

Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Daniel Vojtek

Vyhľadávanie podobných sekundárnych štruktúr RNA

Kabinet software a výuky informatiky

RNDr. František Mráz, CSc.

Studijní program: Informatika, správa počítačových systémů (ISPS)

2007

Predovšetkým ďakujem RNDr. Františkovi Mrázovi, Csc za vedenie bakalárskej práce i ročníkového projektu. Ďalej chcem poďakovať Kataríne Regáľiovej za jej pomoc s otázkami z biológie. Nakoniec by som chcel poďakovať všetkým zamestnancom fakulty, mojím rodičom, spolužiakom a priateľom, bez ktorých by to nikdy nebolo možné.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 8.8.2007

Daniel Vojtek

Obsah:

1 Úvod	5
2 RNA	7
2.1 Typy štruktúr	7
2.2 Fylogenetický strom	12
2.3 Konzervované regióny nekódujúcej RNA	13
3 Výpočet sekundárnej štruktúry	16
4 Výpočet primárnej štruktúry	20
5 Porovnávanie sekundárnych štruktúr	22
6 RNA databázy	26
7 Súvisiace implementácie	27
7.1 Vienna RNA Package	27
7.2 BioPerl	28
8 Moja aplikácia	30
8.1 Algoritmus	30
8.2 Dátový model	32
8.2.1 RNA	32
8.2.2 Logovanie	34
8.3 Grafické rozhranie	34
8.4 Funkcie tretích strán	35
9 Užívateľská dokumentácia	36
9.1 Inštalácia	36
9.2 Užívateľské rozhranie	36
9.3 Načítanie sekvencií	38
9.3.1 Načítanie zo súboru	38
9.3.2 Vyhľadávanie v databáze GenBank	39
9.3.3 Ručné zadanie RNA	40
9.4 Návrh vzoru	40
9.4.1 Načítanie zo súboru	40
9.4.2 Návrh v dizajnéri	46
9.5 Praktické použite	51
10 Záver	57
Literatúra	58
Obsah disku CD-ROM	60

Názov práce: Vyhľadávanie podobných sekundárnych štruktúr RNA
Autor: Daniel Vojtek
Katedra: Kabinet software a výuky informatiky
Vedúci bakalárskej práce: RNDr. František Mráz, CSc.
e-mail vedúceho: mraz@ksvi.mff.cuni.cz

Abstrakt: Reťazce RNA na rozdiel od DNA môžu vytvárať zložité sekundárne štruktúry, ktoré významne ovplyvňujú ich vlastnosti. Cieľom tejto práce bolo vytvorenie aplikácie, ktorá umožňuje do detailov navrhnuť vzor sekundárnej štruktúry RNA a následne v danej množine sekvencií RNA vyhľadať reťazce s podobnými sekundárnymi štruktúrami. Aplikácia má aj priamy prístup do databázy GenBank, z ktorej sa na základe užívateľom zadaného dotazu sťahujú sekvencie RNA. Sekvenciám bez sekundárnej štruktúry sa táto dopočíta pomocou funkcií prevzatých z Vienna RNA Package. Súčasťou aplikácie je grafické rozhranie, v ktorom je možné vzor pohodlne navrhnuť, ako aj prehliadať nájdené podobné sekundárne štruktúry. Práca obsahuje úvod do problematiky hľadania sekundárnych štruktúr RNA a jeho možné využitie pri tvorbe fylogenetických stromov. Súčasťou práce je aj popis všetkých najdôležitejších postupov, ktoré boli použité.

Kľúčové slová: RNA, sekundárne štruktúry.

Title: Searching of similar secondary structures of RNA
Author: Daniel Vojtek
Department: Department of Software and Computer Science Education
Supervisor: RNDr. František Mráz, CSc.
Supervisor's e-mail address: mraz@ksvi.mff.cuni.cz

Abstract: In contrast to DNA, the RNA strings fold into complicated secondary structures, which are highly important for their function. Main goal of this work was to create an application, which allows designing a detailed pattern of RNA secondary structure and then finding similar ones in the given set of RNA. The application has also direct access to GenBank database, from which it can download RNA sequences according to the user-defined query. Sequences without secondary structure are folded using a function from Vienna RNA Package. A part of the application is a graphic user interface, which provides comfortable way to design a pattern and allows the browsing found similar secondary structures. The work also presents basic principles of searching of similar RNA secondary structures and its usage in building phylogenetic trees. Description of all used algorithms is present too.

Keywords: RNA, secondary structure.

Kapitola 1

Úvod

Obrovské množstvá dát, ktoré sú v súčasnosti získavané rôznymi formami sekvenovania organizmov, je veľmi zložitá a prakticky nemožné spracovávať bez použitia výpočtovej techniky. Základnou vedou v tejto oblasti je bioinformatika. Zaoberá sa ako návrhom tak aj implementáciou algoritmov schopných riešiť otázky a problémy biológov [1]. Jeden z takýchto problémov riešila aj pani RNDr. Helena Štorchová, CSc. z Ústavu experimentálnej botaniky AV ČR. Dr. Štorchová sa snažila na základe podobnosti sekundárnych štruktúr RNA určiť príbuzné druhy rastlín, najmä z rodu *Silene*. K tomu ju viedol objav jedného vzoru sekundárnej štruktúry, ktorý aj pri niekoľkých zmenách v sekvencii báz ostal prakticky neporušený. Pri výskume si pomáhala viacerými aplikáciami, avšak nemala k dispozícii komplexnú aplikáciu schopnú riešiť jej problém, čím by sa jej výskum výrazne zefektívnil a odbremenil sa o značnú časť rutínnej práce. Cieľom tejto práce bolo vytvoriť práve takúto aplikáciu. Keďže požiadavky neboli špecifikované veľmi striktné, rozhodol som sa riešiť problém čo najobecnejšie. Teda aby bolo možné v rámci jednej aplikácie vykonávať nasledujúce operácie:

- navrhnúť ľubovoľný vzor sekundárnej štruktúry,
- získať sekvencie RNA z databáz dostupných na internete,
- vypočítať k sekvenciám ich najpravdepodobnejšie sekundárne štruktúry,
- nájsť podobné sekundárne štruktúry k danému vzoru.

K tomuto účelu bolo nutné využiť rôzne podporné programy [2]. V druhej kapitole sú vysvetlené základy RNA a ich sekundárnych štruktúr potrebné k porozumeniu ďalších kapitol. Tretia kapitola popisuje výpočet sekundárnej štruktúry pomocou implementovanej funkcie z Vienna RNA Package a jeho nevhodnosť na dlhšie sekvencie. Štvrtá obsahuje opačný proces k výpočtu sekundárnej štruktúry, a to určenie primárnej štruktúry zo sekundárnej, ktorý je dôležitý pre optimalizáciu doby výpočtu. V piatej kapitole je rozobraný mnou navrhnutý spôsob porovnávania sekundárnych štruktúr. Šiesta kapitola obsahuje popis hlavných RNA databáz, kde sú v súčasnosti uchovávané takmer všetky známe RNA. Siedma kapitola je prehľadom najdôležitejších implementácií z oblasti bioinformatiky, ktoré som aj ja

využil na riešenie rôznych podproblémov. V ôsmej kapitole je predstavená moja aplikácia spolu s popisom architektúry a jej hlavného algoritmu. Deviata kapitola je užívateľská dokumentácia. Desiata kapitola je záver, kde sú uvedené výsledky práce a možné smery jej ďalšieho vývoja.

Kapitola 2

RNA

Táto kapitola obsahuje základný úvod do RNA [1] a jej sekundárnej štruktúry. Podkapitola 2.1 obsahuje popis jednotlivých štruktúr s dôrazom na sekundárnu štruktúru, elementy, z ktorých sa skladá, a taktiež spôsoby akým sa sekundárna štruktúra najčastejšie reprezentuje. Podkapitola 2.2 približuje fylogenetické stromy a ich spojitosť so sekundárnou štruktúrou RNA. Časť 2.3 pojednáva o konzervovaných a nekódujúcich regiónoch RNA.

Ribonukleová kyselina (RNA) je nukleová kyselina skladajúca sa z vlákna kovalentne naviazaných nukleotidov. Je biochemicky rozlíšiteľná od DNA, jednak prítomnosťou dodatočnej hydroxylovej skupiny, pripojenej ku každej pentózovej molekule reťazca, jednak využitím uracilu namiesto tymínu. Jednou z najpodstatnejších funkcií RNA je okopírovanie genetickej informácie z DNA (pomocou transkripcie) a fyzický prenos na miesto, kde následne dôjde k jej preloženiu (translácii) na výsledný proteín (priamo túto funkciu zabezpečuje mediátorová RNA). RNA má štyri rozdielne bázy: adenín (A), guanín (G), cytozín (C) a uracil (U). Prvé tri sa zhodujú s tými, ktoré sa nachádzajú v DNA, ale uracil nahrádza tymín v jeho komplementárnej funkcii k adenínu. DNA vytvára dlhé dvojvlákno, ale RNA vytvára relatívne krátke jednoduché vlákna. Jednotlivé nukleotidy v rade za sebou sú viazané pomocou fosfátových reziduí, ktoré spájajú uhlík 3' jednej deoxyribózy s uhlíkom 5' druhej deoxyribózy. Smer vlákien sa označuje práve podľa orientácie deoxyribózy v nej, teda 3'→5' a opačný smer 5'→3'. Pre nás tento smer nebude podstatný, preto sa ďalšími jeho detailami nebudeme zaoberať.

2.1 Typy štruktúr

U RNA sa rozlišuje primárna, sekundárna, terciálna a kvartérna štruktúra.

Primárna určuje poradie a typ báz v sekvencii: ACUGGGUUTTA. Primárnu štruktúru je teda možné porovnávať ako štandardné reťazce, pričom najčastejším spôsobom zobrazenia rozdielov a podobnosti dvojice primárnych štruktúr je

zarovnanie (alignment). Jedná sa vlastne o editačnú vzdialenosť, kde výsledkom môže byť napríklad minimálny počet operácií nutných na zmenu jedného reťazca na druhú. Povolené operácie sú vkladanie, mazanie a substitúcia jedného znaku za iný. To je možné robiť aj pre viac než dve primárne štruktúry naraz (multiple alignment). Príklad zarovnania dvoch primárnych štruktúr AACTGUAAU a ACCTGCAU je:

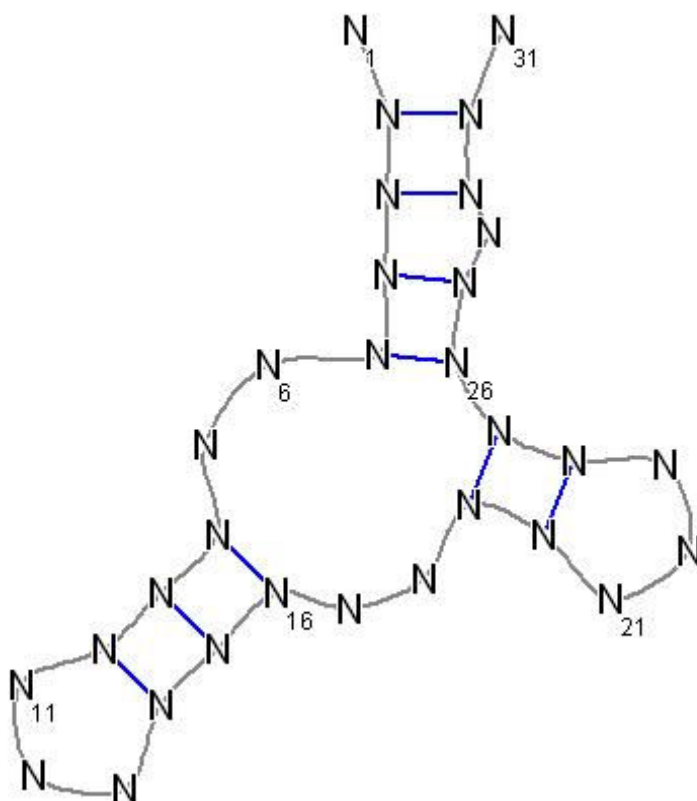
AA-CTGUAAU

A-CCTGCA-U

Na každom riadku je uvedená jedna primárna štruktúra. Odpovedajúce si bázy sú uvedené vždy v rovnakom stĺpci. Podčiarknuté bázy znamenajú substitúciu. Pomlčka v tomto prípade symbolizuje odlišnosť sekvencií, teda že k nukleotidu z prvej sekvencie nie je v druhej odpovedajúci nukleotid. Parametre zarovnania sa líšia podľa potreby. Môžeme totiž hľadať také zarovnania, ktoré majú celkový počet nezrovnalostí sekvencií väčší, ale nesmú byť vo veľkých zhlukoch, teda napríklad štyri po sebe idúce nezrovnalosti.

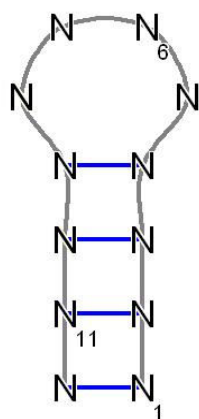
V sekundárnej štruktúre už dochádza k vytvoreniu bázičských párov medzi dvojicami báz z vlákna, podobne ako medzi dvojicami báz v dvojvláknovej DNA. V prípade RNA sa ale páry tvoria medzi bázami jedného vlákna, čím vznikajú zložené dvojrozmerné útvary. Nasledujúci obrázok č.1 obsahuje príklad jednoduchšej sekundárnej štruktúry popísanej grafom, kde N je konkrétna báza a páry sú určené modrou väzbou (vodíkové väzby, ktorých počet závisí na konkrétnych nukleotidoch). Bázičské páry medzi sebou tvoria len niektoré dvojice nukleotidov, a to A-U, C-G a U-G. Popis grafom je názorný, ale nie je vhodný pre výpočty. Pre tento účel bola zavedená zátvorková notácia, ktorej je niekoľko typov. Ďalej si tieto reprezentácie popíšeme podrobnejšie.

a) Popis sekundárnej štruktúry pomocou grafu:



Obr.1: Príklad sekundárnej štruktúry RNA. Čísla označujú poradie nukleotidov v sekvencii. Modrou farbou sú označené bázičné páry.

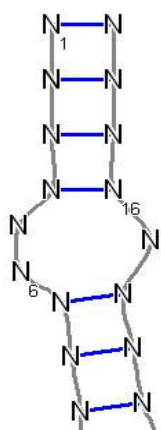
Sekundárne štruktúry tvorí niekoľkých základných elementov. Sú to:



← Vlásoka (Hairpin)

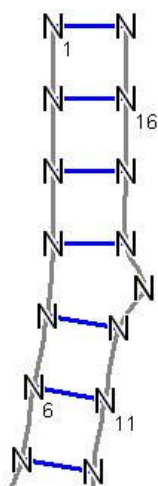
← Kmeň, kmienok (Stem) – bežne 5 až 10 párov

Obr.2



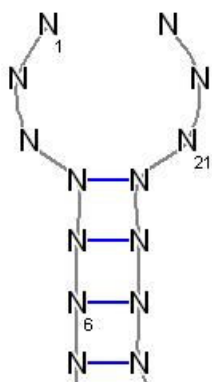
← Vnútný cyklus (Interior loop)

Obr.3



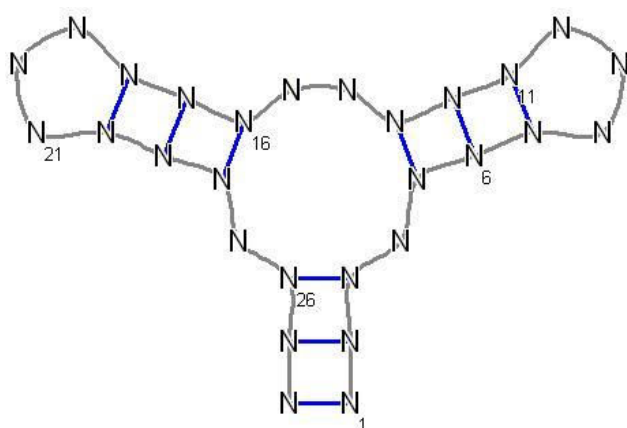
← Výduť (Bulge) – od vnútorného cyklu sa líši tým, že je len na jednej strane

Obr.4



← Koncové bázy (Ending bases, dangling ends).

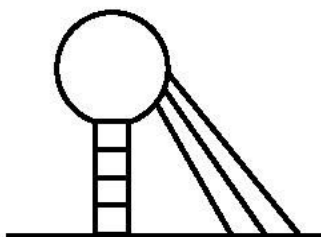
Obr.5



← Multicyklus (Multibranched loop) – rozpoznávame v ňom spárované a nespárované bázy.

Obr.6

Zložitejšími elementami sú tzv. pseudouzly. Tie sú ale častokrát považované už za elementy terciálnej štruktúry. Dochádza k nim, keď sa spárujú bázy z cyklov s bázami mimo cyklus:



← 3 pseudouzly (pseudoknot) z vlásenky.

Obr.7

Konkrétne sekundárna štruktúra na obrázku č.1 obsahuje 3 kmene, multicyklus, 2 vlásenky a výduť. Neobsahuje pseudouzly ani vnútorný cyklus.

b) Popis pomocou zátvorkovej notácie (bracket notation, dot-bracket form):
Je to štandardná reprezentácia sekundárnej štruktúry, kde každý pár zátvoriek reprezentuje jeden bázičný pár a nespárované bázy sa znázorňujú bodkami. Štruktúra na obr.1 má zápis: „(((((((.....))))).((.....))).“ – prvá bodka odpovedá prvému nukleotidu na obrázku (Symbol N označený číslom 1). Používa sa aj notácia s len dvomi symbolmi: P (paired, spárovaný) a U (unpaired, nespárovaný). Bodka je nahradená symbolom (U) a uzatvárajúca zátvorka je doplnená symbolom P. Zápis obrázku 1 pomocou tejto notácie by vyzeral nasledovne: „(U)((((U)U)(((U)U)U)P)P)P)U)U)U)U)U)P)P)P)P)U)P)P)U)“ – opäť

naľavo je nukleotid s označením 1. Bruce Shapiro navrhol ďalší spôsob. Reprezentuje jednotlivé štrukturálne elementy jedným párom zátvoriek a symbolom označujúcim element: H (hairpin, vlásenka), I (interior loop, vnútorný cyklus), M (multibranched loop, multicyklus), S (stack, kmienok), E (external, externé bázy). Pre obr.1 : „((((((H)S)((H)S)M)S)B)S)“, respektíve ak zahrnieme aj koncové bázy ako externé elementy tak „((((((H)S)((H)S)M)S)B)S)E)“. Tieto identifikátory môžu byť nasledované váhou, ktorá určuje počet nespárovaných báz alebo bázičkých párov. Teda opäť obr.1 : „((((((H3)S3)((H3)S2)M4)S2)B1)S2)E2)“. Často sa celá štruktúra uzatvára do virtuálneho koreňa R : „((((((((H3)S3)((H3)S2)M4)S2)B1)S2)E2)R)“.

Jednotlivé formy zápisu nie je vždy možné jednoznačne previesť na iný typ. Napríklad prevod z formátu podľa Bruce Shapira je len málokedy jednoznačne prevediteľný do dot-bracket formy. Táto forma neobsahuje informácie typu, na ktorej strane sa nachádza výduť.

Vo vyššom stupni organizácie, terciálnej štruktúre, pôsobia medzi jednotlivými elementmi sekundárnej štruktúry navyše van der Waalsove sily, ktoré tak vytvárajú 3D model RNA.

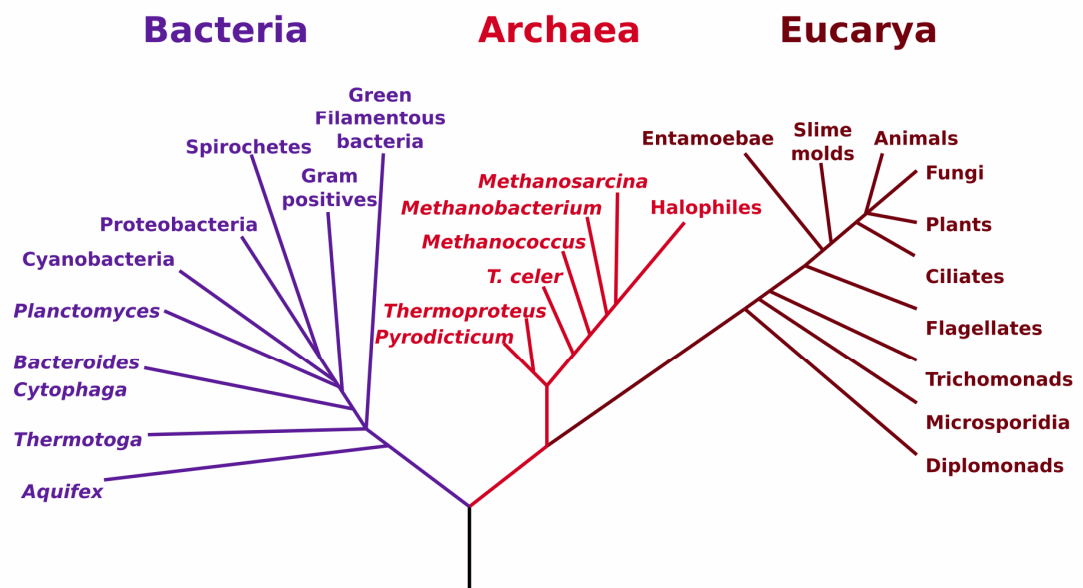
Kvartérnu štruktúru tvoria interakcie medzi molekulami RNA, ktoré vedú ku vzniku supramolekulárnych komplexov.

2.2 Fylogenetický strom

Fylogenetický strom, taktiež nazývaný evolučný strom, je strom, ktorý zobrazuje vzájomné evolučné vzťahy medzi rôznymi taxonomickými skupinami, u ktorých sa predpokladá spoločný predok [3,4]. Pojem fylogénéza predstavuje evolúciu, historický vývoj rastlinných druhov, živočíšnych druhov a taktiež ľudí. Taxonómia je systém rozdelenia rastlín a živočíchov, kde sú tieto zatriedené do kategórií na základe podobností. Fylogenetický strom je špeciálnym typom kladogramu, kde dĺžky vetiev sú proporcionálne k predpokladaným, hypotetickým evolučným časom medzi organizmami alebo sekvenciami. Bioinformatické nástroje vytvárajú kladogramy reprezentujúce vzájomné vzťahy medzi sekvenciami, či už DNA alebo sekvenciami aminokyselín. Keďže sú na výpočty použité jedine dáta v podobe sekvencií, sú tieto diagramy veľmi vhodné na určenie príbuznosti druhov.

Avšak kladogramy nemôžu byť považované za kompletne správny a presný popis evolučného vývoja organizmov, pretože v každom kladograme je niekoľko možných scenárov evolúcie, z ktorých každý vedie k rovnakému vzoru obsiahnutému v danom kladograme. Kladogram len zobrazuje, že dva organizmy, sekvencie sú s väčšou pravdepodobnosťou príbuzné navzájom ako s nejakým iným organizmom, sekvenciou. Nemusí však nutne objasňovať ako daný vzťah vznikol. Je ho ale možné použiť na vytvorenie nových hypotéz ohľadom vývoja a taktiež poukázať na niektoré vzťahy, ktoré by bez neho ostali nepovšimnuté. Príkladom je objavenie predpokladanej samostatnej vetvy vývoja medzi baktériami a eukaryotami, tzv. archae, ktorej prítomnosť bola určená pomocou vysoko konzervovanej časti vyskytujúcej sa v 16 rôznych mitochondriálnych RNA. Túto zakonzervovanú oblasť len s malými zmenami obsahujú všetky známe organizmy. Preto je na nej možné aj vypočítavať niektoré evolučné procesy.

Phylogenetic Tree of Life



Obr. 8: Fylogenetický strom života (zdroj: www.wikipedia.org)

2.3 Konzervované regióny nekódujúcej RNA

Zistilo sa, že nie všetky RNA molekuly, poprípade niektoré ich regióny, sa využívajú pri translácii proteínov. Takéto molekuly RNA a ich regióny sa nazývajú nekódujúce. DNA sekvencie, z ktorých sú ako koncový produkt transkribované

nekódujúce RNA, sa nazývajú nekódujúce RNA gény. Pri porovnávaní odpovedajúcich sa regiónov medzi príbuznými organizmami sa zistilo, že sa častokrát odlišujú len vo veľmi malom počte báz z celkovej dĺžky nekódujúcej časti. Táto vlastnosť je označovaná ako konzervovanosť. Konzervovaná býva nie len primárna štruktúra, ale taktiež sekundárna štruktúra. Výskum konzervovaných sekundárnych štruktúr naberá hlavne v poslednej dobe na dôležitosti, vďaka množstvu objavov v tejto oblasti [5-7]. Na uľahčenie práce biológom pri hľadaní sekundárnych štruktúr existuje veľké množstvo aplikácií, ktorých prístup býva často dosť odlišný [8]. Ako príklad uvediem dve aplikácie.

Alifold vyhodnocuje zarovnania viacerých sekvencií a hľadá v nich nezvyčajne energeticky stabilné a konzervované sekundárne štruktúry RNA [9]. Základ tvorí skript v programovacom jazyku Perl, ktorý prijíma zo štandardného vstupu zarovnania viacerých sekvencií vo FASTA formáte [10]. V tomto formáte sa štandardne popisujú sekvencie nukleových kyselín alebo proteínov, pričom jednotlivé bázičné páry, respektíve proteínové reziduá sú reprezentované jednopísmennými kódmi. Pred sekvenciou je taktiež možné na jednom riadku, ktorý musí začínať znakom '>', uviesť názov sekvencie a ďalšie komentáre, ktoré sa označujú ako hlavička záznamu. FASTA formát, hlavne informácie uložené v hlavičke, využívajú viaceré databázy sekvencií na automatické získavanie informácií o sekvenciách.

Príklad záznamu vo FASTA formáte:

```
>AB000263 |acc=AB000263|descr=Homo sapiens for prepro cortistatin
mRNAACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTG
CCCTGCCCCCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAG
GAAAAGCAGCCTCCTGACTTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGGCC
CTCATAGGAGAGGAAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGC
GCCGGGACAGAATGCCCTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAACCTCAC
CCATGAATGCTCACGCAAGTTTAATTACAGACCTGAA
```

Výstupom programu Alifold je zoznam energeticky najstabilnejších častí sekundárnej štruktúry v zátvorkovej notácii a taktiež PostScriptový obrázok s predpokladanou sekundárnou štruktúrou.

RNAProfile je názov algoritmu a taktiež aj aplikácie prístupnej pomocou formulára priamo na internete, ktorá vyhľadáva spoločné sekundárne štruktúry v nezarovnaných sekvenciách RNA [11,12]. Vstupné RNA sú vo FASTA formáte. Jediným vstupným parametrom je počet hairpinov, ktoré má hľadaný motív [1] obsahovať. Motív je v tomto prípade vysoko konzervovaná sekundárna štruktúra

vyskytujúca sa v skupine RNA. Tento parameter sa zadáva z dôvodu optimalizácie procesu vyhľadávania. Výsledkom sú regióny, ktoré sa vyznačujú najväčšou mierou konzervovanosti. Tá sa určuje na základe podobnosti, ktorá berie do úvahy sekvencie báz a taktiež sekundárne štruktúry, ktoré sa na nich tvoria. Algoritmus je možné rozdeliť na dve časti. V prvej sa z každej sekvencie určí množina kandidátskych regiónov, ktorých sekundárne štruktúry obsahujú určený počet vlásieniek (hairpin). V druhej časti sú tieto regióny navzájom porovnávané a určujú sa skupiny najpodobnejších. Aby sa predišlo úplnému prehľadávaniu sú použité heuristiky.

Kapitola 3

Výpočet sekundárnej štruktúry

Táto kapitola obsahuje popis dvoch algoritmov, ktoré je možné použiť na výpočet sekundárnej štruktúry. Jeden je triviálny, založený na maximalizácii počtu bázičných párov. Druhý optimalizuje veľkosť voľnej energie sekundárnej štruktúry a dosahuje tak ďaleko lepšie výsledky ako prvý algoritmus. Tento je v súčasnosti najpoužívanější. Pre ilustráciu zložitosti problému uvádzam aj reálnu dobu výpočtu algoritmu.

Existuje viacero spôsobov ako získať sekundárnu štruktúru RNA, pričom každý spôsob má svoje výhody aj nevýhody. Najpresnejšia metóda je vypočítavať sekundárnu štruktúru malých RNA pomocou komplikovaných elektronických zariadení. Táto metóda je však príliš nákladná na bežné použitie. Najpoužívanější spôsob je preto použitie výpočtovej techniky na výpočet vysoko pravdepodobných sekundárnych štruktúr danej RNA. Najlepšie výsledky dosahuje metóda, kedy sa za najpravdepodobnejšiu sekundárnu štruktúru považuje tá, ktorá za daných podmienok nadobúda minimálnu voľnú energiu, a teda je energeticky najstabilnejšia.

Jeden z prvých spôsobov predikcie sekundárnej štruktúry navrhla Ruth Nussinová [24]. Použila metódu dynamického programovania pre maximalizáciu počtu bázičných párov. Jej postup má viacero nedostatkov, pričom najdôležitejší je nejednoznačnosť nájdeného riešenia. Zjednodušený algoritmus vyzerá nasledovne:

1. inicializácia štvorcovej matice o veľkosti $n \times n$, kde n je dĺžka sekvencie.
2. Postupné vyplnenie matice, zaznamenávaním maximálnej možnej párovosti.
3. Spätné rekurzívne určenie konkrétnych bázičných párov.

Značne vylepšenú metódu poskytol Zucker a kol. [13-16]. Je založená na minimalizácii voľnej energie. Prevláda totiž názor, že RNA nadobúda také sekundárne štruktúry, ktoré sú energeticky najstabilnejšie a teda tie s najmenšou voľnou energiou. Štandardne sa vypočítaná energia štruktúry uvádza ako záporné číslo. V skutočnosti je to rozdiel energie oproti stavu, keby neboli spárované žiadne dvojice báz. Jeho algoritmus využíva energie, ktoré sú získané pomocou empirických kalorimetrických experimentov. Pre presnejšie výpočty sú hodnoty energií vylepšené aj o energie známych sekundárnych štruktúr. Jednotlivé bázy sú

očíslované tak ako za sebou nasledujú v sekvencii. Sekundárnu štruktúru potom popisujeme ako zoznam bázičských párov (i,j) kde $i < j$ také, že pre každú dvojicu bázičských párov (i,j) a (k,l) , kde $i \leq k$ platí:

$$i = k \Leftrightarrow j = l$$

$$k < j \Rightarrow i < k < l < j$$

Prvá podmienka implikuje, že každý nukleotid sa zúčastňuje na tvorbe maximálne jedného bázičského páru. Druhá zakazuje prítomnosť pseudouzlov. Táto podmienka je dôležitá pre použitý algoritmus dynamického programovania.

Bázičský pár (k,l) je vnútorný k páru (i,j) , pokiaľ $i < k < l < j$. Je priamo vnútorný pokiaľ neexistuje taký pár p,q , pre ktorý by platilo, $i < p < k < l < q < j$. V prípade koncového páru u vlásenky neexistuje k tomuto priamo vnútorný. Pre každý bázičský pár (i,j) a k nemu priamo vnútorný (pokiaľ existuje) sa definuje im odpovedajúci cyklus (nesúvisí s cyklami definovanými pri elementoch sekundárnej štruktúry) ako taký, ktorý obsahuje (i,j) , priamo vnútorný pár k (i,j) a všetky nespárované regióny pripojené k obidvom týmto párom zároveň. Energia sekundárnej štruktúry sa potom určuje ako suma príspevkov energií všetkých cyklov (pár v kmeni tvorí cyklus o veľkosti nula, pretože medzi pármami nie je žiadny nespárovaný región). Ako dôsledok tejto aditivity príspevkov energií, môže byť minimálna voľná energia vypočítaná rekurzívne dynamickým programovaním. Experimentálne určené energetické parametre, využité na učenie príspevku cyklu sú funkcie jeho veľkosti, typov určujúcich bázičských párov a čiastočne aj sekvenciami nespárovaných vláken. Tieto sú štandardne merané pri teplote $T = 37^\circ\text{C}$. Bázy, ktoré sú na koncoch vlákna a nie sú súčasťou žiadneho elementu neprispievajú do celkovej energie štruktúry.

Je to algoritmus založený na dynamickom programovaní, v ktorom sa postupne vyplňujú dvojrozmerné polia F , C a FM . $F[i,j]$ označuje minimálnu energiu pre subsekvenciu zloženú z báz od i do j . $C[i,j]$ je energia keby sa i a j spárovali. Pole FM je prítomné z dôvodu možných multicyklov. Energetické parametre pre všetky cykly okrem multicyklov sú definované ako funkcia $Interior(i,j; p,q)$ označujúca príspevok energie cyklu uzatvoreného dvojicou bázičských párov $i-j$ a $p-q$. Príspevok multicyklu sa určí ako $F = cc + ci * I + cu * U$, kde I je počet vnútorných bázičských párov a U je počet nespárovaných jednotiek v cykle. Cc , ci a cu sú konštanty určené empiricky. Postupuje sa od najkratších sekvencií po najdlhšie (narastá premenná d), kde sa na výpočet dlhších používajú výsledky spočítané u kratších. Celková voľná

energie bude na konci algoritmu v poli $F[1,n]$. Časová zložitosť je $O(n^4)$. Je ju ale možné o jeden rád znížiť obmedzením cyklov konštantou, napríklad 30. Dôležitá časť algoritmu:

```

For (d = 1 .. n)
  For (i = 1 .. d)
    j = i + d
    C[i,j] = MIN( Hairpin(i,j),
                  MIN( i<p<q<j : Interior(i,j;p,q)+C[p,q] ),
                  MIN( i<k<j : FM[i+1,k]+FM[k+1,j-1]+cc) )
    F[i,j] = MIN( C[i,j], MIN( i<k<j : F[i,k]+F[k+1,j]))
    FM[i,j] = MIN(C[i,j]+ci, FM[i+1,j]+cu, FM[i,j-1]+cu,
                  MIN(i<k<j : FM[i,k] + FM[k+1,j] ) )
Free_energy = F[1,n]

```

Naznačený algoritmus ešte nepočíta priamo sekundárnu štruktúru. Na jej určenie sa používa špeciálna funkcia (partition function). Táto funkcia dokáže určiť pravdepodobnosť, že dvojica báz tvorí pár. A z pravdepodobností sa už určujú tie štruktúry, u ktorých je celková pravdepodobnosť veľmi vysoká.

Práve tento algoritmus som využil na skladanie sekvencií v mojej aplikácii. Nevýhodou je ale jeho náročnosť na čas. Nasledujúca tabuľka zobrazuje reálne časy skladania závislé na dĺžke sekvencie získané z mojej aplikácie:

Dĺžka sekvencie (počet báz)	Doba výpočtu (sec)
250	0,16
400	1
600	2
1000	5
2000	20
4000	150 (2.5 min)
6000	480 (8 min)
8000	1000 (>16 min)
10000	1800 (30 min)
12000	3100 (>51 min)

Tab. 1: Závislosť doby výpočtu od dĺžky sekvencie (u času menšieho ako jedna sekunda bol meraný čas pre 1000 opakovaní a výsledkom je podiel celkového času a počtu opakovaní). Časy sa líšia mierne v závislosti od sekvencie a viac od výkonu počítača, predovšetkým CPU. Testovací stroj: notebook Lenovo Z61m s CPU Intel Centrino Duo T2400, s jadrami taktovanými na 1.83GHz a 2.00 GB RAM.

Z nameraných časov vyplýva napríklad, že 1000 krát poskladať sekundárnu štruktúru sekvencie o dĺžke 250 báz zaberie približne 3 minúty, čo odpovedá jednému poskladaniu sekvencie dlhej 5000 báz a desatine doby skladania sekvencie o 10000 bázach. Preto som sa rozhodol skladaniu rozsiahlych RNA vyhnúť a navrhol som optimalizáciu, ktorá vytipuje vhodné regióny v dlhých RNA a až na tie aplikuje algoritmus skladania.

Kapitola 4

Výpočet primárnej štruktúry

Táto kapitola sa zaoberá výpočtom primárnej štruktúry zo sekundárnej a dostupnou aplikáciou, ktorá tento výpočet vykonáva.

V prípade, že máme sekundárnu štruktúru RNA bez určených báz, tak sa nám môže hodiť určiť, ktoré sekvencie takúto sekundárnu štruktúru skutočne nadobúdajú. Keďže funkcia mnohých RNA závisí veľkou mierou na ich štruktúre, tak je týmto spôsobom možné konštruovať RNA molekuly so špecifickými štrukturálnymi vlastnosťami. Praktické využitie je pri tvorbe nových ribozómov alebo konštruovaní umelých RNA nanoštruktúr. Obecné sa tento prístup získavania sekvencií zo sekundárnej štruktúry označuje ako inverzné skladanie (inverse folding), keďže ide o opačný proces ako v prípade skladania (folding) sekvencií do sekundárnych štruktúr.

Ja som tento postup využil ako heuristiku na vytipovanie regiónov sekvencie, u ktorých je vysoká pravdepodobnosť, že danú sekundárnu štruktúru skutočne nadobúdajú. U vzoru sekundárnej štruktúry sú totiž častokrát dôležité aj konkrétne typy báz, čo je aj prípad výskumu dr. Štorchovej, ktorej hľadaný vzor obsahoval aj obmedzenia na konkrétne typy báz. Napríklad môžeme definovať kmienok vo vzore ako taký, ktorý okrem toho, že obsahuje práve tri bázické páry, tak musí obsahovať aj dva páry A-U. Tým sa vlastne kladú isté požiadavky aj na primárnu štruktúru, a to som pri tejto heuristike využil.

Jeden z algoritmov, ktorého implementácia je dostupná cez formulár internetového prehliadača a taktiež aj vo forme zdrojových kódov, je INFO-RNA (INverse-FOLDing of RNA) [17]. Najnovšia verzia 2.0 už umožňuje navyše bližšie určiť sekvenciu, ktorá má danú sekundárnu štruktúru nadobúdať. Napríklad napevno určiť nukleotidy alebo bázické páry vo výslednej sekvencii. Algoritmus sa skladá z dvoch hlavných krokov. Prvý krok zahŕňa návrh dobrej štartovacej sekvencie. V druhom kroku nasleduje vylepšené stochastické lokálne prehľadávanie.

Vstupom algoritmu je cieľová sekundárna štruktúra a sekvencia, ktorá by mala byť z väčšej časti tvorená neurčenými bázami N, ale môže obsahovať aj niektoré konkrétne definované bázy. Počas prvého kroku, je za pomoci dynamického programovania získaná RNA sekvencia s minimálnou energiou akú môže sekvencia

v cieľovej štruktúre nadobudnúť. Avšak nie je garantované, že táto sekvencia nenadobúda inú štruktúru s ešte nižšou voľnou energiou, a teda v skutočnosti by RNA s takouto sekvenciou pravdepodobne nenadobudla nami zvolenú sekundárnu štruktúru. Toto sa snaží riešiť druhý krok algoritmu. Kvalitu sekvencie zlepšuje iteratívne, a to stochastickým lokálnym prehľadávaním, v ktorom minimalizuje štrukturálnu vzdialenosť medzi cieľovou štruktúrou a štruktúrou, v ktorej má vytvorená sekvencia minimálnu voľnú energiu. Susedia sekvencie sú testované buď v náhodnom poradí alebo v poradí podľa rozdielu energií aktuálnej sekvencie a susednej sekvencie pri poskladaní do cieľovej štruktúry. Čím väčší rozdiel, tým skôr je táto susedná sekvencia preskúmaná.

Reálna rýchlosť tohto algoritmu je však ešte horšia ako algoritmu skladania, ktorý táto aplikácia taktiež využíva. Rýchlosť nezáleží len na dĺžke sekundárnej štruktúry, ale do značnej miery aj na jej zložitosti. Teda či napríklad obsahuje väčšie množstvo multicyklov. Stávalo sa mi dokonca, že pri zložitejších sekundárnych štruktúrach táto aplikácia zhavarovala. Taktiež som explicitne musel obmedziť veľkosť vlásenky na 3 až 25 báz, pretože pri väčších veľkostiach aplikácia spotrebovala všetku dostupnú pamäť a zhavarovala. Reálne časy pri štruktúrach do 200 báz sa pohybovali na mojom testovacom stroji od 30 sekúnd až do rádovo niekoľkých minút.

Kapitola 5

Porovnávanie sekundárnych štruktúr

Táto kapitola je o vzájomnom porovnávaní sekundárnych štruktúr. Predstavím najpoužívanejší spôsob a vysvetlím, prečo som sa rozhodol implementovať svoj vlastný postup porovnávania sekundárnych štruktúr.

Sekundárnu štruktúru RNA je možné zapísať aj vo forme stromu. Uzly tvoria jednotlivé základné elementy sekundárnych štruktúr. Koreňom je ten element, ktorý začína na prvej báze sekvencie. Vlásoka nemá nikdy potomka, multicyklus má vždy viac než jedného a zvyšné elementy majú práve jedného potomka. Takto vybudované stromy jednotlivých sekundárnych štruktúr je možné porovnávať. Tu je možné zvoliť niekoľko-úrovňové porovnávanie. Na vyššej úrovni sa určuje, či si odpovedajú samotné typy štruktúrnych elementov. Popríklad je možné povoliť operácie mazania, respektíve pridávania celých elementov. Na nižšej úrovni je možné určovať vzdialenosť stromov aj podľa odlišnosti odpovedajúcich si dvojíc elementov. V takomto prípade môžeme brať do úvahy rôzne faktory podobnosti, ako napríklad počet báz v elemente alebo konkrétne typy bázičkých párov. Na určenie vzdialenosti dvoch reťazcov je vhodné použiť metódu zarovnávanie reťazcov. Porovnávané reťazce môžu byť tvorené bázami (napríklad v prípade porovnávanie nespárovaných báz v multicykle) alebo bázičkými párami (v kmienku). Takéto porovnávanie stromov má však zmysel jedine v prípade, že nepripúšťame pseudouzly. Ich prítomnosť zabráňuje konštrukcii stromu zo sekundárnej štruktúry, pretože spôsobuje cykly v grafe štruktúrnych elementov.

V súčasnosti najpoužívanejším spôsobom je porovnávanie zarovnávaním sekundárnych štruktúr zadaných v notáciách popísaných v časti 2.1. Vstupom sú typicky dve sekundárne štruktúry vo vienna notácii a výsledkom je číslo, ktoré udáva ich odlišnosť. Takéto porovnávanie poskytuje napríklad Vienna RNA Package funkciou `RNAdistance` [22]. U tohto porovnávanie nie je možné žiadnym spôsobom určiť, ktorá časť sekundárnej štruktúry je pre nás dôležitejšia, napríklad, ak sa jedná o veľmi konzervovaný región. A teda, aby v prípade, že si v tejto časti štruktúry neodpovedajú, bolo výsledné číslo porovnania väčšie ako v prípade, že si neodpovedajú menej dôležité časti.

Navrhol som teda svoj vlastný spôsob porovnávania, v ktorom sa presne definuje priorita u každého jedného elementu, penalizácie za nedodržanie definovaných podmienok a taktiež hranica, po ktorú je ešte element považovaný za podobný. Po prekročení tejto hranice je sekundárna štruktúra algoritmom odmietnutá a nezahrnutá medzi nájdené podobné sekundárne štruktúry. Výškou penalizácie je možné určiť, ktorý element je dôležitejší ako iný, teda u dôležitejšieho elementu sa nastaví vyššia penalizácia za odlišnosti od vzoru. Typy elementov si však musia odpovedať, teda ak je napríklad vo vzore sekundárnej štruktúry po prvom kmienku vnútorný cyklus, tak aj všetky jej podobné sekundárne štruktúry musia mať po prvom kmienku vnútorný cyklus. Taktiež počet vetiev u multicyklu si musí odpovedať. Priority sa líšia v závislosti od typu elementu a určujú, čo je dôležité u daného elementu. Určujú, ktoré odlišnosti majú byť pri porovnávaní odpovedajúcich si dvojíc elementov brané do úvahy. U elementov sú povolené nasledujúce typy priorít:

- NO, „No“ – žiadna priorita, musia si odpovedať len typy elementov, túto prioritu je možné nastaviť u všetkých elementov. Iba u tejto priority sa neberie u výdute do úvahy strana na ktorej sa v štruktúre nachádza. Ľavou stranou som označil stav, keď je zo začiatku sekvencie, pravou keď bližšie ku koncu sekvencie. Príklad s výduťou na ľavej strane je (((...(((...)))))) (prvé tri bodky od začiatku sekvencie sú výduť).
- PAIR, „Pair“, „Bases“ – prioritou sú typy bázičných párov/báz, porovnávať sa bude zarovnávaním reťazcov, aby sa zabránilo veľkej chybe v prípade vypustenia jedného páru/bázy. Túto prioritu nie je možné nastaviť v prípade multicyklu. U elementov iných než kmienok sa ale jedná o typy báz, pretože tam bázičné páry nie sú. U opačného poradia báz v páre sa jedná o rozličné páry, teda A-U a U-A nie sú považované za rovnaké.
 - S touto prioritou súvisí penalizácia za každú odlišnú bázu, poprípade bázičný pár, ktorá sa udáva v celých kladných číslach. Výsledok porovnania elementov s touto prioritou bude počet nezrovnalostí zarovnania krát výška penalizácie.
- SIZE, „Size“ – prioritou je celková veľkosť elementu. U kmienku sa jedná o počet bázičných párov a u multicyklu sa počítajú nespárované

bázy medzi jednotlivými vetvami. Je ju možné nastaviť u všetkých typov elementov.

- S touto prioritou súvisí penalizácia za každú chýbajúcu alebo prevyšujúcu bázu, pár, oproti definovanej veľkosti elementu. Výsledok porovnania elementov s prioritou SIZE je absolútna hodnota rozdielu ich veľkostí krát penalizácia.
- Je možné určiť aj maximálnu absolútnu hodnotu rozdielu veľkosti elementov, po ktorej prekročení už nie je sekundárna štruktúra označená za podobnú.
- MATCH, „Match“ – prioritou je presná zhoda. Berú sa do úvahy všetky parametre porovnávania. U tejto priority je nutné aby si elementy odpovedali úplne, teda výsledok ich porovnania bol nula. V opačnom prípade nie je štruktúra označená ako podobná.
- ALL, „All“ – podobne ako u priority MATCH sú zohľadňované všetky parametre porovnávania. Narozdiel od MATCH ale nie je vyžadované aby výsledkom porovnania bola nula.
- UNPBASES, "Unpaired bases" – túto prioritu je možné nastaviť iba u multicyklu. Ide o typy nespárovaných báz v regiónoch medzi jednotlivými vetvami. Každý región je porovnaný samostatne metódou zarovňovania.
 - S touto prioritou súvisí penalizácia za každú odlišnú bázu, podobne ako pri penalizácii pri prioritě PAIR.
- STARTING_BP, „Starting pairs“ – je prístupná iba u multicyklov a zohľadňuje prvé báziké páry jednotlivých vetiev.
 - Aj u tejto priority sa určuje penalizácia za každý odlišný pár.

Dvojitě pomenovanie priorít v podobe názvu veľkými písmenami a názvu v úvodzovkách je spôsobené viacerými spôsobmi zadávania. V prípade definovania vzoru v textovej forme sa používajú názvy uvedené veľkými písmenami. Pri navrhovaní vzoru priamo v aplikácii sa používajú názvy v úvodzovkách. Bližší popis zadávania v aplikácii je uvedený v kapitole s užívateľskou dokumentáciou.

Takýto spôsob definovania konkrétnych elementov umožňuje ďaleko presnejšie vyhľadávanie podobných sekundárnych štruktúr. Návrh je taktiež možné na programátorskej úrovni relatívne jednoducho obohatiť o ďalšie parametre

porovnávania, prípadne upraviť stávajúce priority podľa skutočných potrieb genetikov. Je totiž pravdepodobné, že mnou intuitívne navrhnuté priority a parametre nepokrývajú všetky ich potreby. Implementácia porovnávania je popísaná v kapitole 8 pojednávajúcej o technickej stránke mojej aplikácie.

Kapitola 6

RNA databázy

Najväčším a zároveň najdostupnejším zdrojom DNA a RNA sekvencií sú databázy výskumných organizácií, ktoré sú voľne prístupné na Internete. Najväčšou je zoskupenie pod názvom International Nucleotide Sequence Database Collaboration, pod ktoré spadá GenBank od NCBI (National Center for Biotechnology Information), EMBL (European Molecular Biology Laboratory) a DDBJ (DNA Data Bank of Japan) [18]. Ich databázy je možné prehľadávať pomocou webových stránok, špeciálnych programov a taktiež je možné stiahnuť si celé časti databázy cez FTP. Z týchto databáz je nielen možné čerpať, ale taktiež aj prispievať do nich vlastnými sekvenciami. To ale spôsobuje, že obsiahnuté dáta majú rôznu kvalitu v závislosti na konkrétnom prispievateľovi. Ja som v aplikácii využil voľne dostupné nástroje na získavanie sekvencií práve z databázy GenBank.

Kapitola 7

Súvisiace implementácie

V oblasti bioninformatiky sú najrozšírenejšími programovacími jazykmi C/C++, pre svoju rýchlosť a Perl pre svoje schopnosti spracovávania textu. V jazyku C patrí k najpoužívanejším súbor knižníc pod názvom Vienna RNA Package – RNA Secondary Structure Prediction and Comparison (prekl. Vienna RNA Balíček – RNA Predikcia a porovnávanie sekundárnych štruktúr RNA). Najväčším projektom z tejto oblasti v jazyku Perl je BioPerl projekt, ktorý obsahuje rozsiahlu sadu nástrojov z oblasti bioinformatiky. Obidva tieto súbory programov využíva môj projekt, preto si ich v nasledujúcich dvoch podkapitolách stručne popíšeme.

7.1 Vienna RNA Package

Vienna RNA Package je súbor knižníc zdrojových kódov napísaných v jazyku C pre predikciu a porovnávanie sekundárnych štruktúr RNA [19]. Najpoužívanejšou funkciou z balíčka je funkcia na predikciu sekundárnej štruktúry RNA zo sekvencie pomocou minimalizovania voľnej energie štruktúry. Balíček poskytuje tri rôzne algoritmy dynamického programovania na predikciu štruktúry:

- algoritmus minimálnej energie, ktorý určí jedinu optimálnu štruktúru,
- algoritmus využívajúci funkciu rozdelenia (partition function), ktorého výstupom sú pravdepodobnosti bázičných párov,
- suboptimálny algoritmus skladania, ktorý vracia na výstupe všetky suboptimálne štruktúry, ktorých energia spadá do určeného intervalu energie.

Pre porovnávanie sekundárnych štruktúr poskytuje tento balíček rôzne spôsoby určovania vzdialeností (odlišností) sekundárnych štruktúr, či už na základe zarovnávanie reťazcov alebo pomocou editačnej vzdialenosti stromov. Do Vienna RNA Package patrí taktiež algoritmus na navrhovanie sekvencií s preddefinovanou štruktúrou. Celý balíček je distribuovaný ako voľný software vo forme zdrojových kódov, ktoré je následne možné bez väčších problémov preložiť na rôznych operačných systémoch.

Nasleduje prehľad niekoľkých základných funkcií:

- RNAfold – predikuje sekundárnu štruktúru s minimálnou energiou a pravdepodobnosti bázičických párov,
- RNAeval – vypočítava voľnú energiu sekundárnej štruktúry RNA,
- RNAinverse – inverzné skladanie sekvencie s preddefinovanou štruktúrou,
- RNAdistance – porovnáva sekundárne štruktúry,
- RNAplot – vykresľovanie RNA štruktúr.

Ja som vo svojom projekte využil RNAfold a RNAplot.

7.2 BioPerl

BioPerl sú nástroje vo forme modulov určených na riešenie bioinformatických problémov v skriptovacom jazyku Perl [20,21]. Využíva objektovo orientovaný návrh, takže množstvo modulov navzájom na sebe závisí a spolupracuje pri dosiahnutí výsledku. Inštalácia je trochu zložitejšia a je nutné mať už nainštalovaný Perl. Operačné systémy na báze Unix/Linux Perl obsahujú typicky v základnej inštalácii, zatiaľ čo na iných operačných systémoch je nutné si ho doinštalovať.

Nižšie je uvedený príklad skriptu, ktorý stiahne sekvencie z GenBank databázy a uloží ich do xml súboru s názvom tmpFileRes.xml. Skript sa volá s jediným parametrom a tým je dotaz do databázy – napríklad „Arabidopsis[ORGN] AND topoisomerase[TITL] AND 0:3000[SLEN]“, ktorý má za úlohu stiahnuť všetky RNA organizmu s názvom Arabidopsis, ktoré majú v popise slovo „topoisomerase“ a ich dĺžka je v rozmedzí 0 až 3000 báz.

```
use Bio::Seq;
use Bio::SeqIO;
use Bio::DB::GenBank;
use Bio::DB::Query::GenBank;
# otvorí sa súbor tmpFileRes.xml na zápis
open (strFile, ">tmpFileRes.xml");

# načíta dotaz do databázy z prvého parametra skriptu
my $query = $ARGV[0];

# hľadáme v GenBank sekvencie nukleotidov
```

```

my $query_obj = Bio::DB::Query::GenBank->new(-db      =>
'nucleotide', -query => $query );

# pripravíme objekt s odpoveďou
my $gb_obj = Bio::DB::GenBank->new;

# otvoríme stream s odpoveďou
my $stream_obj = $gb_obj-
>get_Stream_by_query($query_obj);

# zapíšu sa výsledky do XML súboru
print strFile "<?xml version=\"1.0\"?>\n";
print strFile "<RNAs
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"RNAStructures.xsd\">\n";

while ($seq_obj = $stream_obj->next_seq) {
    print strFile "\t<structure>\n";
    print strFile "\t\t<name>", $seq_obj->display_id,
"</name>\n";
    print strFile "\t\t<sequence>", $seq_obj->seq(),
"</sequence>\n";
    print strFile "\t\t<vienna>N/A</vienna>\n";
    print strFile "\t</structure>\n";
}

print strFile "</RNAs>\n";
# zatvorenie XML súboru a ukončenie skriptu
close(strFile);

```

Kapitola 8

Moja aplikácia

Táto kapitola približuje technickú stránku aplikácie. V prvej podkapitole je popísaný celý algoritmus, ktorý som implementoval. Druhá obsahuje dátový model. V tretej je popísaný návrh grafického rozhrania. Štvrtá podkapitola obsahuje použité funkcie, ktoré som neimplementoval sám, ale využil som voľne dostupné implementácie od tretích strán.

Aplikácia rieši problém vyhľadávania podobných sekundárnych štruktúr RNA. Na vstupe je sekundárna štruktúra RNA, ktorá bude slúžiť ako vzor a množina RNA, v ktorých sa bude tento vzor hľadať. Výstupom majú byť regióny prehľadávaných RNA, ktoré nadobúdajú sekundárnu štruktúru podobnú so vzorovou a číselne vyjadrená odlišnosť od vzorovej.

Jednou z požiadaviek bola cieľová platforma, a to operačný systém Windows. Je to dosť netradičné vzhľadom k značnej dominancii unixových operačných systémov v oblasti bioinformatiky. Z dôvodu čo najväčšej použiteľnosti som uvažoval o programovacom jazyku Java, ktorý je do veľkej miery prenositeľný. Od tohto zámeru som však upustil, keďže v tomto jazyku nie sú napísané najdôležitejšie utility, ktoré som plánoval vo svojej aplikácii využiť. Aj keď Java umožňuje využiť iné programovacie jazyky, stratila by sa výhoda prenositeľnosti. Práve kvôli rozsiahlemu množstvu rôznych pomocných programov som sa rozhodol pre programovací jazyk C/C++. V neskoršej fáze vývoja som ešte využil jeden nástroj, ktorý však vyžaduje programovací jazyk Perl. Užívateľské rozhranie je implementované pomocou štandardného WIN32 API.

8.1 Algoritmus

Algoritmus, ktorý som implementoval sa skladá z týchto hlavných krokov:

- 1) Definovanie vstupného vzoru pomocou parametrov rozpoznávaných aplikáciou, ako sú priority a ďalšie atribúty porovnávania.
- 2) Vybudovanie stromu odpovedajúceho sekundárnej štruktúre vzoru.
- 3) Určenie pravdepodobnej primárnej štruktúry vzoru, a to buď priamo z návrhu, alebo ak obsahuje neurčené bázy "N" tak vypočítanie

predpokladanej sekvencie s ohľadom na už definované bázy pomocou funkcie z INFO-RNA.

- 4) Načítanie RNA s prípadnou sekundárnou štruktúrou vo vienna formáte.
- 5) Pre RNA, kratšie ako zvolená konštanta 2000, poskladanie celej sekundárnej štruktúry pomocou Vienna RNA Package. Toto obmedzenie na dĺžku sekvencie RNA je z dôvodu časovej náročnosti skladania rozsiahlych sekundárných štruktúr.
- 6) U dlhších RNA sa určia kandidátske regióny pomocou zarovňovania s predpokladanou primárnou štruktúrou vzoru. Týmto regiónom je následne poskladaná sekundárna štruktúra. Dĺžku hľadaných regiónov som zvolil ako predpokladanú dĺžku vzoru, ktorú som ešte predĺžil v závislosti od dĺžky vzoru. Množstvo porovnaní som znížil tak, že som neporovnával s každou podsekvenciou, ale len s podsekvenciami v pravidelných rozostupoch, závislých na veľkosti vzoru. Teda napríklad u vzoru veľkom 100 báz a RNA o 10000 bázach som zvolil veľkosť regiónu na 110 báz (110% predpokladanej dĺžky). Sekvenciu vzoru som potom porovnal s podsekvenciou 1-110, ďalšou porovnávanou ale nie je 2-111, pretože u nej bude výsledok zarovňania prakticky rovnaký ako u prvej. Rozostup by u sekvencie o 100 bázach bol 10 báz (10% z predpokladanej dĺžky), a teda druhá porovnávaná podsekvencia je 11-121, čím sa prakticky zníži celkový počet zarovnaní z 9891 na necelých 1000. Týmto postupom je ale možné, že niektoré štruktúry, ktoré by správne mali byť označené ako podobné, nebudú objavené. Napríklad tie, ktoré sú dlhšie o viac než 10% oproti predpokladanej sekvencii vzoru. Zvolené konštanty ale zaručujú, že všetky podsekvencie o rovnakej a menšej dĺžke ako predpokladaná sekvencia vzoru, budú zarovnávané (ako súčasť predĺženej podsekvencie) s hľadanou sekvenciou. Aby bol región označený ako kandidátsky, nesmie výsledok zarovňania prekročiť 25% dĺžky predpokladanej primárnej štruktúry vzoru.
- 7) RNA a regióny s určenými sekundárnymi štruktúrami sa prevedú do stromovej reprezentácie.
- 8) V stromoch z prehľadávanej množiny sú vyhladané všetky podstromy obsahujúce hľadaný vzor nasledujúcim spôsobom:

- Strom sa najprv prehľadá a určia sa vhodné počiatočné elementy vzoru.
 - Každý počiatočný element sa porovná s počiatočným elementom vzoru a, ak vyhovuje, tak sa pokračuje porovnávaním jeho potomkov s potomkami vzoru. Výsledok porovnania je súčet výsledkov porovnaní jednotlivých dvojíc elementov.
 - Ak sa narazí na prvý neodpovedajúci si element, či už samotným typom alebo parametrami, tak vyhľadávanie pokračuje ďalším vhodným počiatočným elementom.
- 9) Vyhovujúcim sekundárnym štruktúram RNA sa vypočítajú súradnice $[x,y]$ každej bázy, potrebné pre vykreslenie sekundárnej štruktúry. To sa vykoná pomocou funkcie `RNAplot`.

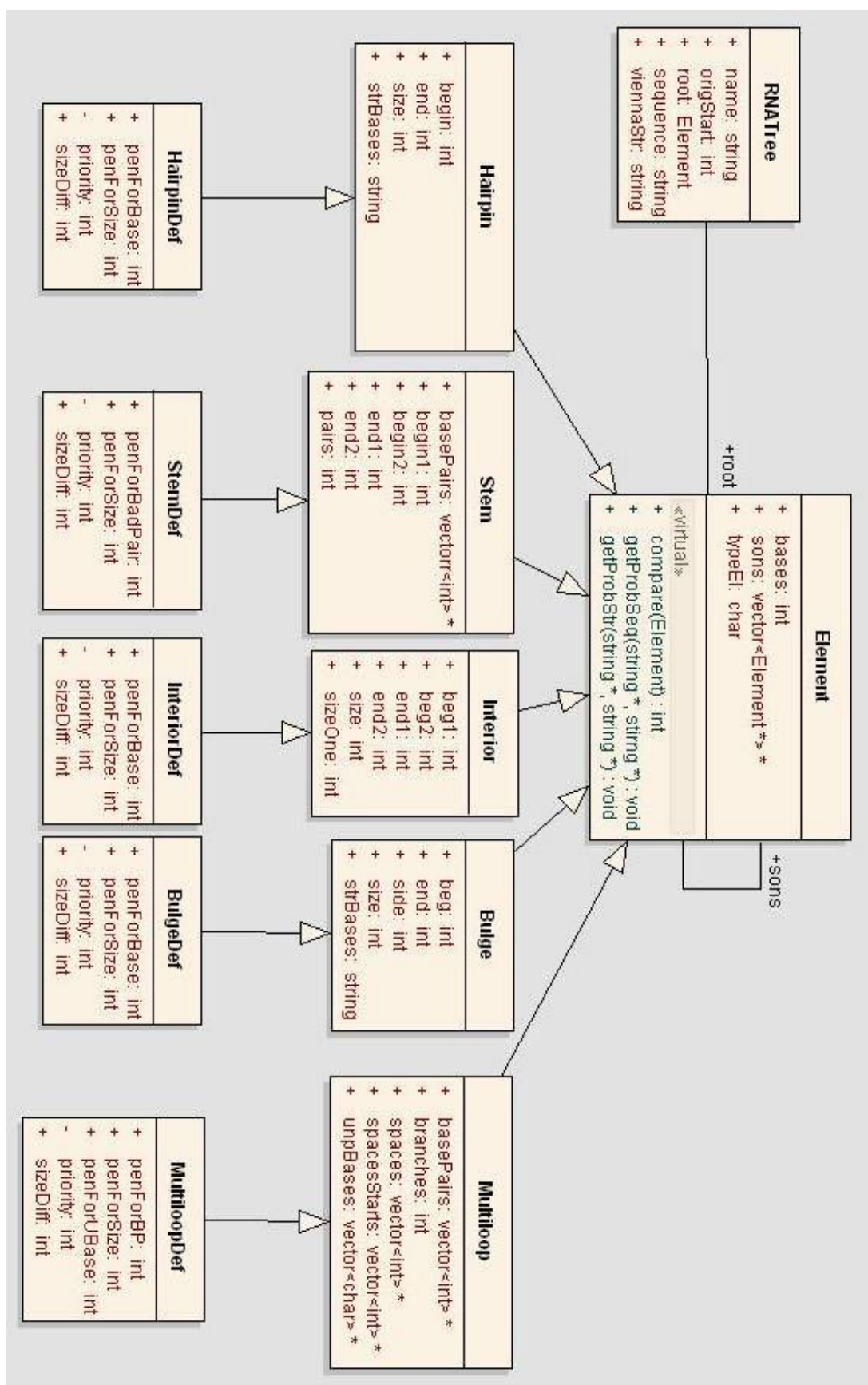
Konštanty, ktoré som použil v uvedenom algoritme sú obsiahnuté v triede `Constants`, v hlavičkovom súbore `Utils.h`.

8.2 Dátový model

Zvolený programovací jazyk C++, je objektový, a preto som sa snažil využiť pri programovaní niektoré praktiky objektového programovania. V nasledujúcich podkapitolách popíšem niektoré najhlavnejšie objekty.

8.2.1 RNA

Keďže som sa rozhodol porovnávať sekundárne štruktúry ako stromy, navrhol som nasledujúci dátový model.



Obr. 9 Dátový model aplikácie

RNA je reprezentovaná ako trieda `RNATree`, ktorá obsahuje odkaz na koreň jej sekundárnej štruktúry. Triedy s príponou „Def“ reprezentujú elementy vzorovej sekundárnej štruktúry RNA. Je u nich možné určiť miery odlišnosti, ako aj priority pri porovnávaní. V prípade vlastného spôsobu porovnávania štruktúr stačí prepísať odpovedajúce metódy `compare()` na jednotlivých elementoch, alebo vydediť ďalších potomkov od týchto tried. Implementácia dátového modelu sa nachádza v súbore `elements.h`.

8.2.2 Logovanie

Logovanie je automatické zaznamenávanie akcií, ktoré vykonáva program. V prípade mojej aplikácie logovanie prebieha do súboru s názvom `log.txt` umiestneného v koreňovom adresári aplikácie. Ak nastanú problémy v aplikácii, tak jednou z možností lokalizovania problému je spätná analýza výstupu logovania. Taktiež v prípade, že aplikácia prestane reagovať je vhodné skontrolovať logovací súbor. Ak narastá, tak aplikácia nereaguje pretože vykonáva náročné výpočty a je nutné počkať na dokončenie týchto výpočtov.

Logovanie zabezpečuje trieda `Logger`, ktorá poskytuje statickú metódu `getLogger()`. V prípade logovacej triedy bol využitý návrhový vzor `Singleton`, vďaka ktorému je v aplikácii prítomná práve jedna inštancia loggeru. To je výhodné z dôvodu jednej inicializácie, ktorej súčasťou je prístup k logovaciemu súboru a taktiež jednotné nastavenie logovacej úrovne v rámci celej aplikácie. Nastaviť logovaciu úroveň (`level`), je doposiaľ možné len úpravou zdrojového kódu. Ten sa nachádza v súbore `logger.h` a `logger.cpp`. Finálna verzia, určená pre normálne použitie má nastavenú úroveň logovania tak, aby zaznamenávala iba kritické chyby.

8.3 Grafické rozhranie

Základ tvorí jedno okno s menu, do ktorého v prípade nájdenia podobných štruktúr budú tieto vykreslené a potom množstvo ďalších okien pre ostatné operácie. Na získanie súradníc, kam sa majú umiestniť jednotlivé bázy, je využitá časť

z Vienna RNA Package. Na takto získaných súradniciach je následne možné vykonávať transformácie, ktorých výsledkom je posun obrázku, ako aj zväčšenie, resp. zmenšenie. Tieto operácie sú umiestnené v súbore `Drawings.cpp`, ktorý následne využívajú okná, kde sa vykresľujú sekundárne štruktúry. Hlavné okno má zdrojové kódy v súbore `RNAwin2.cpp`. Ostatné okná sú umiestnené v adresári `Dialogs`. Na validácie užívateľom zadaných vstupov som vytvoril triedu `FormatValidator` umiestnenú v súbore `Utils.h`. Odstránil som tým opakovanie kódu validácií a zjednodušil opravy.

8.4 Funkcie tretích strán

Dôležitou súčasťou aplikácie sú využité funkcie od tretích strán. Prednostne som sa snažil využívať implementácie v jazyku C++, poprípade C, keďže tieto som mohol pripojiť priamo do svojej aplikácie. Jedinou výnimkou je utilita zodpovedná za sťahovanie sekvencií z databázy GenBank. Tá je v jazyku Perl a vyžaduje nainštalovaný projekt BioPerl a aktívne pripojenie do Internetu. K jej využitiu ma viedla predovšetkým jej vysoká pridaná hodnota v podobe možnosti vyhľadávania a sťahovania RNA sekvencií z najväčšej voľne dostupnej databázy RNA.

Celkovo som využil tri rôzne implementácie v jazykoch C/C++ :

- Vienna RNA Package a konkrétne funkcie :
 - RNAfold pre skladanie sekundárnych štruktúr,
 - RNAplot pre určovanie súradníc báz v štruktúre.
- INFO-RNA na predikciu primárnej štruktúry k sekundárnej.
- XML Parser pre parsovanie XML súborov.

Pre prvé dve som vytvoril špeciálne triedy takzvané obalovače (wrapper). Vytvára to akúsi medzi-vrstvu medzi volaniami funkcií a umožňuje to napríklad jednoduchší prechod na novšie verzie funkcií tretích strán. Využil som to napríklad u novej verzie INFO-RNA.

Kapitola 9

Užívateľská dokumentácia

Táto kapitola obsahuje užívateľskú dokumentáciu k môjmu projektu. V prvej podkapitole je návod na inštaláciu. Nasleduje užívateľské rozhranie. Tretia podkapitola popisuje možnosti načítania sekvencií do aplikácie. V štvrtej podkapitole je rozobraný návrh vzoru. Posledná podkapitola obsahuje štandardný scenár použitia.

Aplikácia RNAwin je určená na hľadanie podobných sekundárnych štruktúr RNA. Vstupom je vzor, ktorý hľadáme a množina RNA, v ktorej tento vzor budeme hľadať. Súčasťou aplikácie je dizajnér, v ktorom je možné detailne špecifikovať požiadavky na jednotlivé elementy. Pridávať RNA do prehľadávanej množiny je možné načítaním zo súboru, ručným zadaním do formulára alebo priamym stiahnutím z databázy GenBank. Výsledné nájdené sekundárne štruktúry sú vykreslené spolu s číselným vyjadrením odlišnosti od vzorovej sekundárnej štruktúry. Výsledky je možné si uložiť vo forme XML súboru. Proces vyhľadávania môže trvať aj dlhšiu dobu v závislosti na veľkosti a počte vstupov. Počas neho nebude dostupná väčšina funkčnosti aplikácie a je nutné počkať na dokončenie výpočtu. Priebežný stav výpočtu sa bude zobrazovať v titulku hlavného okna.

9.1. Inštalácia

Aplikáciu nie je nutné inštalovať, stačí len skopírovať hlavný adresár s aplikáciou tam, kde ho chceme mať. Je nutné mať ale nainštalovaný Perl, spolu s knižnicami projektu BioPerl. Podrobný postup ich inštalácie je dostupný na [21]. Nie je ho nutné inštalovať celý, postačí verzia označená ako core.

9.2. Užívateľské rozhranie

Užívateľské rozhranie je štandardné okienkové s hlavnými funkciami v menu. Akonáhle už sú získané nejaké výsledky, je možné medzi nimi navigovať pomocou klávesnice, poprípade si ich nechať uložiť do súboru vo formáte XML.



Obr. 10. Hlavné okno aplikácie s menu.

Kompletný popis menu:

- File
 - Exit – ukončenie aplikácie.
- RNA
 - Find structures – spustí vyhľadávanie podobných sekundárnych štruktúr.
 - Show results – v novom okne vypíše výsledky.
 - Save results – uloženie výsledkov do súboru.
- Pattern
 - Load from file – načítanie vzoru zo súboru.
 - Design own pattern – navrhovanie vlastného vzoru v novom okne.
 - View pattern – zobrazenie aktuálneho vzoru.
 - Save pattern – uloženie vzoru do XML súboru.
- Structures
 - Load from file – načítanie RNA zo súboru.
 - Load online sequence – načítanie RNA z online databázy GenBank.
 - Add RNA – ručné pridanie RNA v novom okne.
 - Show RNAs – zobrazenie aktuálnych RNA.
- Help
 - About ... – okno s popisom ovládania pomocou klávesnice.

9.3. Načítanie sekvencií

Sekvencie RNA je možné dostať do aplikácie tromi spôsobmi. Načítaním zo súboru, stiahnutím z online databázy GenBank alebo ručným zadáním do formulára v aplikácii. Jednotlivé možnosti si teraz rozoberieme v samostatných podkapitolách.

9.3.1 Načítanie zo súboru

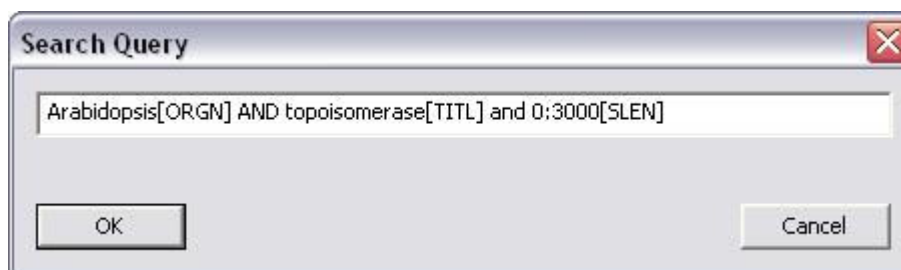
Najrýchlejšia možnosť je načítať sekvencie z dopredu pripraveného XML súboru, ktorého formát je presne daný. Základ tvorí tag <RNAs>, ktorý obsahuje popis schématu. Do tohto tagu sa vkladajú tagy <structure>, ktoré obsahujú samotné RNA. V nich sa v samostatných tagoch určuje názov RNA (tag <name>), sekvencia báz (tag <sequence>) a sekundárna štruktúra zapísaná vo vienna formáte (tag <vienna>). Názov a sekvencia musia byť povinne vyplnené. Sekundárna štruktúra, ak je uvedená, tak jej dĺžka musí byť zhodná s dĺžkou sekvencie. V prípade, že nepoznáme sekundárnu štruktúru, tak je nutné ju označiť ako "N/A" a aplikácia si ju následne sama dopočíta. Nasledujúci príklad je obsah súboru s dvomi RNA:

```
<RNAs xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="RNAStructures.xsd">
  <structure>
    <name>Name of first RNA</name>
    <sequence>UAGUCGCGCUA</sequence>
    <vienna>(((.....)))</vienna>
  </structure>
  <structure>
    <name>Sample RNA two</name>
    <sequence>UAAAAGUAAAGG</sequence>
    <vienna>N/A</vienna>
  </structure>
</RNAs>
```

9.3.2 Vyhľadávanie v databáze GenBank

Druhou možnosťou získania RNA je ich priamo z aplikácie vyhľadať v databáze GenBank. Je ale nutné definovať dotaz na základe ktorého budú následne vyhľadané. Základné pravidlá syntaxe dotazu sú:

- Sú povolené logické spojky: AND (konjunkcia), OR (disjunkcia), BUTNOT (vo význame „ale nie“, napríklad dotaz typu: A AND B BUTNOT C vráti záznamy splňujúce A aj B ale nesplňujúce C). Zátvorky podľa potreby.
- Element má formu text[POLE].
- Hviezdička (*) je povolená napríklad na konci textu ako zástupný znak, pre všetky možné zakončenia textu.
- POLE môže byť :
 - [TITL] – vyhľadá v popisoch,
 - [AUTH] – autor záznamu,
 - [ORGN] – meno organizmu,
 - [PROP] – vlastnosti,
 - [SLEN] – dĺžka sekvencie.

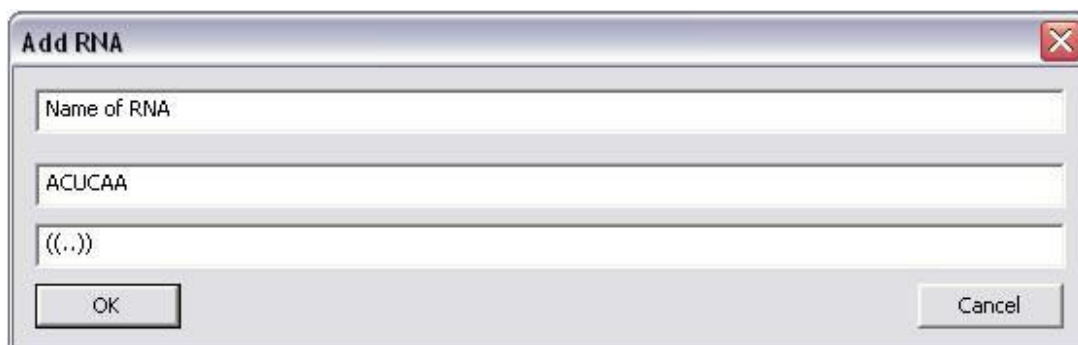


Obr. 11 : Príklad vyplneného okna pre prehľadanie databázy GenBank. Dotaz bude hľadať RNA organizmu Arabidopsis, ktoré majú v popise slovo “topoisomerase” a sú dlhé od 0 do 3000 báz.

Samotné stiahnutie je pomocou externého skriptu, takže sa dočasne zobrazí čierna obrazovka. Pre správne vyhľadanie je nutné mať aktívne pripojenie do Internetu a úspešne nainštalovaný Perl spolu s balíčkami BioPerlu.

9.3.3 Ručné zadanie RNA

Posledná možnosť je ručne pripísať sekvenciu pomocou formulára. Názov a primárna štruktúra sú povinné. Ak nepoznáme sekundárnu štruktúru tak je nutné uviesť do políčka pre sekundárnu štruktúru „N/A“.



Obr. 12 : Formulár na pridanie vlastnej RNA.

9.4. Návrh vzoru

Vzor je možné zadať dvomi spôsobmi. Vo forme XML súboru, ktorého štruktúra je presne daná a v dizajneri, ktorý je súčasťou aplikácie. V nasledujúcich podkapitolách podrobne popíšem obidve možnosti.

9.4.1 Načítanie zo súboru

Súbor so vzorom je vo formáte XML. Kompletný vzor je zaobalený do tagu <RNAPattern>, ktorý obsahuje podelementy pre názov: <name>, primárnu štruktúru <sequence>, sekundárnu štruktúru vo vienna formáte <structure-vienna> a pre sekundárnu štruktúru zadanú vo forme stromu <structure-tree>. Ak sekvenciu alebo sekundárnu štruktúru vo vienna formáte nepoznáme, je nutné ju uviesť ako „N/A“. Z informácií uvedených v elemente <structure-tree> sa potom tieto dopočítajú. Ak ich ale vyplníme, tak musia presne odpovedať údajom uvedeným v elemente <structure-tree>. Do sekvencie je možné zadať namiesto neurčených báz „N“. Elementu <structure-tree> obsahuje atribút externals, v ktorom sa zadáva počet nespárovaných báz na koncoch sekvencie. Veľkosť vzoru nesmie presiahnuť 250 báz, musí začínať kmienkom a všetky vetvy musia byť zakončené vlásenkou.

Pr.:

<RNAPattern>

<name>Silene vulgaris CE33</name>

```

<sequence>UAAAAGUAAAGGGGCAAUAAAAAUUUCUUGUUCG
AUCAAGAAAUUAGUAUUGCCCCUUUACUAGUU</sequence>
<structure-
vienna>....((((((((((((((((...((((((((((.....))))))))))..))))))))))))....</structure-vienna>
<structure-tree externals="8">
.... – bude popísané ďalej
</structure-tree>
</RNAPattern>

```

Element <structure-tree> obsahuje podelement <sub-element>, do ktorého sa vkladá konkrétny element sekundárnej štruktúry. Aplikácia rozlišuje 5 základných elementov sekundárnej štruktúry: <stem> (kmienok), <interior> (vnútorný cyklus), <bulge> (výdut'), <hairpin> (vlásenka), <multiloop> (multicyklus). Pr.

```

<structure-tree externals="8">
  <sub-element>
  ...
  </sub-element>
</structure-tree>

```

Elementy sekundárnej štruktúry obsahujú nasledujúce atribúty:

- priority = obsahuje typ priority porovnávania pre daný element, môže nadobúdať nasledovné hodnoty:
 - „NO“ – žiadna priorita, musia si odpovedať len typy elementov, túto prioritu je možné nastaviť u všetkých elementov. Iba u tejto priority sa neberie u výdute do úvahy strana na ktorej sa v štruktúre nachádza. Ľavou stranou som označil stav, keď je zo začiatku sekvencie, pravou keď bližšie ku koncu sekvencie. Príklad s výdut'ou na ľavej strane je (((...(((...)))))) (prvé tri bodky od začiatku sekvencie sú výdut').
 - „PAIR“ prioritou sú typy bázičných párov/báz, porovnávať sa bude zarovnávaním reťazcov, aby sa zabránilo veľkej chybe v prípade vypustenia jedného páru/bázy. Túto prioritu nie je možné nastaviť v prípade multicyklu. U elementov iných než kmienok sa ale jedná o typy báz, pretože tam bázičné páry nie sú. U opačného poradia báz

v páre sa jedná o rozličné páry, teda A-U a U-A nie sú považované za rovnaké.

- „SIZE“ prioritou je celková veľkosť elementu. U kmienku sa jedná o počet bázičických párov a u multicyklu sa počítajú nespárované bázy medzi jednotlivými vetvami. Je ju možné nastaviť u všetkých typov elementov.
- „MATCH“ – prioritou je presná zhoda. Berú sa do úvahy všetky parametre porovnávania. U tejto priority je nutné aby si elementy odpovedali úplne, teda výsledok ich porovnania bol nula. V opačnom prípade nie je štruktúra označená ako podobná.
- „ALL“ – podobne ako u priority MATCH sú zohľadňované všetky parametre porovnávania. Narozdiel od MATCH ale nie je vyžadované aby výsledkom porovnania bola nula.
- „STARTING_BP“ – je prístupná iba u multicyklov a zohľadňuje prvé bázičické páry jednotlivých vetiev.
- „UNPBASES“ – túto prioritu je možné nastaviť iba u multicyklu. Ide o typy nespárovaných báz v regiónoch medzi jednotlivými vetvami. Každý región je porovnaný samostatne metódou zarovňovania.
- sizeDiff – maximálna absolútna hodnota rozdielu veľkostí elementov, po ktorej prekročení už nie je sekundárna štruktúra označená za podobnú. Priamo súvisí s prioritami SIZE, MATCH a ALL. Tento atribút musí obsahovať každý element.
- penForSize – penalizácia za každú chýbajúcu alebo prevyšujúcu bázu, pár, oproti definovanej veľkosti elementu. Výsledok porovnania elementov s prioritou SIZE je absolútna hodnota rozdielu ich veľkostí krát penalizácia. Tento atribút musí obsahovať každý element.
- penForBase – penalizácia za každú odlišnú bázu, poprípade bázičický pár, ktorá sa udáva v celých kladných číslach. Súvisí s prioritou PAIR. Výsledok porovnania elementov s touto prioritou bude počet nezrovnalostí zarovňovania krát výška penalizácie. Tento atribút obsahujú elementy <hairpin>, <bulge> a <interior>
- penForBadPair- penalizácia za každý odlišný typ bázičického páru. Súvisí s prioritou PAIR a nastavuje sa u elementu <stem>.

- penForUBase – súvisí s prioritou UNPBASES a určuje výšku penalizácie za každú odlišnú bázu.
- penForBP – súvisí s prioritou STARTING_BP a určuje výšku penalizácie za každý odlišný pár.

Element <stem> reprezentuje kmienok. Obsahuje podelementy <pairs> a <sub-element>. Element <pairs> obsahuje pre každý bázičný pár element <pair>, v ktorom je vložený konkrétny pár báz. U neurčeného páru je možné zadať „NN“ Pr.

```
<stem priority="PAIR" sizeDiff="2" penForBadPair="2" penForSize="3">
  <pairs>
    <pair>GU</pair>
    <pair>UC</pair>
    <pair>AG</pair>
  </pairs>
  <sub-element>
  ...
  </sub-element>
</stem>
```

Element <interior> reprezentuje vnútorný cyklus. Obsahuje podelementy <bases> a <sub-element>. <bases> obsahuje podelementy <left> a <right>, kde sa špecifikujú bázy na konkrétnej strane vnútorného cyklu. Pr.

```
<interior priority="SIZE" sizeDiff="3" penForBase="1" penForSize="1">
  <bases>
    <left>AAA</left>
    <right>AG</right>
  </bases>
  <sub-element>
  ..
  <sub-element>
</interior>
```

Element <bulge> reprezentuje výduť. Obsahuje podelementy <bases> a <sub-element>. <bases> obsahuje právě jeden podelement, a to buď <right> alebo <left>, kde sa určia bázy. Pr.

```
<bulge priority="SIZE" sizeDiff="3" penForBase="1" penForSize="1">
  <bases>
    <left>AAA</left>
  </bases>
  <sub-element>
    ..
  <sub-element>
</bulge>
```

Element <hairpin> reprezentuje vlásenku. Obsahuje podelement <bases>, v ktorom sú určené konkrétne bázy vlásenky. Pr.

```
<hairpin priority="ALL" sizeDiff="2" penForBase="5" penForSize="3">
  <bases>UUCGAU</bases>
</hairpin>
```

Element <multiloop> reprezentuje rozvetvenie sekundárnej štruktúry, tzv. multicyklus. Obsahuje podelement <branches>, do ktorého sa vkladajú elementy <unpaired> nasledovaný vždy elementom <stem> a na koniec jeden element <unpBases>. Pr.

```
<multiloop priority="ALL" sizeDiff="3" penForUBase="2" penForSize="4"
penForBP="3">
  <branches>
    <unpaired>UAA</unpaired>
    <stem ...
    <unpaired>UAA</unpaired>
    <stem ...
    <unpBases>ACCA</unpBases>
  </branches>
</multiloop>
```

Kompletný príklad súboru so vzorom sekundárnej štruktúry:

```

<?xml version="1.0" encoding="windows-1250"?>
<RNAPattern xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="RNAPattern.xsd">
<name>Silene vulgaris CE33</name>
<sequence>N/A</sequence>
<structure-vienna>N/A</structure-vienna>
<structure-tree externals="8">
  <sub-element>
    <stem priority="PAIR" sizeDiff="2" penForBadPair="2" penForSize="3">
      <pairs>
        <pair>GU</pair>
        <pair>UC</pair>
        <pair>AA</pair>
        <pair>AU</pair>
        <pair>AU</pair>
        <pair>GU</pair>
        <pair>GC</pair>
        <pair>GC</pair>
        <pair>GC</pair>
        <pair>CG</pair>
        <pair>AU</pair>
        <pair>AU</pair>
        <pair>UA</pair>
        <pair>AU</pair>
        <pair>AG</pair>
      </pairs>
    <sub-element>
      <interior priority="SIZE" sizeDiff="3" penForBase="1"
penForSize="1">
        <bases>
          <left>AAA</left>
          <right>AG</right>
        </bases>
      <sub-element>
        <stem priority="SIZE" sizeDiff="4" penForBadPair="4"
penForSize="5">
          <pairs>
            <pair>AU</pair>
            <pair>AU</pair>
            <pair>UA</pair>
            <pair>UA</pair>
            <pair>UA</pair>
            <pair>CG</pair>
            <pair>UA</pair>
            <pair>UA</pair>
            <pair>GC</pair>
          </pairs>
        <sub-element>
          <hairpin priority="NO" sizeDiff="2" penForBase="0"
penForSize="0">

```

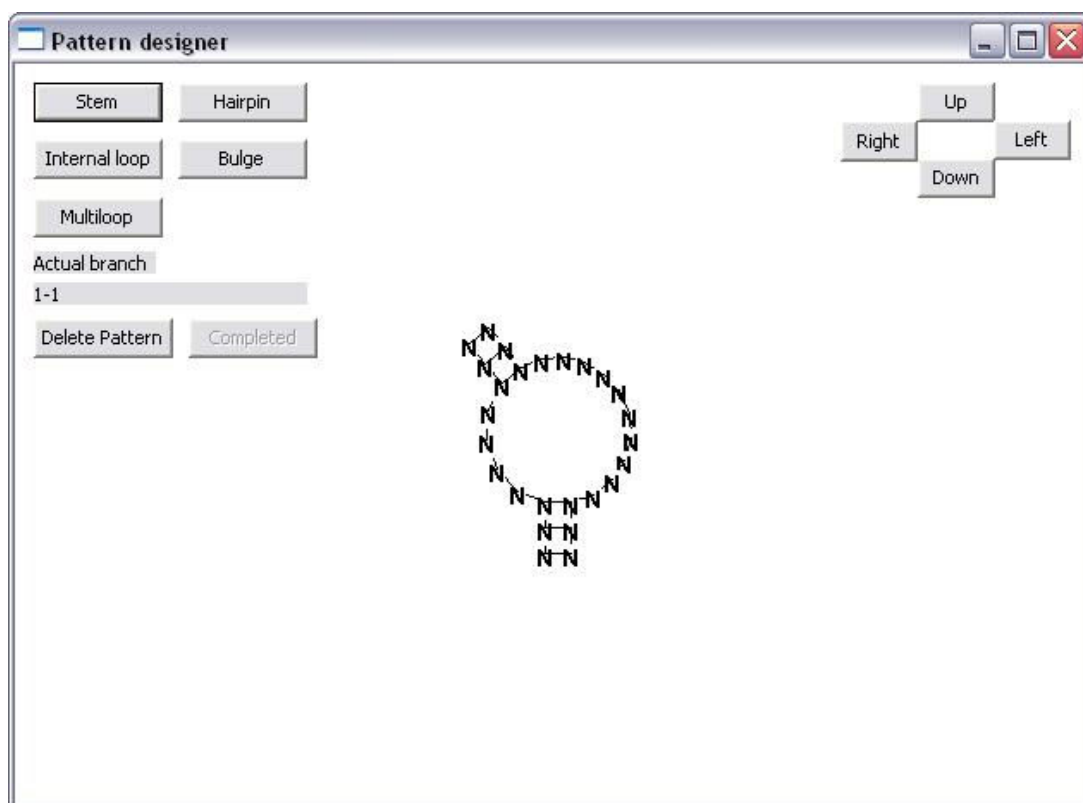
```

        <bases>NNNNNN</bases>
      </hairpin>
    </sub-element>
  </stem>
</sub-element>
</interior>
</sub-element>
</stem>
</sub-element>
</structure-tree>
</RNAPattern>

```

9.4.2 Návrh v dizajnéri

Druhá, pohodlnejšia možnosť, je navrhnúť vzor sekundárnej štruktúry priamo v aplikácii. Zadáva sa vo forme celých elementov, u ktorých je možné špecifikovať rozličné parametre, ktoré sa zohľadnia pri hľadaní. Nespornou výhodou je, že aktuálne navrhovaný vzor je okamžite viditeľný. Ten je možné pomocou tlačidiel posúvať do všetkých smerov. Stredné koliesko myšky približuje a oddiaľuje pohľad. Vzor sa navrhuje postupne po elementoch, pričom aktuálne navrhovaná vetva je určená v textovom políčku “Actual branch”. Navrhnuté elementy nie je možné späťne upravovať. Je ale možné zmazať celý doposiaľ navrhovaný vzor a začať nový, alebo dokončiť návrh a uložiť si ho do súboru, v ktorom prevedieme požadované zmeny. Akonáhle je vzor správne dokončený je možné kliknúť na tlačidlo “Completed”. Týmto sa nový vzor potvrdí a automaticky sa určí predpokladaná primárna štruktúra. Táto bude následne použitá ako heuristika na určenie regiónov, ktoré by mohli tvoriť podobnú sekundárnu štruktúru. Je to nutné z dôvodu optimalizácie u príliš dlhých sekvencií, kde namiesto počítania kompletnej sekundárnej štruktúry sa vyhľadajú len menšie regióny a až týmto bude spočítaná sekundárna štruktúra. Práve z tohto dôvodu je veľkosť vzoru explicitne obmedzená na 250 báz.



Obr. 13 : Okno, kde sa navrhuje vzor sekundárnej štruktúry.

Elementy, ktoré je možné zadať:

- Stem – kmienok,
- Hairpin – vlásenka,
- Internal loop – interný cyklus,
- Bulge – výduť,
- Multiloop – multicyklus.

Actual branch – aktuálne navrhovaná vetva vzoru (pri rozvetveniach). Jednotlivé vetvy u multicyklu sa zadávajú v takom poradí, v akom nasledujú v sekvencii (na obrázku zľava do prava po obvodě multicyklu tvorenom nespárovanými bázami). Ak vzor neobsahuje žiaden multicyklus tak má tento parameter stále hodnotu 1. Akonáhle pridáme prvý element multicyklu, tak sa zmení na 1-1, čo znamená že momentálne navrhujeme prvú vetvu multicyklu. Po dokončení prvej vetvy (vetvy sú zakončené vlásenkou) sa zmení hodnota na 1-2.

Delete pattern – zruší navrhovaný vzor.

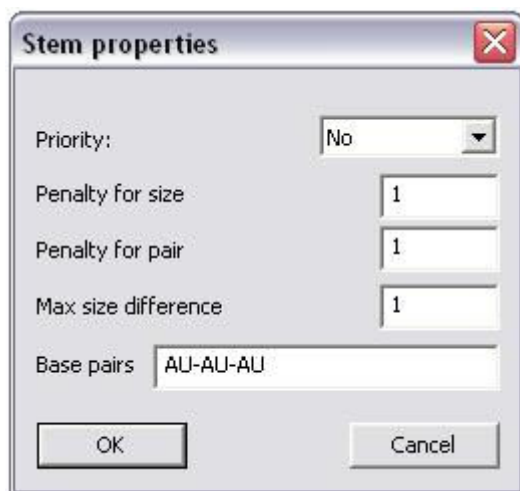
Complete – uloží dokončený vzor.

Up – posunie vzor nahor.

Left – posunie vzor doľava.

Right – posunie vzor doprava.

Down – posunie vzor dole.



Obr. 14: Formulár pre určenie vlastností kmienku.

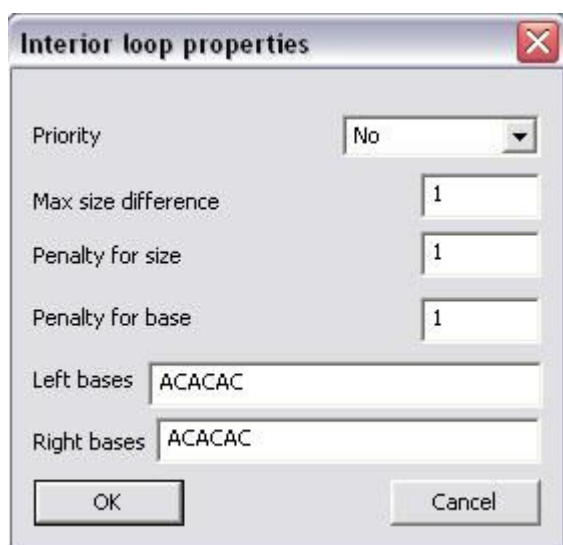
- Priority – ktoré vlastnosti elementu sú dôležité.
- Penalty for size – penalizácia za každý pár nad stanovenú veľkosť.
- Penalty for pair – penalizácia za každý pár s nesprávnymi nukleotidmi.
- Max size difference – maximálna odchýlka veľkosti, pri prekročení už nie je sekundárna štruktúra považovaná za podobnú.
- Base pairs – bázičné páry oddelené pomlčkou.



Obr 15.: Formulár pre určenie vlastností hairpin cyklu.

- Priority – ktoré vlastnosti elementu sú dôležité.
- Max size difference - maximálna odchýlka veľkosti, pri prekročení už nie je sekundárna štruktúra považovaná za podobnú.
- Penalty for size – penalizácia za každú bázu nad stanovenú veľkosť.
- Penalty for base – penalizácia za každú nesprávnu bázu.

- Bases – jednotlivé bázy hairpinu.



Interior loop properties

Priority: No

Max size difference: 1

Penalty for size: 1

Penalty for base: 1

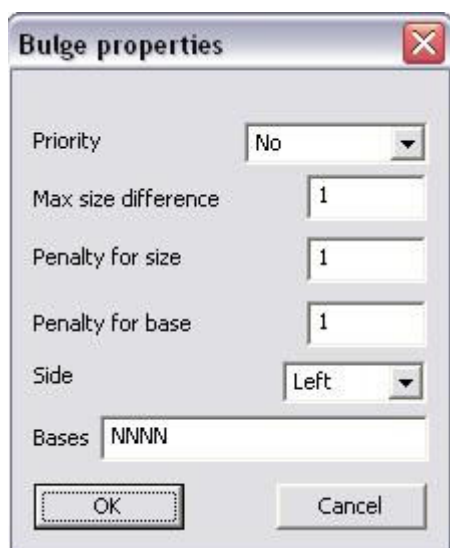
Left bases: ACACAC

Right bases: ACACAC

OK Cancel

Obr. 16: Formulár pre určenie vlastností interného cyklu.

- Priority - ktoré vlastnosti elementu sú dôležité.
- Max size difference - maximálna odchýlka veľkosti, pri prekročení už nie je sekundárna štruktúra považovaná za podobnú.
- Penalty for size - penalizácia za každú bázu nad stanovenú veľkosť.
- Penalty for base - penalizácia za každú nesprávnu bázu.
- Left bases – bázy na ľavej strane interného cyklu.
- Right bases – bázy na pravej strane interného cyklu.



Bulge properties

Priority: No

Max size difference: 1

Penalty for size: 1

Penalty for base: 1

Side: Left

Bases: NNNN

OK Cancel

Obr. 17: Formulár pre určenie vlastností výdute.

- Priority - ktoré vlastnosti elementu sú dôležité, strana výdute sa musí zhodovať pri každej priorite okrem možnosti NO.
- Max size difference - maximálna odchýlka veľkosti, pri prekročení už nie je sekundárna štruktúra považovaná za podobnú.
- Penalty for size - penalizácia za každú bázu nad stanovenú veľkosť.
- Penalty for base - penalizácia za každú nesprávnu bázu.
- Side – na ktorej strane má byť výdut'.
- Bases – jednotlivé bázy výdute.

The image shows a software dialog box titled "Multiloop properties". It has a standard Windows-style title bar with a close button. Inside, there are several configuration options:

- Priority:** A dropdown menu currently showing "No".
- Max size difference:** A text input field containing the number "1".
- Penalty for size:** A text input field containing the number "1".
- Penalty for base pair:** A text input field containing the number "1".
- Penalty for unpaired base:** A text input field containing the number "1".
- Unpaired Bases:** A text input field containing the string "NNNN-NNNN-NNNN".

 At the bottom of the dialog are two buttons: "OK" and "Cancel".

Obr 18.: Formulár pre určenie vlastností multicyklu.

- Priority – ktoré vlastnosti elementu sú dôležité.
- Max size difference - maximálna odchýlka veľkosti, pri prekročení už nie je sekundárna štruktúra považovaná za podobnú.
- Penalty for size – penalizácia za každú nespárovanú bázu nad stanovenú veľkosť.
- Penalty for base pair – penalizácia za odlišný typ bázeckého páru (tieto páry sa definujú v nasledujúcich kmienkoch).
- Penalty for unpaired base – penalizácia za odlišný nukleotid u nespárovaných báz.
- Unpaired bases – nespárované bázy, pomlčka určuje rozvetvenie.

Pozn. Ak nemáme požiadavku na typ bázy tak je možné použiť špeciálny typ N = Neurčená.

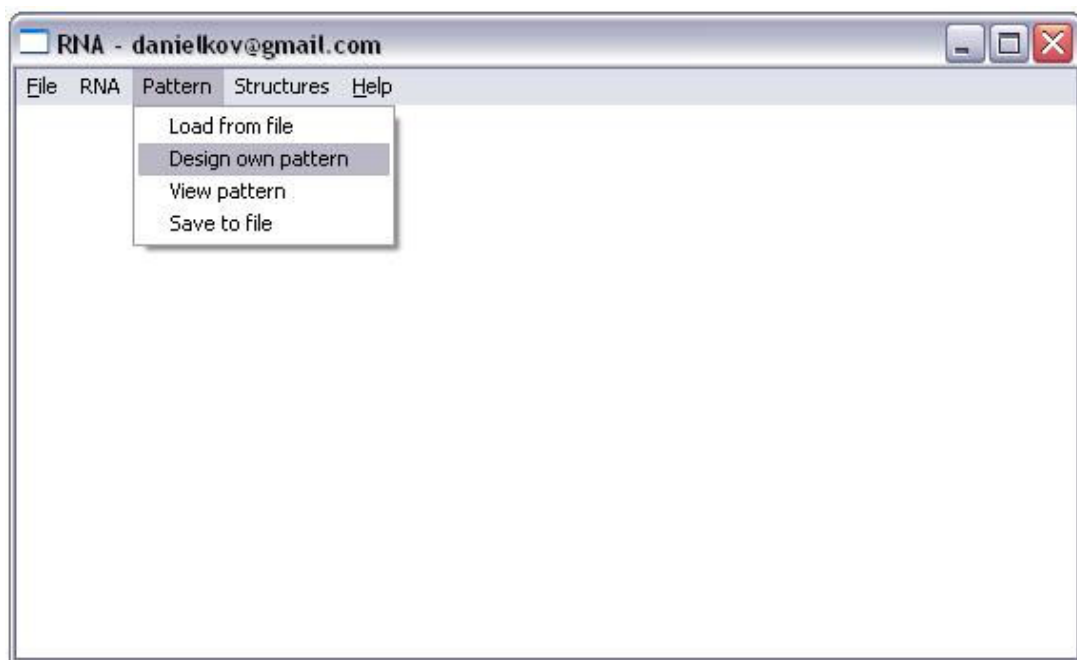
9.5 Praktické použitie

V tejto podkapitole si predvedieme praktický príklad použitia aplikácie. Aplikácia sa spúšťa zo súboru RNAwin2.exe. Po spustení sa zobrazí hlavné okno aplikácie:



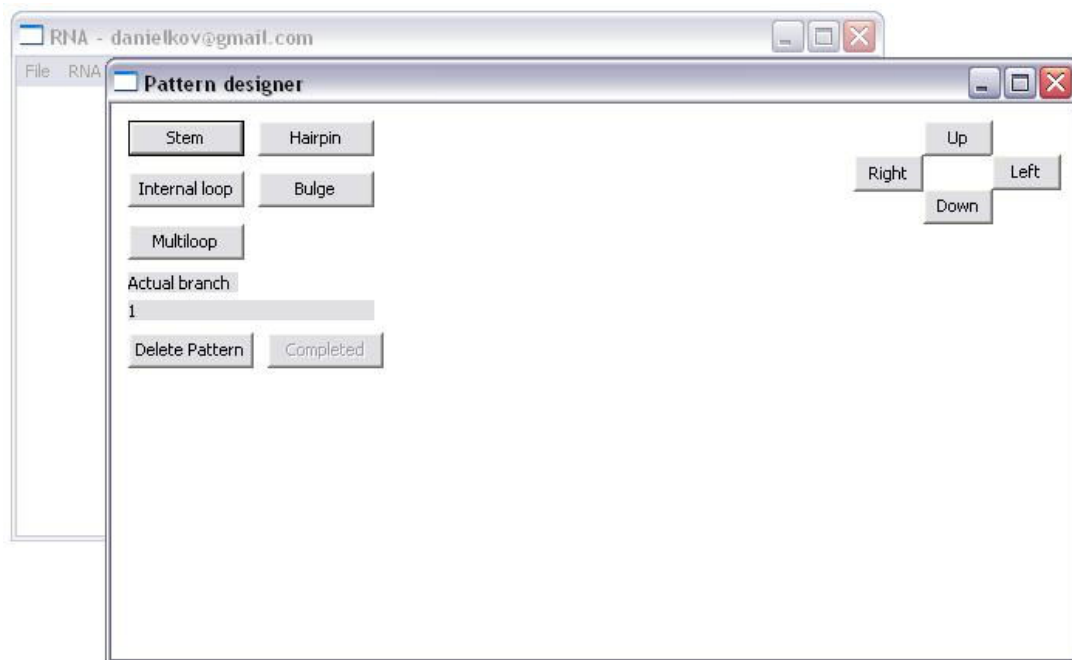
Obr. 19

Z menu spustíme dizajnér, kde budeme navrhovať vzor:



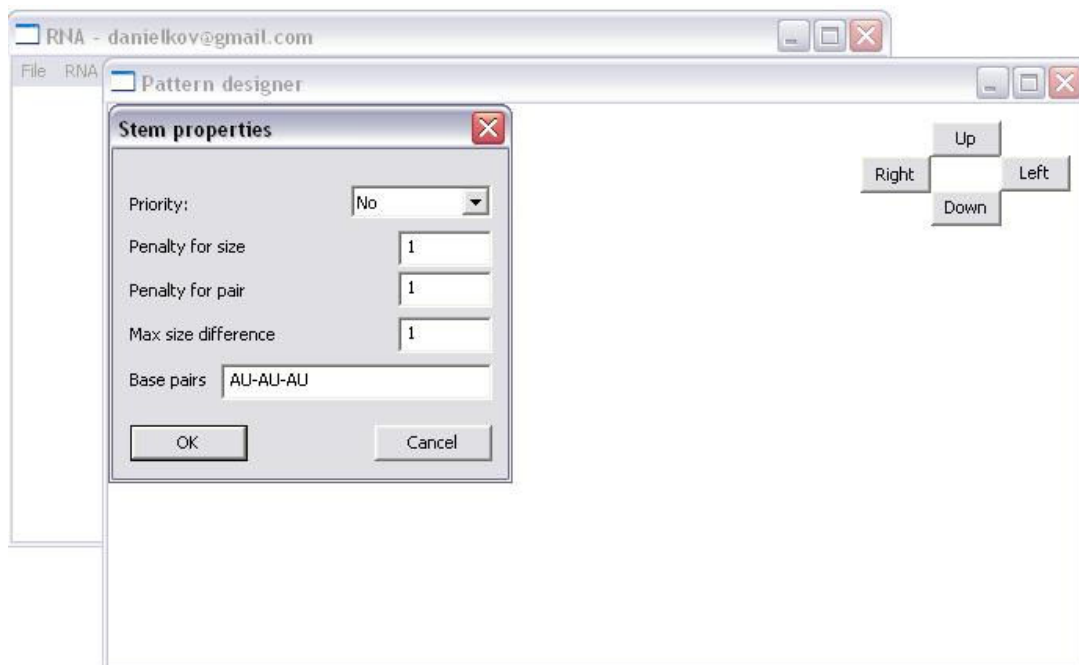
Obr. 20

Ten sa nám otvorí do ďalšieho okna:



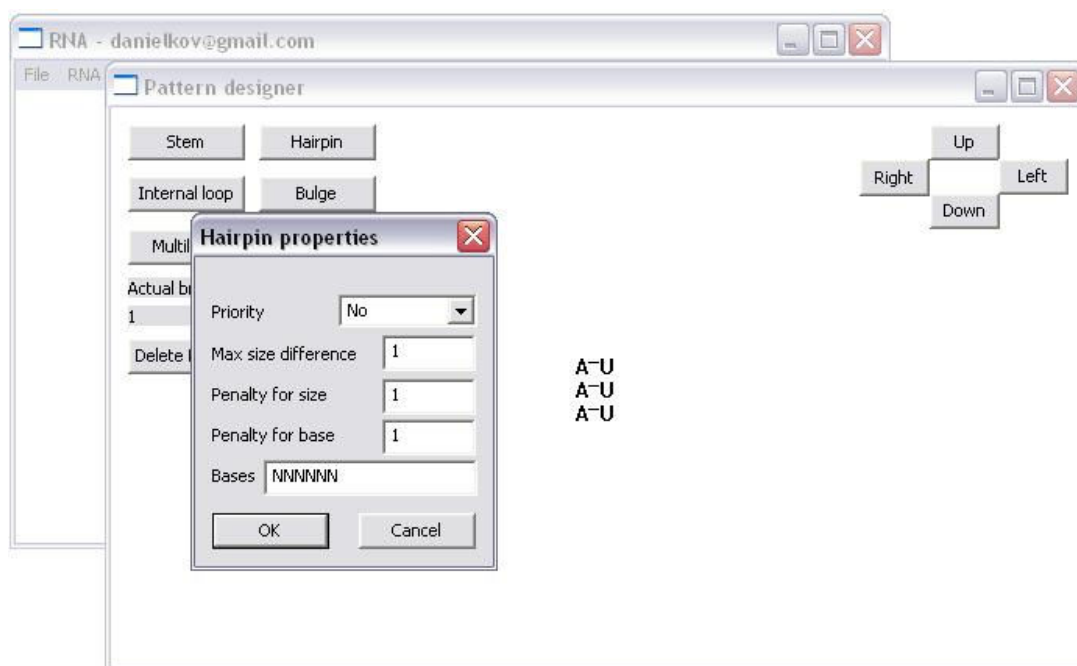
Obr. 21

V tomto okne navrhne jednoduchý vzor, ktorý sa bude skladať iba z jedného kmienku (Stem) a jednej vlásenky (Hairpin). Kliknutím na tlačidlo Stem sa otvorí okno, v ktorom je možné upraviť parametre kmienku:



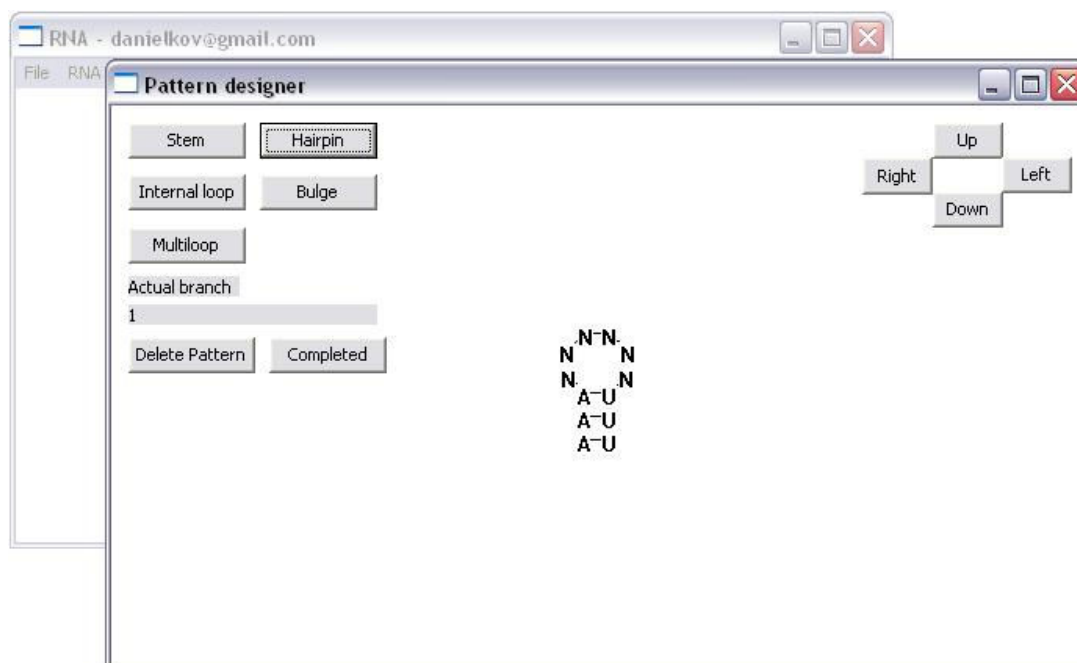
Obr. 22

Obdobne navrhujeme vlásenku kliknutím na tlačidlo Hairpin:



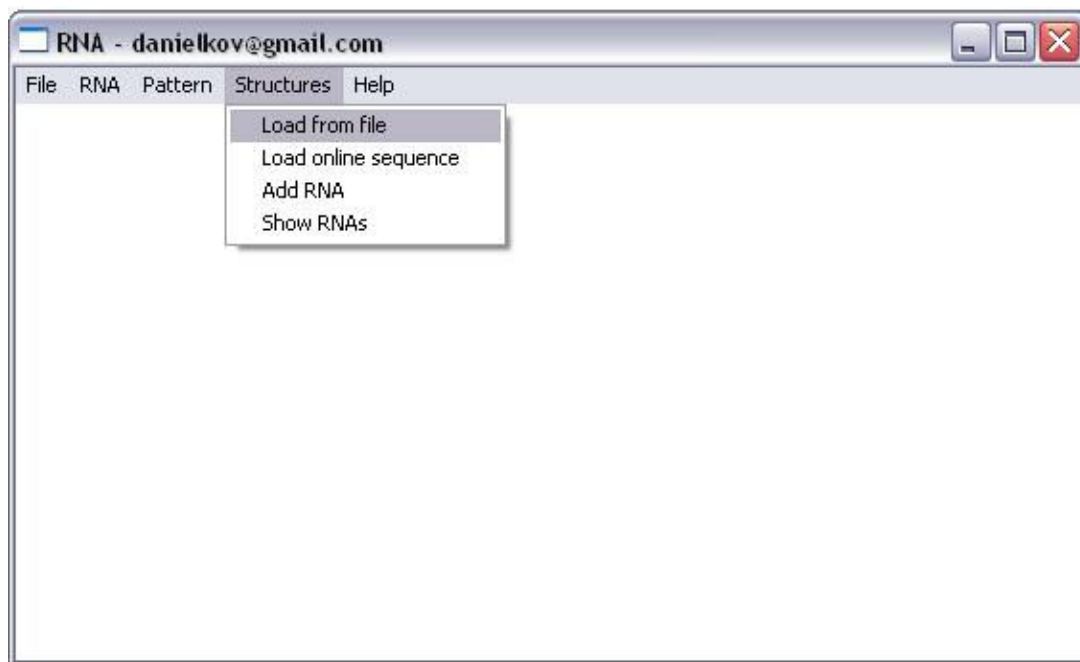
Obr. 23

Po dokončení vlásenky už je vzor kompletný. Potvrdíme ho stlačením tlačidla Completed:



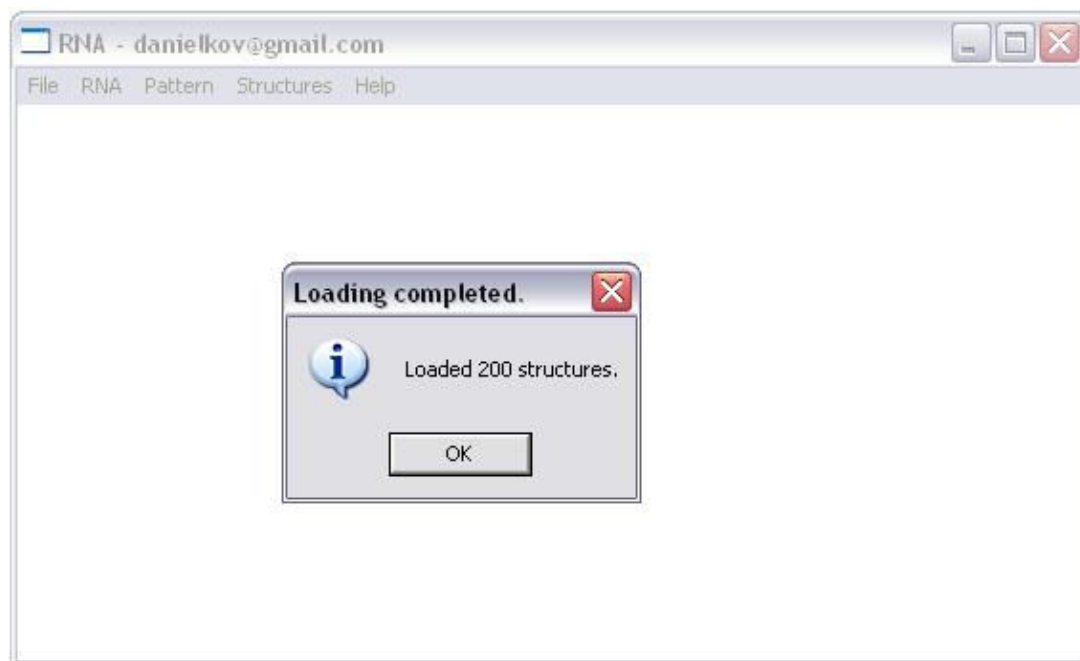
Obr. 24

Teraz načítame zo súboru sekvencie RNA v ktorých budeme tento vzor hľadať.
Načítanie sa spúšťa z menu:



Obr. 25

Po výbere správneho súboru v štandardnom otváracom okne sa zobrazí okno s počtom načítaných RNA:



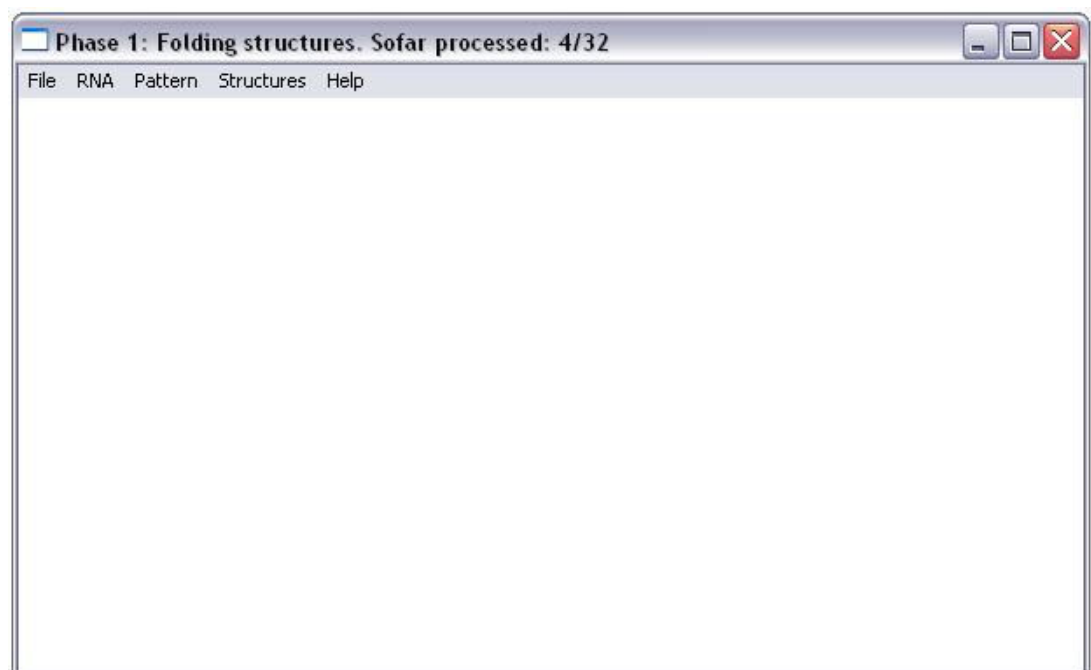
Obr. 26

Teraz už môžeme spustiť samotné vyhľadávanie. To sa spúšťa z menu:



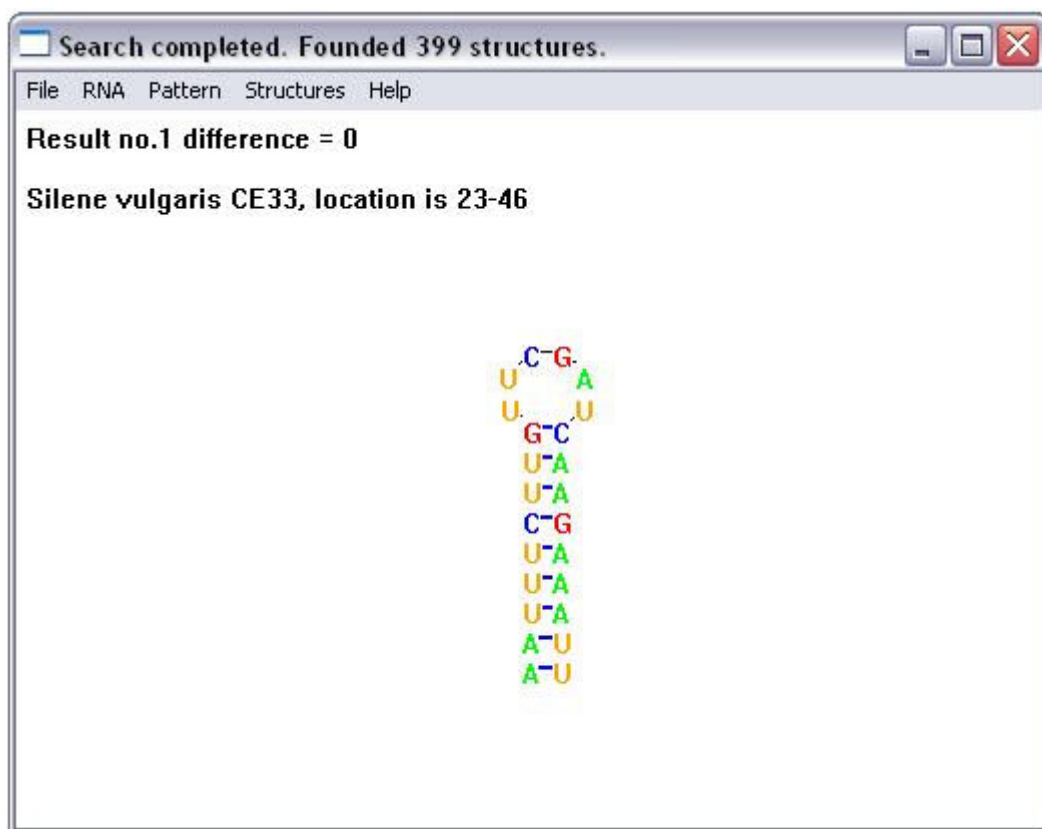
Obr. 27

Priebežný stav vyhľadávania sa zobrazuje v titulku okna:



Obr. 28

Po dokončení výpočtu sa vykreslí prvá nájdená sekundárna štruktúra:



Obr. 29

Okrem samotnej sekundárnej štruktúry je zobrazené aj poradové číslo výsledku. Číslo za „difference“ udáva, o koľko sa odlišuje táto sekundárna štruktúra od vzorovej. V ďalšom riadku je uvedený názov a pozícia vzoru v pôvodnej RNA sekvencii. Medzi výsledkami sa pohybuje pomocou šípiek na klávesnici. Posun vzoru po obrazovke je pomocou nasledujúcich písmen na klávesnici:

- A – posun doľava
- D – posun doprava
- W – posun hore
- D – posun dole

Vzor je možné aj priblížiť a oddialiť pomocou stredného kolieska myši.

Kapitola 10

Záver

Problém vyhľadávania v reťazcoch RNA podľa danej sekundárnej štruktúry je zložitý. Prvotný nápad, ktorý predpokladal poskladanie celej sekundárnej štruktúry pre všetky RNA, sa ukázal ako nepoužiteľný, kvôli časovej náročnosti algoritmu skladania. Po oboznámení sa s možnosťami porovnávania sekundárnych štruktúr som sa rozhodol navrhnúť vlastné riešenie, ktoré som následne implementoval. Explicitne som obmedzil dĺžku sekvencie, ktorej sekundárna štruktúra sa vypočítava priamo a pre dlhšie som navrhol iný prístup. Ten spočíva vo vytipovaní oveľa kratších regiónov sekvencie na základe podobnosti s predpokladanou primárnou štruktúrou vzoru. A až týmto regiónom je poskladaná sekundárna štruktúra.

Prínosom mojej aplikácie oproti iným je predovšetkým možnosť detailne navrhnúť vyhľadávaný vzor v aplikácii, kde už pri návrhu je graficky zobrazovaná aktuálne navrhnutá časť. Štandardné porovnávacie aplikácie umožňujú len zadať sekundárne štruktúry v zátvorkovej notácii a ich výstupom je počet nezrovnalostí. Moja aplikácia umožňuje pre každý jeden element definovať priority porovnávania a aj niektoré limity, po prekročení ktorých už sekundárna štruktúra nie je považovaná za podobnú. Pri návrhu som dbal na to, aby bolo možné typy parametrov porovnávania zmeniť bez väčších problémov. To by však vyžadovalo úpravu zdrojových kódov aplikácie.

Ďalšie pokračovanie projektu vidím predovšetkým v jeho poskytnutí na reálne použitie. Výsledkom by boli presnejšie definované požiadavky na porovnanie a parametre jednotlivých použitých algoritmov. Tie totiž zásadným spôsobom ovplyvňujú jeho efektivitu pri odhaľovaní podobných sekundárnych štruktúr, a teda aj použiteľnosť aplikácie.

Literatúra

- [1] Neil C. Jones a kol.(2004): An Introduction to Bioinformatics Algorithms, The MIT Press
- [2] Matthias Hochsmann (2005): The Tree Alignment Model: Algorithms, Implementations And Applications for the Analysis of RNA Secondary Structures, PhD thesis
- [3] Fylogenetický strom, http://en.wikipedia.org/wiki/Phylogenetic_tree.
- [4] Susan Cates: Phylogenetic Trees, <http://cnx.org/content/m11052/latest/>
- [5] David Morrison (2003): Carl Woese and New Perspectives on Evolution, http://nai.arc.nasa.gov/news_stories/news_detail.cfm?ID=274
- [6] R. Kominami, M. Hatanaka (1979): Conserved Region Of Mammalian Retrovirus RNA, <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=525941>
- [7] Knaut H. a kol (2002): An evolutionary conserved region in the vasa 3'UTR targets RNA translation to the germ cells in zebrafish http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&list_uids=11909530&dopt=Abstract
- [8] RNA gene-finding, http://wikiomics.org/wiki/RNA_gene-finding
- [9] Alifoldz, <http://www.tbi.univie.ac.at/papers/SUPPLEMENTS/Alifoldz/>
- [10] FASTA formát , http://en.wikipedia.org/wiki/Fasta_format
- [11] Giulio Pavesi a kol.(2004): RNAProfile: an algorithm for finding conserved secondary structure motifs in unaligned RNA sequences, <http://nar.oxfordjournals.org/cgi/content/full/32/10/3258>
- [12] RNAProfile, web formulár, <http://159.149.109.16:8080/RNAProfile/index.html>
- [13] I.L. Hofacker, a kol. (1994): Fast Folding and Comparison of RNA Secondary Structures. Monatshefte f. Chemie 125: 167-188
- [14] M. Zuker, P. Stiegler (1981): Optimal computer folding of large RNA sequences using thermodynamic and auxiliary information, Nucl Acid Res 9: 133-148
- [15] J.S. McCaskill (1990): The equilibrium partition function and base pair binding probabilities for RNA secondary structures, Biopolymers 29: 1105-1119

- [16] A New Algorithm for RNA Secondary Structure Design – Mirela Andronescu, Anthony P. Fejes, Frank Hutter, Holger H. Hoos, Anne Condon, Journal of Molecular Biology Volume 336, Issue 3, 20 February 2004, Pages 607-624
- [17] Anke Busch and Rolf Backofen (2007): INFO-RNA – a server for fast inverse RNA folding satisfying sequence constraints
- [18] NCBI, <http://www.ncbi.nlm.nih.gov/>
- [19] Vienna RNA Package: <http://www.tbi.univie.ac.at/RNA/>
- [20] BioPerl, <http://www.bioperl.org>
- [21] Stajich JE a kol.: The Bioperl toolkit: Perl modules for the life sciences. Genome Res 2002 Oct; 12 (10) 1611-8. doi:10.1101/gr.361602 pmid:12368254. PubMed HubMed [bioperl2002]
- [22] Vienna RNAdistance:
<http://bioweb.pasteur.fr/seqanal/interfaces/rnadistance.html>
- [23] Bruce Shapiro,
<http://www.tbi.univie.ac.at/~ivo/RNA/RNALib/notations.html>
- [24] Nussinov R a kol. (1978): Algorithms for loop matchings. SIAM Journal on Applied Mathematics

Obsah disku CD-ROM

CD obsahuje nasledujúce adresáre:

- app – Obsahuje aplikáciu RNA a testovacie dáta.
- doc/thesis – Text bakalárskej práce vo formáte .doc a .pdf a taktiež použité obrázky.
- doc/user – Užívateľská dokumentácia.
- doc/prg – Programátorská dokumentácia.
- src – Zdrojové kódy aplikácie.