the functional similarity of RNA sequences than its primary sequences, given that the structure tends to be more conserved than the actual sequence. Additionally, comparison tools can be used to compare the performance of different predictions tools.

- **Web Service-** Although Web portals are very useful for quick analysis of a few sequences, they are not very efficient for the automatic prediction of very large sets of sequences. On the other hand, Web services allow programmers to create their own prediction applications, either desktop or other Web applications, that can take advantage of the particular Web services without having to use a browser. This is essential for high throughput and automation.

- **Alignment-** Many programs align RNA primary sequences. Very few environments consider the alignment of secondary structures even if it has been observed that such alignments can provide scientists with additional insights on the RNA segment functions.

### 2.3.2 Overview of the Web Portals

The idea behind a consolidated computational environment is to allow scientists to use a single tool for submitting a sequence or sequences, notifying them via e-mail or the browser when the prediction is done, and analyzing the results to reach accurate conclusions. Unfortunately none of the existing environments provide scientists with such a rich, unified environment.

The **Vienna Package** [7] in Figure 2.4 is one of the most well known packages for prediction and analysis. It consists of a C library and several stand-alone

programs for the prediction, comparison, and alignment of RNA secondary structures. The stand-alone programs are not integrated in an unified environment and do not address multiple prediction approaches but instead deploy the Zuker and McCaskill [13, 35] algorithms. Thus, the Vienna Package does not include the prediction and analysis of pseudoknots. Lastly, the package does not integrate grid technology and advanced visualization. In **DynAlign** [11], there is no Web portal to submit sequences online. The only way to access the program is to download the code and compile it for a specific machine. This is one of the main problems we solved in RNAVLab.

**PknotsRG** [19] does provide many features for prediction pseudoknots including batch processing and multiple prediction codes. However, it still lacks the feature of allowing the scientists to save their structures online and to align them. These two features are very important for scientists. **NuPack** [18] has a user friendly Web Interface for using its several software packages. It also adds partition and visualization features. Still, it lacks Web services that allow more advanced users to automate prediction submissions.

**ContraFold** [3] provides the most basic functionality of single sequence predictions and visualization. Although it has been shown to be more accurate than most simpler SCFGs and MFE approaches, without the appropriate features, users may choose to use a less accurate program based on the extensive features. **PknotsRE** [20] is only available by downloading the source and compiling from scratch. Although this is the most compute and memory intensive program, no resources are provided with it (e.g., interfaces to grid resources) and users have to rely on their own resources to get the predictions. Probably the most well known Web application is **MFold/UNAFold** [34] by Zuker, one of the pioneers in RNA secondary structure prediction. Although it has been shown that several other programs are

more accurate and can explore a wider range of secondary structures (e.g., pseudoknots) than MFold, because of its popularity and Web application, it is one of the most used programs. RNAVLab aims to bring the most powerful applications in one place and let the scientists decide what application best fits their needs.

**GTFold**, the parallel implementation of the MFold algorithm, does not have a Web application for online submission [12]. It requires the user to download and compile the application from source. The code requires OpenMP support on the machine on which it is executed. This may overwhelm the user who can ultimately decide to take a easier path by using traditional Web portals. In the case of **STAR** [6], there is no site available for the download of the source and binaries or even a Web portal, making it difficult to obtain. Authors of papers that compare their implementation accuracy and performance versus STAR usually asked for the source directly from the authors.

Although not a prediction algorithm, PseudoViewer is listed in Figure 2.5 because it is very useful for fast visualizations of RNA secondary structures with and without pseudoknots [2]. PseudoViewer also provides scientists with a Web service that allows other prediction sites as well as application developers to take advantage of the auto-generated pictures in their own applications.

Although existing portals are normally very useful, none include all the features listed in Section 2.4 and thus do have some limitations. For instance, only PknotsRG, MFold, NuPack, Vienna, and ContraFold have online Web portals. All allow the scientists to submit one sequence for prediction and each has a max length in nucleotides for the sequence that can be submitted and predicted. To sum up, while they all provide some of these useful features; none of them provide a consolidated environment that incorporate all of the features required by a scientist.

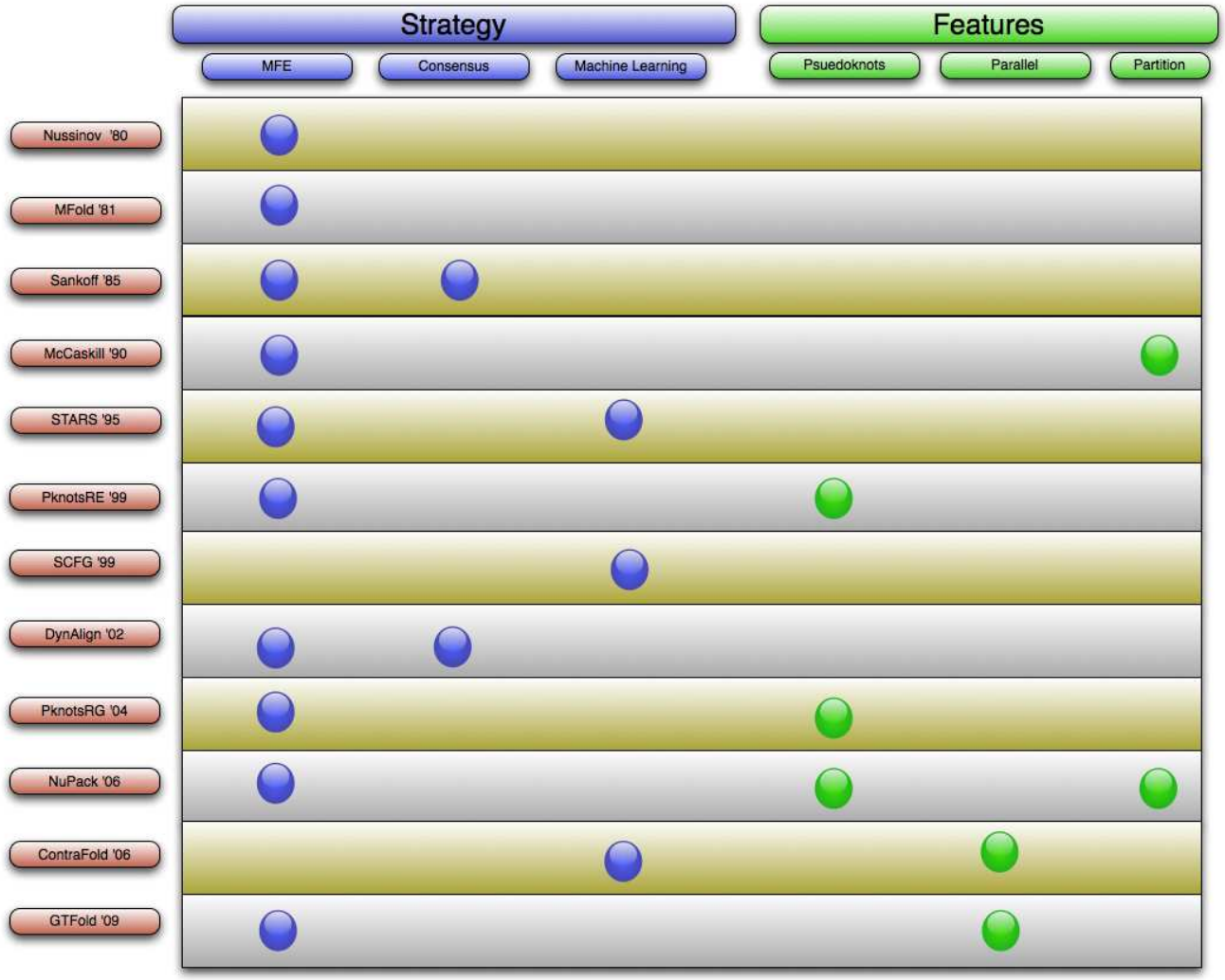



**Figure 2.2:** *Chronology of Prediction Codes*

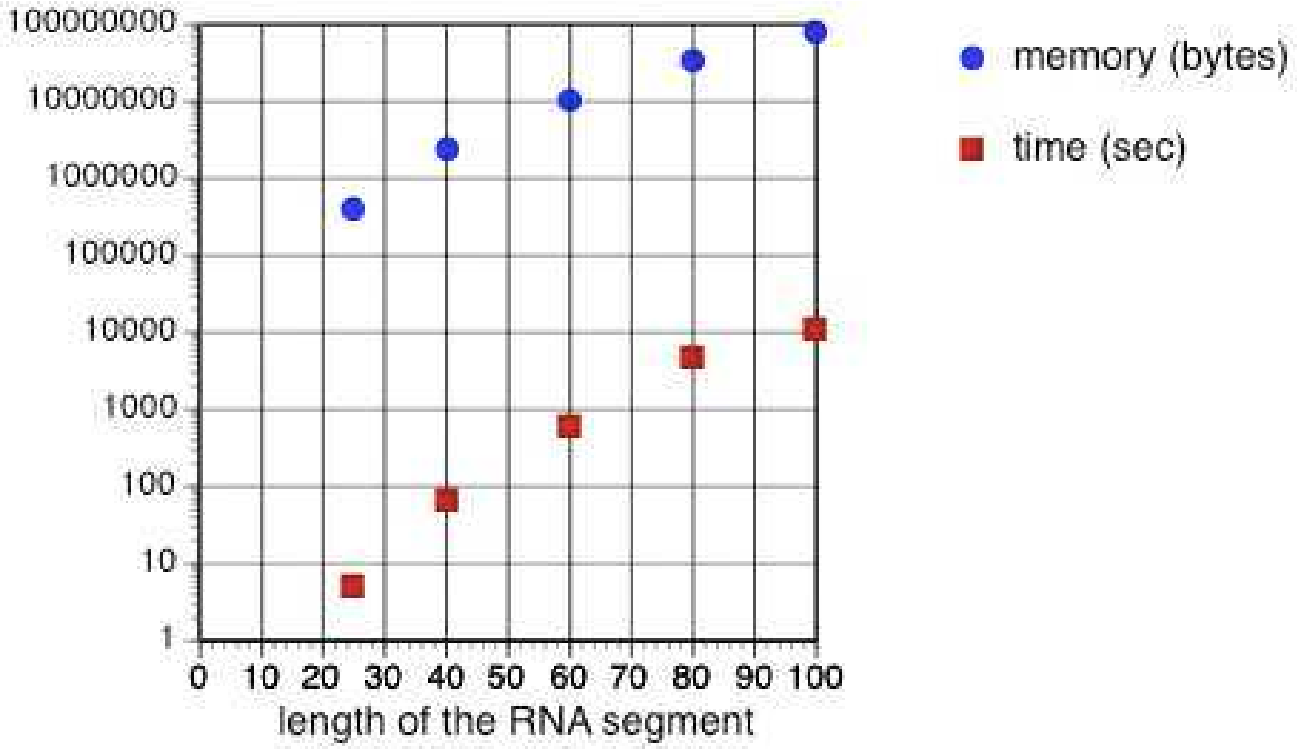



**Figure 2.3:** *Running time and memory consumption of PknotsRE with respect to input sequence length.*

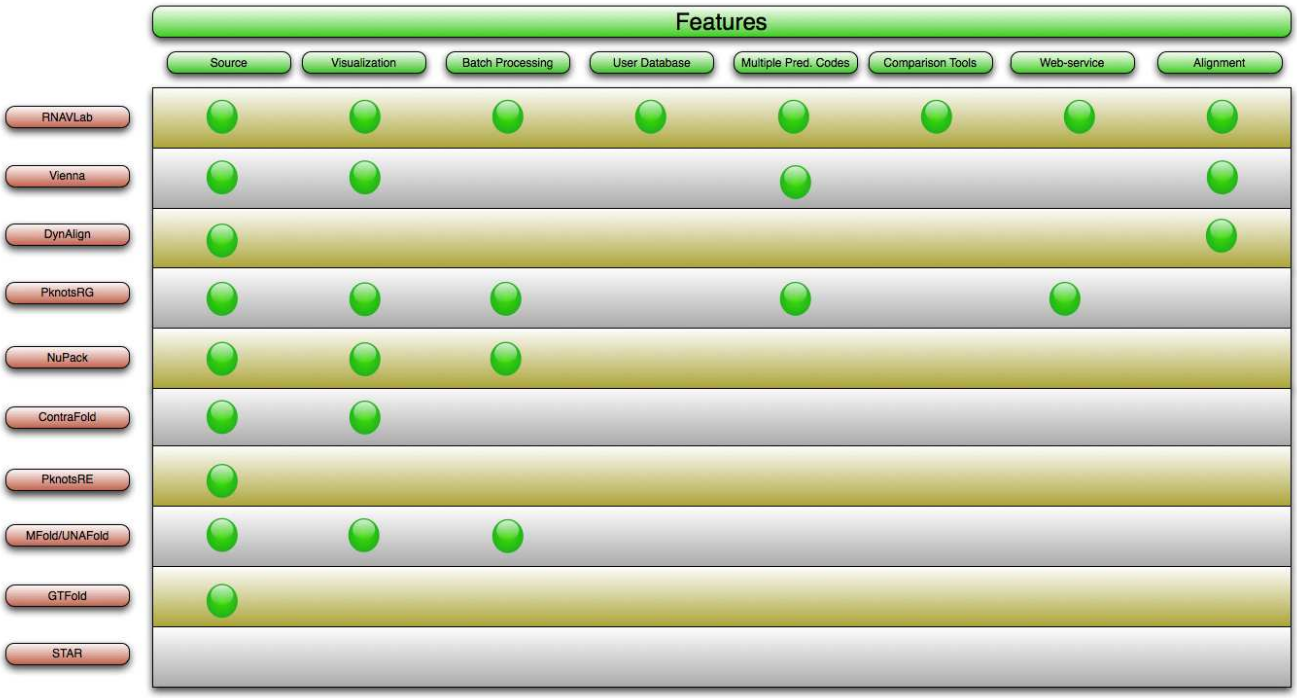



**Figure 2.4:** *Characterization of existing Web portals.*

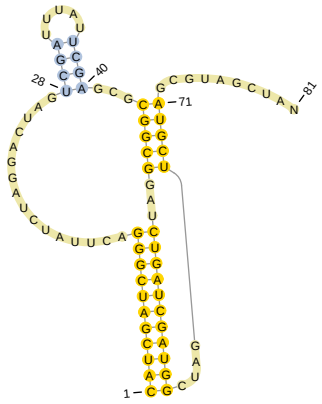


**Figure 2.5:** *Example of a pseudoknot shown via the PseudoViewer application.*

# Chapter 3

# RNAVLAB

## 3.1 From RNAVLab 1.0 to RNAVLab 2.0

In previous work, we built RNAVLab 1.0 [27] to facilitate the prediction and analysis of RNA secondary structures by providing several tools and resources in a uniform environment. To address the limitations presented in Chapter 2, we extended RNAVLab 1.0 from a simple desktop application to a complete, unified framework for the prediction and analysis of RNA secondary structures, called RNAVLab 2.0.

### 3.1.1 RNAVLab 1.0

RNAVLab 1.0 is a Java desktop application. Predictions are performed either locally on the desktop or on a heterogeneous pool of resources using Condor statically configured by reading setting files prepared by the user [26]. In order to predict long RNA structures, a sliding window with a fixed length and step size is used to sample the sequence into chunks. These chunks are sent to the Condor pool for prediction. The predictions are recombined into a single long structure using a rebuilding algorithm based on structural motifs. In Figure 3.1, we show the layout of RNAVLab 1.0. RNAVLab 1.0 had a modular structure including three major components:

- A segment sampler component to sample RNA sequence segments; a sliding window with a fixed length and step size is used to sample the sequence into chunks.

- A structure prediction component to predict the structures of the sampled segments using different programs on heterogeneous computers, i.e., PknotsRE [20] and PknotsRG [19].

- A structure analysis component to compare and contrast predictions with observed structures as well as identify similarities and other characteristics across predictions.

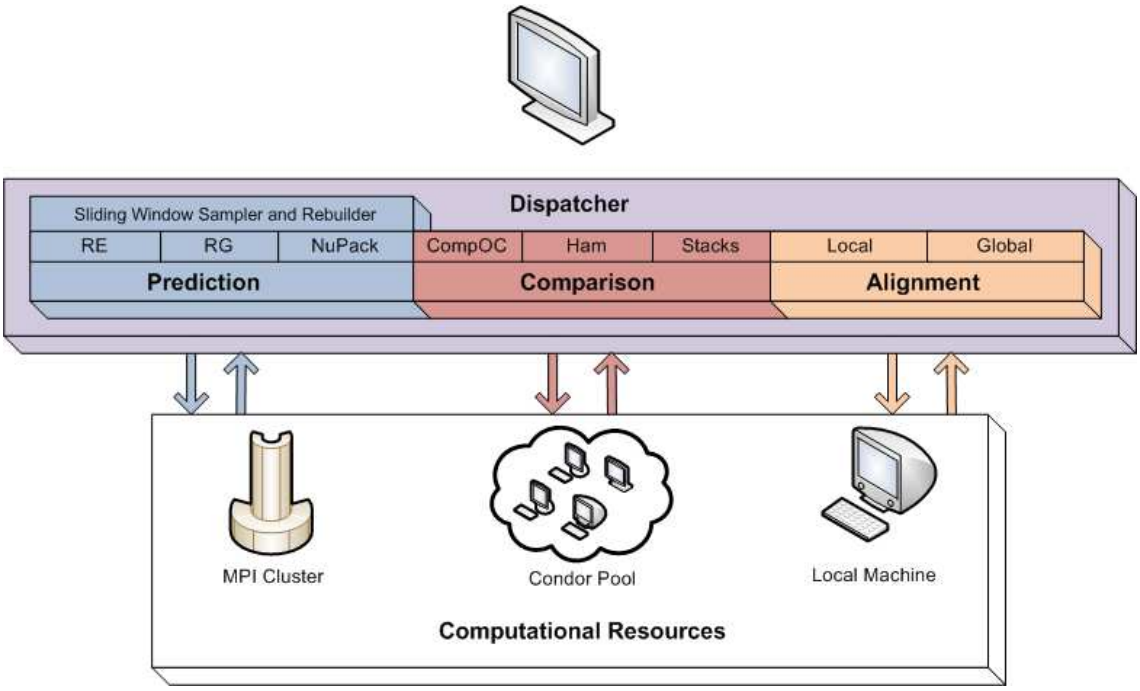



**Figure 3.1:** *RNAVLab 1.0 before improvements*

### 3.1.2 RNAVLab 2.0

In this thesis, we extended RNAVLab 1.0 as follows:

- We built a Ruby on Rails (RoR) Web application front-end including browser and Web service interfaces for scientists to access the RNAVLab 2.0 tools over the Internet (Section 3.2).

- We extended the Java back-end to a multi-threaded daemon that communicates with the front-end via a shared MySQL user database and a TCP control connection (Section 3.3).

- We replaced the fixed sliding window sampler and rebuilder with a new optimal segmentation algorithm based on dynamic programming (Chapter 4).

Figure 3.2 shows the layout of RNAVLab 2.0 with the new and old features.

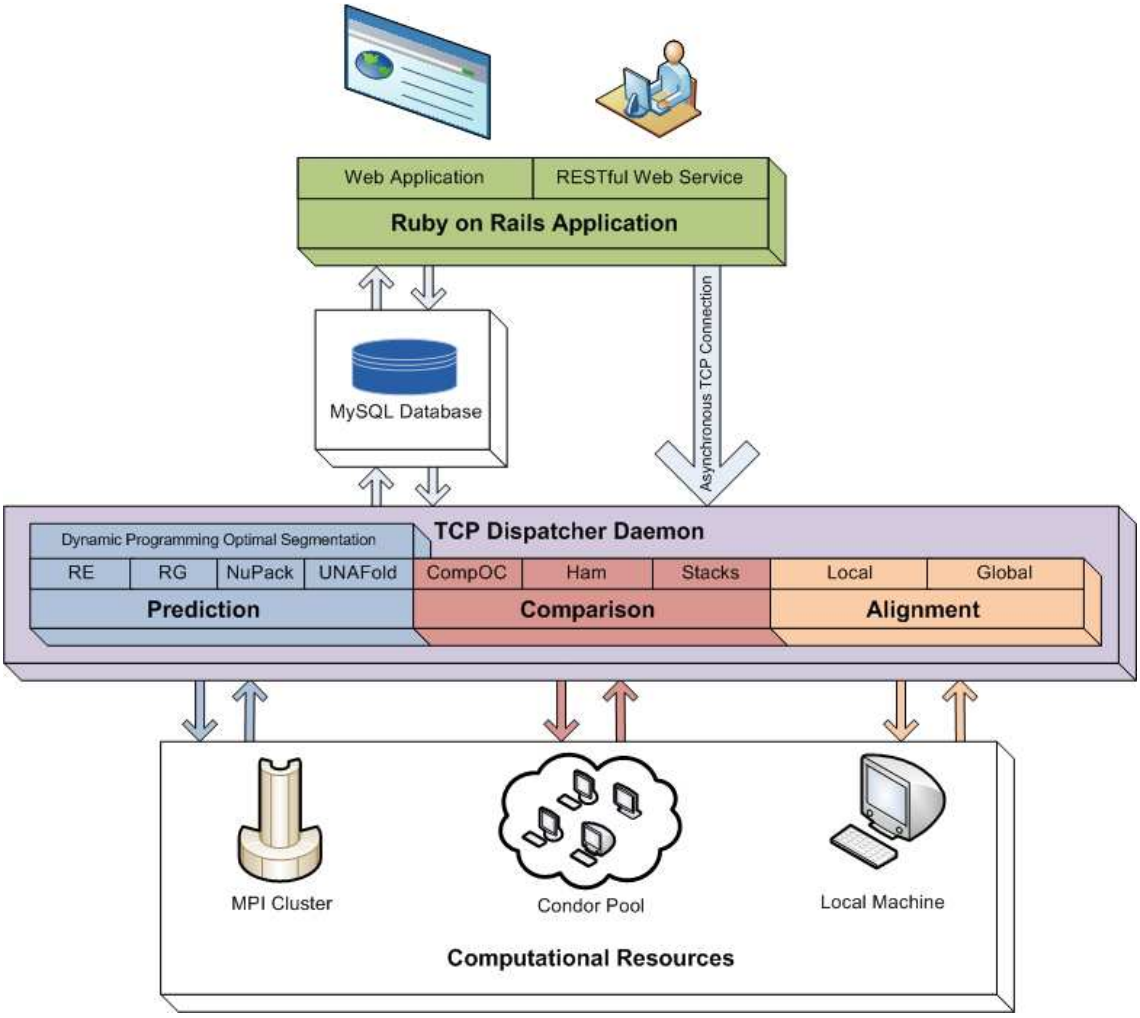



**Figure 3.2:** *RNAVLab 2.0 integrating the new front-end and back-end features.*

## 3.2 RoR Web Application Front-end

The Web application front-end in RNAVLab 2.0 is built using the Ruby on Rails (RoR) framework. The RoR framework is based on the Model-View-Controller (MVC) paradigm, a paradigm in which the domain-specific data (the model of the data) is separated from its presentation (the view of the data) and from its control (the logic that dictates the control flow). This separation allows for changes in the model without affecting the view or the control and vice versa. It also results in applications that are easier to modify, maintain and extend.

The Web application front-end and its Web service allow scientists to access the RNAVLab back-end capabilities through the Internet or with a browser. Our approach addresses and solves portability issues. The web application includes three main parts:

- The Web interface that users interact with when using a browser
- The Web services that are interfaced with via XML requests
- The actual RoR application that services the pages and handles communication with the dispatcher daemon

### 3.2.1 Web Interface

The RNAVLab 2.0 Web application provides an easy-to-use, consolidated environment for the prediction and analysis of RNA secondary structures. The main pages of the site (Figure 3.3) represents the main tools used in RNA prediction and analysis as well as a way for the users to save their predictions on an online user database hosted at the University of Texas at El Paso (UTEP).

Other key pages of RNAVLab 2.0 are: the home and sign in page, the page for submission of new sequences, the page for prediction of structures, the page for comparisons and analysis of predictions.

- **Home and Sign In-** In this page (Figure 3.3), the users are introduced into the RNAVLab project and are given several options for RNA structure analysis. This includes a non-login option for user who need a quick prediction or want to try RNAVLab 2.0. It also includes the possibility to create a login account (Figure 3.4). Once a user has an account, she is allowed to access the database, store her predictions, and create a virtual work laboratory for working with her sequences of interest.


**Figure 3.3:** *Screen shot of Home Page.*

- **New Sequences**- In this page (Figure 3.5), users are able to upload one or many sequences to their online virtual laboratory. The supported formats are FASTA and BPSEQ. These sequences can be later selected for prediction and analysis using the various RNAVLab tools.

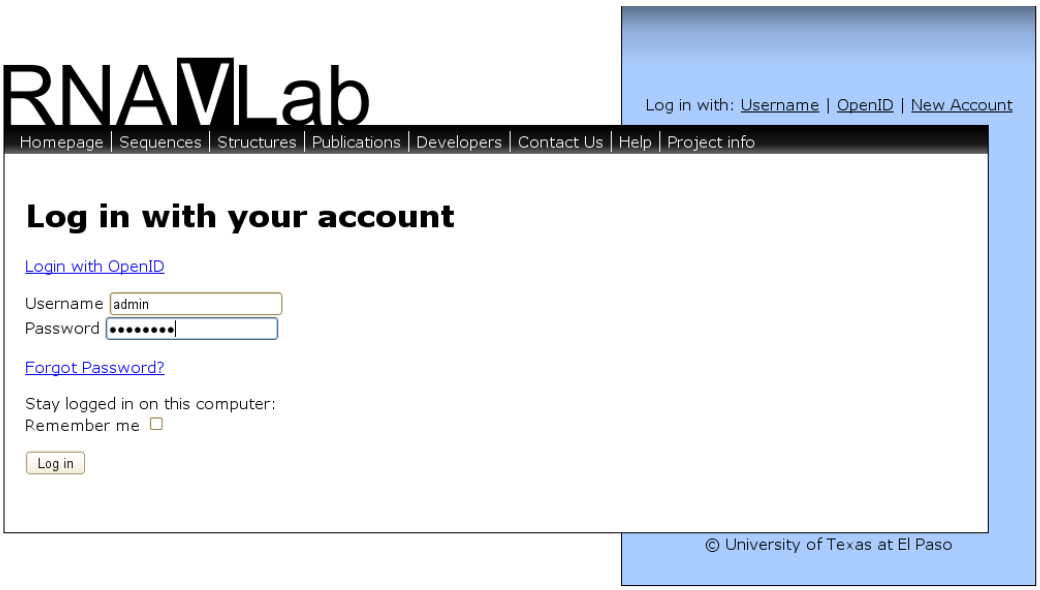



**Figure 3.4:** *Screen-shot of Sign-in page.*

- **New Predictions and Structures-** The new prediction page, shown in Figure 3.6, allows the user to predict secondary structures using the several prediction tools available. Prediction tools can work on the nucleotide sequence as a whole (if the computation is within the resource constraints) or can use the the optimal segmentation algorithm presented in Chapter 4 to segment the sequence in shorter chunks and predict the single long structure. The structure page uses the Pseudoviewer web-service to show the resulting predicted structure as shown in Figure 3.7.

- **Alignment and Comparisons-** This page allows the user to compare a given secondary structure to other structures in her virtual laboratory, as shown in Figure 3.8. This allow users to choose from the three comparison tools available (Section 3.3.3.1) as well as the two alignment tools available(Section 3.3.2). The page orders the structures by similarity, with the most similar first.

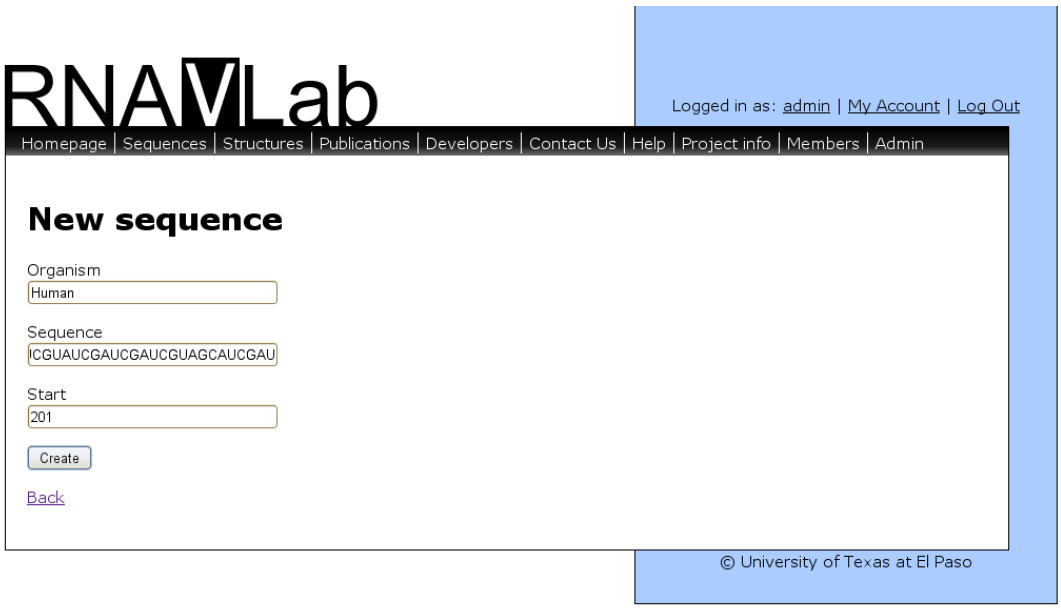



**Figure 3.5:** *Screen-shot of sequence page where you can create a new sequence.*

### 3.2.2 Web Service

When creating a project in Ruby on Rails by following the RESTful (Representational State Transfer) protocol guidelines [4], the application is automatically available as a RESTful Web service. For a site to be RESTful, it has to follow certain criteria:

- The site has to be client-server based.
- The server has to contain information about the client.
- The site has a uniform way to manipulate and retrieve data on the server.
- All the information necessary to manipulate data is contained in an individual request.

HTTP (Hypertext Transfer Protocol) meets these criteria. Thus, if we map the HTTP request types to the actions provided in the controller for a given resource,

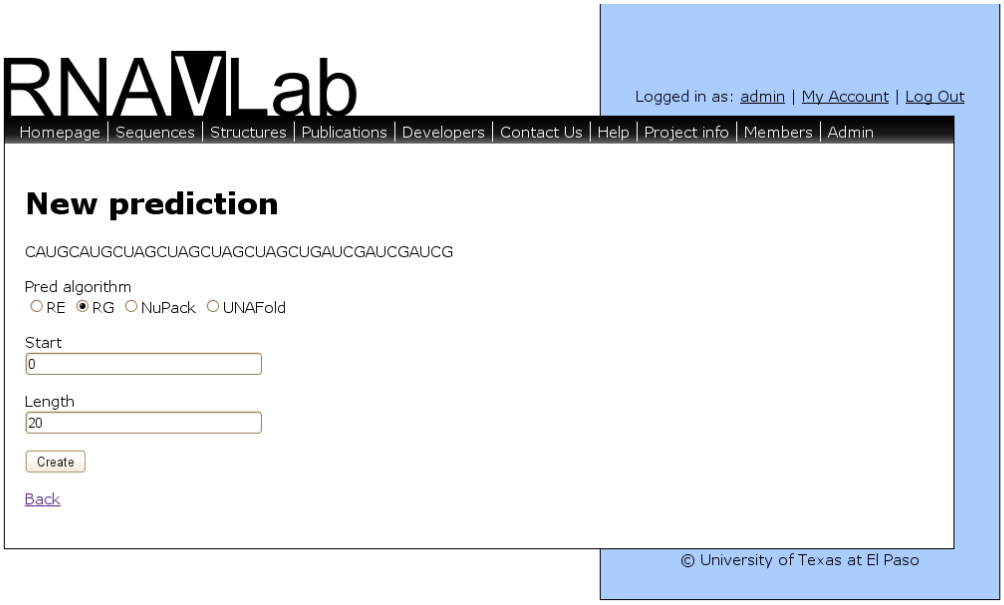



**Figure 3.6:** *Screen-shot of prediction page where you can submit a new prediction.*

we can easily supply a RESTful Web service. In RNAVLab, we do that with the CRUD (create, read, update, and delete) actions. All the resources available in RNAVLab have the CRUD actions available for the application developer to access. For instance, given a sequence resource with id = 22, which represents a single RNA primary sequence, and the data associated with it, there is a mapping for each of the following HTTP request types and URLs to Ruby on Rails actions in the controller:

- A **Get** request to http://rnavlab.utep.edu:/sequences/22 retrieves a single sequence and all the information about it. This corresponds to the read action.
- A **Post** request to http://rnavlab.utep.edu/sequences creates a new instance of a sequence in the user database. This corresponds to the create action.
- A **Put** request to http://rnavlab.utep.edu/sequences/22 update an existing instance of a sequence in the user database. This corresponds to the update action.

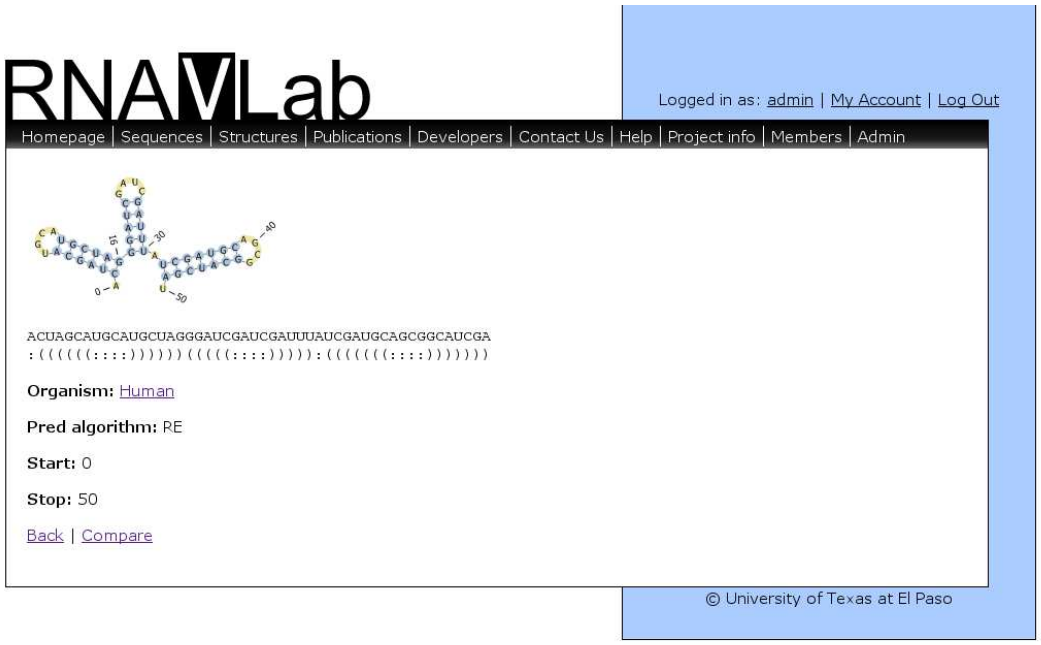



**Figure 3.7:** *Screen-shot of structures page where you can view your predicted structure.*

- A **Delete** request to http://rnavlab.utep.edu/sequences/22 removes the corresponding sequence record in the user database. This corresponds to the delete action.

When the user creates a new application, she creates a program that connects to the site, makes the appropriate HTTP request to the corresponding URL, and parses the XML response. The communication is handled entirely in HTTP, so there is no need for an additional protocol like XML/RPC or SOAP.

Providing a RESTful Web service in this manner via RNAVLab has several advantages:

- The server does not have to keep track of the client state, making the site less complex and more scalable.

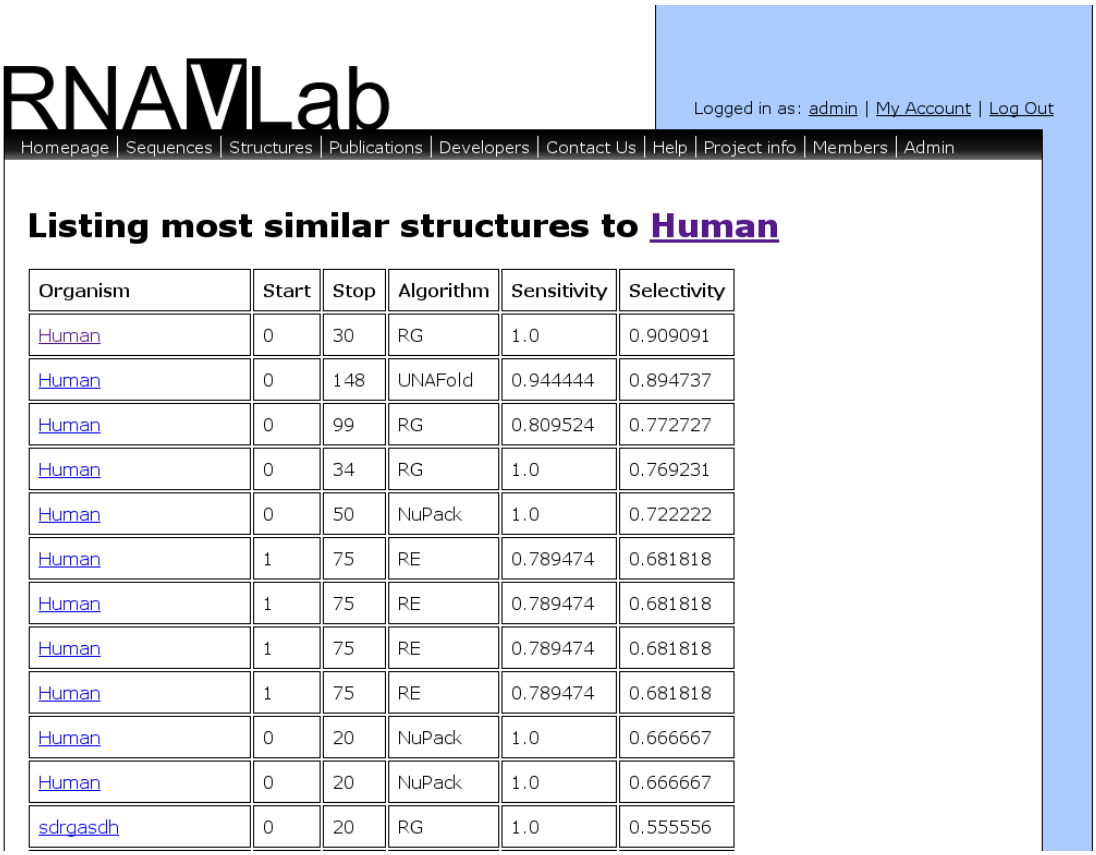



**Figure 3.8:** *Screen-shot of comparisons page where you can find similar predictions.*

- The site provides a uniform way to access and manipulate every resource. This means that an application developer can always expect the same set of actions for all the resources, making something such as a WSDL (Web Services Description Language) unnecessary.

- No additional protocols like XML/RPC or SOAP are needed to access the Web service.

### 3.3 Java Back-end

The Java back-end is in charge for the compute intensive tasks such as the RNA secondary structure predictions. It consists of a multi-threaded TCP daemon (dispatcher daemon) and several analysis tools. The tools have access to a pool

of heterogeneous resources at UTEP (a MPI cluster and a Condor pool). The communication between back-end and front-end takes place via a TCP connection and a shared MySQL user database.

### 3.3.1 Dispatcher Daemon

The dispatcher daemon harnesses and manages heterogeneous computing resources provided by a Condor pool and a Beowulf cluster at UTEP. RNAVLab uses these resources to predict RNA secondary structures (predictions are among the most time demanding tasks in RNAVLab). Requests for predictions are submitted by the user via the Web application and performed using one or multiple prediction programs available. Currently RNAVLab supports the following prediction programs: PknotsRE [20], PknotsRG [19], NuPack [18] and UNAFold[10] [1].

When receiving a request with one or multiple sequences to predict, the Web application sends a message to the daemon, via a TCP connection, signaling that there are prediction jobs pending in the user database. The daemon retrieves the jobs and checks whether its local prediction database already contains the predictions of the sequences submitted. If a prediction does not already exist in the database, it is submitted to the heterogeneous resources via the Condor submission, i.e., using Condor job matchmaking and job scheduling policies. Once Condor returns the results, it saves them in the user database and updates the job status of the submission. One advantage of not sending the actual sequences or structures to the daemon is that if the daemon is offline, the jobs are not lost, but wait in the database for the next time the daemon is started and accesses the data. Having a TCP daemon also gives the potential benefit of hosting the Web application and the Java back-end on separate machines, which can enable running several instances of the submission daemon.

[1] UNAFold was added as part of the RNAVLab 2.0 extension

To setup a grid computing environment for the daemon to submit jobs to, RNAVLab relies on the Condor framework [28], for a variety of reasons. Condor provides all of the functionalities needed to implement a seamless grid layer that allows for faster prediction of RNA secondary structures by harnessing the idle cycles of computers, i.e., workstations and clusters. The pool of machines on which RNAVLab is currently relying consists of 23 single-, double-, and quad-processor 32- and 64-bit machines, and while all these machines run Linux (i.e., Fedora, Kubuntu, SuSE), Condor can also be installed on Unix, Windows (e.g., 2000, XP), and Macintosh (OS X) machines, among others. Condor handles all the details of sending executable and data files to computing resources and retrieving the computation results. Furthermore, Condor provides other useful features, such as checkpointing and job migration that only require re-linking the prediction programs used by RNAVLab with Condor libraries. These features can be very helpful, especially with predictions that take a long time: if the application is interrupted, checkpointing saves the computation state so it can be resumed later (instead of starting from scratch), and job migration allows a saved state to resume execution on a different machine. RNAVLab successfully re-links PknotsRE, NuPack, and UNAFold but not PknotsRG, due to its use of pthreads. Therefore, PknotsRG cannot use checkpointing and cannot benefit from job migration, but it can still be dispatched to the computing resources using the Condor policies.

To interface RNAVLab with Condor and dispatch jobs, a Condor submit file describing the jobs is generated when the user invokes the use of global resources through the Web application. Each job consists of a unique identifier, the name of the analysis program to be used (e.g., PknotsRE, PknotsRG, NuPack, UNAFold, alignment, comparison), and the command-line parameters (i.e., the input files with the RNA sequences or type of alignment to use). The file is then submitted to the pool. Condor also provides the functionality needed to check whether a submitted

job has completed execution: a single thread of the dispatcher daemon calls this function and updates the shared MySQL user database with the results once the job has been completed. The Web application updates the browser via an AJAX (Asynchronous JavaScript and XML) call and the user can visualize the structure as soon as the job is complete. At this point, the user can perform further analysis of the predictions if needed.

### 3.3.2 Prediction Tools

The most costly computations are the predictions of the RNA secondary structures. In order to provide a consolidated environment for predicting secondary structures RNAVLab provide access to four well-known prediction programs. These are:

- **UNAFold** Zucker's latest version of MFold which has a running time of $O(N^3)$
- **PknotsRE** Riva's and Eddys application that includes a constrained set of pseudoknots and has running time of $O(N^6)$
- **PknotsRG** Reider and Geigrichs application that further constrains the types of pseudoknots predicted and has running time of $O(N^4)$
- **NuPack** Pierce's application which is an extension of the McCaskils algorithm for also computing the partition function of the predictions and includes pseudoknots. It has a running time of $O(N^4)$.

In case a sequence is too long to predict given the provided resources, RNAVLab provides a dynamic programming algorithm that optimally segment a sequence based on a max window length and energy score (Chapter 4). This algorithm can be used with any of previous prediction algorithms and is part of this thesis's contribution.

### 3.3.3 Analysis Tools

The analysis of RNA sequences does not stop at predicting their secondary structure. More can be learned from alignments and comparisons of these structures such as the structural conservation along evolutionary history and performance amongst different applications. Thus, RNAVLab includes tools for comparison and alignment of secondary structures.

#### 3.3.3.1 Comparison Tool

Comparisons of observed and predicted structures, or across predicted structures using different programs are performed on aligned or non-aligned strings of brackets. Three different algorithms can be used for comparisons in RNAVLab:

- **A variant of the Hamming Distance-** the algorithm assigns each kind of nucleotide bond a numerical tag. Bonds GC, CG, UA, and AU are assigned tags from 1 to 4, respectively. The closures of the bonds are all assigned 0. The resulting numerical strings are compared and when two non-zero numbers and their respective closing 0 match, it is counted as a true pair. This approach is useful when the types of nucleotide bonds are important.

- **A stack based algorithm-** the algorithm uses stacks to keep track of the positions of open parenthesis and brackets in secondary structures with and without pseudoknots. When an open bracket or parenthesis is encountered, its position is pushed into a stack associated to a stem-loop. Pseudoknots are considered to be a combination of two stem- loops and therefore use two stacks. When a closed bracket or parenthesis is encountered, the position is popped from the associated stack. If a bracket or parenthesis is encountered at the same time in both structures and the position popped from both stacks is the same, this is counted as a true pair. This approach is useful when the identification of exactly alike structures is important.

- **A lenient algorithm-** the algorithm uses simple string comparisons that allows for similar yet shifted structures to receive high comparison scores. The algorithm traverses two bracket strings and counts how many times an open bracket or parenthesis is in the same position and how many times a closed bracket or parenthesis is in the same position for the strings, without considering the type of nucleotides involved. The smaller of these two values is the amount of true pairs.

#### 3.3.3.2 Alignment Tools

The alignment of two or more secondary structures is performed by aligning their bracket strings using variants of well known alignment algorithms such as the SmithWaterman [22] and NeedlemanWunsch [14] algorithms. Unlike the aforementioned algorithms that align string of nucleotides, i.e., A, U, C, and G, the variant algorithms align strings of brackets, i.e., ":", "(, and "[. Shifts are indicated with an underscore, i.e., _. The alignment of secondary structure is very important to identify secondary structures that are similar in their shape but are shifted: this can happen when, e.g., a predicted structure is compared with an experimentally observed structure or when the genome structures of two viruses belonging to the same family are compared.

## 3.4 User Database

None of the applications mentioned in Chapter 2 allow users to store submitted data for long period of time. This can be frustrating for scientists when working on sequences for several days, weeks, months or even years. Without storage, the user is forced to either save the structures on their own machine in flat files or some kind of database, or just re-compute the structures whenever they are needed again, which can be a waste of computing resources. In order to facilitate long term study

of RNA sequences, RNAVLab 2.0 provides users with a database that can store previously submitted sequences for future references.

The user database associates each user with her own set of submitted sequences, predictions, alignments, and comparisons. Once a structure is predicted with a given algorithm, it will not have to be predicted again, and the user will not have to wait for a prediction as she would if she just re-submitted it. Furthermore, because the dispatcher daemon checks the database to see if the sequence has been submitted before, if two users happen to be studying the same sequence, when the second user requests the predicted structure, she immediately gets back the prediction submitted by the first user, making RNAVLab 2.0 more efficient than single session prediction applications.

# Chapter 4

# OPTIMAL SEGMENTATION OF LONG RNA SEQUENCES

## 4.1 Search for Optimal Segmentations

All the computer prediction programs described in Chapter 2 can predict the secondary structure of an RNA sequence if given its primary sequence. Most of these programs use thermodynamic methods to find the structure with the lowest possible free energy (global energy minimum): they employ known energy values for short RNA sequences (determined empirically in wet laboratories as a function of the temperature required to completely denature a given RNA sequence) [2] and use a dynamic programming algorithm to build the secondary structure prediction. Although these programs are accurate for sequences a few hundred bases in length, their accuracies diminish for longer sequences due to a lack of experimental energy results for long RNA sequences. At the same time, because of the large polynomial running times of these programs, especially those that consider pseudoknots, these programs cannot predict the secondary structures of long RNA sequences of thousands of bases because of computer resources constrains. Therefore we sought to discover an alternative method that can find the optimal segmentation of these long prediction given poor scoring function and resource constrains.

In our previous work [26], we used an alternative approach for predicting secondary structures in which we divided long sequences into overlapping chunks, predicted the structure of each chunk, and rebuilt the whole secondary structure

from these component parts. Using this method, our predictions remain supported by the empirical thermodynamic evidence yet allow us to operate within existing limitations in computing power and memory. However, this raised the fundamental questions of how and where to subdivide the long sequence into chunks.

The problem of finding the best sequence segmentation to use for such a prediction is similar to some problems in natural language processing, i.e., problems with word recognition in recorded audio, finding paragraph breaks in a body of text, or any other optimal segmentation of non-overlapping information given a metric for scoring each segment. In the 1960s, Vintsyuk first proposed the use of dynamic [32] programming methods for time-aligning a pair of speech utterances. Although the essence of the concepts of dynamic time warping, as well as rudimentary versions of the algorithms for connect-word recognition, were embodied in Vintsyuk's work, it was largely unknown in the West and did not come to light until the early 1980s – long after more formal methods were proposed and implemented by others.

In [26], we used a brute force approach for subdividing long sequences in chunks in which we conducted a search of a small sub-space of overlapping chunks with fixed sizes by considering all the possible segmentations of the sub-space. Although successful in terms of its accuracy, the approach was not optimal and was extremely time demanding. The work in this thesis expands upon our previous results and presents an optimal method using dynamic programming to segment a long sequence into non-overlapping sub-segments. Not only does the method produce more accurate secondary structure predictions with respect to the similarity to the structure found in nature, it also conducts the search more efficiently. It is more efficient because, although the possible combinations of the segmentations is $O(2^N)$, which is intractable for any RNA sequence of a practical length N, we can use dynamic programming techniques to search this space in $O(N^2)$ time.

## 4.2 Algorithm Overview

Dynamic programming is an optimization technique that can be used when the optimal solution of the overall problem is composed of optimal solutions to subproblems. Here, we want to find the optimal non-overlapping segmentation of a long primary nucleotide sequence into chunks. In this case, the optimal segmentation is one that will provide us with the greatest similarity to the associated observed secondary structure once the secondary structure of each segment has been predicted and combined into an overall secondary structure prediction for the long sequence. An initial approach to solve this problem is to enumerate all the possible segmentations and search for the segmentation that maximizes the similarity with the observed secondary structure among the $2^{N-1}$ alternatives, where $N$ is the number of nucleotides in the RNA sequence. Since the search space grows exponentially, this approach is intractable for any practical value of $N$, even on supercomputers. As an alternative approach, we can use dynamic programming to search the segmentation space in polynomial time, where optimal solutions for each chunk are part of the optimal solution for the whole sequence.

In our dynamic programming approach, given a nucleotide sequence $x$ of length $N$ with $N >> 1$ and $j >= i$, we first build a matrix we call *predicted* that is filled as follows:

$$predicted(i,j) = predict_{prediction\ code}(i,j) \quad with\ j - i < Max\_C << Nandj >= i \qquad (1)$$

The function $predict_{prediction\ code}(i,j)$ takes the chunk starting at nucleotide $i$ and terminating at nucleotide $j$ and returns the predicted secondary structure of the chunk using a prediction algorithm. Any code for prediction can be used and, since the predictions are independent of one another, they can be performed in parallel and stored in a database. However, not all the chunks can be predicted for

a very long sequence, due to resource constraints, such as limits in memory size and computing power. Each code has a different $Max\_C$ length, which is the longest chunk length predictable. For example, the Pknots-RE [20] code can predict the structure of sequences up to 200 nucleotides in length, while Pknots-RG [19] can predict the structure of sequences up to 800 nucleotides long. Thus the predicted matrix is an upper right triangular matrix.

The rebuilding process uses the upper right triangular matrix, selects non-overlapping chunks and their predicted secondary structures from within the matrix, and combines them to build a secondary structure prediction for a nucleotide sequence longer than can be predicted otherwise. The selection of the chunks can be an a-posteriori or an a-priori selection.

A-priori selections are based only on the minimum energies of the chunks and can be used for blind predictions. In an a-priori selection using the prediction matrix, we build a score matrix and initialize its first row as follows:

$$score(1,j) = min\_energy(predicted(1,j)) \quad \forall j\ with 1 \leq j \leq Max\_C << N \quad (2)$$

where *min_energy()* is the lowest predicted energy of the chunk starting at nucleotide 1 and terminating at nucleotide $j$. Here we assume that $j \leq Max\_C << N$. Using a recurrence relation, we complete the upper right triangle of the score matrix as follows:

$$score(i,j) = min\_energy(predicted(i,j)) + \\ + min(score(k,i-1)) \\ \forall 1 \leq k \leq i-1 \qquad (3)$$

For each cell of the score matrix, $score(i,j)$, we also store a pointer to the cell that gave the best score among the matrix cells $score(k,i-1) \forall 1 \leq k < i-1$.

A-posteriori selections use experimentally obtained secondary structures as references for the scoring and is used for finding the best possible segmentation in terms of similarity to the known structure in nature.[1] In an a-posteriori selection using the prediction matrix, we build a *score* matrix and initialize its first row as follows:

$$score(1,j) = compare(predicted(1,j), observed(1,j))$$
$$\forall j \; with 1 \leq j \leq Max\_N << N \quad (4)$$

where *compare()* secondary structure of the sub-segment starting at nucleotide 1 and terminating at nucleotide $j$ to the corresponding experimentally observed secondary structure, *observed(1,j)*. The comparison can be based on different criteria, e.g., sensitivity and selectivity. Using a recurrence relation, we complete the upper right triangle of the score matrix, assuming that $j \leq Max\_C << N$ as follows:

$$score(i,j) = compare(predicted(i,j), observed(i,j)) +$$
$$+ max(score(k,i-1))$$
$$\forall 1 \leq k \leq i-1 \quad (5)$$

Again, as for the a-priori approach described above, for each cell of the score matrix we store a pointer to the cell that gave the best score among the matrix cells $score(k,i-1) \forall 1 \leq k <= i-1$.

[1] The a-posteriori method uses the known structure in nature in its scoring function, therefore it can not be used as a stand alone prediction algorithm. However, this data could potentially be used for training data in Machine Learning methods, but that is not covered in this thesis.

At this point, we backtrack through the pointers and retrieve the segments that give the optimal similarity (best score). Figure 4.1 shows a mock example of a score matrix and the backtrack used to identify the optimal segmentation for a sequence of 4 nucleotides ($N$=4), with a code with maximum predictable length equal to 3 ($Max\_N$=3). The values use are for demonstration purposes only and do not represent a real prediction scenario.

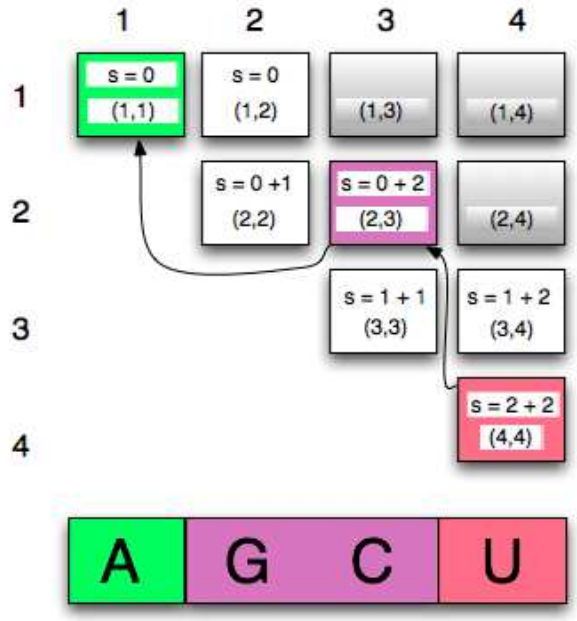



**Figure 4.1:** *Example of score matrix with backtrack to rebuild the optimal segmentation with $N = 4$ and $Max\_C = 2$*

## 4.3 Algorithm Complexity

The search for this optimal non-overlapping segmentation can be performed in time $O(N^2)$, where $N$ is the length of the sequence. Both the prediction and score matrices have sizes of $O(N^2)$; thus completing them takes time $O(N^2)$. Furthermore, it is important to note that, although every cell depends on the maximum score of the $i-1$ column, the maximum score for every row is the same. This is because

a row $k$ represents all segments starting at nucleotide $k$ and thus we need only to compute the maximum score of the $k-1$ column, representing all the segments that end at nucleotide position $k-1$. Once the maximal score for column $k-1$ is computed, the value can be copied to every other cell in row $k$, keeping the running time at $O(N^2)$.

# Chapter 5

# EVALUATION RESULTS

## 5.1 Evaluation Goals

As part of the evaluation, we use RNAVLab 2.0 to understand whether predictions of a sequence set that consider each sequence as a whole is more or less accurate than predictions that consider each sequence as multiple, non-overlapping chunks. Chunks are obtained using our optimal segmentation algorithm presented in Chapter 4. We also seek to understand whether the folding process favors local minimum energies rather than global minimum energies. This is relevant when the prediction of structures formed by very long sequences is not feasible. Therefore, when the global energy cannot be determined experimentally, the search for global energy must be replaced by the search across local minimum energies of shorter chunks. Last but not least, we compare the results using RNAVLab 2.0 and our optimal segmentation algorithm versus our previous work in [27] where we used a sliding window search of overlapping chunks to rebuild long secondary structures.

## 5.2 Prediction Program, Scoring Metrics, and Datasets

Given the definition of our dynamic programming algorithm, any prediction code and score metric for RNA secondary structure prediction can be used. We choose to use a popular prediction tool such as PknotsRG [19] because of its excellent performance in predicting the structure of sequences of up to 800 nucleotides and its ability to predict pseudoknotted structures. We use this code to address questions

regarding the accuracy of our approach. Given both a predicted structure and an observed structure in parenthetical format, we measure the accuracy of the predicted structure in terms of sensitivity (i.e., ability to predict all true pairs) and selectivity (i.e., ability to only predict true pairs). Secondary structures of long RNA sequences, i.e., of the order of thousands of nucleotides, that have been experimentally validated are rare. Thus, for our analysis in this thesis we used the longest nucleotide sequences from Group A in CONTRAfold [3], which have lengths ranging from 200 to 482 nucleotides.

## 5.3 Single-Segment Predictions vs. Predictions with Non-overlapping Chunks

To address whether predictions of a sequence set that consider each sequence as a whole are more or less accurate than predictions that consider each sequence as multiple, non-overlapping chunks, we select chunk sizes for our data set that are not limited by the prediction program or by computing resources. Therefore, our chunk sizes or $Max\_C$ range from 1 to $N$, where $N$ is the length of each sequence in terms of nucleotides. We also select sequences whose secondary structures have been experimentally-determined and are thus available for building the score matrix in Chapter 4, based on an a-posteriori approach. Finally, we use the average value of sensitivity and selectivity of predicted secondary structures versus experimental secondary structures as our metrics.

We first predict the secondary structure of each entire sequence using Pknots-RG. We then use our method and the same prediction code to identify the set of non-overlapping chunks with highest sensitivity and selectivity, as described in Chapter 4, Equations 4 and 5. The comparison of the two techniques, i.e., prediction based on the entire sequence and prediction based on non-overlapping chunks, can result in two possible outcomes. One possible outcome is that our approach based on chunks always converges towards the predictions based on the whole sequence,

meaning we get a solution with one large chunk. Alternatively, solutions that have a higher or equal score to whole sequence predictions are combinations of several smaller chunks. Note that since the algorithm is optimal and we set $Max_C = N$, it will search through all configurations, even the one where the solution is just a single chunk (which corresponds to the single sequence prediction). Also, the scores based on an a-posteriori approach are not driven by energy values in our selections but by the similarity to the known structure in nature.

For each sequence, Table 5.1 presents the sequence name and length; the sensitivity and selectivity of the prediction considering the sequences as a single sequence (a single chunk); the sensitivity, selectivity, the number of optimal non-overlapping chunks used to rebuild the secondary structure, and the maximum chunk length (or maximum window size) used with our method.

**Table 5.1:** *Comparison of sensitivity and selectivity for secondary structures predicted considering the sequence of nucleotides as a whole and as a set of non-overlapping chunks using an a-posteriori approach.*

| | | Single Sequence | | | Multiple Chunks - unlim. window | | | |
|---|---|---|---|---|---|---|---|---|
| Name | length(nt) | sen. | sel. | #segm. | sen. | sel. | #segm. | max window |
| RF00458_A | 202 | 0.58 | 0.39 | 1 | 0.78 | 0.64 | 4 | 146 |
| RF00193_A | 273 | 0.79 | 0.60 | 1 | 0.99 | 0.96 | 17 | 49 |
| RF00231_A | 275 | 0.71 | 0.41 | 1 | 0.97 | 0.71 | 9 | 68 |
| RF00503_A | 293 | 0.70 | 0.44 | 1 | 0.95 | 0.92 | 16 | 40 |
| RF00030_A | 297 | 0.68 | 0.48 | 1 | 0.67 | 0.59 | 13 | 59 |
| RF00216_A | 302 | 0.40 | 0.21 | 1 | 0.60 | 0.46 | 5 | 154 |
| RF00010_A | 312 | 0.77 | 0.63 | 1 | 0.77 | 0.63 | 3 | 306 |
| RF00009_A | 320 | 0.57 | 0.22 | 1 | 0.75 | 0.43 | 9 | 112 |
| RF00100_A | 330 | 0.40 | 0.23 | 1 | 0.81 | 0.70 | 8 | 107 |
| RF00036_A | 337 | 0.94 | 0.86 | 1 | 0.94 | 0.86 | 3 | 335 |
| RF00209_A | 379 | 0.75 | 0.46 | 1 | 0.90 | 0.76 | 10 | 218 |
| RF00024_A | 451 | 0.80 | 0.48 | 1 | 0.86 | 0.54 | 3 | 427 |
| RF00210_A | 462 | 0.80 | 0.56 | 1 | 0.91 | 0.74 | 7 | 295 |
| RF00177_A | 482 | 0.93 | 0.74 | 1 | 0.93 | 0.74 | 3 | 480 |

With unlimited chunk sizes and no resource limits, the chunk sets range from 3 to 17 sub-segments and their sizes are always smaller that the total sequence length. Only in 4 cases out of the 14 sequences considered (i.e, RF00010_A, RF00036_A, RF00024_A, and RF00177_A) did a single chunk cover the majority of the sequence. For these four cases, $Max\_C = N$. Our approach for these cases converges toward the whole-sequence prediction. In all the other cases, we observe equal or better sensitivity and selectivity when rebuilding the secondary structures from shorter non-overlapping sequence chunks. The better predictions can be either due to insufficiently accurate thermodynamic models for longer sequences (since the wet laboratory experiments are still missing or, in some cases, not feasible), or to the tendency for structures when folding to favor multiple localized minimum free energy structures rather than the global minimum free energy structure of the whole sequence, or it can be a combination of both.

### 5.3.1 Dealing with Resource Limits

To address the question of whether structures, when folding, tend to favor multiple localized minimum free energy structures rather than the global free energy structure, we explore all possible values for each sequence in our data set, of $Max\_C$ from 1 to N in Equations 2 and 3. This results in $N$ score matrices, each exploring window sizes only up to the $Max\_C$ associated with the matrix. Figure 5.1 shows a simple example for a sequence with 4 nucleotides and four score matrices obtainable with this sequence (The scores are mock examples). For each score matrix we rebuild its lowest energy secondary structure given the limitation of the window size. Note that this time our scoring approach is an a-priori approach based on energy values only and not a comparison to the experimentally known structures as in the previous section.

In Figure 5.2, we present two case studies from our data set to show how sensitivity and selectivity grow as a function of the window size. In both examples, we

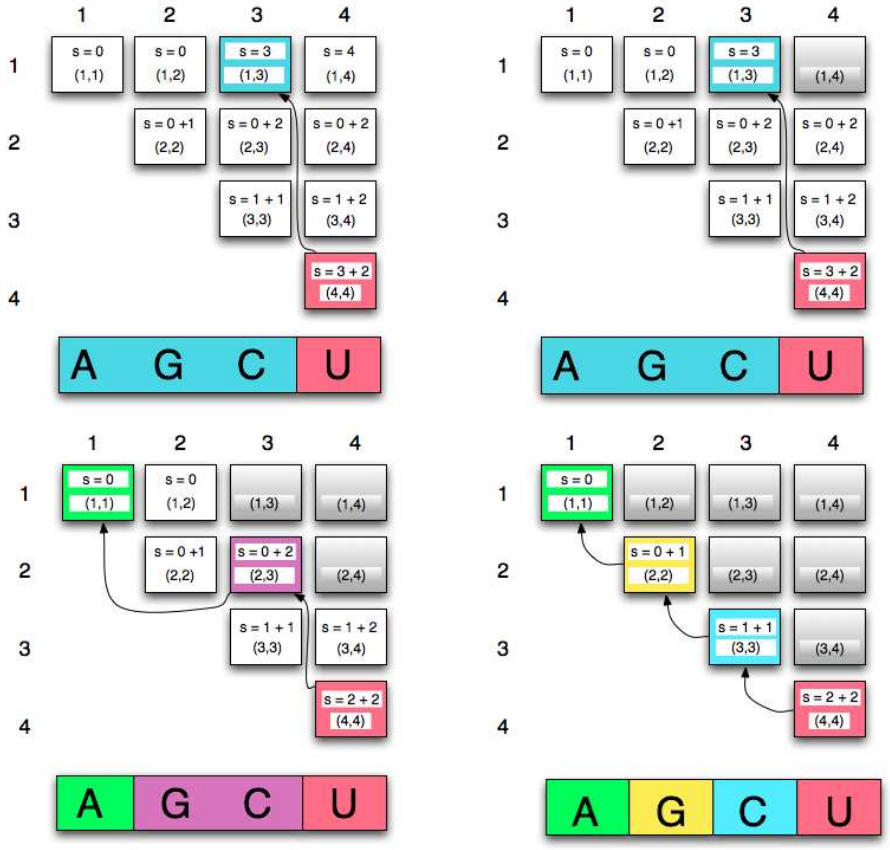
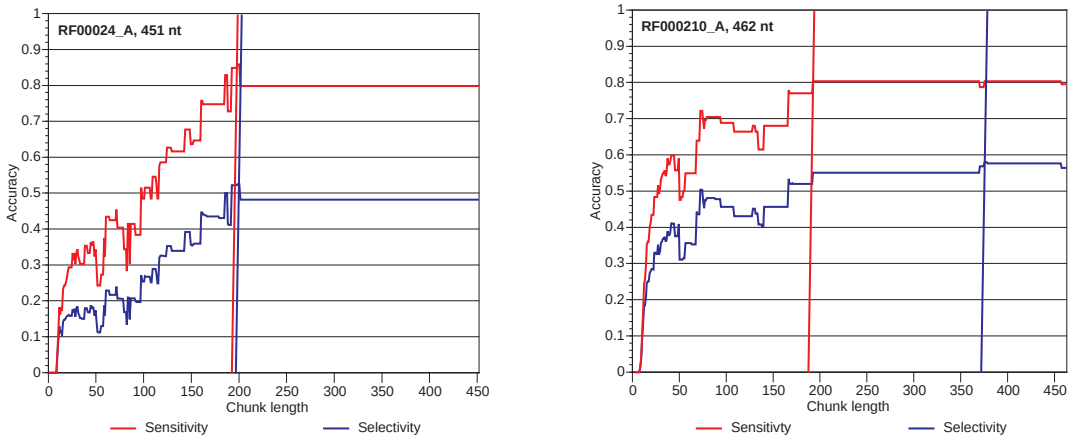



**Figure 5.1:** *Examples of 4 score matrixes obtainable with a sequence of 4 nucleotides.*

see that the whole sequence is not necessary to predict the best secondary structure. For Sequence RF00024_A with length 451 nucleotides and Sequence RF00210_A with length 462, with windows of 198 and 375 nucleotides respectively, we can already capture the best structures for the overall sequence. No further accuracy is gained by using longer chunks.

In Table 5.2, we compare sensitivity and selectivity of the prediction for all the 14 sequences in our data set using a global energy minimization (by feeding the whole sequence into the prediction code) with the sensitivity and selectivity of the prediction built from the best set of chunks obtained with our method. For the latter prediction, the table presents the number of chunks and the length of the




(a) Sequence RF00024_A, 451 nt

(b) Sequence RF000210_A, 462 nt

**Figure 5.2:** *Sensitivity and selectivity as a function of the chunk length for two sequences in our data set.*

longest chunk ($Max\_C$).

Using chunks that are limited in size (i.e., within the limitations of the computer resources) allows us to overcome computing and memory constraints. At the same time, we observe that for all 14 sequences we examined, the combination of non-overlapping chunks can predict secondary structures with either equal or better sensitivity and selectivity than those determined using the entire sequence. For only two sequences out of the 14 (RF00177_A and RF00036_A), the value of $Max\_C$ is almost as long as $N$, indicating the convergence of our method to the search for the global energy minimum.

## 5.4 Sliding Overlapping Chunks vs. Optimal Non-overlapping Chunks

In a previous attempt to solve the rebuilding of long RNA sequences using the prediction of their subsegments, we used a segmentation approach that searches the space of overlapping chunks [27]. Since the search space for overlapping chunks is exponential, the complexity of an algorithm that explores all the possible chunks is O($2N^2$), where N is the length of the input sequence. Because such an approach

**Table 5.2:** *Comparison of sensitivity and selectivity for secondary structures predicted considering the sequence of nucleotides as a whole and as a set of non-overlapping chunks selected using an a-priori approach.*

| | | Single Sequence | | | Multiple Chunks - lim. window | | | |
|---|---|---|---|---|---|---|---|---|
| Name | length(nt) | sen. | sel. | #segm. | sen. | sel. | #segm. | max window |
| RF00458_A | 202 | 0.58 | 0.39 | 1 | 0.71 | 0.53 | 17 | 71 |
| RF00231_A | 275 | 0.70 | 0.41 | 1 | 0.96 | 0.66 | 19 | 73 |
| RF00193_A | 273 | 0.79 | 0.65 | 1 | 0.82 | 0.65 | 46 | 33 |
| RF00503_A | 293 | 0.70 | 0.44 | 1 | 0.94 | 0.77 | 49 | 43 |
| RF00030_A | 297 | 0.67 | 0.48 | 1 | 0.67 | 0.48 | 9 | 228 |
| RF00216_A | 302 | 0.39 | 0.20 | 1 | 0.60 | 0.41 | 37 | 49 |
| RF00010_A | 312 | 0.76 | 0.63 | 1 | 0.76 | 0.63 | 6 | 307 |
| RF00009_A | 320 | 0.56 | 0.21 | 1 | 0.68 | 0.30 | 55 | 23 |
| RF00100_A | 330 | 0.40 | 0.22 | 1 | 0.70 | 0.48 | 20 | 107 |
| RF00036_A | 337 | 0.93 | 0.85 | 1 | 0.93 | 0.85 | 3 | 335 |
| RF00209_A | 379 | 0.74 | 0.45 | 1 | 0.79 | 0.50 | 15 | 218 |
| RF00024_A | 451 | 0.79 | 0.48 | 1 | 0.85 | 0.52 | 30 | 198 |
| RF00210_A | 462 | 0.79 | 0.56 | 1 | 0.80 | 0.57 | 18 | 375 |
| RF00177_A | 482 | 0.92 | 0.74 | 1 | 0.92 | 0.74 | 3 | 480 |

is not feasible even for short sequences, in our previous work we only searched a sub-space of the possible subsegments by using heuristics. In particular, each set of segments had a fixed size (window size) and a fixed sliding step (window step). The segments in a set were generated by progressively sliding the fixed-size window forward for a fixed number of steps. At each step, the nucleotides within the window formed a segment. For an extensive sampling, this procedure was repeated to generate several sets by increasing the window sizes and/or the window steps, each time generating a new set of segments. In our experiments the window sizes were increased by five bases. The maximum length of a window was n/2, where n is the length of the RNA sequence we wanted to rebuild the secondary structure for. Window steps ranged from 1 base to $w - 1$ bases, where $w$ is the window size.

Accuracy is determined using the lenient algorithm described in chapter 3. It

uses simple string comparisons that allows for similar yet shifted structures to receive high comparison scores. The algorithm traverses two string in dot-par format and counts how many times an open bracket or parenthesis is in the same position and how many times a closed bracket or parenthesis is in the same position in both predicted and known structures. The lesser of these two values is the amount of true pairs.

In this section we compare the sensitivity and selectivity for the predictions obtained using the sliding overlapping chunks versus the approach in this thesis based on optimal non-overlapping chunks, with an a-posteriori approach. Table 5.3 presents the results of the comparison. Our optimal a-posteriori approach always matches or outperforms our previous approach. In addition, the number of chunk predictions required with the optimal, non-overlapping approach is significantly less that with the sliding overlapping technique.

**Table 5.3:** *Comparison of sensitivity and selectivity for secondary structures predicted considering the sequence of nucleotides as a set of sliding overlapping chunks and a set of non-overlapping chunks.*

| | | Multiple Chunks - sliding and overlapping | | Multiple Chunks - optimal and non-overlapping | |
|---|---|---|---|---|---|
| Name | length(nt) | sen. | sel. | sen. | sel. |
| RF00458_A | 202 | 0.78 | 0.64 | 0.78 | 0.64 |
| RF00193_A | 273 | 0.86 | 0.73 | 0.99 | 0.96 |
| RF00231_A | 275 | 0.97 | 0.71 | 0.97 | 0.71 |
| RF00503_A | 293 | 0.95 | 0.87 | 0.95 | 0.92 |
| RF00030_A | 297 | 0.70 | 0.53 | 0.67 | 0.59 |
| RF00216_A | 302 | 0.60 | 0.46 | 0.60 | 0.46 |
| RF00010_A | 312 | 0.68 | 0.57 | 0.77 | 0.63 |
| RF00009_A | 320 | 0.77 | 0.35 | 0.75 | 0.43 |
| RF00100_A | 330 | 0.75 | 0.58 | 0.81 | 0.70 |
| RF00036_A | 337 | 0.63 | 0.5 | 0.94 | 0.86 |
| RF00209_A | 379 | 0.77 | 0.54 | 0.90 | 0.76 |
| RF00024_A | 451 | 0.86 | 0.53 | 0.86 | 0.54 |
| RF00210_A | 462 | 0.91 | 0.69 | 0.91 | 0.74 |
| RF00177_A | 482 | 0.82 | 0.63 | 0.93 | 0.74 |

# Chapter 6

# CONCLUSION AND FUTURE WORK

## 6.1 Conclusion

In this thesis we describe a Web application for the study and analysis of RNA secondary structures that we call RNAVLab 2.0. It expands and extends the existing virtual laboratory for the study of RNA secondary structures RNAVLab 1.0 with a Web application and Web service front-end and a multi-threaded Java back-end. This portal is aimed at bringing several powerful prediction tools together previously scattered across several portals and sites. Within RNAVLab 2.0, these applications can be accessed from anywhere on the Internet using a browser or via an XML RESTful Web service.

An important component of RNAVLab 2.0 is the dynamic programming algorithm for predicting the optimal segmentation of long sequences into chunks using non-overlapping segments. This feature is particularly relevant when scientists try to predict RNA sequences whose length is too long and thus cannot be predicted as a whole because of resource constraints. We show in this thesis that predictions based on rebuilding long secondary structures from the prediction of the single, non-overlapping chunks can get comparable accuracy to predictions using the whole sequence. More in particular, our results show that our approach can outperform MFE methods using dynamic programming to search for global energy minima 12 times out of 14 with the longest sequences in Group A in CONTRAfold.

To address resource constraints, chunk lengths in RNAVLab 2.0 can be set up less than $Max\_C$, where $Max\_C$ is the maximum window length a PC can handle

before it runs out of memory or exceeds a desired max running time. Besides overcoming the computational and memory constraints of a single computer, our approach of segmenting a long sequence in chunks and predicting the chunks before merging them in the long secondary structure makes it possible to predict the smaller structures in parallel and thus makes it possible to use parallel computers across campus and high performance computers across the Internet, such as TeraGrid platforms.

## 6.2 Future Work

We identify several main tasks that can be considered for future work:

- In this thesis we explored the accuracy of our dynamic programming algorithm when the rebuilding of the single chunks was performed using PknotsRG. Since our algorithm is application-independent, it is worthwhile to study whether the results presented in this thesis are reproducible across several other prediction programs, including those that do not use the traditional energy minimization method.

- Although the current pool of machines used by RNAVLab consists of Linux machines only, future work includes its extension to clusters available across the whole UTEP campus as well as the integration and support of BOINC (Berkeley Open Infrastructure for Network Computing) [1] to allow researchers to deploy desktop and laptop PCs owned by students or administration personnel outside the campus when these computers are idle. Work in [25] shows that adding idle cycles of PCs significantly increases available computing power.

- Machine learning and artificial intelligent technique can be used as part of the optimal segmentation algorithm to train a model to detect where a window should be sampled and predicted, given only the input sequence. If such a model can be found and it provides good accuracy, we can further decrease

the prediction running time by only predicting those chunks that the model outputs and by not having to explore the $N^2$ space of sub-segments, where $N$ is the length of the input sequence. An example of such a scenario could be designing and training neural network based tools to identify optimal segmentations when the experimental secondary structure is not available and the memory and computing resources are limited.

- Robustness of an application is extremely important as the number of users grows. We would like to examine the durability of the RoR Web Application as well as the Java backend under several common use cases and load strains. The lack of bugs in a site and its consistant and correct functioning will help attract new users as well as keep existing ones.

- Help items such as tool tips and documentation make it easy for a user to understand the tool that they are using in a straight foreward way. Future work will include adding tool tips and comprehensive documentation in order to further facilitate the easy use of RNAVLab 2.0.

- Although powerful and simple, the RESTful web-service is still a relatively new technology and paradigm. It would be worthwhile to include a template program in order to show users how to use the available services and give them a foundation to extend their own implementations.

# BIBLIOGRAPHY


[1] D.P. Anderson. BOINC: A system for public-resource computing and storage. In *proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, page 10. IEEE Computer Society, 2004.

[2] Y. Byun and K. Han. PseudoViewer: web application and web service for visualizing RNA pseudoknots and secondary structures. *Nucleic Acids Research*, 34(Web Server issue):W416, 2006.

[3] C.B. Do, D.A. Woods, and S. Batzoglou. CONTRAfold: RNA Secondary Structure Prediction without Physics-Based Models. *Bioinformatics*, 22(14):e90, 2006.

[4] R.T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, Citeseer, 2000.

[5] PP Gardner and R. Giegerich. A comprehensive comparison of comparative RNA structure prediction approaches. *BMC bioinformatics*, (5), 2004.

[6] A.P. Gultyaev, FHD Van Batenburg, and C.W.A. Pleij. The computer simulation of RNA folding pathways using a genetic algorithm. *Journal of Molecular Biology*, 250(1):37–51, 1995.

[7] I.L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Research*, 31(13):3429, 2003.

[8] B. Knudsen and J. Hein. RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15(6):446, 1999.

[9] AV Lukashin and M. Borodovsky. GeneMark. hmm: new solutions for gene finding. *Nucleic Acids Research*, 26(4):1107, 1998.

[10] NR Markham and M. Zuker. UNAFold: software for nucleic acid folding and hybridization. *Methods in molecular biology (Clifton, NJ)*, 453:3, 2008.

[11] D.H. Mathews and D.H. Turner. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *Journal of Molecular Biology*, 317(2):191–203, 2002.

[12] A. Mathuriya, D.A. Bader, C.E. Heitsch, and S.C. Harvey. GTfold: A scalable multicore code for RNA secondary structure prediction. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 981–988. ACM New York, NY, USA, 2009.

[13] JS McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Peptide Science*, 29(6-7):1105–1119.

[14] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[15] R. Nussinov, G. Pieczenik, J.R. Griggs, and D.J. Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied Mathematics*, 35(1):68–82, 1978.

[16] J.T. Pedersen and J. Moult. Genetic algorithms for protein structure prediction. *Current Opinion in Structural Biology*, 6(2):227–231, 1996.

[17] M. Petrillo, G. Silvestro, P.P. Di Nocera, A. Boccia, and G. Paolella. Stem-loop structures in prokaryotic genomes. *BMC genomics*, 7(1):170, 2006.

[18] NA Pierce. NuPack: A Software Suite for the Analysis and Design of Nucleic Acids, 2006.

[19] J. Reeder, P. Steffen, and R. Giegerich. PknotsRG: RNA Pseudoknot Folding Including Near-Optimal Structures and Sliding Windows. *Nucleic acids research*, 2007.

[20] E. Rivas and S. R. Eddy. A Dynamic Programming Algorithm for RNA Structure Prediction Including Pseudoknots. *Journal of Molecular Biology*, 285(5):2053–2068, February 1999.

[21] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM Journal on Applied Mathematics*, 45(5):810–825, 1985.

[22] TF Smith and MS Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[23] M.C. Su, C.T. Chang, C.H. Chu, C.H. Tsai, and K.Y. Chang. An atypical RNA pseudoknot stimulator and an upstream attenuation signal for-1 ribosomal frameshifting of SARS coronavirus. *Nucleic acids research*, 33(13):4265, 2005.

[24] G. Tan, N. Sun, and G.R. Gao. A parallel dynamic programming algorithm on a multi-core architecture. In *Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, page 144. ACM, 2007.

[25] M. Taufer, C. An, A. Kerstens, and CL Brooks. Predictor@ Home: A" Protein Structure Prediction Supercomputer" Based on Global Computing. *IEEE Transactions on Parallel and Distributed Systems*, 17(8):786, 2006.

[26] M. Taufer, T. Solorio, A. Licon, D. Mireles, and M.Y. Leung. On the Effectiveness of Rebuilding RNA Secondary Structures from Sequence Chunks. In *Proceedings of 7th IEEE Intl Workshop on High Performance Computational Biology (HiCOMB)*, pages 1–8, 2008.

[27] Michela Taufer, Ming-Ying Leung, Thamar Solorio, Abel Licon, David Mireles, Roberto Araiza, and Kyle L. Johnson. RNAVLab: A Virtual Laboratory for Studying RNA Secondary Structures Based on Grid Computing Technology. *Parallel Computing*, 34(11):661 – 680, 2008. High-Performance Computational Biology.

[28] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The Condor experience. *Concurrency and Computation Practice and Experience*, 17(2-4):323–356, 2005.

[29] V. Thiel, K.A. Ivanov, A. Putics, T. Hertzig, B. Schelle, S. Bayer, B. Weissbrich, E.J. Snijder, H. Rabenau, H.W. Doerr, et al. Mechanisms and enzymes involved in SARS coronavirus genome expression. *Journal of General Virology*, 84(9):2305, 2003.

[30] I. Tinoco, O.C. Uhlenbeck, and M.D. Levine. Estimation of secondary structure in ribonucleic acids. 1971.

[31] FHD Van Batenburg, AP Gultyaev, CWA Pleij, J. Ng, and J. Oliehoek. PseudoBase: a database with RNA pseudoknots. *Nucleic Acids Research*, 28(1):201, 2000.

[32] TK Vintsyuk. Speech Discrimination by Dynamic Programming. *Cybernetics and Systems Analysis*, 4(1):52–57, 1968.

[33] S.R. WILKINSON and M.D. BEEN. A pseudoknot in the 3' non-core region of the glmS ribozyme enhances self-cleavage activity. *RNA*, 11(12):1788, 2005.

[34] M. Zuker. Mfold Web Server for Nucleic Acid Folding and Hybridization Prediction. *Nucleic acids research*, 31(13):3406, 2003.

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