ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Le** Jméno: **Anh Vu** Osobní číslo: **420346**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Datové vědy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Hledání sekundarních struktur v primarních strukturách nukleových kyselin

Název diplomové práce anglicky:

Secondary structure search in primary nucleic acid structures

Pokyny pro vypracování:

1. Seznamte se s funkcí a strukturou ribonukleových kyselin (RNA).
2. Proveďte rešerši stávajících nástrojů predikce sekundarní struktury nukleových kyselin (princip, složitost, dostupnost, apod.).
3. Nástroje uvedené výše (resp. vybraného kandidáta) použijte pro odhad složitosti vyhledání výskytu motivů interního vazebného místa pro ribozóm viru hepatitidy C v transkriptomu lidské buňky.
4. Úlohu uvedenou výše implementujte ve vhodné parametrizaci včetně hierarchické aplikace na vybraná místa lidského genomu.
5. Reportujte a diskutujte/předběžně ověřte kvalitativně uspořádané nalezené kandidátní výskyty motivů.

Seznam doporučené literatury:

Durbin, R. et al.: Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge university press, 1998.
Mathews, D.H.: Revolutions in RNA secondary structure prediction. Journal of molecular biology, 359(3):526–532, 2006.
Churkin, A et al: Design of RNAs: comparing programs for inverse RNA folding. Briefings in bioinformatics, 19(2), 350-358, 2017.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Jiří Kléma, Ph.D., Intelligent Data Analysis FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **23.01.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

doc. Ing. Jiří Kléma, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

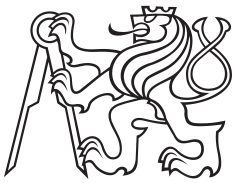
III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Secondary structure search in primary nucleic acid structures

Anh Vu Le

Supervisor: doc. Jiří Kléma
May 2019

Acknowledgements

I would like to thank my supervisor Mr. Kléma for guidance and useful advice throughout this work. I am also grateful for the resources he has given me in the final phase of the thesis. My gratitude also belongs to my parents, whose support and words of encouragement has been tremendous.

Declaration

I declare that this thesis has been composed solely by myself and except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

In Prague, 15. May 2019

Abstract

Structure of RNA molecules is often important for their function and regulation. Discovering a particular functional structure within a genome could then be viewed as discovering the associated function. Cap-independent translation is one of those functions. It is a well known mechanism with which viruses seize protein production capacities of cells. Though typical for viral RNA, cap-independent translation is not only a domain of viruses - a fraction of human genes was shown *in vitro* to use this mechanism as well. Tests leading to these discoveries are however both time and resource consuming. This work proposes a computational pipeline, that searches the human genome and outputs potential IRES candidates. The main steps of the pipeline involve inverse folding, BLAST and structural search. As a proof of principle, we present findings within 5'UTR region of DRC3 gene that contain structural motifs, which may exhibit transcription regulatory capabilities.

Keywords: HCV IRES, translation regulation, inverse RNA folding, BLAST, RNA structure search, human genome

Supervisor: doc. Jiří Kléma

Abstrakt

Struktura RNA molekul je významná z hlediska jejich funkce a regulace. Na objev konkrétní funkční struktury v genomu pak může být nahlíženo jako na objev funkce s ní spojené. Translace nezávislá na 5' čepičky je jednou z těchto funkcí. Je to známý mechanismus, který umožňuje virům převzít buněčné kapacity na výrobu proteinů. Přestože je to mechanismus typický zejména pro viry, není pouze jejich doménou - u části lidských genů bylo ukázáno *in vitro*, že tento způsob translace také používají. Testy vedoucí k těmto objevům jsou však náročné jak na čas, tak na zdroje. Tato práce proto představuje výpočetní pipeline, ve které je prohledáván celý lidský genom a úseky podobné IRES viru HCV nahlášeny. Hlavními kroky pipeline je inverzní skládání sekvencí, BLAST and vyhledávání na základě struktur. Jako ověření konceptu je předložena oblast v 5'UTR oblasti genu DRC3, ve kterém lze najít strukturní motivy mající potenciálně schopnost regulovat genovou translaci.

Klíčová slova: HCV IRES, inverzní skládání RNA, BLAST, strukturní vyhledávání RNA, lidský genom

Contents

Figures

Tables

Chapter 1

Introduction

Protein synthesis is an extremely important process for all living organisms. However, any organism or cell in the multicellular body does not synthesize all possible proteins at the same time. The actual palette of proteins in a given cell and given time period is limited. Therefore it is not surprising that protein synthesis is highly regulated and diverse process.

Proteins are mainly synthesized by so called *cap-dependent* manner in eukaryotes and their viruses. However, other possibilities exist. One of them is direct binding of a ribosome to an mRNA via so called *internal ribosome entry site (IRES)*. Many viral and cellular IRESs have been described so far. It should be noted that all best studied IRESs are of viral origin and that known putative cellular IRESs often exhibit much weaker activity then their viral counterparts. Experimental demonstration of IRES function is technically demanding and thus many of the published cellular IRESs have been challenged by other authors.

Despite of displaying weaker activity, cellular IRESs carry out a significant role in gene expression, as they can help manifesting important proteins, even under conditions of reduced protein synthesis. These proteins are often involved in cell proliferation, apoptosis, antiviral response, etc. Finding such an IRES would bring a valuable insight into protein synthesis regulation. Therefore new approaches for searching and validation of novel cellular IRESs are needed.

Another motivation for this project is that many viral functional structures can have their origin in cellular molecular machineries or conversely, could have developed to fit into an already established cellular function. In the world of RNA, discoveries of functional ribozymes, that are frequently used by viruses, in many genomes including human ([70], [77]) can serve as examples.

IRES too is a structure, for which a strong analogy in viruses can be found. In this work, Hepatitis C virus (HCV) is taken as a model - the HCV IRES is well studied and is therefore suited as a subject of searching. A hypothesis, that similar structure can be found within a human genome will be tested computationally and results handed to biologists for *in vitro* verification.

Identifying potential IRES-like structures will at some point require prediction of RNA structure from its sequence. As the performance and accuracy of prediction algorithms do not scale well with sequence sizes, searching for

entire IRES structures would make the task nearly infeasible. Fortunately, the nature of HCV IRES permits us to focus on smaller parts of the structure, as some of those *domains* are more vital for IRES functioning. Primary interest will be put on *domain III*, the secondary on *domain II*.

However, even with such a task reduction, a search based on pure RNA structure comparison would still be extremely difficult. The goal would be therefore to first employ faster and well established sequence based methods to obtain partial results and already then make use of structural search. This core idea will be developed into a pipeline, where first, a large number of sequences having the required structure will be generated by a process called *inverse folding*. Such a step enables departure from the demanding structural search to significantly faster sequence based search. Generated sequences will be subsequently compared with the human genome through well known BLAST. Finally, partial results obtained form this search will be processed by an *NA2Dsearch* tool to check, which of them can possibly acquire the IRES-like structure.

Regarding the thesis structure - the content is organized into 4 blocks:

- *Biological background*: First, biological foundations are laid down, on which the subsequent reasoning on the computational methodology is built. HCV IRES structure will be described in the context of its impact on the element's function.
- *RNA structure prediction and search*: Here, a research on the existing algorithmic solutions to RNA structure prediction and structure search will be presented. Inverse folding tools, related to the subject, will be included as well.
- *Previous work*: The task of locating IRES elements *in silico* has been approached by several solutions, whose summary will be given in this section.
- *Results*: A novel solution in a form of a pipeline will be proposed, consisting of inverse folding, BLAST and structural search. Each stage of the pipeline will be discussed separately. This section will in addition contain benchmark on selected inverse folding programs as prior step to building the pipeline.

Chapter 2

Biological background

DNA is known to hold information about proteins an organism can produce. Each such protein has a *gene* holding instructions how to build it. These instructions are encoded into DNA as a sequence of elementary units called *nucleotides*. There are four of them - adenine, cytosine, thymine and guanine with single-letter designations A,C,T and G. The so called "code of life" is then written as a combination of those four letters.

2.1 Mechanism of protein synthesis

When a protein is needed, the information on the corresponding gene has to be manifested. The protein is however not assembled directly from the DNA - instead, a copy of its gene is made onto an RNA. Already then, the protein is synthetized by a ribosome via a process called *translation*, when for each RNA nucleotide triplet an corresponding amino-acid residue is added to a growing polypeptide chain to form a compelete protein [3].

2.1.1 Role of mRNA structure in protein synthesis

So far, RNA was perceived as a linear sequence, whose only purpose was to code protein building instructions. As we see later, RNA does not need to be this simple string of nucleotides, it can be more structuraly diverse. There excist non-coding RNAs, which can fold into complex 3D structures, which then catalyse various bio-chemical reactions [20]. Coding, or messenger RNAs (mRNAs) can too harbor structures with a potential to play a vital role in translation initiation.

After an eukaryotic messegner RNA (mRNA) is transcribed, it has its ends altered with elements increasing its stability and marking it fit for translation. The 5' end is treated with a so called *5' cap* and its 3' end extended with a sequence of adenine bases called *poly(A) tail*. Thanks to the 5'cap, the mRNA is then recognized by *(eukaryotic) translation initiantion factors* (eIFs). The purpose of these factors, apart from recognizing mRNAs, is to then help assembling ribosome at the start codon.

Between mRNA's 5' cap and a start codon lies a region of nucleotides, which do not code aminoacids, called *5' untranslated region (5' UTR)*. This

region is in eukaryotes several hundreds bases long [48] and serves as antoher tool for gene expression regulation. It has been found, that secondary structures within 5'UTR can effectively inhbit mRNA translation - for instance a sufficiently stable "hairpin" structure located close to the cap can block translation altogether (reviewed at [8]).

The untranslated regions and their structure also serves as means for viruses to take hijack the host cell protein synthesis pathways. This will be discussed further later.

2.1.2 Other RNA functions

RNA structure can carry other roles, than the post-transcriptional regulation in mRNAs. A structured non-coding RNA has diverse catalytical roles. Important catalytical functions can be found just within the protein synthesis machinery. For instance, the activity of peptidyl transferase in ribosomes, an important enzyme in charge of adding amino acid residues to a growing polypetide chain, is thought to be catalyzed by ribosomal RNA. Next, the removal of non-coding regions in mRNA (*splicing*) before entering ribosomes, is catalysed by an RNA/protein complex called spliceosome [20].

2.1.3 mRNA translation initiation

The initiation of protein synthesis plays an essential role in celluar biology. The initiation of the translation process relies on recognition of mRNA 5' cap by a group of *translation initiation factors* [15]. The process is performed by the joint effort of small ribosomal subunit (40S) along with several other proteins called eukaryotic initiation factors (eIFs). These factors bind to the 5' end of an mRNA in a specific order and enable the translation to begin [32].

2.2 Viral interference with cell protein synthesis

Viruses tend to disrupt this pathway and inhibit the host's translation initiation, so that the host's ribosomes are used for synthesis of viral proteins. They do so in various ways - for instance influenza virus RNA carries endonucleases that cleave the 5' cap from host cell mRNAs, effectively making the mRNA unrecognizable by translation machinery. Some go even further and use the cleaved 5' caps to synthesize their own mRNAs with a process called *cap snatching*. Others skip the need for 5' cap-dependent translation using a so called *internal ribosome entry sites (IRESs)* [15].

2.2.1 IRES

As outlined earlier, IRES is an RNA feature within its 5'UTR, enabling cap-independent translation. The structure of IRESs allows RNA to recruit directly 40S ribosomal subunit wihout the need for cap-recognition or even

some eIFs [32]. IRES is commonly leveraged by RNA viruses, which can in some cases take over host-cell's protein synthesis [62].

IRES groups

Depending on requirements for cap-independent translation (various factors and proteins), viral IRES can be divided into four separate groups [43]. The biological characteristics of the groups are not relevant for this work. It suffices to discuss their differences on the level of primary and secondary structures.

Although the IRES elements in viral RNA perform a similar function, they vary greatly on the sequence level. This applies even for the representatives within the same group. Taking members of group 3, the portion in which their RNA differ can as high as 50% as in case of encephalomyocarditis virus (EMCV) and foot-and-mouth disease virus (FMDV) picornavirus. On the other hand, both IRES elements adapt rather similar secondary structure and to a large extent use the same mechanism to recruit ribosomal subunits [50] [43]. The RNA structure is indeed vital for IRES to function. This can be evidenced by so called compensatory mutations that help to preserve structural and thus also functional features of IRESs in highly variable viruses such as FMDV, hepatitis C (HCV) and others [38].

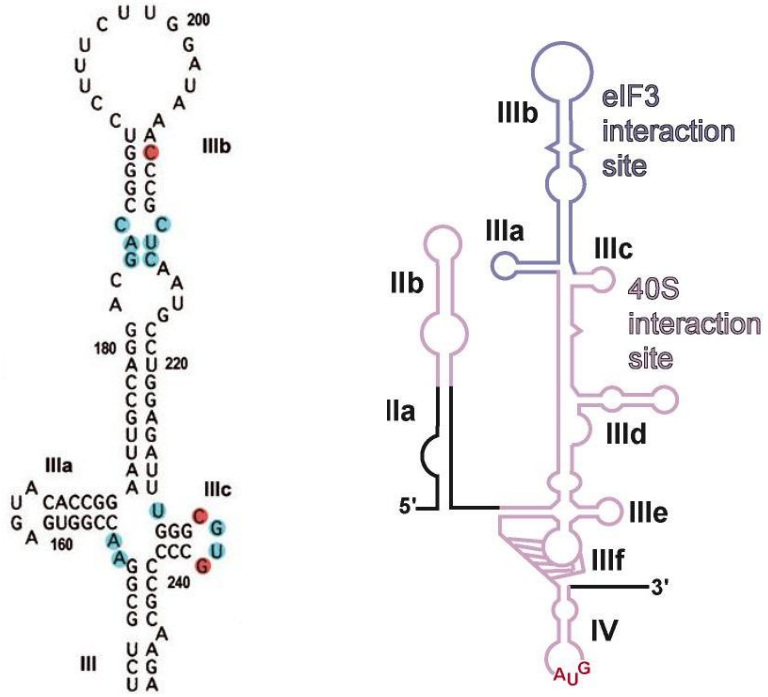
The structural conservation however applies only for viruses within the same group. For instance, well-established IRES elements, such as the IRES of HCV virus and those of picornaviruses, which belong to different families of RNA viruses, not only lack sequence homology, but also manifest different structural organization. The cause is that IRES elements can differ in the requirement of factors to assemble 48S initiation complexes [63]. This also implies that there is no universal IRES structural motif. In fact, IRES elements present in the genome of different families of RNA viruses even lack overall conserved features [55].

2.2.2 Hepatitis C IRES

The 5' cap-independent translation is also a mechanism employed by Hepatitis C (HCV) virus. Its IRES directly recruits a small ribosomal subunit (40S) at the start AUG codon wihout the need of the entire set canonical eIFs. The subset it targets includes eIF2 and most importantnly eIF3 [43] [67], which has a special function of controling access of other initiation factors [74]. The principles driving this recruitment are given by the structure of HCV IRES.

The IRES region is formed by approximately 341 nucleotides organized into four domains, designated as I, II, III, and IV. Each performs a distinct function during initiation. The most intersting for our purposes is *domain III*, the principal domain of HCV IRES, which interacts with both eIF3 and 40S subunit. It consists of a multi-bulged structure organized in sub-domains IIIa, b, c, d, e, and f. The key part is the junction of sub-domains abc, playing an important role of recognizing eIF3 and 40S - introduction of mutations at the junction has been shown to be fatal for IRES efficiency

[42]. Subdomain IIIb binds eIF3, but is less critical for the ability of IRES to form elongation-competent ribosomes [31]. Its structure near the apical loop is also the least explored and seems rather disorderly in the absence of stabilizing eIF3 [31][71]. IIIb on the other hand does contain a conserved motif - the intrahelical C186–C211 mismatch and surrounding base pairs affect the efficiency of eIF3 binding [51]. Lastly, the functional start codon of the HCV RNA is located at domain IV [37].



(a) : Domain IIIabc [42] **(b) :** HCV IRES schema [51]

Figure 2.1: Secondary structure diagram of the HCV IRES domain IIIabc. Nucleotides colored in cyan indicate highly conserved regions. Red nucleotides can be neglected in our context. On the second figure, the domain III is put into the context of the entire IRES with ribosomal proteins interaction sites denoted in distinct colors. Note the start codon at the domain IV.

2.2.3 IRES in eukaryots

Cap-independent translation is not only a domain of viruses. It has been found, that under certain stress conditions, when cap-dependent translation is inhibited (such as during nutrient limitation or mitosis), cellular IRES-mediated translation takes over. It should be noted, that compared to their viral counterparts, cellular IRES elements appear to be much more diverse in their structures and less stable in terms of the Gibbs free energy of the folded mRNA (review in [45]). A recent study experimentally determined that approximatelly 10% of human 5'UTRs can attract the ribosome through IRES elements. 583 novel human IRESs have been discovered across the

genome. [78] The IRESes unfortunately have no indication, which viral family they resemble.

It should be noted that known putative cellular IRESs often display much weaker activity then their viral counterparts and experimental demonstration of their IRES function is technically demanding. Many of published cellular IRESs thus have been challenged by other authors. [61]

Chapter 3

RNA structure representation and comparison

RNA, as well as proteins, can form complex 3D structures. The basic phenomenon governing the RNA structure formation is base pairing of nucleotides. In canonical (also known as Watson-Crick) base pairing, RNA building blocks connect with each other via hydrogen bonds. Guanine (G) pairs with cytosine (C) and adenine (A) pairs with uracil (U]. RNA reaches its final, biologically active 3D conformation through Van der Waals forces and in a presence of magnesium ions [79].

3.1 Levels of RNA structural organization

Considering only effects of base pairing and enumerating such base-pairs, one can capture a planar representation of RNA, its *secondary structure*. It is an intermediate representation between the *primary structure* (solely string of nucleotides) and the final *tertiary structure*.

The tertiary structure is formed by arranging secondary structure elements in space. This process is driven by forces weaker, than the ones creating the secondary structure. RNA folding can be therefore viewed as a hierarchical process, in which secondary structure forms before tertiary structure [76].

Since this assumption also greatly decreases computational efforts, from now on we will be considering only the secondary level of RNA structural organization.

3.2 RNA secondary structure elements

In contrast to DNA, which exists in a form of two complementary nucleotide strands bound together, RNA molecules do not have a counterpart and fold just on themselves. This leads (on the level of secondary structure) to double stranded helices interrupted by various single stranded regions. By comparing RNA structures, several of repeated structural elements have been revealed [3]. We now describe the most essential motifs.

A *hairpin loop* is formed when an RNA strand folds back on itself. In an *internal loop*, two paired regions are separated by at least one base on each

strand. On the contrary, a *bulge* has unpaired nucleotides only on one strand. A *multi-branched loop* occurs when multiple double-stranded regions join via any number of unpaired bases [79]. A figure below pictures the described elements:

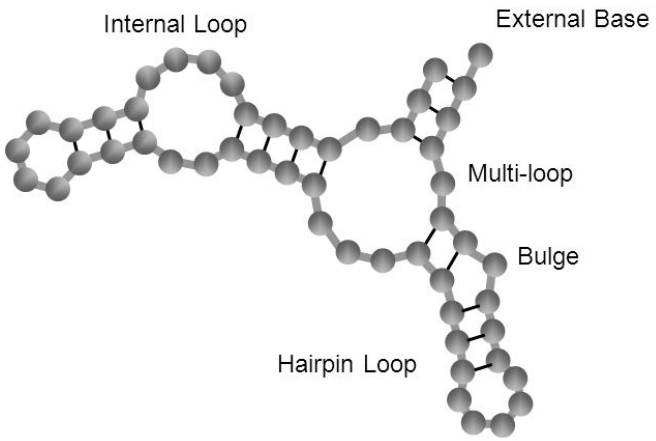


Figure 3.1: Secondary structure elements [1]

3.3 Secondary structure representations

The figure above is also an example of one representation of the RNA secondary structure. It takes form of a graph with nodes representing bases and with two kinds of edges - one denoting an adjacency of nucletides along the RNA sequence (on the figure this edge is reduced to two circles touching each other) and one denoting base-pairing [36].

Apart from this representation there exist less visual ones, but more suitable for machine processing. The most used ones will be presented.

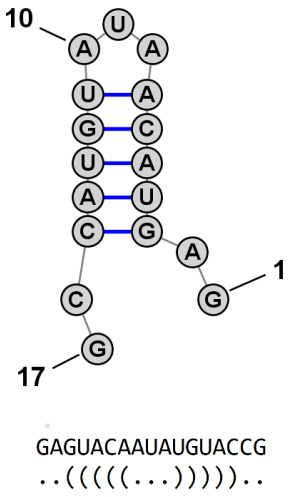
3.3.1 Dot-bracket and CT notation

The *bracket* or *dot-bracket* notation is the most common and most simple representation. In this notation, RNA secondary structures are encoded as linear strings with balanced parentheses representing the base pairs, and dots representing unpaired positions. The simplicity comes with the usual price, which is expressiveness - without post-processing, there is no direct information about the structural context of a nucleotide [59]

Connectivity table (CT) notation is the second unofficial standard for structure representation [28]. It has a form of a table, where each row holds information about a single base in the sequence. The columns (in their respective order) are following [2]:

1. base number (n)
2. base (one of letters A, C, G, T, U, X)

3. index $n - 1$
4. index $n + 1$
5. the number of the base to which n is paired (no pairing is indicated by zero)
6. base number n again



(a) : Dot bracket notation

1	G	0	2	0	1
2	A	1	3	0	2
3	G	2	4	15	3
4	U	3	5	14	4
5	A	4	6	13	5
6	C	5	7	12	6
7	A	6	8	11	7
8	A	7	9	0	8
9	U	8	10	0	9
10	A	9	11	0	10
11	U	10	12	7	11
12	G	11	13	6	12
13	U	12	14	5	13
14	A	13	15	4	14
15	C	14	16	3	15
16	C	15	17	0	16
17	G	16	0	0	17

(b) : CT notation

Figure 3.2: Comparison of the three secondary structure representations - a visual graph, dot-bracket notation and connectivity table. All encode the same structure. In the visual graph we can also notice different edge types.

3.3.2 Tree representation

RNA secondary structures can be represented as trees. A secondary structure is converted into a tree by assigning an internal node to each base pair and a leaf node to each unpaired base (see figure 3.3). The tree is rooted in a node, which does not represent any base-pair, nor a single nucelotide. The virtual root is introduced to prevent the formation of a tree forest. A forest would not form only in rare cases, when start and end RNA bases are paired.

Tree edit distance

A tree is transformed into another tree by a series of editing operations (insert, delete, update) with predefined costs. These costs are also a measure of (dis)similarity between two structures called *tree edit distance*. The distance between two trees is the smallest sum of the editing costs. An interesting side note is that for trees consisting solely of leaves (unstructured string of nucleotides), tree editing becomes standard sequence alignment [35].

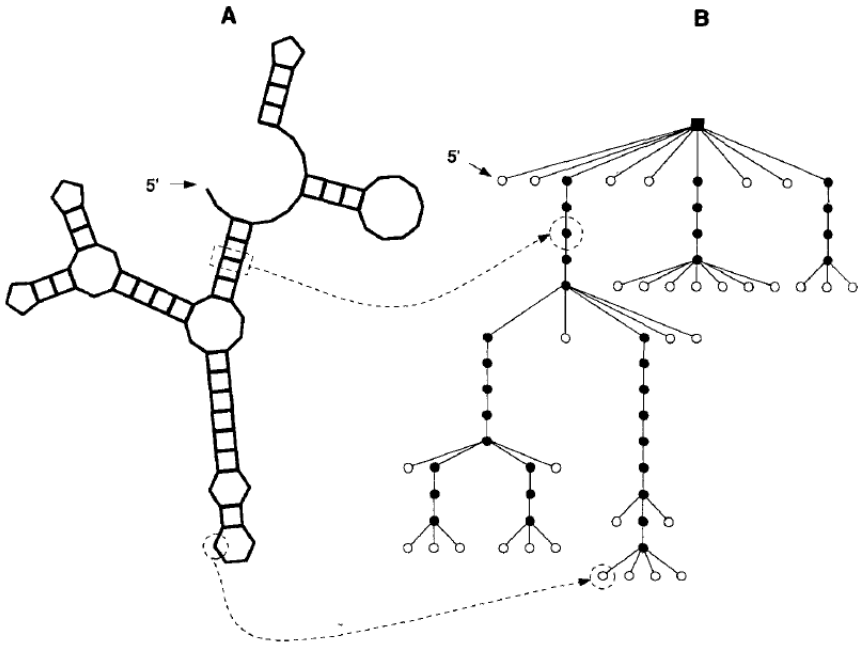


Figure 3.3: The equivalence between a secondary structure (A) and an *ordered rooted tree (B)*. An internal node (black) of the tree corresponds to a base pair, a leaf node (white) corresponds to one unpaired nucleotide, and the root node (black square) is a virtual parent to the elements along the external loop [35].

3.4 Performance metrics for structure prediction methods

Recall and precision

The *recall* or *sensitivity* of a prediction algorithm is the percentage of base pairs in the actual structure (or a structure considered as the ground truth) that are also present in the predicted structure.

The *precision* (or sometimes called *positive predictive value* - PPV) is in contrary the percentage of base pairs in the predicted structure that are in the actual structure [16].

Chapter 4

RNA structure search

As outlined in the biological section, structure of RNA molecules is often important for their function and regulation. Discovering a particular functional structure within a genome could then be viewed as discovering the associated function. Searching for structures or structural motifs can then enhance the understanding of RNA functional and regulatory properties.

In this field the most well known program is *RNAmotif* [53]. It performs structural search in primary sequences and uses its own descriptor language to characterize the structure to be queried. Another approach is manifested in *NA2Dsearch* [34], which is a CUNI's internal tool employing similar descriptor language as *RNAmotif*, but differentiates itself by performing the search in secondary, instead of primary structures.

In essence, the named tools alone can be employed for discovering IRES-like motifs within human genome. The later, for instance, has a sliding window mode of working. It can scan through the entire genome to obtain hits. Computational efforts would be however enormous - the human DNA is 3 billion base pairs long and RNA *folding* (a necessary step discussed later) is $O(N^3)$ in complexity. For those reasons, presented methods will only be one component of a solution proposed in this thesis.

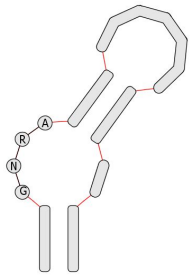
4.1 RNAmotif

RNAmotif's work-flow can be divided into three stages. In the first stage, user defines a queried structure using program's rich descriptor language; global parameters, such as what base-pairs are permitted, can be also defined. Next, the actual search within provided sequences is conducted. A currently examined sequence is checked for compatibility with the query structure (i.e. checked, whether any forbidden base pair is present, when the sequence is "fit" into the structure). This step can result in multiple hits. Lastly a *scoring* stage processes resulting hits, applying filters, which could not be performed during the search (e.g. GC content requirements, structure stability), and ranking them. This final step's purpose is to limit the number of possible solutions to biologically meaningful ones.

RNAmotif's main feature is its descriptor language. Its building blocks are symbols for *single strands* and *helices* - *ss*, *h5* or *h3* (*h5* denotes the 5'

```
params
    wc+=gu;
descr
    h5(minlen=1,maxlen=7)
        ss(seq="^GNRA$")
        h5(minlen=1,maxlen=1)
            ss(minlen=3,maxlen=7)
        h3(minlen=1,maxlen=1)
        ss(minlen=2,maxlen=2)
    h3(minlen=1,maxlen=7)
```

(a) : *RNAmotif* descriptor



(b) : *NA2Dsearch* query

Figure 4.1: The *RNAmotif* descriptor with two out of three sections (**params** and **descr**; **score** omitted). In parameters section, GU pairing is allow globally. In description section we can notice *ss* and *h* elements with sequence constraints (N and R are IUPAC codes for 'any nucleotide' and 'A or G' respectively). The second picture shows the *NA2Dsearch* query corresponding to the descriptor

strand of a helix and *h3* the 3' strand). An simple example of a structure described by these blocks is shown in the figure above.

The language outclasses its predecessors in length and complexity of definable structures. One can define, in addition to the mentioned parameters, also nucleotide and length constraints for *ss* and *h* elements or custom scoring rules. The rules provide some form of compensation for the apparent drawback, that the structural search is performed on sequences.

4.2 NA2Dsearch

This drawback has motivated a development of *NA2Dsearch*, which prefers the idea, that structures should be searched within structures. The descriptor language is otherwise to a large extent similar. The scoring system evaluates the hits on how much they violate the target structure represented by the query. It introduces various penalties for extra or missing bases, base pair, bulge openings etc. The scores are therefore most of the time negative.

The method itself has to deal with one large obstacle - it conducts searches in structures, but verified RNA structures are not easily available. Inputs will be often only primary structures which then forces the program to estimate their structure computationally. It does so through a *RNAsubopt*[49] routine. Since *NA2Dsearch* will be later incorporated to our pipeline and since, in turn, RNA structure prediction is such a crucial step in the tool, we discuss sequence folding in the following chapter [1].

This search tool is considerably slower, than *RNAmotif*. It is however expected, given that sequences need to be folded first. On the other side, from the user perspective, it is incomparably more convenient to work with *NA2DSearch*, thanks to its GUI with drag-and-drop capabilities.

[1] In fact, structure prediction plays a considerable role also in the first step of the pipeline, in *inverse folding*

Chapter 5

RNA structure prediction

The RNA structure is formed by folding the chain of ribonucleotide bases involving multiple acting forces - mainly base-base interaction through hydrogen bonds with addition of hydrophobic effects and electrostatic interactions between the negatively charged phosphate groups with ions in solution.

The number of sources affecting the resultant fold makes the full structural prediction problem extremely difficult. RNA has however shown, that it is a reasonable approximation to focus on the secondary structure, where only bonds between pairs of nucleotide bases are considered. The results have been proven useful in the following tertiary structure prediction as well as biological function of RNA [60].

In this section, works on these methods will be summarized.

The landscape of the RNA prediction approaches can be roughly divided into three categories based on biological data available as an input. The first category is the most straightforward one an requires only an RNA sequence to obtain a prediction. Due to the lack of biological constraints that could direct the prediction process, methods from this category tend to be inaccurate. The second category has information on the structures of other potentially similar RNAs and exploit this RNA homology data to guide the prediction process. Finally the third and the most recent category outputs RNA structure from a knowledge of so called *structure profiling* measurements. This measurement can quickly tell, to what degree is each nucleotide part of a stem, or a loop.

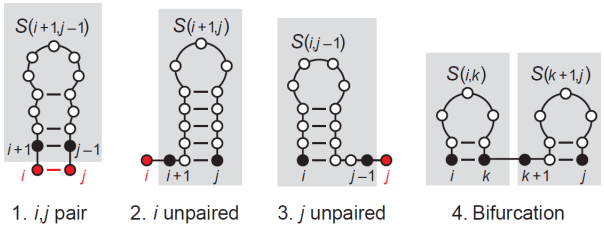
5.1 Free Energy Minimization methods

In a previous chapter, interactions between RNA bases A–U and G–C forming base-pairs, were described. Each such a pair lowers the thermodynamic free energy of the structure (also called the *Gibbs free energy*). The first group of structure prediction algorithms we are going to discuss here, then builds on an assumption, that ribonucleotide chains favor conformations with the minimal such free energy (and maximal stability) [47]. Since these approaches assume the bare minimum about the resulting structure, they often serve as a baseline for accuracy comparison with other, more informed folding methods.

5.1.1 From Nussinov towards minimum free energy methods

The task of finding a secondary structure can be reduced to a problem of finding the structure with the maximum number of base pairs. Although this criterion is too simplistic, this is in essence what almost all other complex energy minimization and stochastic context free grammar (SCFG) approaches to at least some degree attempt to achieve [20].

This base-pair maximization method was implemented in the early Nussinov algorithm, a fairly straightforward dynamic programming algorithm, whose recursion steps are visualized and formalized below:



.

Figure 5.1: Schema of how dynamic programming algorithm decides the optimal structure of a sub-sequence *i,j*. It considers all 5 possible conformatinos of nucleotides *i,j* and chooses the one with the best score [21]

$$S(i,j) = \max \begin{cases} S(i+1,j-1)+1 \text{ if } i,j \text{ base pair} \\ S(i+1,j) \\ S(i,j-1) \\ \max_{i<k<j} S(i,k) + S(k+1,j) \end{cases} \tag{5.1}$$

The fill step is $O(L^3)$ in time. The trace-back is linear in time

Simple base pair maximization however, is a poor scoring scheme for RNA structure prediction. An RNA more likely adopts a globally *minimum energy structure*, rather than the structure with the maximum number of base pairs [21]. Taking a hairpin as an example - increasing the number of pairing will narrow and stretch the ending loop. This is clearly not the minimal free energy state [56]. As we see further, even the minimum energy state assumption is too simplistic

To adjust to this new minimum energy paradigm, the dynamic programming has to be modified. The score (previously equivalent to the number of base-pairs) now takes form of an overall *free energy of a secondary structure*. The terms contributing to this score (previously ones and zeros indicating paired/not paired) then take form of *independent energy increments* of different loops and base pairing interactions.

A thermodynamic model (outlined in a following section) has been developed in conjunction with the dynamic programming folding algorithms to estimate these individual increments. With the described adjustments, structure prediction algorithms become somewhat complex with more detailed

recursions, that distinguish different lengths and types of loops and base pair interactions, since each of them contribute differently to the structure's free energy [56].

5.1.2 Free energy nearest-neighbour minimization

To predict the folding free energy of a given secondary structure, an empirical *nearest-neighbor model* is used. The name *nearest neighbor* comes from the assumption, that the free energy change for each motif depends on its sequence composition and only the most adjacent base-pairs. The order of the base-pairs is also important here - e.g. CG-GC pairs will result in different stability, than GC-CG. This is called *stacking context*.

The free energy change at $37°C$ is denoted as ΔG_{37} and the sample calculation is shown below.

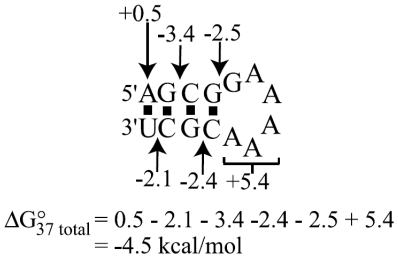


Figure 5.2: The base-pair stacking are each favorable with various negative increments. The helical model gives a +0.5 kcal/mol penalty for each AU or GU pair that terminates a helix. Lastly, a +5.4 kcal/mol increment for hairpin loop closure is an entropic penalty for constraining nucleotides in a loop [56]

The most used programs with the nearest-neighbor technique incorporated in one way or another include *Mfold* [86] (currently evolved to *UNAFold* [54]), *RNAStructure* [58] and *RNAsubopt* with *RNAFold* [35], which are the part of the well known *Vienna package*.

Taken *Mfold* as a representative, the accuracy the MFE methods would fall somewhere between 40 to 70% depending on the type of RNA sequence (e.g. ribosomal or tRNA) [30]. This is useful for many purposes, but not as reliable as we need.

Further benchmarks of the methods above did not reveal any significant and consistent superiority of one implementation over the others. [49]

Energy models

The thermodynamic parameters for predicting the free energy change of base pairs and loop regions have been compiled based on empirical data (melting experiments) and adjusted by various computational techniques such as linear regression or comparative sequence analysis [57] (details on comparative folding in section 5.2).

Up to date, two standard energy models are used - Turner'99 and Turner'04. It is widely accepted, that Turner'04 are more accurate, even though for some cases the '99 one has proven to be a better choice [27]. In addition to these two models, there is a third, distinct one. This model was assembled by creators of a curated database of secondary structures *RNA STRAND* [6]. Here the the thermodynamic parameters were determined by applying maximum likelihood parameter estimation on a set of 2200 experimentally verified RNA structures. With the new set of parameters a folding program *SimFold* (equivalent to *mfold 3.1*) has achieved remarkable improvements in prediction (8 and 10% increase in sensitivity and selectivity) in compare to Turner'99 parameters.

5.1.3 Sub-optimal folding

We have seen, that determining an RNA secondary structure by free energy minimization has its limits in accuracy. First, the principle of nearest-neighbor independent stacking of energy increments does not completely hold - there are experimental evidences, that stability of motifs depend also their sequence composition (*non-neighboring effects*). Some dynamic algorithms at least take into account several such motifs [57]. Furthermore not all RNAs are in the minimum free energy equilibrium due to folding kinetics; some functional RNAs have even bistable structures, each having a different catalytic activity.

Given all above limitations, it is clear, that the predicted lowest energy structure does not always provide the most accurate solution and that sub-optimal folding would bring significantly more insight to the true strucutre [56].

Statistical sampling with partition function

First, McCaskill [60] introduced his *partition function*

$$Q = \sum_{S \in structures} e^{\Delta G_{37}(S)/RT} \tag{5.2}$$

over all possible structures S of a given sequence with $\Delta G_{37}(S)$ being the energy of the structure S, R the gas constant and T temperature.

Having the partition function, it is possible to evaluate probability of a particular secondary structure σ:

$$p(\sigma) = \frac{1}{Q} e^{\Delta G_{37}(\sigma)/RT} \tag{5.3}$$

Since the number of structures increase exponentially with sequence length, the function is partitioned for dynamic programming computation. The partition $Q_{i,j}$ is determined for all sub-sequences between positions i and j, starting with the shortest sequences, filling the dynamic programming table. One can then use the partition function values and calculate a probability of each base pair i, j being paired.

This was further developed by Ding [16], who modified the traceback procedure from MFE algorithms to utilize the base-pair probabilities. Instead of choosing base-pairs deterministically, base-pairs are chosen probabilistically based on values determined previously by the partition function. The set of predicted secondary structures is a statistically representative sample of the complete *ensemble of structures*.

An *ensemble centroid* is introduced to help picturing the landscape of predicted structures space. The centroid is defined as the predicted secondary structure with the least total base-pair distance (number of base-pairs in which two structures differ) to all the structures in the set. Authors also distinguish individual cluster centroids.

With ensemble centroids, significant accuracy improvements have been achieved compared to the MFE structure. The number of correctly predicted base-pairs (selectivity) increased by 30% (the improvements in sensitivity were marginal).

Another interesting finding regarding MFE structure and clusters - in only 45% of sequences is the structure present in the most dominant cluster, indicating that the MFE structure does not always represent well the ensemble [56].

5.1.4 Troublesome pseudoknots

Lastly, it is worth mentioning the greatest limitation of the discussed algorithms. In addition to nested stem-loop base pairing interactions, RNA can also make nonnested base pairs a so-called RNA *pseudoknot*. It is formed when bases outside a hairpin structure pair with bases within the hairpin or internal loop [23] [21].

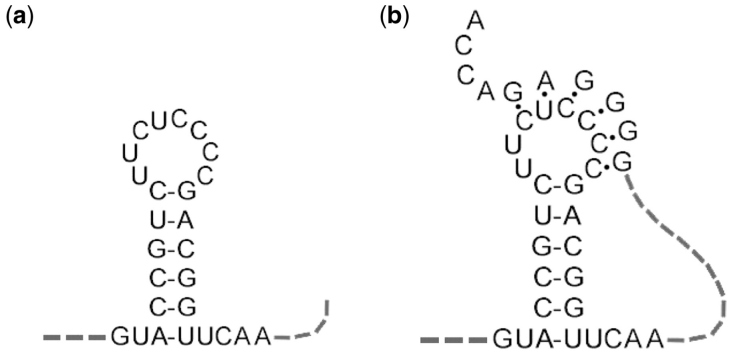


Figure 5.3: Stem-loop and pseudoknot comparison [75]

Pseudoknots is what the dynamic programming algorithm cannot deal with - it has not kept track of which nucleotides are available for pseudoknot pairing and which ones are already a part of a stem [21]. In fact determination of the MFE pseudoknoted structure was proven to be NP-complete [52]. There are RNA folding algorithms that deal with pseudoknots, but each of them has serious limitations of its own (e.g. the $O(N^6)$ scaling) [21].

It would be therefore tempting to omit pseudoknots from consideration as a rare structural element and focus on less constrained and faster folding methods. However, according to the data from RNA STRAND, a carefully curated database of secondary structures, pseudoknots occur rather commonly, especially in longer molecules - 74% of all entries with 100 or more nucleotides contain pseudoknots [6] . Even though the content of pseudoknoted base-pairs is on average relatively low (<2%) [57], their presence can significantly modify overall structure.

5.2 Comparative folding methods

In the previous section, obastacles limiting the accuracy of single sequence prediction methods were outlined - folding kinetics, bistable structure, base modifications to name few. This is why researchers, when possible, always resort to comparative methods [29].

5.2.1 Multiple sequence comparative methods

This family of methods was the first one developed for purposes of secondary structure prediction. This knowledge-based technique assumes, that homologous RNA sequences fold into similar secondary and tertiary structure, while their primary structure can change significantly [17]. The idea is then to determine *canonical pairs*, that are common across multiple homologous sequences.

Comparative folding can be considerably accurate, when sufficient homologous sequences are available (over 97% of predicted base-pairs in ribosomal RNA were experimentally confirmed [30]). The greatest limitation is then the requirement for multiple sequences, which are not always available and which require significant insight [56].

Gardener and Giegerich [29] categorized the existing comparative methods into three *plans*. Each plan suited for different sequence conservation degrees of underlying homologous RNAs. A graphic outline of the categorization is shown on figure 5.4.

Plan A: Folding of a sequence alignment

Plan A first obtains sequence alignments from standard multiple sequence alignment tools (e.g. ClustalW). Then using this knowledge derives a consensus structure, usually with the combination of MFE fold and *covariation score* [29]. CM is considered one of the most accurate prediction methods, but it requires large numbers of orthologous sequences that can be unambiguously aligned [65]. And this the major pitfall of Plan A - multiple-alignment step assumes highly preserved primary structure, which is often not fulfilled in fast evolving non-coding RNA. Incorrect sequence alignments can then blur any covariation signal. This plan is represented by *RNAalifold*, *Pfold* and

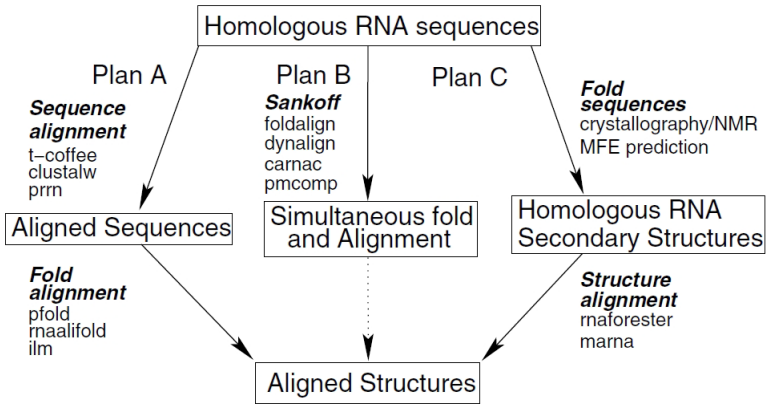


Figure 5.4: Overview of the three main methodology classes for solving the comparative folding task. Each class assumes a certain level of conservation in homologous sequences, for which the structure is estimated. Plan A expects high level of sequence conservation. Plan B expects weaker sequence conservation, but rather well preserved secondary structure. Plan C is used when sequences are highly variable [29].

ILM. Note, that only the most successful programs will be briefly discussed; for descriptions of other ones please refer to the original review.

Plan B: Simultaneous folding and alignment

This is when Plan B steps in. Here, sequence alignment and folding is done simultaneously for a set of homologous structural RNA sequences - an approach realized in the *Sankoff algorithm*. It combines sequence alignment and Nussinov (maximal pairing) folding. The algorithm requires extreme computational resources ($O(n^{3m})$ in time, and $O(n^{2m})$ in space, where n is the sequence length and m is the number of sequences. Practical implementations (*Carnac*, *Foldalign*) of the algorithm restrict sizes and shapes of substructures to make the task feasible.

Plan C: Structure alignment

Researches resort to Plan C, when no helpful level of sequence conservation is observed. The sequence alignment step common for the previous plans can be then skipped altogether. Instead, structures of the sequences are predicted and directly aligned. Structure alignment is built on the property of RNA secondary structures, which can be represented as trees (see 3.3.2). Similarity between two structures is measured in terms of tree edit operation of their corresponding tree representations.

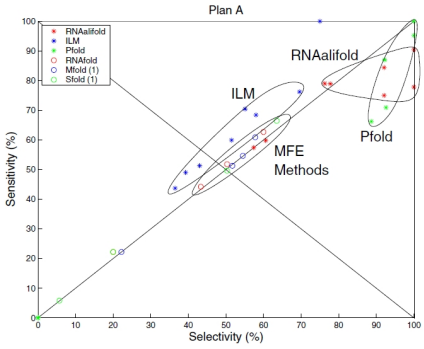
One can immediately spot a weak point of this plan. Folding tends to be rather inaccurate and it is a question whether this step produces structures aligning well enough to unveil the hidden consensus structure when one exists.

Representatives of this plan are *RNAforester* and *MARNA*

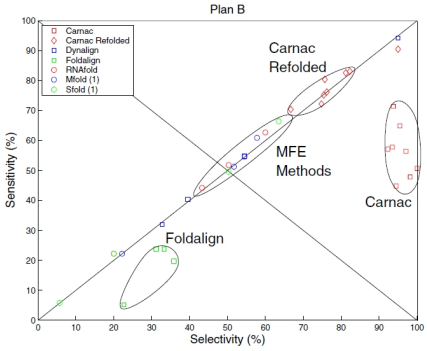
Plans comparison

Gardner has compiled a test data-set to benchmark implementations of all plans. In addition, energy minimization methods discussed earlier were included as baseline methods. He has articulated the need for objective and impartial testing, since authors of algorithms tend to present their work in a slightly better light, than it actually is.

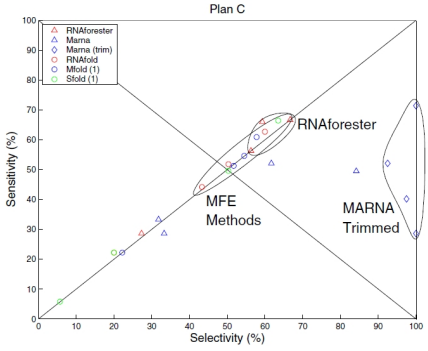
The choice of data-set sequences was aimed to truly test the limits of bench-marked algorithms. Most of them are difficult for structure analysis, including ribosomal RNA (shape influenced by ribosomal proteins), RNase P (little sequence and structure conservation; pseudoknots) or transfer RNAs (contain modified bases, which influence structure formation).



(a) : Plan A



(b) : Plan B



(c) : Plan C

Figure 5.5: Benchmark results for comparative folding plans. The plan's accuracy was measured in terms of an ROC plot displaying both sensitivity and selectivity. Values for MFE methods are added for comparison [29].

RNAalifold and *Pfold* are clear winners. For well aligned sequences the speed and performance of *RNAalifold* was excellent, reaching high scores in both selectivity and sensitivity. *RNAalifold* is a implementation of an extended version of Zuker-Steigler algorithm (see section 5.1.2) for computing consensus structure from RNA alignment. The algorithm is remarkably efficient $O(N \cdot n^2 + n^3)$ in time and $O(n^2)$ in memory.

Pfold, another member of Plan A, has achieved similar accuracy despite being built on a different paradigm, than *RNAalifold*. The program imple-

ments a *stochastic context free grammar* (SCFG) designed to produce a prior probability distribution of RNA structures for an RNA alignment input.

As expected, pan C did not perform well. An notable observation is however worth mentioning - both programs from this plan perform better on the medium similarity data than on high similarity data. The reason behind this paradoxical statement lies in the fact these programs work with predicted structures. If sequences are very similar, they may jointly fold into the wrong MFE structure. With greater sequence variation then rises a chance some structures have good predictions. This means that especially in the case of low sequence similarity, where nothing else works, plan C has a certain promise [29].

5.2.2 Template-based prediction

This family of comparative methods aims to determine an unknown structure through an initial *structure template*. The template can be viewed as a consensus structure from the previous algorithms. The only difference is, that the consensus structure the methods in Plans A to C tried to obtain in one way or another, is in this case assumed to be given. This allows structure prediction with just one sequence as an input and no need for alignment. It may seem as an improvement to previous comparative approaches. One has to however realize, it is not structure prediction in the true sense, since the approximate shape has to be known beforehand.

One method tries to map an input RNA into a template structure, while keeping consistent parts and folding the inconsistent ones. The authors summarized the procedure in a graphic chart below.

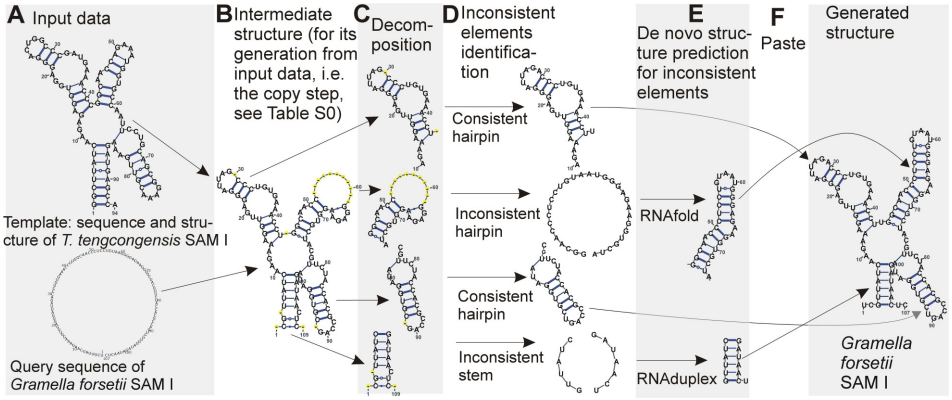


Figure 5.6: First, a pairwise alignment of the query and template sequence is computed. The template structure is then mapped into the query sequence to form an intermediate structure, which preserves base pairs that the alignment maps to complementary nucleotides. All other bases are marked as unpaired. Second, the intermediate structure is decomposed into basic structure elements: individual hairpins and stems. Inconsistent structure elements are identified and *de novo* predicted [66].

This method is not only useful for predicting RNA secondary structures,

but also for estimating an ability of sequences to adopt the investigated structure.

TRAVeLer

TRAVeLer is strictly speaking not a structure prediction tool. It is primarily a visualization tool, which lays out complex secondary structures with a guidance of a homologous structural template. It solves an issue when long, highly structured RNAs are being poorly visualized with stems and loops overlapping each other, making the whole representation unreadable. The template structure then serves as a skeleton for a correct layout.

Despite being aimed for visualization, TRAVeLer can be used in secondary structure prediction. It is useful in moments when multiple predictions of the same sequence need to be visualized in a consistent manner to enable visual analysis of differentially predicted regions.

To briefly outline TRAVeLer's work flow: at the beginning, it converts the target and template structures into their corresponding tree representations. Secondly, tree edit distance (TED) is used to obtain mapping between the trees. - apart from telling a score telling the dissimilarity of the input trees, TED can generate a minimal sequence of tree edit operations (insert, update, delete) which turn a template tree into the target one. Following those opperations will then lead to construction of an appropriate RNA structure layout [22].

5.3 Methods using RNA structure profiling data

RNA structure profiling is a bio-chemical measurement, which serves to provide insights about RNA secondary structure at nucleotide resolution. This is achieved through different reactivity of profiling reagents with nucleotides in single-strand and double-strand conformation. Based on the measurements, each ribonucleotide can be then assigned a quantitative degree, to which it is single-stranded or double-stranded. Recent improvements to this method enabled higher rate of processed RNA, providing structural profiling on genome-wide level. On the figure below are demonstrated principles of this improved profiling method titled *Parallel Analysis of RNA Structure (PARS)*.

The measured single/double-stranded conformation estimates can be then fed into *constraint folding algorithms*. These are folding algorithms that use profiling data to restrict the predicted RNA structure [41]. With increasing availability of high-throughput secondary structure profiling data, it is ever so important to do so [65].

SeqFold

SeqFold grasps this idea - it incorporates data from multiple profiling methods and integrates them with existing computational approaches. For instance, it has achieved considerable improvement to an MFE program *RNAStructure*

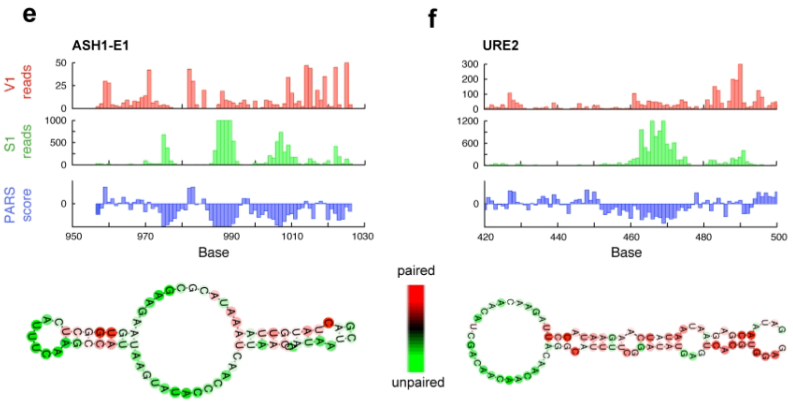


Figure 5.7: (e-f) Raw number of reads obtained using profiling agents RNase V1 (red bars) or RNase S1 and the resulting PARS score (blue bars) along one inspected domain of ASH1 (e) and URE2 (f). Also shown are the known structures of the inspected domains with nucleotides color-coded according to their computed PARS score

predictions when folding of certain RNAs where constrained by SHAPE [1] reactivity profiling data.

The ultimate goal is to take genome-wide, noisy experimental data directly as input without manual curation. However promising this idea is, it comes with a significant issue. Recent analyses indicate, that even dozens of focused SHAPE experiments on individual RNAs may not generate sufficient measurement density for such modeling strategies leading to considerable modeling errors.

Authors propose an approach to counter this noise sensitivity of MFE methods. They sample from the given RNA's Boltzmann weighted ensemble and cluster the structures. They then compare the experimental *structure preference profile* (an array of single/double-strand signal values for each nucleotide - SPP) with an SPP of sampled structures. The centroid of the nearest cluster is then set to be the predicted structure. The proposed framework indeed increased the prediction robustness to noise in compare to the MFE approach. It was also highly successful in predicting ncRNA covariance model-based structures (sensitivity/selectivity of 0.88/0.78) in conjuction with previously mentioned PARS structure profiling method/data [65].

[1] from *Selective 2'-hydroxyl acylation analyzed by primer extension*; biological details on this profiling method available at [80]

Chapter 6

RNA inverse folding

RNA *inverse folding* is a reverse problem to the structure prediction. So far, the problem was to find a correct structure for a given sequence. In case of inverse folding however, the task is to search for sequences capable of attaining a predefined given structure. I.e. finding sequences from a structure as an input. To be precise, the criterion is for these sequences to form the given structure as their minimum free energy conformation. For that reason, the majority of algorithms employ MFE folding in some step of their work flow. Some of the programs even go as far as directly plugging in existing MFE solvers (e.g. *RNAfold*). This makes them dependent and also susceptible to thermodynamic parameters alike the single sequence folding programs [14].

6.1 Implementations

There are several strategies for solving the RNA inverse folding problem - *RNAinverse*, *RNA-SSD* [7], *INFO-RNA* [11] or *NUPACK* [83] to name few. The common approach in all of them is to start with an arbitrary *compatible sequence* or *seed sequence*. Compatible in a sense, that it can form base-pairs wherever the input structure requires. Being compatible however does not mean the sequence will actually fold into the structure. Therefore, the seed sequence is mutated in various ways to minimize an objective function (e.g. a distance between the target structure and the structure of the tested sequence) [25].

Named programs are not the complete list of inverse folding solutions. For the purposes of the thesis, not all of them can be examined, but a more narrow selection has to be made. *Churkin, et al., 2017* [14] have reviewed almost all approaches and from their results the selection will be made. The criteria for the program choice are the following: the method has been used in 'wet laboraty' experiments and it internally works with the partition function (for reasons outlined in the section 5.1.3). From the named programs, *RNAinverse* and *NUPACK* fit these criteria, together with *RNAiFOLD*, a program differing considerably from the previous two in its approach. We will look at each one more closely.

RNAInverse

Written in the 90's, the pioneer in this field is *RNAInverse* [35]. The objective function, the internal algorithm is minimizing, is the tree edit distance between two structures - the MFE structure of the tested sequence and the target structure.

From observations that sub-structures contribute additively to the total energy of the structure, the program proceed to optimize small structural elements first, which effectively reduces the number of full sequence folding.

For the optimization itself, *RNAInverse* introduces random mutations to the initial sequence, accepting those reducing the objective function.

As for input, the program accepts a structure of interest in the dot-bracket notation. User can further force certain positions to hold a fixed nucleotide - the output sequences will then vary in all but the constrained positions. Generated sequences can be assembled based on the *MFE criteria*, on the *partition function criteria* (see section 5.1.3) or based on both.

Listing 6.1: Example of RNAinverse input and output. The structure on the first line is constrained to have a GC pair closing the stem and a left bulge to contain only adenines. Letter N indicates no restriction on nucleotides. The returned output follows after a newline

```
((((....((.(((((.........)))))).)))))))
gNNNaaaaNNNNNNNNNNNNNNNNNNNNNNNNNNNNc

length = 37
gGCUaaaaUCGCGCAACGCCCCCCCUUGCGGGAGGUc
```

RMAiFOLD

Together with *RNAInverse*, *RNAiFOLD* [26] is another inverse folding program, that have been used in biologically meaningful way to precede 'wet laboratory' experiments [14]. Internally however, *RNAiFOLD* works on a different principle. It employs constraint programming to arrive at a target objective - a sequence folding into a given structure.

Two modes of running are available - *CP (Constraint Programming)* implementation, that performs an exhaustive exploration of the search space and *LNS (Large Neighborhood Search)* implementation, which is heuristic and calls the CP sub-routine on smaller tasks. This method hierarchically breaks down the structure and mutates partial solutions just as the previous group of programs, here however the differences between iterations are greater (thus the name "Large Neighborhood"). Due to its heuristic nature LNS cannot prove no solution exists (as CP can), but is more suited for larger structures as it has shorter runtimes.

For target sequences, *RNAiFOLD* is able to optimize against one or more of the following criteria: *MFE structure* (solutions minimum free energy structure is the target structure), *minimum free energy* (minimize solution's free energy when folded into target structure; relaxed first criteria) and

minimum ensemble defect (the application of partition function; more closely described on following pages).

The program, thanks to its constraint programming design guarantees optimal solution (within defined constraints) and can, given enough time, prove no solution exists, in the case that none does exist. *RNAiFOLD* stands out also for encompassing Andronescu's energy model, which may provide more accurate folding results as indicated previously in 5.1.2 .

NUPACK-design

NUPACK-design [83] works out the idea set by *RNAinverse* - an iterative mutation of a candidate sequence, accelerated by hierarchical structural decomposition to avoid full-length objective function calculation. After a discussion on the validity of various objective functions (including the ones involving MFE) the authors propose a so called *ensemble defect minimization*. This objective function represents the average number of incorrectly paired nucleotides over the ensemble of structures Γ. The ensemble defect of a sequence ϕ and its particular structure s is denoted as $n(\phi, s)$ and defined as:

$$n(\phi, s) = \sum_{\sigma \in \Gamma} p(\sigma, \phi) d(\sigma, s) \tag{6.1}$$

where $p(\sigma, \phi)$ is the probability of a structure σ derived from the partition function as described in section 5.1.3 and $d(\sigma, s)$ is the distance between secondary structures σ and s measured in the number of nucleotides paired differently.

The program seeks to design a sequence ϕ of a length N with ensemble defect satisfying the stop condition:

$$n(\phi, s) <= f_{stop} \cdot N \tag{6.2}$$

with $f_{stop} \in (0, 1)$ being the user specified value (0.01 by default).

As for general framework of the program - *NUPACK* starts from an arbitrary sequence compatible with a required structure. The structure itself is decomposed into a tree, with leafs nodes being individual sub-structures. Sequences for those sub-structures are optimized independently by defect-weighted sampling (positions causing the most defect has the biggest chances of being mutated). An emerging sequence is further optimized during leaf merging, when the sub-structures are joint together.

Comparison

Each presented program represents a distinct philosophy in inverse sequence generation. While *RNAinverse* and *NUPACK* are heuristic, *RNAiFOLD* is exhaustively exact. *NUPACK* in turn differs from *RNAinverse* with its ensemble defect minimization. At this point we cannot say, which one is more suited for our application - thorough experimental comparison of the programs will be given in the practical part of the thesis. In general however, the upper range estimate for the sequence length that these programs are useful for is around 150 nucleotides. [14]

Chapter 7

BLAST

BLAST is arguably the most popular tool in computational bioinformatics. It features similarity search in sequence databases. The strength of BLAST comes from the fact, that this search has an exceptionally favorable trade-off between time complexity and sensitivity. It can well approximate results of dynamic programming methods (Needleman-Wunsch [64]) while being by an order of magnitude faster. This is mainly thanks to its heuristic design, which will be described below. Two versions of the algorithm will be discussed - the original one from 1990 (BLAST90) and the refined version from 1997.

7.1 BLAST principles

BLAST[4] does not attempt to find an optimal score as alignment methods do - this would be too slow for database searches. Instead, it splits a query sequence into *words* of fixed length and scans database for words, that score at least T against some of them. The score between two words is calculated using *substitution matrices*, e.g. BLOSUM [33]. Hits are then extended character by character in both directions, until the score drops too much under the best score seen so far. This way a *high scoring segment pair* (*HSP*) is obtained. In this process, the hit extension is the most costly operation. Some of the extension operations will be "wasted", meaning they lead to score drop. The number of wasted extension can be reduced by tuning the score cut-off T and word length w values. Authors empirically found $T = 17$ and $w = 4$ to be the best, even though different values are used today[19].

BLAST attempts to collect all HSPs with *significant* enough scores. The significance is expressed in terms of so called *e-value*.

7.1.1 E-value

Let's have a query sequence of length m with a hit scored S against some database. E-value of this query is then defined as the expected number of HSPs with score at least S obtained by searching a random sequence of length m. In other words, e-value can be understood as the number of false positives expected from BLASTing a query of length m, having the score S. The

formula is the following:

$$E = Kmne^{-\lambda S} \tag{7.1}$$

where n is the database size in number of nucleotides/aminoacids and K with λ can be viewed as parameters for search space size and scoring system [19] [40].

7.2 Gapped two-hit BLAST

BLAST97 [5] is an evolution of BLAST90, bringing new features and faster search. In total, three major improvements were implemented. The speed enhancement was achieved by modifying the extension step - the most time consuming step. In this version it will not be triggered until two words are found in the same diagonal; the words also need to be within a certain distance (see figure 7.1).

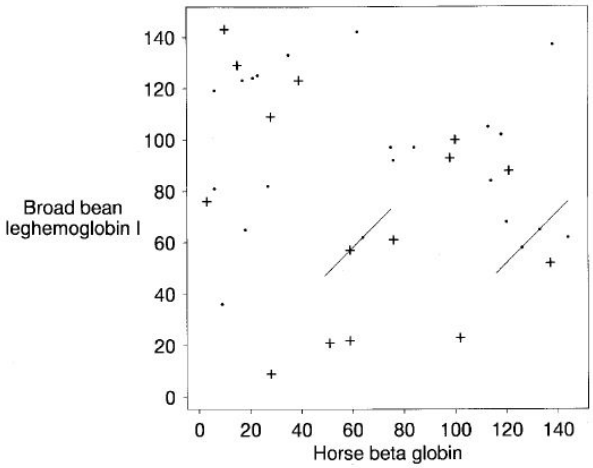


Figure 7.1: Example of the new hit extension policy. Two globins are compared. 15 hits with score at least 13 are indicated by plus signs. Additional 22 hits with score at least 11 are indicated by dots. Of these 37 hits, only the two indicated pairs are on the same diagonal and within distance 40 from each other. Thus the two-hit heuristic with T = 11 triggers two extensions, in contrast to the 15 extensions invoked by the one-hit heuristic with T = 13 [5].

7.2.1 New features

Two features added are first the ability to generate *gapped alignment*, and second, incorporating of *position-specific score matrix (PSSM)* into the search process. The later is only the domain of proteins and is presented only for the sake of completeness.

Gapped alignment was already considered in the original BLAST. Authors decided to leave it out for being an unfavorable trade-off for speed. Now, the gap extension is allowed, however only on a limited number of HSPs, that

exceed a moderate score $S_g > T$. Therefore, even though still time-consuming, this limitation keeps the complexity reasonably low. Together with speed improvements thanks to the two-hit modification, the gapped extension can be newly effectively used.

PSI BLAST

Inclusion of position-specific score matrices (also *profiles*) was motivated by their ability to detect weak relationships in compare to simple sequence queries. The matrices embody both the query and the substitution matrix in a sense, that a query of length L and a substitution matrix with dimensions 20×20 (aminoacids) are replaced by a PSSM of dimensions $L \times 20$. In the PSI BLAST (Position Specific Iterated BLAST), the first iteration constructs such a PSSM from the candidate matches and in subsequent iterations the algorithm compares the candidates with the PSSM from the previous iteration [19].

7.3 BLAST applications

BLAST has been re-implemented in the BLAST+ suite[12], improving usability and efficiency in certain cases (optimization for long queries, extended command line options, etc.). It further splits the program into separate command line applications based on types of molecules of query and subject - *blastn* and *blastp* are manifestations of the original algorithm for searches in nucleotide and protein sequences; *blastx* translates a nucleotide query for protein searches; *megablast* [84] optimized for closely related sequences (which e.g. differ only by sequencing errors). In compare to *blastn* this application can use larger word size, for which it is by an order of magnitude faster.

For the purposes of this work, the standard *blastn* will be used.

7.4 BLAST databases

Sequences are always searched against some database. The database can be one of pre-formatted options such as *nucleotide collection* (*nt*), or can be set up by the user from his/hers own set of sequences. Pre-formatted BLAST databases govern records from archival and annotation projects such as *GenBank* (database joining several sequencing projects)[10] or *RefSeq* (non-redundant, curated RNAs sequences; shares some sources with *GenBank*)[69]. Individual databases can be further dedicated to a particular sequence type (protein, RNA, genomic) or/and to particular organisms (human, mouse, viral, etc.). Apart from the general *nt* database, examples can be extended to *other_genomic* (genomic records of lower eukaryotes) or *refseq_protein* (protein sequences from *RefSeq* project). The complete list available at `https://www.ncbi.nlm.nih.gov/books/NBK62345/`.

Chapter 8

Previous work

Several attempts were made in this area to computationally locate unknown IRES-like elements. Each of them took different approaches utilizing various machine learning and statistical techniques (e.g. support vector machines or logistic regression). Four methods will be summarized in following pages.

8.0.1 Search using inverse folding

Previously, an RNA inverse folding program *RNAiFold* was presented (section 6.1). The creators then put their program to the test. They believed, that since the sequence conservation of IRES elements is low and at the same time its functional dependence on the structural conservation is high, it is worth generating a large number of possible IRES sequences and checking, whether some occur in nature. The picornavirus IRES was chosen as a model, specifically its sub-domain III, an essential region driving IRES activity.

Given the viral IRES domain structure, over 100 000 sequences were generated. The sequences were constrained to contain conserved C-rich and RAAA motifs (IUPAC code R standing for base A or G). The structure and sequence constraints used as input to RNAiFold were the following:

Listing 8.1: Picornavirus IRES domain structure and nucleotide constraints on generated inverse sequences

```
(((((((.....)))(((((((((((.....)))))))))))(((((.......)))))))))
NNNNNNNNNNNNNNNNNNNNNNNNNNNNRAAANNNNNNNNNNNNNNNNAACCCCANNNNNNNNN
```

To reduce search time, the sequences were clustered via a greedy algorithm and random representative of each cluster was BLASTed. Afterwards, the obtained BLAST results were filtered by *keywords* (hits containing words like ribosome, transcription, kinase, etc.) and *location* (5'UTR, beginning of the coding region). From the final 6 candidates, a coding region of *Drosophila melanogaster* (see figure 8.1) was identified as the most fit for experimental testing to confirm or reject the possibility of the region harboring an IRES-like subdomain structure. The *in vitro* tests showed weak, but possitive internal initiation of translation, indicating the subdomain could be present [18].

> **Drosophila melanogaster TBP-associated factor 6 NM_079437.4**

ctcttttgggtgtgtgcacactggccgagcagctcggcgattattatttgtttatttgtaaaaacctgggaa**atg**$_1$agtggaaaaccgtcg
aaaccgagcagcccctccagcagcatgctgtacggctccagcatctcggcggagtccatgaaggtgatcgcggagagcatcggagtgg
gctccctgtcggatgacgccgccaaggaactagcggaggatgtgtccatcaagctgaagaggattgtacaggatgcggccaagttcat
gaaccacgccaagcggcagaagctctcagtgcgggacatcgacatgtcccttaaggtgcgaaatgtggagccgcagtacggtttcgta
gccaaggacttcattccattccgcttcgcatctggcggaggacgggagctgcacttcaccgaggacaag**gaaat**cgacctaggagaaa
tcacatccaccaactctgt**aaaa**attcccctggatctcaccctgcgctcccattggtttgtt*gtggagggagtgcaacccactgtgcccg*
*aa**aaccccc**ctccgctct*cgaaggattcccagttactggactcggtcaatccagttattaag**atg**$_2$gatcaaggcctaaacaaagatgc
ggcaggcaaacccaccaccggcaagatacacaagctgaaaaacgtggagaccattcatgtcaagcaactggccacgcacgagttgtc
cgtggagcagcagttgtactacaaggagatcaccgaggcgtgcgtgggatctgatgagccgcggcgcggggaagcgctgcagtcgct
gggatccgatcctggcctgcacgaa**atg**$_3$cttccccgc**atg**$_4$tgcaccttcattgccgagggagttaaggtcaatgtggttcagaacaac
ttggcgttgcttatttacctcatgcgcatggttcgtgcgcttctggataatccttcgctgtttctggagaaatacctccacgaactgataccc

Figure 8.1: Region in *Drosophila's* genome likely to harbor an IRES-like substructure. The sequence retrieved with RNA inverse folding is indicated in italics; motifs conserved with the IRES subdomain are indicated in bold letters. The region encompassing the IRES-like motif is colored in violet while the control downstream region is indicated in light green [18].

Remarks

Authors have built a pipeline, where they focused on the RNA structure of the searched domain, while leveraging the speed of sequence searching methods. They part in this aspect from other solutions, which rely on some form of a sliding window, that would make genome-wide scans extremely time consuming (one such example will be also presented in this section).

On the other hand, the use of the inverse folding program *RNAiFOLD* might not be the fittest. Recall, that the program is build on constraint programming and can search for all solutions or prove none exists. The method's nature however shows to be rather prohibitive in diversity landscape of sequences it generates. As seen from the sample of outputted inverse sequences (figure 8.2), the program has a clear bias towards GC base pairs and A unpaired nucleotides.

The issue is addressed by program's settings, in which the number of consecutive nucleotides and GC-content ratio in resultant sequences may be adjusted. This however does not remove the bias, just shifts it somewhere else. For instance, when the maximum number of consecutive As were restricted to 3, the program steadily replaced every fourth consecutive A with an U (see figure 8.3).

As we can see, the sequences appear highly artificial, however they might perfectly fold into the target structure. Trying to fine-tune program's parameters to make sequences look more 'natural' is a doubtful way of solving this issue, not speaking of the vague definition of the word 'natural'.

Even though the idea behind this approach is reasonable, the discussed flaws limits its exact use for the purposes of this work.

```
((((..((((((....))))))((((((((((..((.(((((..............)))))).))....))))))))))).((((..))))))))
GGGGAAGGGGGGAAAACCCCCCGGGGGGGCGGAAGGAGCGGGAAAAAAAAAAAAAAACCCGCACCAAAACCGCCCCCCCAGGGGAACCCCCCCC
GGGGUAGGGGGGAAAACCCCCCGGGGGGGCGGAAGGAGCGGGAAAAAAAAAAAAAACCCGCACCAAAACCGCCCCCCCAGGGGAACCCCCCCC
GGGGAAGGGGGGAAAACCCCCCGGGGGGGCGGAAGGAGCGGGAAAAAAAAAAAAAACCCGCACCAAAACCGCCCCCCCACGGGAACCCGCCCC
GGGGUAGGGGGGAAAACCCCCCGGGGGGGCGGAAGGAGCGGGAAAAAAAAAAAAAACCCGCACCAAAACCGCCCCCCCACGGGAACCCGCCCC
GCGGAAGGGGGGAAAACCCCCCGGGGGGGCGGAAGGAGCGGGAAAAAAAAAAAAAACCCGCACCAAAACCGCCCCCCCACGGGAACCCGCGC
```

(a) : Structure of the *HCV domain IIIabc* as an input to RNAiFOLD. A sample of returned inverse sequences follow. Notice all unpaired nucleotides in all sequences are marked as As. Also notice the domination of GC pairs and low U-content in general.

```
CCGGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCCGG
GGCGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGCC
GACGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGUC
CACGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGUG
GCCGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGGC
CCCGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGGG
ACCGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGGU
UCCGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGGA
GUCGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGAC
CUCGGCCAAAAAAGGCGCGUGGCGGGGAAAAACCCCGCCGCGCCCGCCAACCCCAGGCGGCGAG
```

(b) : Published [18] sample of sequences used for IRES searching. The same issue may be visible here as well. If wasn't for the C-rich motif constraint at the end of the sequence, there would be As too.

Figure 8.2: Bias of RNAiFOLD towards GC pairs and A single nucleotides.

```
((((..((((((....))))))((((((((((..((.(((((..............)))))).))....))))))))))).((((..))))))))
AGAGAAGCGGGGAAAUCCCCGCGGCGGCGGGGAAGGAGGGCGAAAUAAAUAAAUAACGCCCACCAAAUCCCCGCCGCCAAGGGUACCCUCUUU
UGAGAAGCGGGGAAAUCCCCGCGGCGGCGGGGAAGGAGGGCGAAAUAAAUAAAUAACGCCCACCAAAUCCCCGCCGCCAAGGGAACCCUCUUA
UGAGAAGCGGGGAAAUCCCCGCGGCGGCGGGGAAGGAGGGCGAAAUAAAUAAAUAACGCCCACCAAAUCCCCGCCGCCAAGGGUACCCUCUUA
AGAGAAGCGGGGAAAUCCCCGCGGCGGCGGGGAAGGAGGGCGAAAUAAAUAAAUAACGCCCACCAAAUCCCCGCCGCCAUGGGAACCCACUUU
AGAGUAGCGGGGAAAUCCCCGCGGCGGCGGGGAAGGAGGGCGAAAUAAAUAAAUAACGCCCACCAAAUCCCCGCCGCCAUGGGAACCCACUUU
```

Figure 8.3: New bias of RNAiFOLD when the maximum number of consecutive As is set to 3 and the maximum GC-content set to 60%.

8.0.2 IRESPred:Prediction of cellular and viral Internal Ribosome Entry Site

In this approach a support vector machine (SVM) classifier is built to decide, whether a sequence can harbor an IRES. The main biological set of features used in classification is computationally predicted reactivity of 5'UTR regions with 27 small sub-unit ribosomal proteins (SSRPs). 8 additional features were included, such as number of upstream AUGs or number of various structural elements (hairpin loops, external loops, etc.), totaling them to 35.

As for a training data set, authors extracted 114 sequences from the IRESite database [62], containing experimentally validated IRES elements. These were the positive examples. Negative examples were assumed to be 5'UTR sequences of housekeeping genes and viral/cellular coding sequences chosen by random.

Multiple SVMs were trained on sampled training sets (96 true positives, 96 true negative) and tested (set of 93 TP and TN). The best one was then implemented in the IRESPred web server. Reported accuracy metrics were 75% in both sensitivity and specificity [44].

This tool couldn't be validated and the claims verified as at the time of completion of the thesis, all instances of the server were inaccessible or removed.

8.0.3 IRESfinder:Identifying RNA internal ribosome entry site in eukaryotic cell using framed k-mer features

The most recent tool for discovering novel IRES elements. The authors mention the discussed *IRESPred*, criticizing it for using too small data sets for the purpose of training and testing. They overcome this issue using a newly published set of 583 human IRESs and their motifs [78]. The approach to the task of identifying IRES-like regions is build on k-mers of length 1-5. 19 k-mer motifs were selected to calculate features, which were then used to train a logit model. For training and testing, a selection of the 583 IRESs was extracted, specifically those sequences, which lie entirely within 5'UTR. Authors argue, that different nucleotide frequencies between 5'UTR and coding regions would impair the classification performance.

The reported accuracy and precision of *IRESfinder* on a testing dataset are respectively 65% and 73.08% [85]. We tested the program on random human gene sequences, which were fragmented by a sliding window. From 533 fragments of length 180 (comparable to the length of the 583 human IRES sequences the program was trained on) *IRESfinder* output showed doubtful results indicating 404 fragments are IRES-like. Even thought authors stated the program is suited only for 5'UTRs, these results cast a shadow on the credibility of the tool.

8.0.4 IRSS (2009) and VIPS (2013) IRES secondary structure prediction and searching systems *in silico*

Even though structure prediction programs are available, none of them is suitable to predict the IRES element. To build a specialized tool for this task, a group of researches has first set up an *IRES search system (IRSS)* and later its improved version a *Viral IRES prediction system (VIPS)*. Both combine two RNA structure prediction methods - MFE folding and consequently secondary structure comparison (alignment). VIPS is then extended by pseudoknot support.

IRSS

The procedure of IRES searching consist of a sliding window, that moves along the examined sequence and a so called R metric, that is calculated for each such window fragment. R is defined as a ratio between the fragment alignment score (ALEN) and the distance score of its predicted structure towards the structure of one of the "golden standard" IRESs. An R threshold is set to discriminate between IRES-like and non-IRES-like sequences. As for golden standard IRESs, the of Pestivirus and HCV were implanted into IRSS.

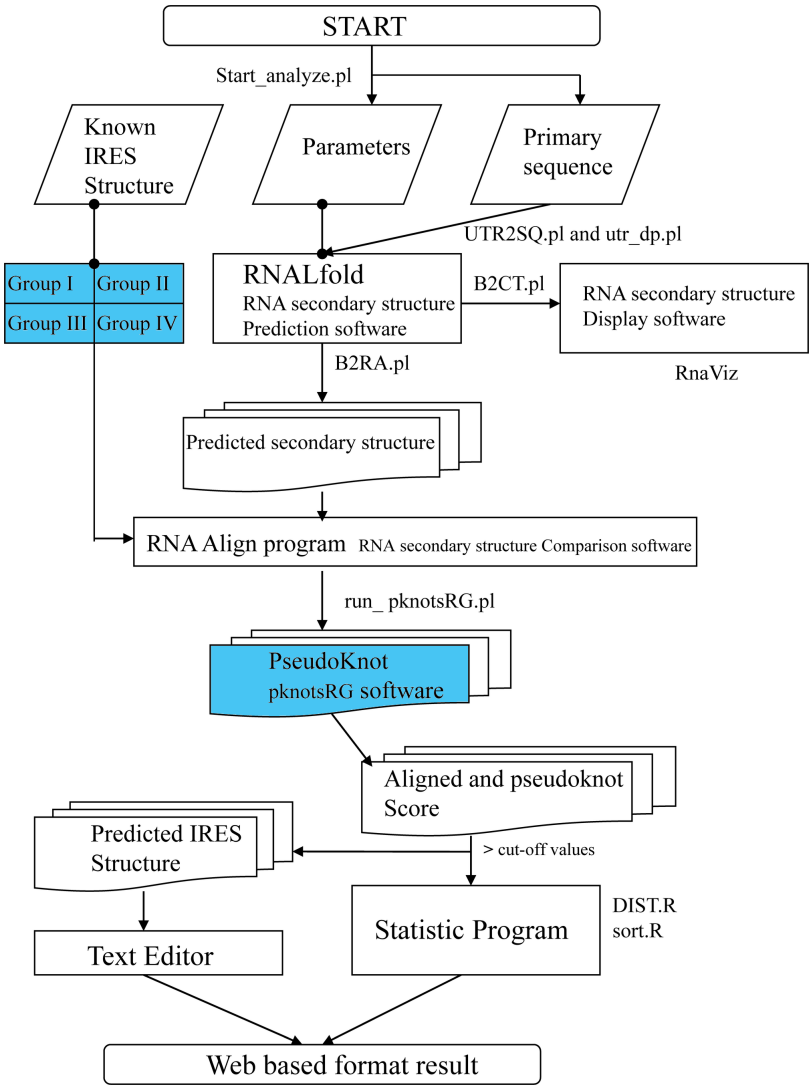


Figure 8.4: Workflow of the IRSS and VIPS search systems. A primary sequence is fragmented by a sliding window and folded. Predicted structures are compared with known IRESs and an R score for each window is calculated. Windows containing a potential IRES should manifest a significantly higher score, than the rest of the sequence. The worklow is identical for IRSS and VIPS apart from highlighted items, which are only a part of VIPS. In IRSS, only a single IRES is chosen as the standart one and pseudoknot score is not included [39]

The accuracy was as high as 72.3%, tested on 152 UTRs of HCV and Pestivirae taken from the *UTRdb* [1] database [81]

[1] http://utrdb.ba.itb.cnr.it/

VIPS

To improve the accuracy of the IRSS, the authors extended its capabilities by considering pseudoknots in prediction. This is a crucial enhancement, since only IRES Group 1 (see section 2.2.1) contains three essential pseudoknots in its structure [43].

The scoring metric was modified to take into account (apart from the R score) a pseudoknot value returned by *pknotsRG* program. To determine the metric's cut-off threshold, individual scores were calculated for a positive group (known IRESs from *IRESite* and *Rfam*) and a negative group (viral coding sequences). Based on these R and pseudoknot values, the threshold was estimated by linear discriminant analysis.

The VIPS has higher accuracy than IRSS even when only R score is considered (77% - 92% depending on the IRES group). Authors contribute this improvement to more competent standard IRES elements, that serve as comparison models. When pseudoknot values are added to evaluation, the accuracy rises for IRES groups 1 and 4 (containing the most pseudoknots) to 80.41% and 98.53%. It is worth noting, that VIPS has near-perfect specificity, but sensitivity can be as low as 64%.

Tested on human cellular 5'UTR database, VIPS was able to find potential IRES candidates only for groups 1 and 4. The system labeled 6% of all 5'UTRs (42 768) as candidates. From those candidates, 21% and 25% of them were verified as IRES elements based on *IRESite* data [39].

Remarks

In the context of previous two methods, the presented procedures recognize the need to shift focus from primary to secondary structure when searching for IRESs. It does so in a straightforward manner - by folding a sequence and comparing the fold with proved IRESs. The limitation however lies in the sliding window, which on one hand provides a very thorough scan and is reasonable for shorter sequences, on the other hand makes genome-wide searches in-feasible.

Regarding the test results on human 5'UTRs - they show lower accuracy of cellular IRESs prediction than the one of viral IRESs. The number may be however higher - by the time of publication, over 500 novel cellular IRES elements uncovered by Weingarten et. al [78] were not yet known. Perhaps some of the false positives were in fact not false at all. It would be interesting to evaluate the results again, see how many true positives were wrongly labeled. These particular results were however not published. They also cannot be reproduced, due to the VIPS server being down (as well as the IRSS one). Authors did provide the corresponding Perl scripts, they are however incomplete, missing essential configuration and template files.

Lastly, in the context of our goal of locating HCV IRES-like elements, it might be worth noting, that VIPS has not predicted any cellular 5'UTR sequence to be of IRES group 2, which the HCV one belongs to.

Chapter 9

Results

9.1 Approach principle - pipeline

The approach this thesis takes in solving this task is a modification of the one presented by *Dotu, et al.*[18]. There the authors dealt with similar task of detecting novel IRES elements. They built a pipeline consisted of inverse folding, greedy clustering of generated sequences, BLAST and finally keyword and biological signal filters on hits.

The method presented here preserves the two key steps of the pipeline - inverse folding and BLAST search, the remaining parts will be however different. The clustering step will be modified to be more complex, than the simple greedy approach. The keyword and biological signals filters will be dropped altogether, since we do not require the domain to be functional and active. Instead, those filters will be replaced by a post-processing step, in which hit regions obtained form BLAST will be folded and checked for similarity with the *domain IIIabc* of HCV. Furthermore, an appropriate inverse folding program/algorithm will be selected by benchmarking existing solutions. The one used by *Dotu, et al.* was their own creation and chosen apriori with no justification.

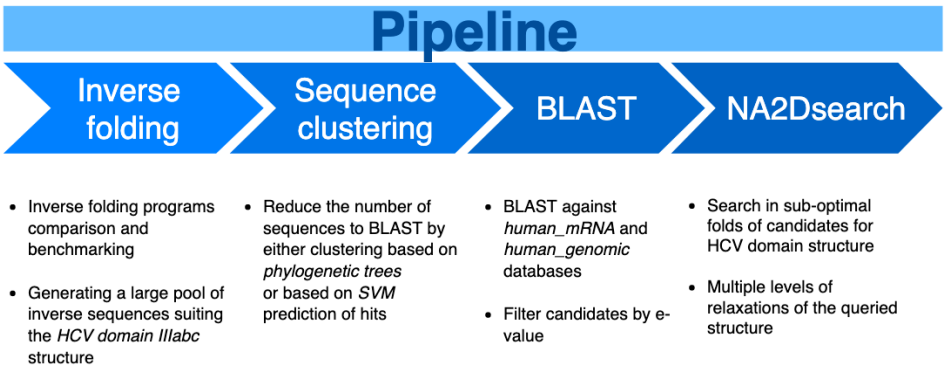


Figure 9.1: Summary of the pipeline

The schema shows the progression of the proposed pipeline together with details on what steps will be made at each stage. The pipeline should in

ideal case serve a funnel, where at the beginning it accepts a large number of inverse sequences and with each consequent step (clustering, BLAST e-value cut-off), the number decreases until at the very end, a handful of candidates is outputted. For instance, the e-value attribute of BLAST hits is suitable for funneling - it allows sorting of candidates by priority, by which we can regulate the number of sequences sent to the next step for further evaluation.

On following pages we will be dealing with each of the four step separately. However, before describing the pipeline itself, we need to choose an inverse folding tool.

9.2 Inverse program selection

Inverse folding is the very first step of the search process. Choosing an appropriate inverse folding program can therefore mean the difference between discovering a hit and loosing it. The candidates here are *RNAinverse*, *RNAiFOLD* and *NUPACK-design*.

While comparing available inverse solutions, we will have two metrics in mind. One metric is measuring the quality of sequences they produce - how close to the target structure do the sequences fold. For the second metric - not only the structure faithfulness is important, but the nucleotide composition of inverse sequences is as well. Later on we will see, that one method can be extremely biased towards certain nucleotides and sequence patterns, which in turn puts unwanted restrictions on the diversity of candidate sequences. Those restrictions are not backed by any domain knowledge, and besides, we naturally want the sequences to be as diverse as possible.

9.2.1 Evaluating structure faithfulness of inverse sequences

In this section, folded inverse sequences will be examined on how consistently the generated sequences fold to the desired structure. Since however the true structure of a given inverse sequence is unknown, we will have to rely on predicted structures as a means of comparison.

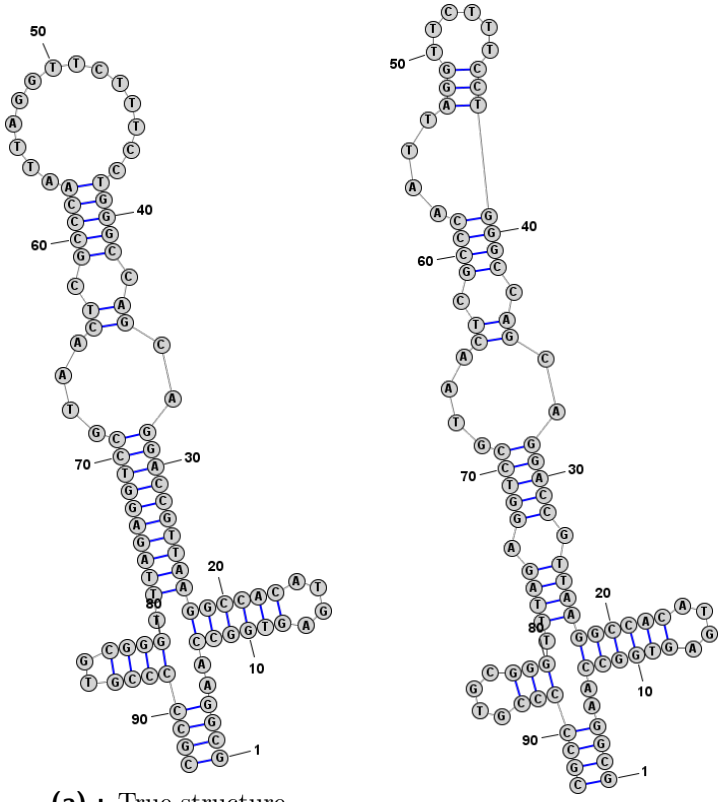
The comparison itself was conducted first by basic *tree edit distance* measurements and then more thoroughly using template matching with the *NA2Dsearch* tool. Results for this selection criteria shown below.

Tree edit distance

The first glimpse on the structure faithfulness is given by the tree edit distance. This metric tells the number of edit operations needed to transform one structure to another when both are represented as trees.

To see, what numbers we can expect, we first fold and examine the original *HCV domain IIIabc* sequence in this fashion. The reason for that comes from the inherent flaw of structure prediction - "correct" sequences (sequences, which actually folds to the target structure) are often misfolded and are assigned structures different from the true one. To at least estimate this

discrepancy, the examined HCV sequence was folded by the structure prediction program *RNAStructure*. The prediction was compared with the true structure to see, how big can the error of folding a "correct" sequence be.



(a) : True structure

(b) : Predicted structure

Figure 9.2: Comparison of the true and predicted domain IIIabc structures. Sequence was folded by *RNAStructure*, folding with Vienna's *RNAFold* would produce the exact same structure. The MFE structure of *RNAsubopt* would differ in bases A30T69, which would be unpaired.

As seen from the figure 9.2, the predicted structure is rather similar to the original one. To have the differences quantified, the tree edit distance (TED) between the true and predicted structures was measured to be 20. Taking this value to the context of inverse sequences - if the TED of a folded sequence is around 20, though non-zero, the sequence might still fold into the desired structure.

Each of the programs was tasked to generate 300 sequences (command line parameters available at the end of section). These were then folded and compared with the original structure in terms of TED. A histogram of TED values was plotted to get a glimpse on the quality distribution of inverse sequences. The results are shown on figure 9.14.

There we see, that *RNAinverse*, though written two decades ago, performed fairly well with the average TED to the target structure of 26.4 ± 19.06. *RNAiFOLD* cut this value almost in half, having an average TED of 14.8 ± 8.02. Both *RNAinverse* and *RNAiFOLD* were run in the partition function mode.

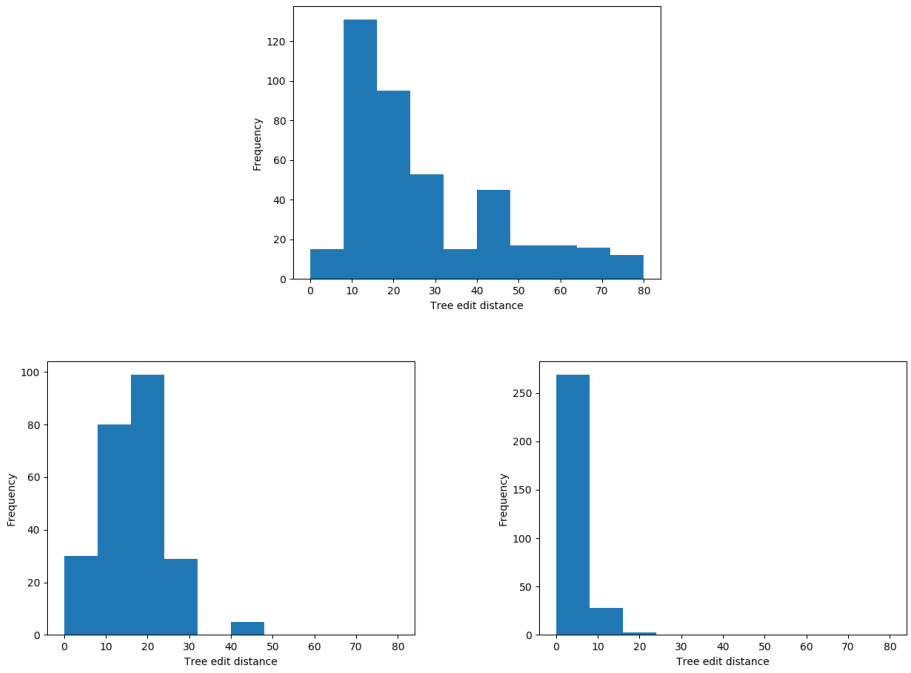


Figure 9.6: Tree edit distance histograms of folded inverse sequences generated from programs RNAinverse, RNAiFOLD and NUPACK. 300 sequences were assembled for each program. The average TEDs are 26.4, 14.8 and 4.6 respectively

NUPACK surprisingly managed to get almost all its sequences to fold into the same structure with only 4.64 ± 2.14 TED. The folded structures differed from the target one just at one pair of nucleotides - CG bases at positions 83 and 86 were supposed to be paired. [1]

NA2Dsearch

Tree edit distance of folded inverse sequences does tell something about quality of those sequences. First of all however, TED is just a number and one TED value can embody a large set of diverse structural changes. Second, only the MFE structures were compared and other conformations neglected. For those reasons, each set of 300 sequences was examined in the *NA2Dsearch* tool.

NA2Dsearch is a program that takes a secondary structure and searches an underlying sequence database for candidates, whose sub-optimal folds might match the target structure. It is an extremely exhaustive search (one sequence can result in hundreds of folds), but this ensures no potential candidates are missed. The search for the *domain IIIabc* within inverse sequences is summarized in table 9.1 with the query structure shown in figure 9.7. There we

[1] This error is however understandable - this pairing is made possible only by a non-standard conformation of the remaining nucleotides in the loop [42]. When this CG pair is forced, energy calculation programs would assign an infinite positive contribution to the structure energy.

can confirm the previous results - *RNAinverse* produces the most inaccurate sequences (3 sequences missed in compare to none missed in case of the other two methods) and with numerous possible conformations (hundred times more folds, than other two methods). *NUPACK* and *RNAiFOLD* both performed similarly well.

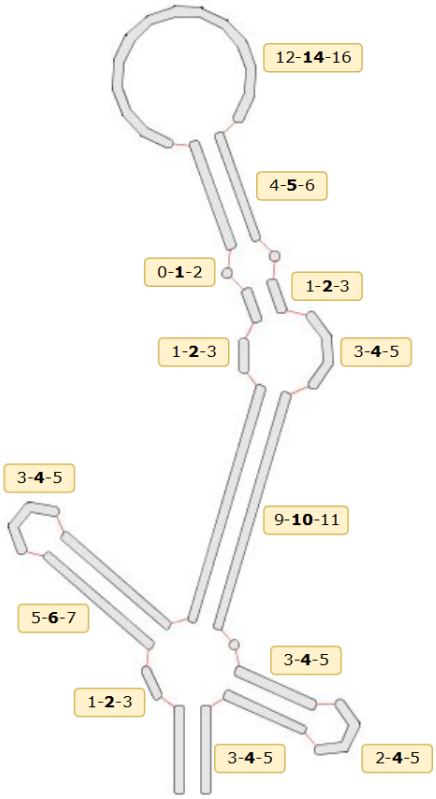


Figure 9.7: Search query structure. The number triplet at each structural element denotes *minimal*, *optimal* and *maximal* number of nt/bp respectively. These constraints allow tolerance of $\pm 1 nt/bp$ in shorter and $\pm 2 nt/bp$ in longer elements

This tool will be used once again later on and will be described more in details.

9.2.2 Evaluating nucleotide distribution of inverse sequences

Next, we explore the sampling capabilities of the three methods. To evaluate, how diverse set of inverse sequences they can produce, the sequences were visualized by the multiple sequence alignment tool *ClustalX* [46]. There (see figure 9.8), one can immediately spot high level of regularities among the sequences produced by *RNAiFOLD*. Some regions were even conserved across all 300 sequences, which shows non-uniformly sampled candidates. The remaining two programs have their conservation levels uniformly low across all nucleotide position.

	searched structures	hits	missed sequences
RNAinverse	$< 10^6$	11196	3/300
RNAiFOLD	< 20000	5421	0/300
NUPACK	< 10000	4133	0/300

Table 9.1: Search for domain IIIabc within 300 inverse sequences using *NA2Dsearch*. The columns show total number of examined structures within the energy increment $\Delta\Delta G = 4$kcal/mol, the number of matched structures and the number of sequences with no sub-optimal fold matching the target.

Repeated runs influence

A question might also be, whether sequences from multiple program executions differ in any way. In other words, whether different runs of inverse programs sample different sequence-structure sub-spaces, or the same one. And if yes, to what degree? This might have crucial consequences - having big differences between runs would mean the sequences are being sampled unevenly.

One way of quantitative determination and visualization of the generated sequences landscape is through *phylogenetic trees*. The original purpose of phylogenetic trees is to depict evolutionary relationships among organisms - their genetic information is compared using *multiple sequence alignment* (MSA) and then the tree (often referred to as "tree of life") is constructed by joining the most similar sequences or clusters of sequences.

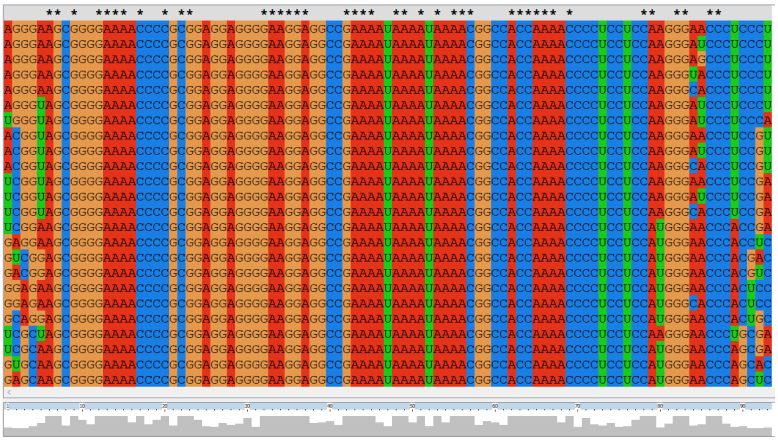
For purposes of this work, we will not study the evolutionary relationship of the inverse sequences, we just need the hierarchical clustering feature the tree is providing. In this test, 300 sequences were generated in batches of 20-30 (*RNAinverse*, *RNAiFOLD*), or designed one-by-one in case of *NUPACK*, which doesn't support batch generation. Corresponding trees are assembled via *ClustalW* [46] phylogeny.

From the figure 9.9 we can recognize, that *RNAiFOLD* sequences belonging to the same run form distinct clusters. This is an undesirable characteristic, since it indicates a different sequence-structure sub-space is sampled each time the program is executed. It is possible, that given enough time, the algorithm would explore the entire space. From the practical standpoint however, we cannot rely on such assumption.

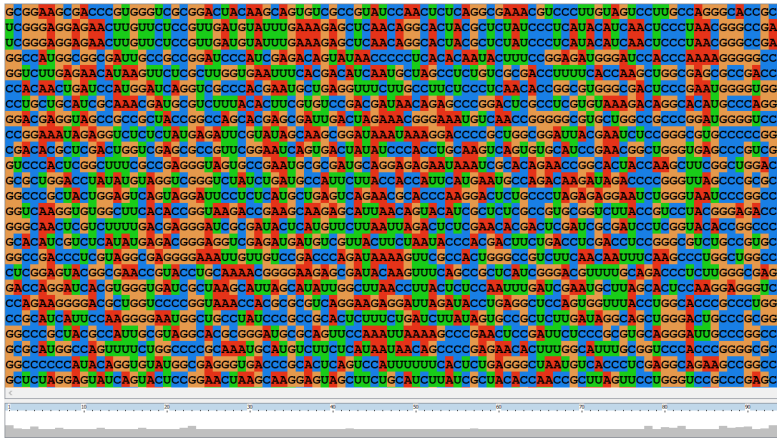
The remaining two methods have their output across the runs more evenly distributed and also individual sequences more distant from each other. Note however that having such uniform distribution of sequences doesn't necessarily mean that it is the true distribution - it is just an indication, that the space the samples are drawn from is large enough to appear uniform in the context of a small sized sample.

9.2.3 Comparing inverse folding solutions

Implementations discussed in the inverse folding section were examined on how quality sequences they produce. As mentioned in the theoretical part 6.1, the presented candidates were pre-selected based on the following criteria:



(a) : RNAiFOLD

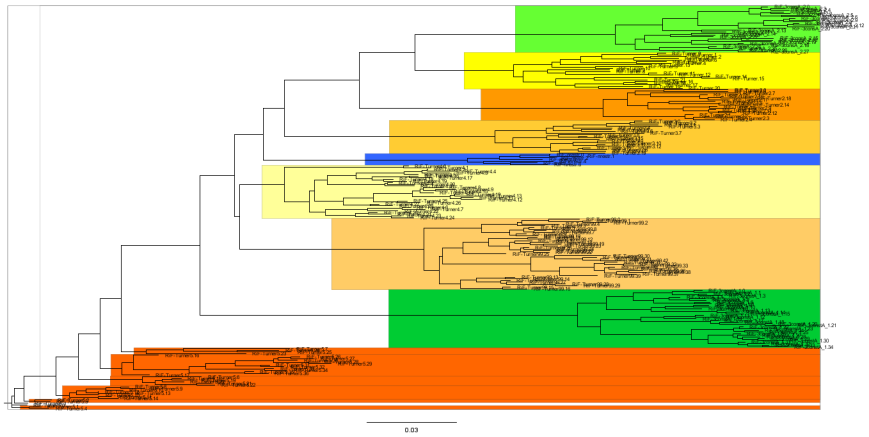


(b) : NUPACK

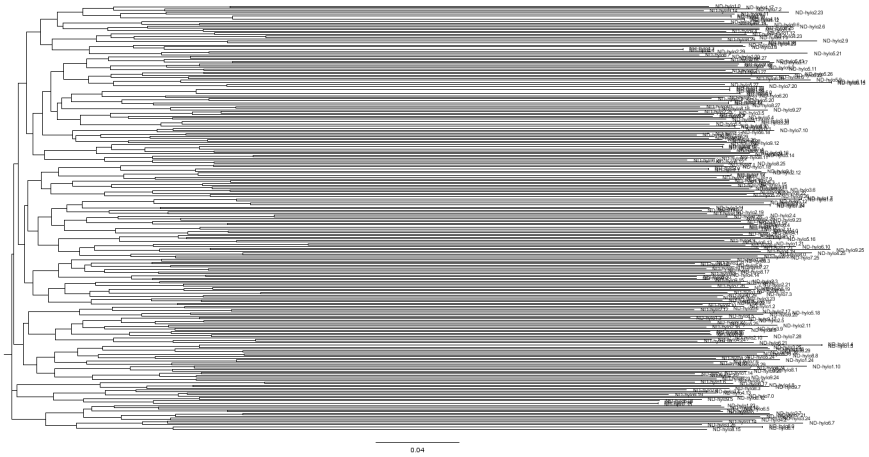
Figure 9.8: Visualization of the sequences composition in *ClustalX*. *RNAiFOLD* sequences manifest strong conservation indicated by the star marks on first line, that mark positions with identities. The conservation is also noticeable from the bottom graph indicating the degree of conservation of each column (high score for well-conserved columns and vice versa). Other two programs have this graph almost perfectly smooth (data shown only for *NUPACK*)

the ability to generate inverse sequences based on partition function and whether or not they have had an experience in 'wet laboratory' experiments. These criteria yielded inverse programs *RNAInverse* (the first implementation), *RNAiFOLD* (featuring constraint programming) and *NUPACK-design* (optimized for ensemble defect).

The programs were first examined in terms of how well do the sequences fold with respect to the target structure. Here, *NUPACK* achieved the best results, followed by *RNAiFOLD* - both had no sequence missed in the *NA2Dsearch* search and low average TED towards the target. The closeness of folded sequences towards the target structure is however not all. Even though *RNAiFOLD* performed better than *RNAinverse* in this regard, the sequences it outputs suffered from severe undesirable traits such as high



(a) : RNAiFOLD phylogenetic tree



(b) : NUPACK phylogenetic tree

Figure 9.9: Inverse sequences from multiple program runs clustered into phylogenetic trees. *RNAiFOLD* sequences formed a tree with distinct clusters highlighted in different colors. The clusters hold only sequences from one program run, indicating high dependence on the initial proto-sequence. Furthermore, different energy parameters were tested - green/blue clusters represent Andronescu's set; orange/yellow clusters are populated with sequences generated with Turner 99' set. *NUPACK* sequences are in contrary all distant from each other. The same goes for *RNAinverse* (data not shown for the tree is very similar). The trees are drawn by *FigTree*

Listing 9.1: Command line argument for examined programs from which the 300 inverse sequences were generated. Inputs to each program are separated with a blank line. *Fp* argument of RNAinverse tells the algorithm to use partition function. In case of RNAiFOLD the partition function is set by *-MinimizeEnsDef "1"*. Arguments *-maxGCcont, -consA, -consC* restricts the corresponding nucleotides to reduce RNAiFOLD's bias toward them. Andronescu07 energy model was used, heuristic LNS (Large Neighborhood Search) was turned off to search exhaustively. NUPACK accepts input structure and sequence constraints from a file (the first argument; the content matching the input to RNAinverse), the latest energy parameters available (Turner99) were used, allowed ensemble defect was set to 3% to allow more variation, than the default 1%

```
RNAinverse -R300 Fp
((((..((((((....))))))(((((((((..((.(((((..............))))).))
    ....)))))))))).(((....)))))))
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNN

RNAiFold2 -RNAscdstr "((((..((((((....))))))
    ((((((((((..((.(((((..............))))).))....))))))))))
    .((((..))))))))" -MAXsol 300 -RNAlibrary "Vienna" -
    EnergyModel "2007" -MFEstructure "0" -MinimizeEnsDef "1" -
    maxGCcont "60" -consA "4" -consC "4" -LNS "0"

nupack3.0.6\bin\design domainIIIabc_struct.fa -material rna1999
    -fstop 0.03
```

conservation and bias towards As and GCs (section 8.0.1) making it unusable for this work.

RNAinverse, despite being less accurate, is still faithful enough to have only a small portion of sequences missed (3 out of 300). It is however a reasonable trade-of for its speed, being able to generate one sequence in 1s.

NUPACK produces the most accurate and at the same time well distributed sequences. At the same time though, it is the slowest approach with 3s needed for one sequence. Depending on the application and its emphasis on time-accuracy trade-of, both *NUPACK* and *RNAinverse* are reasonable choices. In this work the emphasis will be on accuracy and the source of inverse sequences will be *NUPACK*.

9.3 Step 1: inverse sequences generation

At this point the inverse folding program is chosen and we may proceed to generate a pool of inverse sequences as the first step of the pipeline. Next step will be to cluster them and then BLAST the cluster representatives.

All inverse folding programs (including the chosen *NUPACK*) support nucleotide constraints, which are there to fix certain bases and preserve required sequence motifs. This option will be however left unused and sequences generated without restrictions. It does not mean there are no conserved bases in HCV IRES, our goal is only to keep the search as general as possible, focusing solely on the structure itself.

NUPACK settings were the following: energy model *Turner99* (the most recent supported) and the ensemble defect stop condition 0.03. The value for ensemble defect, which can be understood as allowing 3% of nucleotides to be paired differently, then in the probable correct structure, may seem small, but it still allows generating of 100 000 unique inverse sequences. Command line arguments are the same as in the benchmarking part and can be seen in the listing 9.1. The target structure definition in dot-bracket notation was:
((((..((((((....))))))((((((((((..((.(((((.....)))))).))....))))))))))).((((..))))))))

9.4 Step 2: inverse sequence clustering

The next step in the solution would be to BLAST the inverse sequences. BLAST is however a costly operation, especially on larger databases, and is the main bottleneck of the proposed procedure. Naturally, finding a way to reduce the number of sequences to BLAST, while loosing only a small chance of discovering hits, would mean a significant boost to the pipeline.

Clustering is the most appropriate candidate for this task. The idea is to generate a large set of inverse sequences, cluster them a BLAST only the representatives of the clusters. The hypothesis here is that the members of the same cluster would produce similar BLAST hits, thus being redundant. One can therefore omit them from further evaluation, selecting only a suitable centroid for such purposes and that all without a significant loss of BLAST hits. With the idea set, two questions remain to solve - how to cluster the sequences and how to choose cluster representatives?

9.4.1 Phylogenetic tree cutting

Main component for solving the first question has been already outlined in the previous section on inverse program choosing - a *phylogenetic tree*. Phylogenetic trees are a natural choice for the clustering step for one reason - as a hierarchical method, it enables variation on clusters granularity without the need of running the clustering again. A dendogram can be simply cut at different distances from the root to obtain different number of clusters. In our setting, this is a convenient property, since it allows for easier trade-offs between cluster quality and computational effort (better clusters - more BLASTing).

The trees discussed in this section look just the same as the one on the figure 9.9, only with denser branching. From its structure one can already say, that clusters originated from tree cutting will have have for a long portion of the tree roughly the same size of 1-2. Then at around 30% of the tree, the number of members will sharply rise - this is the area, where the tree will be most probably cut.

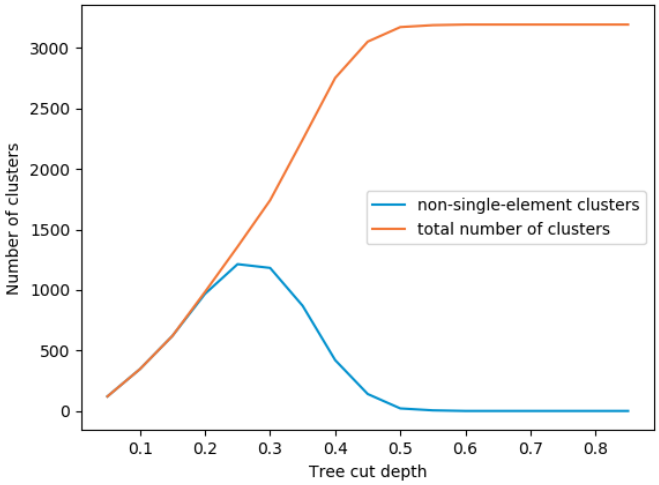


Figure 9.10: Relationship between the number of clusters with more than one element (blue) and the portion of phylogenetic tree (the depth), where the cut is made. The orange line shows the total number of clusters, including single-element ones. For this graph, a sample of 3200 unique sequences was generated, aligned by *ClustalOmega* with default parameters. The tree was build using neighbor-joining algorithm in *ClustalX*.

9.4.2 Clustering results

To explore the proposed clustering approach, over 3000 sequences were generated, aligned and organized into a phylogenetic tree, which was then in turn cut at various spots to obtain clusters. Properties to look at are then following: first the clusters are examined on how their sizes and number change with respect to where a cut is made on a tree. Next, the influence of choosing an MSA method on clustering results is explored. And lastly and most importantly we test, whether members of clusters produce the same (or at least similar set of) BLAST hits. I.e. whether it is even meaningful to perform this clustering step.

Influence of tree cut location on number and size of clusters

The plot 9.10 shows the relationship between the number of clusters with at least two elements and the portion of phylogenetic tree, where the cut is made. Judging from where the blue curve detaches from the x-axis, we can conclude, that up to the half of the tree, there are only single-element clusters. This indicates sequences are in general very far from each other. In some cases, the cut has to be made near the root (0.2), in order for a sequence to be part of any cluster.

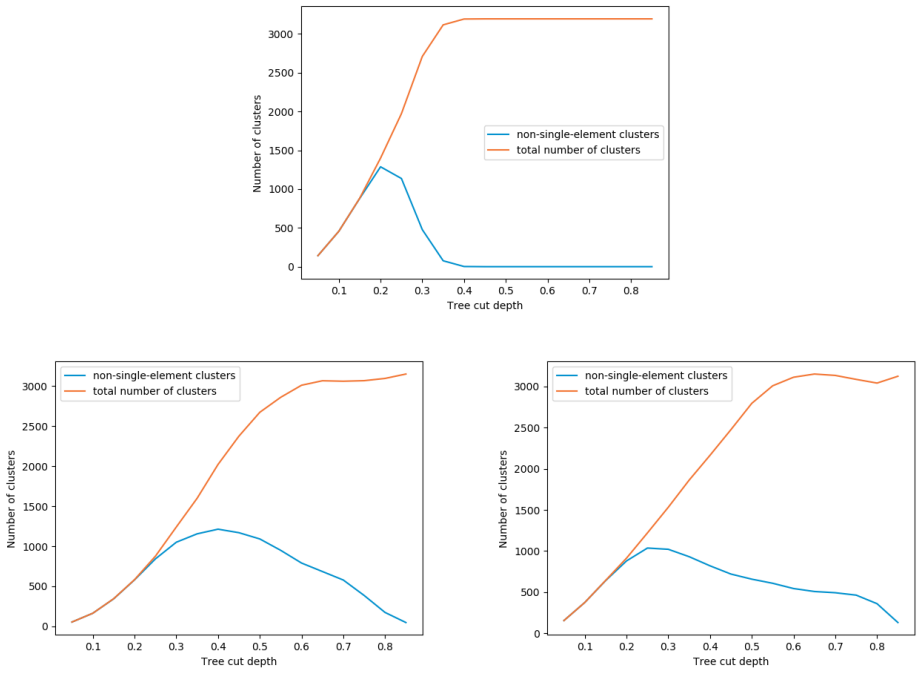


Figure 9.14: Number of clusters w.r.t. the cut position for alignment methods *MUSCLE*, *ClustalOmega(3 iterations)* and *MAFT*. *MAFT* and *ClustalO* differ in a way that they have no sharp breaking point, from which the clustering explodes. This however seem to have no effect on examined metrics - they neither do partition sequences more similarly, nor their cluster members have more similar BLAST hits.

Different alignment methods - different clusters

Preceding to the phylogenetic tree construction, clustered sequences need to be aligned by an MSA method. Popular tools nowadays for handling large data sets include MUSCLE, Clustal Omega and MAFT. Due to the number of sequences and the heuristic nature of MSA, the alignments differ greatly. And with different alignment, also subsequently derived trees and clusters are different. The difference between MSA methods is best quantified via a comparison of the corresponding clusters they produce. A simple metric for such comparison is *Rand score*, which tells to which degree the two clustering methods agree on partitioning of examples - i.e. it takes into account how many examples assigned into the same cluster by one method are grouped together also in the second method (with 1 being full agreement, and 0 the opposite).

Testing on the 3200 sample set, the Rand score between cluster derived from different MSA methods are nearly zero (if the tree is not cut near the root). This indicates almost completely dissimilar partitioning of sequences to clusters derived from various MSA programs. The score is this low even when comparing clusters from the same MSA program, just with different parameters. Furthermore, this difference in clustering results will be only

	MUSCLE	*MAFT*	*ClustalO*	*ClustalO (3iter.)*
MUSCLE	*	$3.7 \cdot 10^{-3}$	$11.7 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$
MAFT	*	*	$3.1 \cdot 10^{-3}$	$0.2 \cdot 10^{-3}$
ClustalO	*	*	*	$9.1 \cdot 10^{-3}$
ClustalO (3iter)	*	*	*	*

Table 9.2: Pairwise adjusted Rand scores for clustering derived from 3 MSA tools. *ClustalOmega* is present twice, once with default parameters and once with arguments for repeated iterations. Trees were cut at 0.3 portion of their depth, were the number of non-single-element clusters is expected to be the highest (see figure 9.10). The score is the highest for *ClustalO* and *MUSCLE* alignments, even though still nearly zero.

greater with larger set of sequences.

One can argue, that cutting the tree further from root would result in fewer, but more "tightly packed" clusters of less distant members, and that those clusters might be more similar across MSA methods. This is however not the case - the scores remained close to zero, which suggest, there are no universally close sequences in the set.

Evaluating the relevancy of the clustering step with BLAST

The reason, why the clustering step was proposed, is due to considerable computational effort BLASTing requires on larger databases. Thus, reducing the number of queries while maintaining the amount of collected hits is highly desirable. The assumption behind this idea of clustering inverse sequences is that the clusters can be well represented by a *prototype sequence*, which when BLASTed would give similar results as BLASTing other cluster members. The question is, whether any partitioning of the inverse sequences can achieve such a property.

To evaluate the relevancy of the clustering step, members of clusters were BLASTed against *human mRNA/prot* database and obtained set hits examined. If our assumption is true, the results set would contain many redundant hits, as the clustered sequences give similar results. The score telling how similar BLAST results are within a cluster, is defined as:

$$score = \begin{cases} 1 - \frac{\text{no. of unique hits}}{\text{no. of hits total}} & \text{if hits total} > 0 \\ 0 \end{cases} \tag{9.1}$$

This simple metric assigns number close to 1 to clusters having high similarity of hits and 0 to those, whose hits are highly heterogeneous. Whether the hits are the same or not can be put many ways - the most strict setting would require hits to have the same accession and (more or less) the same location. It however turned out, that to make a conclusion on this question, it is enough to consider hits to be equal just for having the same accession. Even with such simplistic assumption, the scores were zero for almost all clusters, regardless of the point, where the tree was cut and regardless of the MSA method. In other words, even clusters obtained by cutting the

	BLAST hit accessions	total/unique	score
cluster 1	None	10/10	0
	NM_001007593, NR_134502, NR_027400, XR_002957158, XR_001747940, XM_011520304, NM_001365135, NM_001318088, NM_000543, NM_001318087		
cluster 2	None	6/6	0
	NR_034105, XM_017023562, XM_017023107, XM_024450223, XM_024450222, NM_181718		
cluster 3	XM_017028198	8/8	0
	XM_011536032, XR_001747198, XM_005252591, XM_017016625, XM_005252588, XM_011519659, XM_006717502		

Table 9.3: Clustering evaluation output sample (3 clusters shown). Each row contains a set of BLAST hits of a cluster member (here all clusters have 2 members). Note, that the hits are often redundant - the accession are often only different transcript variants of the same gene. For instance the first cluster consists solely of transcript variants of the *SMPD1* gene (in fact the hit matches a *short genetic variation (SNP)* rs786204733 likely linked with Niemann-Pick disease). The number of unique hit sequences is then usually lower

tree far from the root, which should contain the most similar members, have completely heterogeneous hits. A listing below shows a sample of an output from the examination of clusters. BLAST hits with e-value below 1 were collected from the *human mRNA* database.

From the results above, which show even the smallest clusters produce heterogeneous hits, lead to a conclusion, that clustering is not a relevant step in this setting. Sequences are in general too far from each other to cluster and find a meaningful cluster representative. Great distances are also apparent from the chart 9.10.

Apart from all clusters having a score of 0, we can notice that from the pair of cluster members, one member is often behind all hits of the ensemble. This observation is another argument against clustering. Choosing a wrong representative would potentially result in loosing all hits of a given cluster.

9.5 Predicting existence of BLAST hits with SVM

Clustering was shown to be ineffective in reducing the number of BLAST queries - the sequences were simply too dissimilar. Even though this method failed, it provided potential ground for a different approach. From clusters evaluation above (9.3) we can make an interesting observation - a substantial number of inverse sequences does not produce any BLAST hits (e-value < 1)

- indeed, when 28000 sequences were BLASTed against the mRNA database, approximately half of them did not yield any result [2].

This has motivated another view to the problem - instead of looking at the sequences as a set of multiple clusters, they can be viewed as a set of binary classes - one which does produce hits and one which does not. If we could effectively recognize to which class a sequence belongs, a significant portion of BLAST queries would not need to be even made since they would most probably not provide any results.

9.5.1 Support vector machines

Types of classifiers, that are are well-suited for two-class classification are *support vector machines (SVMs)*. They accept examples labeled with $+1/-1$ denoting the two classes, where the examples themselves are numeric vectors (can be viewed as *features* or *attributes* of an example; in computational biology the vectors can for instance take form of micro-array measurements). The classifier attempts to find a boundary between positive and negative examples, which it then uses to separate unseen examples. The boundary is a line in case of 2-element feature vectors, in multidimensional spaces it is a hyperplane ($n-1$-dimensional sub-space of a n-dimensional one).

Decision boundary

To which space (positive/negative) an example vector belongs, is given by a dot product with a boundary *weight vector*. In a 2D intuition, a boundary line diagonal to the axes would be represented as a weight vector $(-1, 1)$. Mathematically, for an M-dimensional feature vector $\mathbf{x}$ and weight vector $\mathbf{w}$ is the operation denoted as: $\langle \mathbf{x}, \mathbf{w} \rangle = \sum_{i}^{M} x_i w_i$. Continuing the 2D example, points laying exactly on the boundary line would have this dot product with the weight vector equal to 0, points laying above the boundary value greater than zero and points below boundary line value less than zero (figure 9.15). This holds in general spaces as well and by this key, unseen examples are classified. The boundary can be further shifted by a *bias*; the previous expression for the vector "score" is then generalized to $\langle \mathbf{x}, \mathbf{w} \rangle + b$

Large margin classification

Let's assume examples can be separated by a boundary and that there exists a hyperplane capable of correctly classifying all examples. It would be wrong to expect such hyperplane to be the only one fitting this criteria. In fact, there are multiple of them and the question is, which one is optimal.

Since the boundary is formed during training of an SVM, thus not all data points has been seen, it is reasonable to require the boundary to have as much margin towards examples of the separated classes as possible - this criteria

[2] later we will see, that even with e-vale threshold 3, the portion of sequences not yielding any hits would be still around 40%

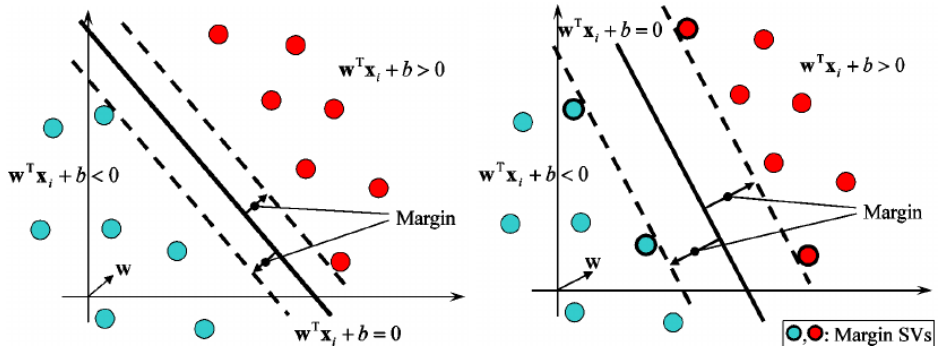


Figure 9.15: Principles of SVMs. The goal is to find a decision boundary separating two classes of examples. The boundary is represented by a weight vector **w**. Data point vectors are assigned a class based on the dot-product with **w** (positive or negative). Though one can find multiple separating hyperplanes (the figure shows two possibilities), the optimal boundary is considered to be the one with the largest margin (second graph) [82].

should ensure greater robustness during testing (figure 9.15). Finding the boundary then turns into an optimization problem of finding the classifier with maximum margin among all classifiers that correctly classify all the input examples. The problem is defined as follows:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$
$$\text{subject to : } y_i(\langle \mathbf{x_i}, \mathbf{w}\rangle + b) \geq 1 \text{ for } i = 1, ..., n$$

To constraint ensures, that all examples are correctly classified (requiring the score and the label to be both positive or both negative) and the minimization term ensures the maximum margin (derivation leading to this term is non-trivial and is out of the section's scope).

In practice, not all examples can be classified correctly and usually it is even not desired to avoid overfitting. The optimization constraint is therefore relaxed by introduction of *slack variables* ξ, that allow some examples fall into the margin or to be misclassified: $y_i(\langle \mathbf{x_i}, \mathbf{w}\rangle + b) \geq 1 - \xi_i$ for $i = 1, ..., n$. To prevent the overuse of slack variables, the variables are added to the minimization term:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i$$
$$\text{subject to : } y_i(\langle \mathbf{x_i}, \mathbf{w}\rangle + b) \geq 1 - \xi_i \text{ for } \xi > 0 \text{ and } i = 1, ..., n$$

The constant $C > 0$ is a regularization term, that can either magnify or diminish the penalty for slack, affecting the resultant margin. Large values forces a narrow, more prohibitive margin, while small values allow wider margin with more errors. SVMs with slack enabled are called *soft margin SVMs* as opposed to *hard margin SVMs*, which do not permit misclassfication.

Similarity kernels

So far, data points were linearly separable, meaning, that there existed a hyperplane, that could perfectly separate examples of the two classes. Such an assumption is convenient, but not always true (in 2D, an example of non-linear data could look like an "island" of positive data points surrounded by a "sea" of negative ones). Fortunately, SVMs can be extended to generate non-linear boundaries and separate even such cases. Data points vectors are in this scenario transformed into a new space with higher dimensionality, where they can be linearly separated using the same principles as before. The dot-product operation then modifies from $\langle \mathbf{x}, \mathbf{w} \rangle$ to $\langle \phi(\mathbf{x}), \phi(\mathbf{w}) \rangle$, where ϕ is the transformation function, that "lifts" vectors to a space with higher dimension.

Dot-product in the transformed space and the transformation itself however greatly increase computational costs. Fortunately, both operations can be circumvented by so called *kernel functions*, which output the transformed dot-product using only vectors and operations in the original space. Kernel function $k(\mathbf{x}, \mathbf{x}')$ is defined as:

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$$

Kernels can be understood as a measure of similarity between two vectors in transformed space. They alone are sufficient to find the weight vector defining the separating hyperplane. The exact method is again out of the scope - it is however worth noting, that the weight vector can be then expressed as combination of certain vectors near the separation line, which are then called *support vectors* (hence the name of this approach).

Weighted degree kernel

There are numerous kernel functions, each suited for different classes of problems. *Polynomial kernels* and *Gaussian kernels* belong among the most known ones. In case of computational biology, *spectrum kernel* and from it derived *weighted degree kernel* are the ones frequently used. Both are able to express the similarity between two sequences and were used for instance in splice site recognition [9]. The spectrum kernel in essence decomposes sequences into substrings (and their occurrences) and compares them on this level. Formally, with $\overline{\mathbf{x}}$ and $\overline{\mathbf{x}'}$ being compared sequences:

$$k_l^{\text{spectrum}}(\overline{\mathbf{x}}, \overline{\mathbf{x}'}) = \langle \Phi_l^{\text{spectrum}}(\overline{\mathbf{x}}), \Phi_l^{\text{spectrum}}(\overline{\mathbf{x}'}) \rangle$$

where $\Phi_l(\overline{\mathbf{x}})$ maps a given sequence to a vector, whose elements are counts of occurrences of all possible substrings of length l. In case of dimers ($l = 2$), the vector would hold 16 features, with trimers 64 features, etc.

Spectrum kernel however does not operate with one crucial information - the location of motifs within the sequence. This is why weighted degree (WD) kernels have been developed. Internally they make use of spectrum kernels, and apart from substring occurrences, they also take into account their position:

$$k_D^{WD}(\overline{\mathbf{x}}, \overline{\mathbf{x}'}) = \sum_{d=1}^{D} \sum_{l=1}^{L-d+1} \beta_d k_d^{\text{spectrum}}(\overline{\mathbf{x}}_{[l:l+d]}, \overline{\mathbf{x}'}_{[l:l+d]})$$

where β_d is a weighting for different substring lengths, D is the maximum considered substring length (refered to as *kernel size*) and $\overline{\mathbf{x}}_{[l:l+d]}$ is a substring of sequence $\overline{\mathbf{x}}$ of length d at position l.

For its properties, the WD kernel will be applied to our task of predicting, whether a sequence has BLAST hits, or not.

9.5.2 Results

In the pursue for the best performing SVM, optimal values need to be found for two hyperparameters - regularization parameter C defining the tolerance for misclassification (or also the "width" of the decision boundary), and *kernel size* of the WD kernel. The misclassification cost C can split into two parameters, each for negative and positive class (e.g. C_{neg}, C_{pos}) in case of unbalanced data [9]. Here however, we have roughly the same number of positive and negative examples and the parameter can be kept as one value.

Estimation of the two hyperparameters was done through *grid search*. In this type of search, ranges of C and *kernel size* values are combined into pairs and for each such pair an SVM is built and evaluated. The pair yielding the best performing SVM is then considered to be the optimal set of parameters. In this case the value ranges were $[0.1, 1, 10, 100]$ for C and $[4, 11, ..., 20]$ for kernel degree. Each parameter combination was evaluated by 5-fold cross-validation. Examples were formed by 28000 sequences as features and $+1/-1$ indications, whether a sequence has BLASTs hit or not as labels.

As a result, the pair $C = 1, \text{kernel degree} = 6$ reached by a negligible margin better accuracy $(60, 9\%)$, than the rest of the pairs. The validation accuracy was is in fact this low across all parameter combinations (see figure 9.17).

Figure shows nearly constant validation accuracy and therefore that no trained model (regardless of its kernel degree and training accuracy) is able to classify correctly unseen examples. This suggests, that it is not possible to predict the existence of hits, given sequence, since there is no relationship between them. Indeed, with 22400 training examples (0.8 of total examples), the number of support vectors was > 20000. I.e. almost all examples were necessary to form a decision boundary, meaning there was virtually no room for generalization. In a sense, this method was an attempt to describe an entire BLAST database with incomparably smaller set of support vectors, which would be a surprise, if it was possible.

One might consider tweaks such as "cloning" examples (sequences) with multiple BLAST hits to corresponding number of copies in order to amplify their contribution during classifier learning (e.g. a sequence producing 4 hits would result in 4 examples, instead of one). Such modification would however not bring any improvement for our case for the previously stated reasons.

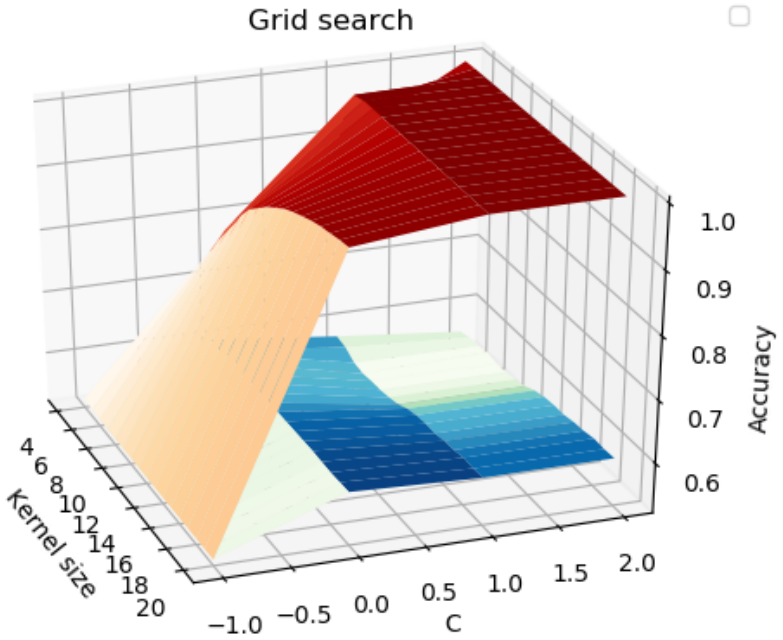


Figure 9.16: Grid search results. We can see clear divergence of training and testing accuracy - the training one surpasses 90% while the cross-validation accuracy stays around 60% and forms a flat plateau. Next, there is a sharp improvement on training accuracy with transition from $C = 0.1$ (logarithmic scale) to $C = 1$. With higher C training accuracy further rises, while the testing one slightly drops - a sign of overfitting

As for technical documentation, the SVM solver used was *LIBSVM* [13] wrapped in *Shogun* [73] ML toolbox. The WD kernel implementation came from the same package.

9.6 Step 3: BLAST

Returning back to the pipeline, the previous results lead us to a conclusion, that no significant reduction of inverse sequences by clustering is not applicable. The sequences will have to BLASTed as they are. BLAST+ [12] version 2.8.1 was used.

9.6.1 Choice of a sequence database

For our application, there are two databases to choose from - one containing only genomic transcripts (human *mRNA/Prot* database[3]) and one holding the full genome (*human genomic*[4]) . The former is considerably smaller and contains only *RefSeq* RNA sequences, while the later holds entire human chromosomes from both *RefSeq* and *GenBank*.

[3] accessible from `ftp://ftp.ncbi.nlm.nih.gov/blast/db/`
[4] accessbile from `ftp://ftp.ncbi.nlm.nih.gov/refseq/H_sapiens/mRNA_Prot`

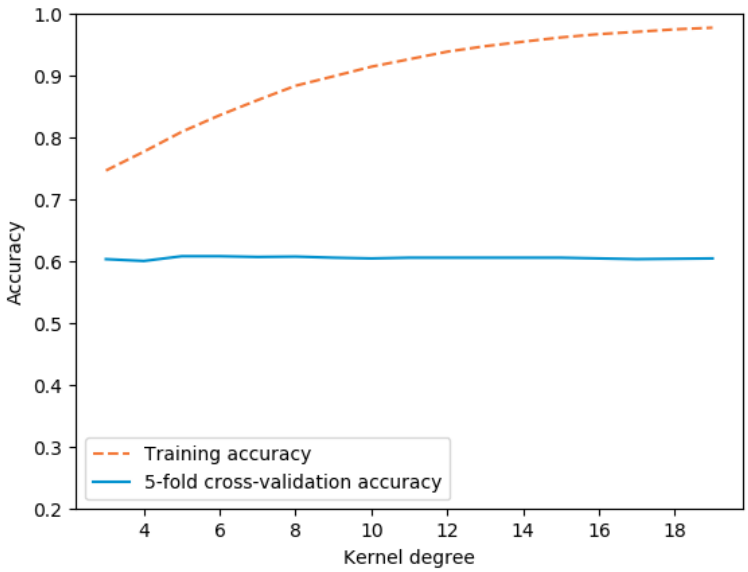


Figure 9.17: A cut of the grid search plot at $C = 1$ (or 0 on logarithmic scale). The training accuracy rises with kernel degree, which is no surprise. The testing one stays the same regardless of the degree For completeness, the area under the ROC curve is on average around 62%. Almost exact results can be achieved also with half of the training set size

The choice of database to search has tremendous effect on computational efforts. For instance, considering only the *human mRNA* database would reduce the search space 100 folds in compare to the entire *human genomic* database ($5 \cdot 10^8$ nt vs $6.4 \cdot 10^{10}$ nt). The search time is reduced roughly by the same magnitude.

The different database sizes have also an effect on the interpretation of a hit's e-value. In larger amount of sequences it is easier to get a hit just by chance, thus making it harder to obtain a significantly low e-value (see e-value definition in 7.1.1). Indeed, when BLASTing a sequence against the *mRNA* database, a hit with a score 0.6 was found. When BLASTing the same sequence against the full genomic database, the same hit was obtained, this time however with e-value of 6.

9.6.2 Searching and filtering

Inverse sequences were BLASTed against both named databases. The e-value threshold was set to 3, allowing hits with rather low similarity. Not all of them can be further examined though - obtained results will have to be filtered and this permissive e-value gives us large enough pool to choose from.

The secondary e-value threshold on BLAST hits was set to 0.5 in case of the *mRNA* database and 0.5 in case of the *genomic* database. By pushing candidates with sufficiently small e-value further through the pipeline we

hope to save folding and structure comparison operation only to sequences, that are meaningful enough.

9.7 Step 4: post-processing hits

At the post-processing step, the chosen BLAST hits are folded and checked for a potential match with the desired structure. Now, the set of candidate sequences is small enough to allow exhaustive search over all sub-optimal structures (within given energy range). By this, the chance for a false negative is minimized.

For this procedure to work, we need the hit sequences to have comparable length (equal or longer) with the original *domain IIIabc*. The length of hits is however hardly ever the same as the length of the query (i.e. hits obtained from BLAST are shorter, than the examined domain). Therefore, before folding, chosen BLAST hits are extended to have the required length, as illustrated below:

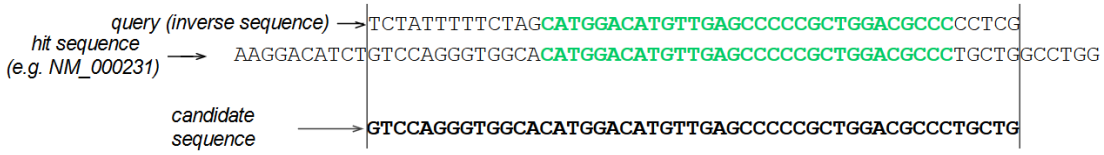


Figure 9.18: Extraction of candidates from BLAST hits. The green regions are hits between the query and subject sequences. The region in the subject sequence is extended to match the length of the query. This yields the final candidate sequence

It is worth noting, that number of extracted candidate sequences is not equal to the number of BLAST hits. As discussed earlier, the hits can be redundant due to multiple transcript variants of the same gene. On average, extracted regions make up only 25% of the original number of BLAST hits.

9.7.1 NA2Dsearch search

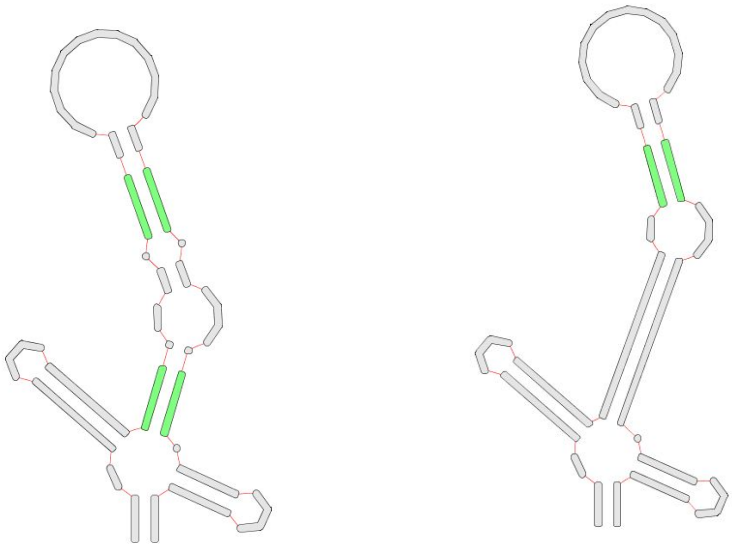
The search for the domain within extracted candidate regions is carried out by *NA2Dsearch*. The program first folds the regions by a *RNAsubopt* routine. Each candidate sequence then results in multiple structures and by this, we seek to overcome the insufficient reliability of folding methods. Candidate regions turn into candidate structures, which are in turn checked for structure similarity with the query structure. Checking itself is through acceptance of a nondeterministic finite state automata, which has been built upon descriptors of the searched structure.

At this stage not only the exact structure of the domain is searched, but also other, relaxed versions. These versions will have more loose requirements on structural elements, allowing some variation in structure of the folded candidates to be tolerated. After all, expecting to find the precise structure of the domain would be rather optimistic assumption. For this, the program

allows query structures to contain so called *variable nodes* (variable stem/-variable loop), which can match a structural motif of arbitrary length and composition. The variable nodes are here utilized to make the query more flexible.

Below, the relaxed queries are presented, starting from the least relaxed one. Green motifs denote variable parts of a query. Grey parts match the query used to benchmark inverse folding programs (see 9.2.1) and represent the true domain structure.

The relaxation is made in accordance with the findings presented in the biological chapter 2 - bottom *IIIabc* junction is vital for IRES function and needs to be preserved; similar importance is given to the intrahelical mismatched pair above the central internal loop (only the first query, as we realize it might be too strict of a constraint and therefore omitted in the second one). The remaining stem and apical loop are relaxed and allowed to be variable, as the structure of this part is functionally more permissive - the stem is replaced by a variable node and the loop is relaxed in terms of wider length range (5 to 16, with 14 being ideal length).



(a) : Minimal query able to detect domain IIIabc from sequence (**S1**)

(b) : Upper stem of sub-domain IIIb relaxed (mismatch omitted) (**S2**)

Figure 9.19: Following queries were assembled to be searched in candidate regions extracted from BLAST hits. Each query represents a different level of domain structure relaxation

The first query is the closest to the true structure, which is able to detect the *domain IIIabc* from its sequence. I.e. when the domain sequence is searched with its true structure as a query, no hit would be found. This is because *RNAsubopt* will always assign nucleotide pairs G176A223 and A179U220 as unpaired (see apendix figure A.1). Hence the lower stem relaxation to cover this error .

9.7.2 NA2Dsearch results - *human mRNA* sequences

The exact *domain IIIabc* was not found within neither *mRNA*, nor *human genomic* database candidates. This was an expected outcome.

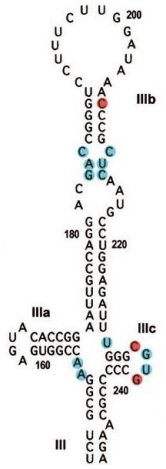
As for the relaxed structures, **S1** did not yield any hits as well. The mismatched nucleotide pair added before the variable hairpin seems to be a too strict requirement. The query **S2** applied on the *mRNA* candidates matched 5 sequences with 18 sub-optimal structures satisfying the query requirements. The best structures from the 5 candidates can be seen in figure 9.20 and are summarized in the table 10.1.

Focusing on the table, probably the most important are the last three columns telling the length of the matched sequence/structure, the tree edit distance and *NA2Dsearch* score telling how much the hit violates the optimal structure (the closer to zero the better). First, we can notice, that TEDs are rather similar across all hits. Looking at the structures however, we can spot eminent differences between them as well as different levels of similarity with the *domain IIIabc*. Even though TED can express dissimilarities between structures, as a metric of closeness to the target it is too general (comparing two arbitrary structures) and thus insufficient for our purposes.

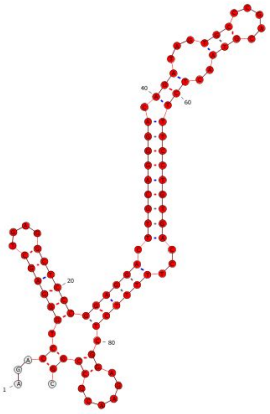
NA2Dsearch score is in this regard a better indicator. It is not suited for comparison of two arbitrary structures, but rather an ideal structure with a candidate one. In the evaluation, it takes into account optimal structural motifs of the target together with a degree to what the candidate structure violates them. Indeed, from the table we see more distinct values. In this regard, regions for genes DRC3, LMNB2 and CORO2B stand out, as their score is the closest to zero. The first two are also the most promising from the perspective of matched lengths - both have almost entire sequence matched with the query. At this point however, we can witness the drawback of using a variable node in the search query. The LMNB2 candidate structure has an extra internal loop with a hairpin in *domain IIIb* region (figure 9.20(b)), heavy violations to the domain structure, which are however not penalized. Thus, despite of the high score, LMNB2 is removed from further evaluation.

RNA accession	gene name	s. nt	5'UTR	e-value	mlen	TED	NA2D
XM_024450962	DRC3 (X5)	10	yes				
XM_024450965	DRC3 (X9)	194	yes	0.01727	89/93	34	-17
XM_011524020	DRC3 (X4)	198	yes				
NM_032737	LMNB2	967	no	0.2104	88/93	38	-17
NM_001190457	CORO2B (X3)	1	yes	0.2104	70/93	37	-13
XR_001739336	non-coding	1953	N/A	0.2104	82/93	25	-34
XM_017006507	CNTN3 (X2)	-521	no	0.2104	64/93	38	-54

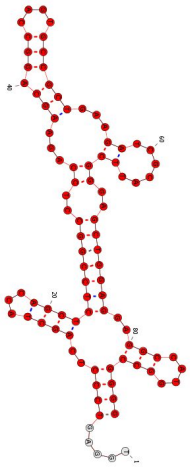
Table 9.4: Summary of hits combining results from BLAST and *NA2Dsearch*. The database of origin is *human mRNA*. Column *s. nt* denotes starting nucleotide position of the hit within the record (negative value indicates reverse/complementary strand). *5'UTR* column indicates, whether the hit lies within RNA's untranslated region, *mlen* shows the matched length with query, *TED* shows tree edit distance towards the actual structure and *NA2D* is the *NA2Dsearch* score. First three hits are grouped as they have the same hit sequence



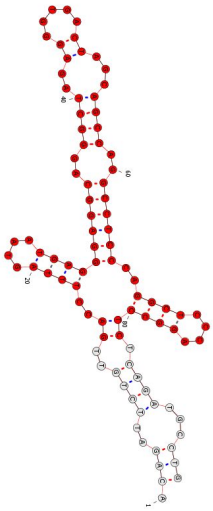
(a) : Domain IIIabc according to [42]



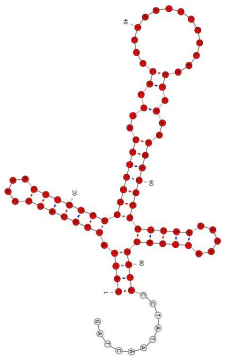
(b) : *XM_011524020 dynein reg. complex subunit 3* (DRC3)



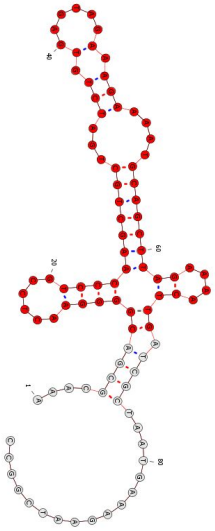
(c) : *NM_032737 lamin B2* (LMNB2)



(d) : *NM_001190457 coronin 2B* (CORO2B)



(e) : *XR_001739336 uncharacterized LOC102723389*



(f) : *XM_017006507 contactin 3* (CNTN3)

Figure 9.20: Hits obtained from searching the **S2** query in candidate sequences from *human mRNA* database. The 18 structures, that matched the query, originate from 5 distinct sequences. The best structure for each sequence is shown along with its NCBI accession. The red part of the structure matches the query descriptor. All sequences/structures have the same length.

DRC3 gene region remains as the best candidate with the highest e-value, match length and the second highest *NA2D* score with no significant violations introduced by the variable part of the query. The hit almost perfectly preserves the structure of the bottom multi-loop (corresponding to the IIIabc junction) so critical for IRES activity. Only the right hairpin (*domain IIIc*) is mismatched, having unpaired A-G bases. Continuing upwards to the stem (beginning of *domain IIIb*), we see, that it is shorter by 4 base pairs from ideal. The follow-up internal loop is shorter by 1nt on its left side. The rest of the structure, corresponding to the variable node of the query, could have been arbitrary. We however see, that the upper stem has almost ideal length of 8 base-pairs (9 in the hit). Final part of the hit is more structured than the actual domain, having an extra left bulge and internal loop (as opposed to a single hairpin loop). This region is however more structured as well in the case of the folded true domain (see figure A.1) and can be a folding discrepancy.

9.7.3 NA2Dsearch results - *human genomic* sequences

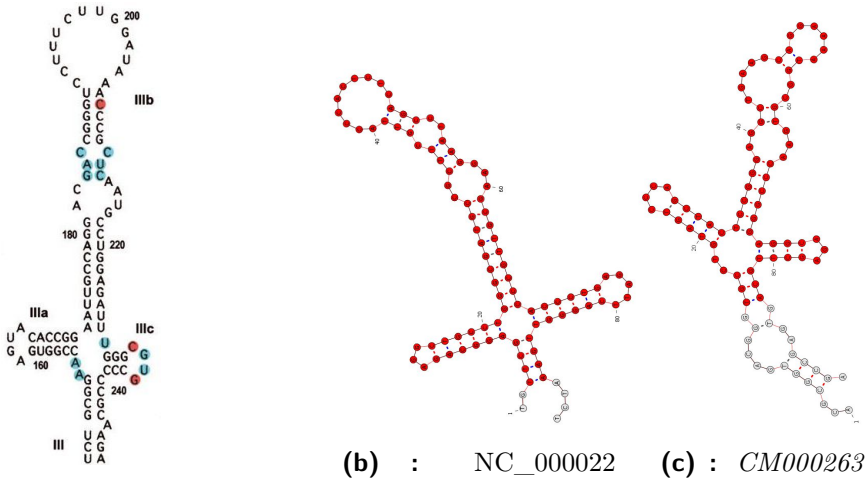
Also in case of candidates from *human genomic* database, only query **S2** (figure 9.19) yielded results. 36 structures matched the query, this time having only 2 sequences as sources. Such a configuration indicates the overall shape of the structures are energetically stable enough to form multiple structural variations matching the query.

rec. accession	gene name	s. nt	5'UTR	e-value	mlen	TED	NA2D
NC_000022	LARGE1 (i)	-34791821	no	0.514	87/93	25	-29
CM000263	EP400 (rci)	-132080532	no	0.147	71/93	34	-50

Table 9.5: Summary of hits combining results from BLAST and *NA2Dsearch*. The accessions code for chromosomes 22 and 12 respectively. The gene names were derived from the hit locations within the chromosomes. *i* and *rci* abbreviations stand from intronic and reverse-complement intronic regions respectively. These results originate from the *human genomic* database and were obtained from searching sub-optimal structures of BLAST hits with e-value < 1.

From the table we can again notice the last three columns. The first hit is superior in the matched length and TED, but lacks in NA2D score. Looking at the figure 9.21 is however evident, that despite the NA2D score, the first candidate resembles *domain III* far more, than the second one. The heavy penalty is a result of missing both single stranded regions at the bottom multi-loop. Even it seems as a small discrepancy, these nucleotides are conserved and are part of the functionally vital part of the domain.

As *human genomic* database contains only large shotgun sequences, the information on what gene the hit lays in needs to be extracted from sequence viewers. The first hit is within intronic region of LARGE1 gene, while the second lays in the reverse-complement to the intronic region of EP400 gene (supplementary figures).



(a) : Domain IIIabc [42] **(b) :** NC_000022 (LARGE1) **(c) :** *CM000263* chromosome 12

Figure 9.21: Hits obtained from searching the **S2** query in candidate sequences from *human genomic* database. 36 structures were matched, originating from 2 distinct sequences. Blue letters in the scheme indicate conserved nucleotides.

9.7.4 Searching for domain II in the vicinity of the hit

The presence of one particular domain, however preserved, is not sufficient to determine a functional IRES element [24]. Therefore, to elevate the hit significance, we look around the hit region for other potential IRES subdomains. In this case we will be looking for *domain II* as it interacts with 40S subunit as well (see schema 2.1). Assuming the obtained candidates are truly a *domain IIIabc* structure, the *domain II* would start approximately 100nts upstream and would span over 70nts. Taking the DRC3 hit as an example, whose supposed *domain III* begins at nt 198 at the DRC3 transcript, we would be interested in its region bounded by nts 98-168. The search area is further extended to nts 70-180 to explore wider surroundings of the suspect region. As the regions of interest are larger than the domain, they will not be folded in their full length, as it has been in the case of *domain IIIabc* search, but instead would be scanned using a sliding window of the *domain II* size.

A query corresponding to the domain structure is assembled according to the *domain II* structure. Individual stems and loops are allowed to be ±1-2 bases longer/shorter. No variable node is used.

A domain II-like structure was indeed found as seen from figure 9.22. The hit spans over nucleotides 115-167 of the DRC3 gene (the expected region was 98-168). The hit therefore not only resembles the *domain II*, but is located where the domain could be expected. The only issue with the candidate might be lower thermodynamic stability - the one of the MFE structure is -19.9 while the candidate's one is -17.6 kcal/mol (for comparison, the MFE structure of the *domain II* sequence matches its true structure).

Applying the search on the remaining 3 *human mRNA* candidates (CORO2B candidate was excluded as the hit starts at nt 1 and has no upstream region) and 2 *human genomic* candidates, no hits were obtained even when the folding

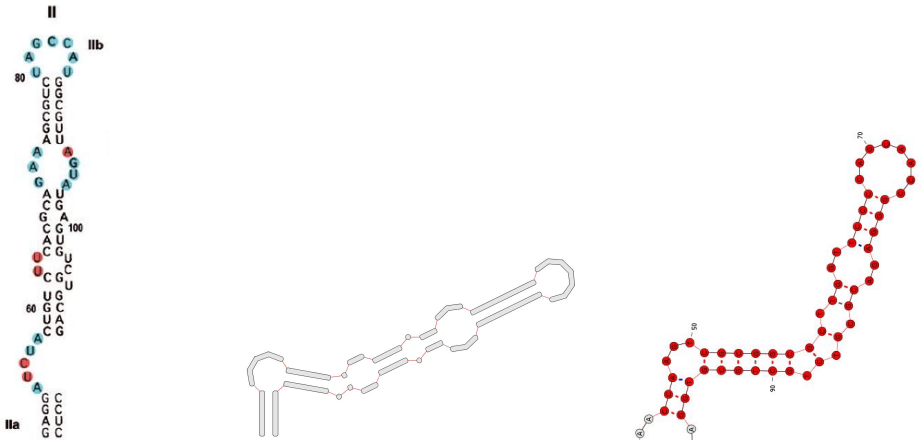


Figure 9.22: *Domain II* schema, corresponding structural query and hit respectively. Blue nucleotides represent conserved regions. The query was set up to allow small variations (1-2nt) in individual structural elements.

energy range was extended by additional 0.5kcal/mol [5]. The fact, that a *domain II*-like structure is present in the DRC3 candidate and simultaneously is absent in the remaining candidates, further boosts the significance of the initial *domain IIIabc* hit.

9.8 Biological details on the DRC3 hit

The best hit in figure 9.20(a) has its sequence contained not in a single, but in multiple *RefSeq* records - in particular RNAs with accession numbers XM_011524020.2 XM_024450962.1 and XM_024450965.1. All these records, being only different transcript variants, belong to the DRC3 gene, which codes a protein *dynein regulatory complex subunit 3*. Its coding region starts at nt position 444-448 (depending on the variant), meaning all the results are located within the gene's 5'UTR (see table 10.1). This gives the hit greater importance, as it indicates the region might have potential transcription regulatory function. The XM prefix of records indicates they fall under predicted mRNAs. That makes the hit position (from which it gains its significance) not entirely confirmed. On the other hand, verification of some parts exists in a form of *EST evidences* (expressed sequence tags - short sub-sequences of sequenced cDNA).

Lastly, the sequence of the DRC3 hit together with the inverse sequence leading to its discovery can be viewed on figure A.2.

[5] In fact, the were no hits even for further relaxed version of *domain II* containing 2 variable nodes (query figure in A.3)

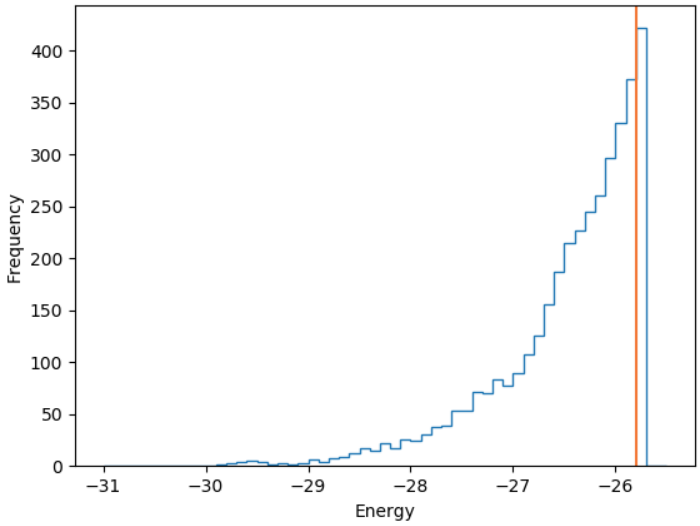


Figure 9.23: The sequence of the DRC3 gene hit was folded within $\Delta\Delta G =$ 5kcal/mol, 3750 structures were obtained. The graph shows energy distributions of the sub-optimal structures. Energy of the hit structure is -25.8 and is indicated by the vertical line. The MFE structure has energy -30.8kcal/mol

9.9 Discussion

A few similar attempts have been made to use inverse folding and BLAST for discoveries of novel IRES or riboswitch elements [18] [72]. All of them were motivated by the need to circumvent searches based on a sliding window, which restricts the search domain to a few genes, and achieve genome wide scans. These scans however comes with obstacles of their own. The major one is an issue with a huge number of potential candidates, either in a form of inverse sequences or subsequent BLAST hits. Each of the methods dealt with the issue in their own way. Either by pre-processing, when the generation of inverse sequences is heavily controlled [72] (to BLAST only the most promising candidates). Or by post-processing, relying on filters and biological indicators applied on hits [18]. Our method took the later path, with unsuccessful attempt for incorporating the first path as well.

The distinction from previous methods is in the strong emphasis on the secondary structure, while completely neglecting the primary one. Inverse sequences are not constrained and hits are processed by an exhaustive structural search. It does not mean our approach is more correct. Perhaps more general, as no sequence motifs are forced into inverse sequences, but at the same time less targeted.

Due to the nature of the task, the method proposed here is not a classical machine learning approach in a sense of training and testing phases. Therefore, it is not possible determine metrics such as selectivity and sensitivity. The only validation, able to confirm or reject the findings, is biological in wet laboratory.

From computational perspective, the validation has a form of evaluating structural similarity, thermodynamic stability of candidate structures and various biological signals (e.g. position within 5'UTR, conservation of vital elements, etc.). And with these tools in hands, our findings in the DRC3 gene seem certainly promising.

Chapter 10

Materials and methods

In this chapter, more details will be given on the methodology of this work. Technical information omitted from the result chapter will be presented as well.

The HCV sequence (genotype 1) used in the work was obtained from the *Virus Pathogen Database (ViPR)* [68]. Except for a handful of nucleotides, the IRES sequence matches the one published by *Khawaja, et al.* [42]

The infrastructure for the pipeline is implemented in Python 3.6 together with *BioPython* framework for easier manipulation with RNA/DNA sequences as well as phylogenetic trees. Key components of the source are covered by unittests. The pipeline integrates other command line tools such as *NUPACK-design* for inverse folding, Vienna package programs for various utility functionality (e.g. *RNAdistance* for tree edit distance, *RNAsubopt* for folding) and BLAST+ applications (mainly *blastn* for searching).

10.0.1 Inverse folding step

The inverse step was straightforward, consisting only of invoking *NUPACK* with desired parameters. Among them, the most important is the structure of the *domain IIIabc* (the rest of them was shown in the section 9.3). Generating of 102 000 sequences took over 100h to complete. The computation was performed on a desktop computer with an Intel Core i7-8700 3.2GHz processor and 46GB RAM.

10.0.2 BLAST step

Next comes the BLAST search. As the most costly operation, we want to perform the search only once - create a large pool hits, which would be then only filtered and processed. For that reason, it is convenient to have BLAST results saved into a persistent database. This step also allows us to pause the search and resume it any time, as the partial results are safely stored.

We made use of the *BioSql* [1] database schema, a native extension to the *BioPython* framework. Only part of the schema was used. On the other hand, the database model needed to be extended by custom classes to suit

[1] accessible from `https://biopython.org/wiki/BioSQL`

the purposes of the pipeline. The full schema is shown below. As for the database itself, a standard MySQL 8.0 was used.

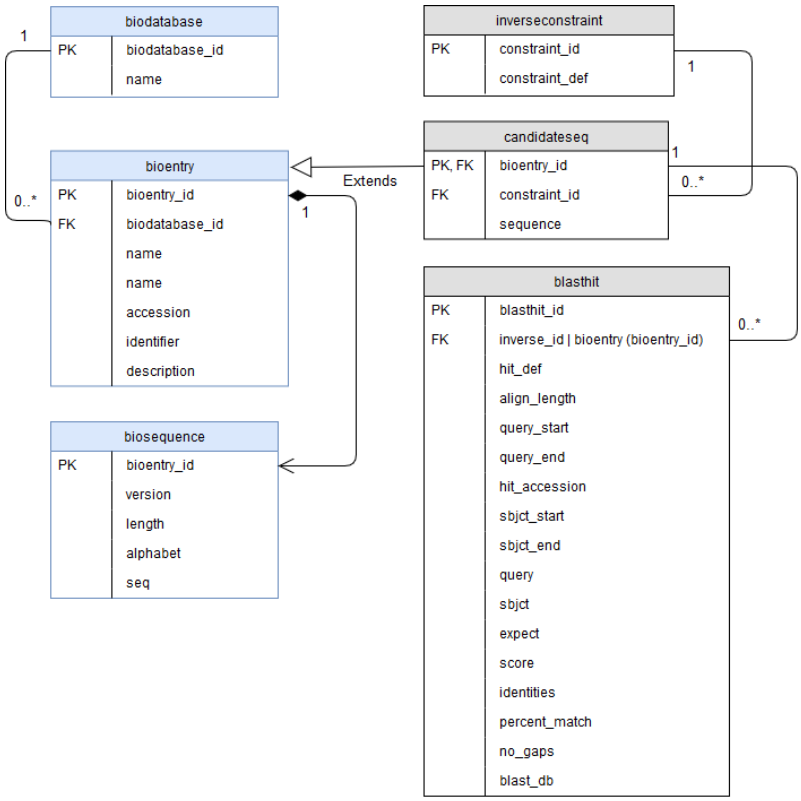


Figure 10.1: Database schema used in the pipeline. Blue classes are native *BioSql* models (only relevant attributes included) and together represent an RNA/DNA record. The fields carry basic information like accession numbers or the sequence itself. Grey models are custom extensions where *candidateseq* represent an inverse sequence, which has been generated by an *inverseconstraint* and could produce a *blasthit*. The *blasthit* record then holds information on the positions of the hit (*query* and *sbjct* starts and ends), its significance (*expect*, *score*) and more. Fields are the mixture of VARCHARs and numeric types, which are not shown for the sake of readability.

From 102 000 sequences searched against the *human mRNA* database, around 40% did not yield any result, even with a highly permissive e-value threshold of 3 [2]. The rest yielded 560 000 hits, from which 116 000 had unique subject sequences. The whole search took 13.7h to complete. Such a low execution time (in compare to the inverse phase) is however due to reduced size of the database. When full *human genomic* was searched, the time required was not less, than 300h. Regarding the *human genomic database*, over 150 000 hits were found originated from over 6500 inverse sequences (6.5% of all sequences). The lower number of hits in compare to *human mRNA* can be addressed to the e-value threshold. The threshold of 3 in a smaller database allows for much more hits, than the same threshold in a bigger one.

[2] this level of significance was inspired by a similar method [18]

Nevertheless, the set threshold is still sufficiently high even for databases of the *human genomic* size as the widely accepted significance values are all below zero.

10.0.3 NA2Dsearch step

Structural search with *NA2Dsearch* was conducted on a selection of BLAST hits filtered by their e-value. The hit sequences were elongated to match the domain length and by this, a list of candidate regions was assembled. The candidates were all folded within $\Delta\Delta G = 5$kcal/mol and checked for similarity with queried domain. This part was completed in 1-2 hours.

database	e-value	candidates	searched	hits	unique	time
human mRNA	0.5	11 877	76.6mil	18	5	98m
human genomic	1	5 126	39.8mil	36	2	58m

Table 10.1: Overview of the *NA2Dsearch* run. The first column represents e-value cut-off for BLAST hits to filter out insignificant ones. The number of resultant candidates is under the second column. Folding them gives number of structures searched (*searched*), from which only a handful match the query (column *hits*). Many of the structures originate from the same sequence, as indicated by the last column *unique*.

For this final stage, human involvement is necessary. Unless a very strict and unambiguous query is compiled, the set of hits needs to be curated. As seen from the results, no scoring (neither TED, nor *NA2Dsearch* one) is an accurate measure of hit quality, in case variable nodes are introduced into a query.

Chapter 11

Conclusion

Numerous RNA structures carry an important role in gene expression. The goal of this thesis was to propose a procedure for searching of such structures in human genome. Model structure chosen to show working of the procedure was HCV IRES *domain IIIabc* - a structure known to attract crucial translation initiation factors.

A pipeline for discovering IRES-like structural motifs was proposed. Due to an enormous search space, an emphasis was put on feasibility of the task. For that, the search was partially moved from structural to sequential domain. The keystones the workflow were inverse folding, BLAST of inverse sequences and structural search in folded candidates. We refrained from an intermediate step of clustering/filtering inverse sequences, as no relevant reduction of sequences could have been achieved. Inverse folding was conducted by *NUPACK-design* program, that has been selected from three candidates after thorough evaluation. Over 100 000 sequences were generated and searched by BLAST against two databases - *human mRNA* and larger *human genomic*. Obtained hits with significant level of e-value were then given to the *NA2Dsearch* tool searched again (this time structurally) for a match with the target *domain IIIabc*.

Several candidates matched the relaxed version of the domain structure. The most significant one matched almost entirely with the structural query. The origin of the corresponding sequence was the DRC3 gene. The hit significance is further supported by its location within the 5'UTR region together with a nearby presence of *domain II*-like structure, which suggests its potential regulatory abilities. Verification of such hypothesis is given to the biologists.

The results shown the pipeline is able to detect structural features in human genome sequences. The method, with regard to the magnitude of the task, cannot be exhaustive and finding a significant hit is therefore not guaranteed. Nevertheless, the pipeline provides an approach, that gives an approximate solution to a complex task, for which currently no tools exist.

11.1 Contributions

- Rigid biological foundations compiled from relevant sources to ensure biologically correct results
- Comprehensive summary of RNA structure prediction methods
- Review and benchmarks of selected inverse folding methods on quality of generated sequences
- Review of all existing solutions to the task of identifying potential IRES elements.
- Computational pipeline combining inverse folding, sequence search and structural prediction to solve the above task. The pipeline can be used to search other types of functional elements as well.
- Discovery of an area within DRC3 5'UTR region containing both HCV *domain IIIabc* and *domain II*-like structures. The hit needs to be and will be validated in laboratory.

11.2 Future work

This pipeline is only a proof of concept demonstrated on a single structure. Many steps can be optimized or done differently. We present several possible improvements.

11.2.1 Inverse folding step

Only in inverse folding there are few - one can constrain the sequences, forcing conserved nucleotide motifs; or in contrary prevent certain nucleotide patterns. Applying these features would make inverse folding more targeted with reduced number of possible sequences. The decision on whether (and what) constraints to use should be left to the expertise of biologists.

Regarding the number of possible inverse sequences - over 100 000 have been generated. The question is, what part of all possible inverse sequences this set makes up? An answer to this question is not straightforward. We know the general working of *NUPACK* - the steps are however probabilistic and involve a great deal of sampling. Without closer examination of the algorithm, a knowledgeable estimate cannot be given. Knowing such a metric could greatly influence the overall strategy of the pipeline.

11.2.2 BLAST step

E-value is the main indicator of significance of BLAST hits. We used this value in our pipeline to choose candidates for structural evaluation. As mentioned in the result section however, this metric turns out to be (at least for our

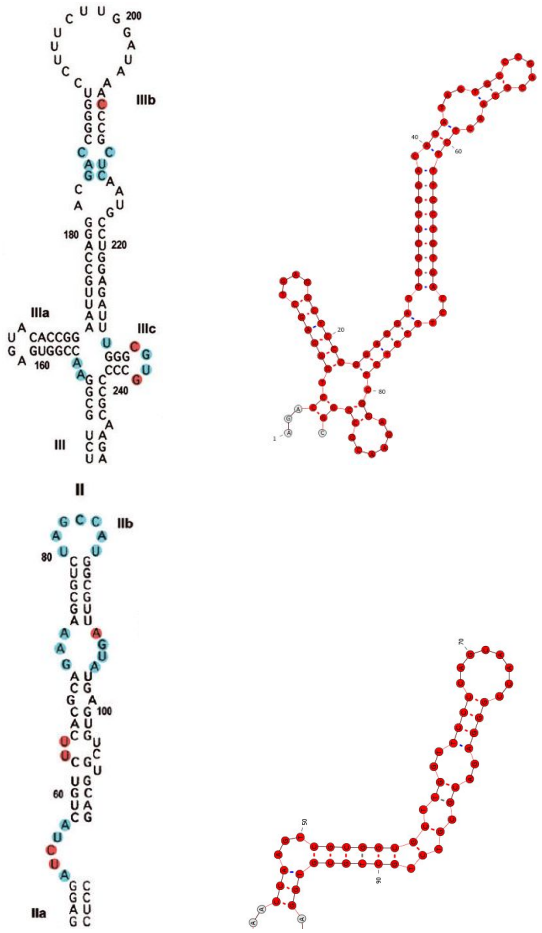


Figure 11.1: Domains II and IIIabc according to[42] with corresponding structures found in DRC3 gene.

purposes) a rather inadequate criterion as low e-values do not guarantee better chances for a structural match. Filtering solely by this metric might be too simplistic. On the other hand, joining it with other structurally-aware indicators can provide more precise estimates on significance. For instance matching a short, but highly structured part of a query (e.g. a hairpin with a bulge) can be more valuable, than matching a long, but unstructured part (e.g. a loop or only a 3' half of a stem).

11.2.3 Transformation to a publicly available tool

Further development would naturally lead to an web application available for public use. As in many biological tools, a user would submit a searched structure with search parameters (number of inverse sequences, e-value cut-off, etc.) and the server would notify him about the result. In this case however, the process would be semi-automatic. The *NA2Dserach* step, with the convenient GUI, would a user conduct on his/her own, defining the set of various (relaxed) structural queries. The minimal viable product would entail

integration of *NA2Dsearch* with the pipeline on at least some basic level and exposing its user interface to a web application.

Bibliography

[1] Internal loops in rna secondary structure prediction. http://slideplayer.com/slide/4798843/15. Accessed: 2019-02-12.

[2] Rnastructure command line help file formats. https://rna.urmc.rochester.edu/Text/File_Formats.html. Accessed: 2019-02-12.

[3] Alberts, B., Bray, D., Hopkin, K., Johnson, A. D., Lewis, J., Raff, M., Roberts, K., and Walter, P. *Essential cell biology*. Garland Science, 2015.

[4] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. Basic local alignment search tool. *Journal of molecular biology 215*, 3 (1990), 403–410.

[5] Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research 25*, 17 (1997), 3389–3402.

[6] Andronescu, M., Bereg, V., Hoos, H. H., and Condon, A. Rna strand: the rna secondary structure and statistical analysis database. *BMC bioinformatics 9*, 1 (2008), 340.

[7] Andronescu, M., Fejes, A. P., Hutter, F., Hoos, H. H., and Condon, A. A new algorithm for rna secondary structure design. *Journal of molecular biology 336*, 3 (2004), 607–624.

[8] Araujo, P. R., Yoon, K., Ko, D., Smith, A. D., Qiao, M., Suresh, U., Burns, S. C., and Penalva, L. O. Before it gets started: regulating translation at the 5 utr. *Comparative and functional genomics 2012* (2012).

[9] Ben-Hur, A., Ong, C. S., Sonnenburg, S., Schölkopf, B., and Rätsch, G. Support vector machines and kernels for computational biology. *PLoS computational biology 4*, 10 (2008), e1000173.

[10] Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Sayers, E. W. Genbank. *Nucleic acids research 37*, suppl_1 (2008), D26–D31.

[11] Busch, A., and Backofen, R. Info-rna—a fast approach to inverse rna folding. *Bioinformatics 22*, 15 (2006), 1823–1831.

[12] Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., and Madden, T. L. Blast+: architecture and applications. *BMC bioinformatics 10*, 1 (2009), 421.

[13] Chang, C.-C., and Lin, C.-J. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST) 2*, 3 (2011), 27.

[14] Churkin, A., Retwitzer, M. D., Reinharz, V., Ponty, Y., Waldispühl, J., and Barash, D. Design of rnas: comparing programs for inverse rna folding. *Briefings in bioinformatics* (2017), bbw120.

[15] Darnell, J. E., Lodish, H. F., Baltimore, D., et al. *Molecular cell biology*, vol. 2. Scientific American Books New York, 1990.

[16] Ding, Y., Chan, C. Y., and Lawrence, C. E. Rna secondary structure prediction by centroids in a boltzmann weighted ensemble. *Rna 11*, 8 (2005), 1157–1166.

[17] Doshi, K. J., Cannone, J. J., Cobaugh, C. W., and Gutell, R. R. Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for rna secondary structure prediction. *BMC bioinformatics 5*, 1 (2004), 105.

[18] Dotu, I., Lozano, G., Clote, P., and Martinez-Salas, E. Using rna inverse folding to identify ires-like structural subdomains. *RNA biology 10*, 12 (2013), 1842–1852.

[19] Durand, D. Database searching and blast tuesday, october 27th.

[20] Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[21] Eddy, S. R. How do rna folding algorithms work? *Nature biotechnology 22*, 11 (2004), 1457.

[22] Elias, R., and Hoksza, D. Traveler: a tool for template-based rna secondary structure visualization. *BMC bioinformatics 18*, 1 (2017), 487.

[23] Eun, H.-M. *Enzymology primer for recombinant DNA technology*. Elsevier, 1996.

[24] Fernández-Miragall, O., de Quinto, S. L., and Martínez-Salas, E. Relevance of rna structure for the activity of picornavirus ires elements. *Virus research 139*, 2 (2009), 172–182.

[25] Gao, J. Z., Li, L. Y., and Reidys, C. M. Inverse folding of rna pseudoknot structures. *Algorithms for Molecular Biology 5*, 1 (2010), 27.

[26] Garcia-Martin, J. A., Clote, P., and Dotu, I. Rnaifold: a constraint programming algorithm for rna inverse folding and molecular design. *Journal of bioinformatics and computational biology 11*, 02 (2013), 1350001.

[27] Garcia-Martin, J. A., Dotu, I., and Clote, P. Rnaifold 2.0: a web server and software to design custom and rfam-based rna molecules. *Nucleic acids research 43*, W1 (2015), W513–W521.

[28] Gardner, P. Rna structure formats. `http://projects.binf.ku.dk/pgardner/bralibase/RNAformats.html`. Accessed: 2019-02-12.

[29] Gardner, P. P., and Giegerich, R. A comprehensive comparison of comparative rna structure prediction approaches. *BMC bioinformatics 5*, 1 (2004), 140.

[30] Gutell, R. R., Lee, J. C., and Cannone, J. J. The accuracy of ribosomal rna comparative structure models. *Current opinion in structural biology 12*, 3 (2002), 301–310.

[31] Hashem, Y., Des Georges, A., Dhote, V., Langlois, R., Liao, H. Y., Grassucci, R. A., Pestova, T. V., Hellen, C. U., and Frank, J. Hepatitis-c-virus-like internal ribosome entry sites displace eif3 to gain access to the 40s subunit. *Nature 503*, 7477 (2013), 539.

[32] Hellen, C. U., and Sarnow, P. Internal ribosome entry sites in eukaryotic mrna molecules. *Genes & development 15*, 13 (2001), 1593–1612.

[33] Henikoff, S., and Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences 89*, 22 (1992), 10915–10919.

[34] Hlubuček, P. Na2dsearch: a fast and easy tool for secondary structure searches through databases of nucleic acids in parallel.

[35] Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M., and Schuster, P. Fast folding and comparison of rna secondary structures. *Monatshefte für Chemie/Chemical Monthly 125*, 2 (1994), 167–188.

[36] Hofacker, I. L., Stadler, P. F., and Stadler, P. F. Rna secondary structures. *Reviews in Cell Biology and Molecular Medicine* (2006).

[37] Honda, M., Ping, L.-H., Rijnbrand, R. C., Amphlett, E., Clarke, B., Rowlands, D., and Lemon, S. M. Structural requirements for initiation of translation by internal ribosome entry within genome-length hepatitis c virus rna. *Virology 222*, 1 (1996), 31–42.

[38] Honda, M., Rijnbrand, R., Abell, G., Kim, D., and Lemon, S. M. Natural variation in translational activities of the 5 nontranslated rnas of hepatitis c virus genotypes 1a and 1b: evidence for a long-range rna-rna interaction outside of the internal ribosomal entry site. *Journal of virology 73*, 6 (1999), 4941–4951.

[39] Hong, J.-J., Wu, T.-Y., Chang, T.-Y., and Chen, C.-Y. Viral ires prediction system-a web server for prediction of the ires secondary structure in silico. *PLoS One 8*, 11 (2013), e79288.

[40] Karlin, S., and Altschul, S. F. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences 87*, 6 (1990), 2264–2268.

[41] Kertesz, M., Wan, Y., Mazor, E., Rinn, J. L., Nutter, R. C., Chang, H. Y., and Segal, E. Genome-wide measurement of rna secondary structure in yeast. *Nature 467*, 7311 (2010), 103.

[42] Khawaja, A., Vopalensky, V., and Pospisek, M. Understanding the potential of hepatitis c virus internal ribosome entry site domains to modulate translation initiation via their structure and function. *Wiley Interdisciplinary Reviews: RNA 6*, 2 (2015), 211–224.

[43] Kieft, J. S. Viral ires rna structures and ribosome interactions. *Trends in biochemical sciences 33*, 6 (2008), 274–283.

[44] Kolekar, P., Pataskar, A., Kulkarni-Kale, U., Pal, J., and Kulkarni, A. Irespred: web server for prediction of cellular and viral internal ribosome entry site (ires). *Scientific reports 6* (2016), 27436.

[45] Komar, A. A., and Hatzoglou, M. Cellular ires-mediated translation: the war of itafs in pathophysiological states. *Cell cycle 10*, 2 (2011), 229–240.

[46] Larkin, M. A., Blackshields, G., Brown, N., Chenna, R., McGettigan, P. A., McWilliam, H., Valentin, F., Wallace, I. M., Wilm, A., Lopez, R., et al. Clustal w and clustal x version 2.0. *bioinformatics 23*, 21 (2007), 2947–2948.

[47] Layton, D., and Bundschuh, R. A statistical analysis of rna folding algorithms through thermodynamic parameter perturbation. *Nucleic acids research 33*, 2 (2005), 519–524.

[48] Lodish, H., Berk, A., Zipursky, S. L., Matsudaira, P., Baltimore, D., Darnell, J., et al. *Molecular cell biology*, vol. 7. WH freeman New York, 2013.

[49] Lorenz, R., Bernhart, S. H., Zu Siederdissen, C. H., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. Viennarna package 2.0. *Algorithms for Molecular Biology 6*, 1 (2011), 26.

[50] Lozano, G., Fernandez, N., and Martinez-Salas, E. Modeling three-dimensional structural motifs of viral ires. *Journal of molecular biology 428*, 5 (2016), 767–776.

[51] Lukavsky, P. J. Structure and function of hcv ires domains. *Virus research 139*, 2 (2009), 166–171.

[52] Lyngsø, R. B., and Pedersen, C. N. Rna pseudoknot prediction in energy-based models. *Journal of computational biology 7*, 3-4 (2000), 409–427.

[53] Macke, T. J., Ecker, D. J., Gutell, R. R., Gautheret, D., Case, D. A., and Sampath, R. Rnamotif, an rna secondary structure definition and search algorithm. *Nucleic acids research 29*, 22 (2001), 4724–4735.

[54] Markham, N. R., and Zuker, M. Unafold. In *Bioinformatics*. Springer, 2008, pp. 3–31.

[55] Martínez-Salas, E., Francisco-Velilla, R., Fernandez-Chamorro, J., Lozano, G., and Diaz-Toledano, R. Picornavirus ires elements: Rna structure and host protein interactions. *Virus research 206* (2015), 62–73.

[56] Mathews, D. H. Revolutions in rna secondary structure prediction. *Journal of molecular biology 359*, 3 (2006), 526–532.

[57] Mathews, D. H., Sabina, J., Zuker, M., and Turner, D. H. Expanded sequence dependence of thermodynamic parameters improves prediction of rna secondary structure1. *Journal of molecular biology 288*, 5 (1999), 911–940.

[58] Mathews, D. H., Turner, D. H., and Watson, R. M. Rna secondary structure prediction. *Current protocols in nucleic acid chemistry 67*, 1 (2016), 11–2.

[59] Mattei, E., Ausiello, G., Ferre, F., and Helmer-Citterich, M. A novel approach to represent and compare rna secondary structures. *Nucleic acids research 42*, 10 (2014), 6146–6157.

[60] McCaskill, J. S. The equilibrium partition function and base pair binding probabilities for rna secondary structure. *Biopolymers 29*, 6-7 (1990), 1105–1119.

[61] Mokrejš, M., Mašek, T., Vopálenskỳ, V., Hlubuček, P., Delbos, P., and Pospíšek, M. Iresite—a tool for the examination of viral and cellular internal ribosome entry sites. *Nucleic acids research 38*, suppl_1 (2009), D131–D136.

[62] Mokrejš, M., Vopálenskỳ, V., Kolenatỳ, O., Mašek, T., Feketová, Z., Sekyrová, P., Škaloudová, B., Kříž, V., and Pospíšek, M. Iresite: the database of experimentally verified ires structures (www. iresite. org). *Nucleic acids research 34*, suppl_1 (2006), D125–D130.

[63] Nakashima, N., and Uchiumi, T. Functional analysis of structural motifs in dicistroviruses. *Virus research 139*, 2 (2009), 137–147.

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

[65] Ouyang, Z., Snyder, M. P., and Chang, H. Y. Seqfold: genome-scale reconstruction of rna secondary structure integrating high-throughput sequencing data. *Genome research 23*, 2 (2013), 377–387.

[66] Pánek, J., Modrák, M., and Schwarz, M. An algorithm for template-based prediction of secondary structures of individual rna sequences. *Frontiers in genetics 8* (2017), 147.

[67] Pestova, T. V., Shatsky, I. N., Fletcher, S. P., Jackson, R. J., and Hellen, C. U. A prokaryotic-like mode of cytoplasmic eukaryotic ribosome binding to the initiation codon during internal translation initation of hepatitis c and classical swine fever virus rnas. *Genes & development 12*, 1 (1998), 67–83.

[68] Pickett, B. E., Sadat, E. L., Zhang, Y., Noronha, J. M., Squires, R. B., Hunt, V., Liu, M., Kumar, S., Zaremba, S., Gu, Z., et al. Vipr: an open bioinformatics database and analysis resource for virology research. *Nucleic acids research 40*, D1 (2011), D593–D598.

[69] Pruitt, K. D., Tatusova, T., and Maglott, D. R. Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research 35*, suppl_1 (2006), D61–D65.

[70] Przybilski, R., Gräf, S., Lescoute, A., Nellen, W., Westhof, E., Steger, G., and Hammann, C. Functional hammerhead ribozymes naturally encoded in the genome of arabidopsis thaliana. *The Plant Cell 17*, 7 (2005), 1877–1885.

[71] Quade, N., Boehringer, D., Leibundgut, M., Van Den Heuvel, J., and Ban, N. Cryo-em structure of hepatitis c virus ires bound to the human ribosome at 3.9-å resolution. *Nature communications 6* (2015), 7646.

[72] Retwitzer, M. D., Kifer, I., Sengupta, S., Yakhini, Z., and Barash, D. An efficient minimum free energy structure-based search method for riboswitch identification based on inverse rna folding. *PloS one 10*, 7 (2015), e0134262.

[73] Sonnenburg, S., Rätsch, G., Schäfer, C., and Schölkopf, B. Large scale multiple kernel learning. *Journal of Machine Learning Research 7*, Jul (2006), 1531–1565.

[74] Sun, C., Querol-Audí, J., Mortimer, S. A., Arias-Palomo, E., Doudna, J. A., Nogales, E., and Cate, J. H. Two rna-binding motifs in eif3 direct hcv ires-dependent translation. *Nucleic acids research 41*, 15 (2013), 7512–7521.

[75] Taufer, M., Licon, A., Araiza, R., Mireles, D., Van Batenburg, F., Gultyaev, A. P., and Leung, M.-Y. Pseudobase++: an extension of pseudobase for easy searching, formatting and visualization of pseudoknots. *Nucleic acids research 37*, suppl_1 (2008), D127–D135.

[76] Thirumalai, D., Lee, N., Woodson, S. A., and Klimov, D. Early events in rna folding. *Annual review of physical chemistry 52*, 1 (2001), 751–762.

[77] Webb, C.-H. T., Riccitelli, N. J., Ruminski, D. J., and Lupták, A. Widespread occurrence of self-cleaving ribozymes. *Science 326*, 5955 (2009), 953–953.

[78] Weingarten-Gabbay, S., Elias-Kirma, S., Nir, R., Gritsenko, A. A., Stern-Ginossar, N., Yakhini, Z., Weinberger, A., and Segal, E. Systematic discovery of cap-independent translation sequences in human and viral genomes. *Science 351*, 6270 (2016), aad4939.

[79] Westhof, E., and Auffinger, P. Rna tertiary structure. *Encyclopedia of analytical chemistry* (2000), 5222–5232.

[80] Wilkinson, K. A., Merino, E. J., and Weeks, K. M. Selective 2'-hydroxyl acylation analyzed by primer extension (shape): quantitative rna structure analysis at single nucleotide resolution. *Nature protocols 1*, 3 (2006), 1610.

[81] Wu, T.-Y., Hsieh, C.-C., Hong, J.-J., Chen, C.-Y., and Tsai, Y.-S. Irss: a web-based tool for automatic layout and analysis of ires secondary structure prediction and searching system in silico. *BMC bioinformatics 10*, 1 (2009), 160.

[82] Yang, C. Y., Yang, J.-S., and Lian, F.-L. Safe and smooth: mobile agent trajectory smoothing by svm. *Int. Journal of Innovative Computing, Information and Control 8*, 2012 (2012), 4959–4978.

[83] Zadeh, J. N., Wolfe, B. R., and Pierce, N. A. Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of computational chemistry 32*, 3 (2011), 439–452.

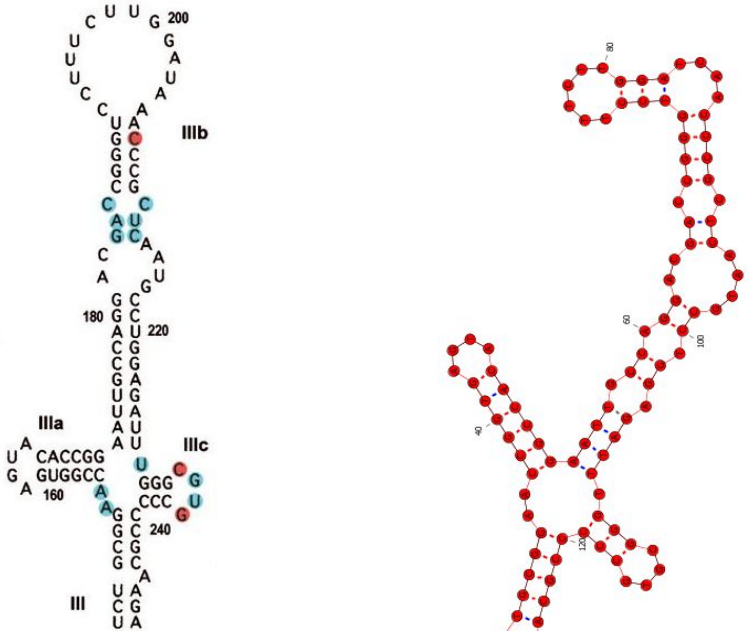
[84] Zhang, Z., Schwartz, S., Wagner, L., and Miller, W. A greedy algorithm for aligning dna sequences. *Journal of Computational biology 7*, 1-2 (2000), 203–214.

[85] Zhao, J., Wu, J., Xu, T., Yang, Q., He, J., and Song, X. Iresfinder: Identifying rna internal ribosome entry site in eukaryotic cell using framed k-mer features. *Journal of genetics and genomics= Yi chuan xue bao 45*, 7 (2018), 403.

[86] Zuker, M. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic acids research 31*, 13 (2003), 3406–3415.

Appendix A

Additional figures



(a) : Domain IIIabc [42]

(b) : Folded domain IIIabc

Figure A.1: Example of a folded domain III abc by *RNAsubopt*. Notice the mismatched pairs G176A223 and A179U220 in the beginning of the domain IIIb stem. Also the apical loop is always replaced by more structured elements.

GCCCAAGGGAGCAGAGGCTCCCGGAGGACTTATGTGCGCAACGTAAGATCGTGGTA
GTTGCTCAATTTTAAGTCCTCCAGGCCGAGGCCGGGC

AGACCTGGGAGCTGAGGCTCCCGGAGGACTTGCAGGGACAGATAGTGCCCGACGTA
ACTCTTTCCTGTGACCTTTTTCTCGGAGAACGCCGGC

Figure A.2: HCV *domain IIIabc* inverse sequence leading to the DRC3 hit and the DRC3 hit itself. The highlighted region was matched by BLAST.

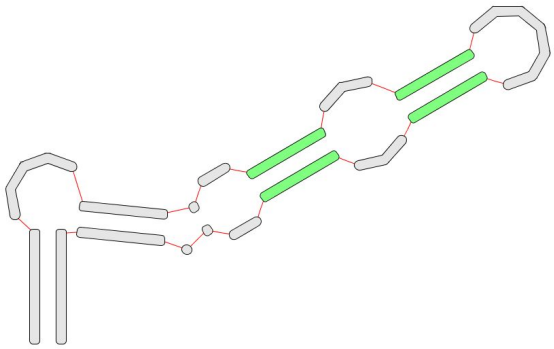
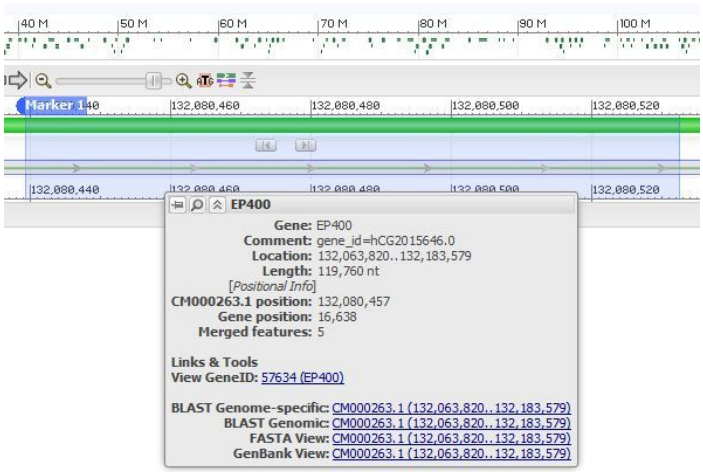
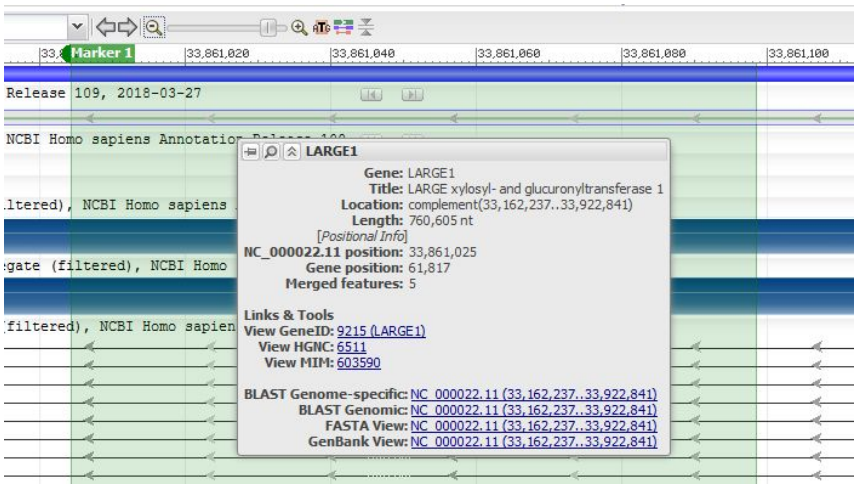


Figure A.3: Relaxed version of a query structure used to locate a potential *domain II*



(a) : EP400 intronic region



(b) : LARGE1 intronic region

Figure A.4: Regions in chromosomes 12 and 22, where hits of *domain IIIabc* structure are located (*human genomic* database). Regions are highlighted in blue and green markers and both lay on the reverse complement strand.

Appendix B

Supplementary data and documentation

B.1 Documentation

The installation steps are described in the README file of this project accessible on GitHub `https://github.com/lequyanh/hcvsearch`

B.2 Content of the CD

- The thesis itself
- Supplementary data
  - HCV sequence, *domain IIIabc* structure and folds
  - Results of *NA2Dsearch* searches for domains *IIIabc* and *II*
  - various miscellaneous data (samples of *RNAiFOLD* sequences, testing inputs and outputs of *IRESfinder*, etc.)
- Project source code