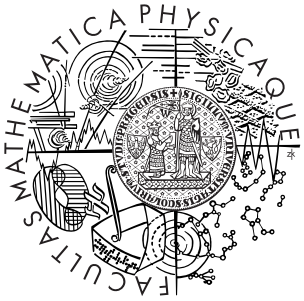
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jan Zimmermann

Porovnávání struktur RNA

Kabinet software a výuky informatiky

Vedoucí bakalářské práce: RNDr. František Mráz, CSc.
Studijní program: Informatika

2007

Rád bych na tomto místě poděkoval za vedení bakalářské práce a inspiraci k tématu RNDr. Františku Mrázovi, CSc. a RNDr. Heleně Štorchové, CSc. za uvedení do problému sekundárních struktur RNA.

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

V Praze dne 2.8.2007 Jan Zimmermann

Obsah

Název práce: Porovnávání struktur RNA
Autor: Jan Zimmermann
Katedra (ústav): Kabinet software a výuky informatiky
Vedoucí bakalářské práce: RNDr. František Mráz, CSc.
e-mail vedoucího: Frantisek.Mraz@mff.cuni.cz

Abstrakt: Sekvence RNA na rozdíl od DNA mohou vytvářet složité sekundární struktury. V předložené práci stanovujeme postupy pro automatické hledání podobností mezi sekvencemi RNA. Podobnosti chápeme nejen na základě sekvence bází tvořící RNA, ale též na základě jejich sekundárních struktur. Cílem práce je vymyslet a implementovat metody pro zkracování různorodých sekvencí, které jsou načteny z různých zdrojů. Definujeme pojmy strukturální alignment, strukturální profil a strukturální konsenzus, které jsou odpovídajícím rozšířením pojmů alignment sekvencí, profil sekvencí a konsenzus sekvencí. Součástí práce je aplikace RNAcut. Umí dávkově ořezat sekvence RNA na základě jejich primární a sekundární struktury a tím by měla usnadnit práci biologům. V aplikaci RNAcut implementujeme většinu algoritmů, které popisujeme v této práci.

Klíčová slova: bioinformatika, sekundární struktura RNA, dávkové zpracování, strukturální konsenzus

Title: Comparing RNA structures
Author: Jan Zimmermann
Department: Department of Software and Computer Science Education
Supervisor: RNDr. František Mráz, CSc.
Supervisor's e-mail address: Frantisek.Mraz@mff.cuni.cz

Abstract: RNA sequences creates complicated secondary structures, unlike DNA. In the presented work we state techniques for automatic similarity searching among RNA sequences. The similarity of RNA sequences we understand not only based on primary structure, but based on secondary structure as well. The goal of this work is to create techniques for cutting of various sequences, which are read from different sources. We define terms as structural alignment, structural profile and structural consensus, which are a generalization of terms as sequence alignment, sequence profile and sequence consensus. Part of this work is an application RNAcut. It is able to

cut RNA sequences based on its primary and secondary structure. In the application we implement most of algorithms, which are described in this work.

Keywords: bioinformatics, RNA secondary structure, batch processing, structural consensus

Kapitola 1

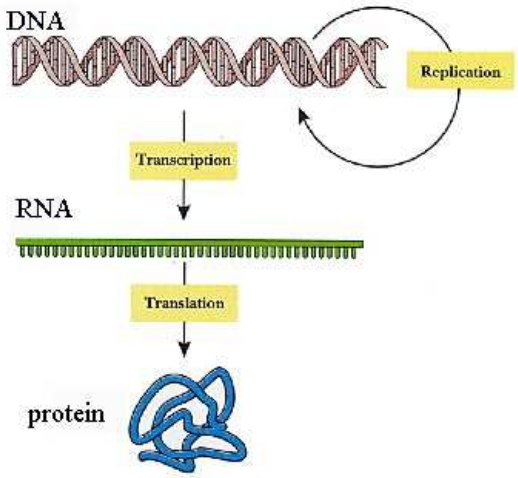
Stručný úvod do molekulární genetiky a bioinformatiky

Na začátku této práce nejdříve seznámíme čtenáře s vybranými pojmy bioinformatiky a genetiky. V první kapitole objasníme některé pojmy, jejichž pochopení je důležité pro čtení ostatních kapitol.

Každý biologický organismus (fenotyp) má v sobě uloženou genetickou informaci (genotyp). Fenotyp představuje výsledek spolupůsobení genotypu a prostředí, čili jak se informace z genotypu projevý v organismu. Genotyp je univerzálně uložen pomocí nukleových kyselin (DNA/RNA). V sekci 1.1 bude rozebrán vztah RNA x DNA a způsob uložení informace v těchto nukleových kyselinách. V dalších částech práce se již budeme zabývat výhradně RNA.

1.1 Co je to RNA?

Ribonukleová kyselina (RNA) je jednoduchý lineární řetězec kovalentně spojených nukleotidů. Nukleotid, jako základní stavební jednotka RNA, se vyskytuje ve čtyřech různých variantách A(adenin), C(cytosin), G(Guanin) a U(Uracil). Genetická informace je uložena právě v posloupnosti těchto nukleotidů (v tzv. sekvenci viz definice 1). Např. ...UAACAAGGU... Směr čtení sekvence se dá určit podle chemické orientace molekul nukleotidů (dohodnutý směr je pak 5'-3'). Více o orientaci sekvencí najde čtenář v odborné literatuře. Nejpodstatnější funkcí RNA je okopírovat genetickou informaci z DNA (pomocí transkripce) a přenést ji do místa, kde dojde k přeložení (translaci) RNA na výsledný protein, viz obrázek 1.1.



Obrázek 1.1: Způsob dekódování DNA na protein, převzato z [13]

Organismus	Počet bází
Escherichia coli (bakterie)	$4 \cdot 10^6$
Saccharomyces cerevisiae (kvasinky)	$1.35 \cdot 10^7$
Drosophila melanogaster(hmyz)	$1.65 \cdot 10^8$
Homo sapiens(člověk)	$2.9 \cdot 10^9$
Zea mays (obilí)	$5 \cdot 10^9$

Tabulka 1.1: Přehled velikostí genetické informace vybraných organismů (jedna báze může nabývat 4 hodnot), zdroj [1]

Chemicky jsou si RNA a DNA velmi podobné. Z hlediska bioinformatiky nám stačí vědět, že se v DNA se místo Uracilu vyskytuje Tymin. Při transkripci z DNA na RNA se v sekvenci nahradí T za U, transkripce je tedy pro bioinformatiku jednoduchý problém. Narozdíl od DNA netvoří RNA dlouhou dvoušroubovici, ale pouze kratší lineární řetězec. Tento řetězec zaujímá kvůli energetické výhodnosti stabilnější pozici. O tom se více zmíníme v sekci 1.3.

Při evoluci dochází v genetické informaci ke změnám. Tyto změny se dějí při mutaci[1] nebo při rekombinaci[2]. Následkem těchto změn dochází k různorodosti organismů a jejich genotypů. Sekvence DNA se skládá z exonů a intronů. Exony jsou oblasti, které kódují informaci o genu a které se přeloží

[1]Mutace je náhodné poškození původní sekvence nejčastěji při replikaci
[2]Rekombinace je sloučení dvou rodičovských sekvencí

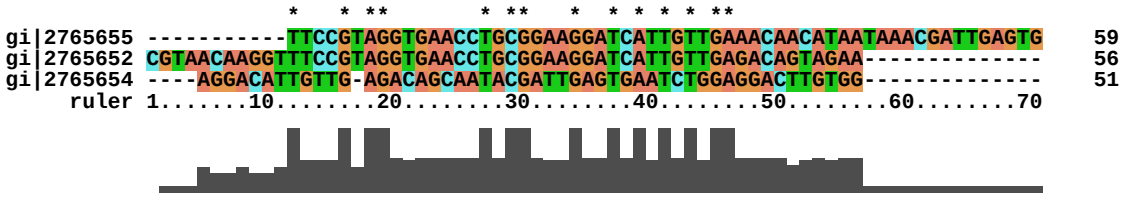
na proteiny. Naopak introny jsou oblasti sekvence, které nic nekódují, slouží jako vycpávka mezi exony.

Tedy organismy se liší svými genotypy. Věda zabývající se růzností organismů na základě analyzování sekvencí DNA se jmenuje molekulární fylogenetika. Exony jsou mnohem více konzervované části sekvence, protože podléhají evolučnímu tlaku. Naopak nekódující části, introny, se mění mnohem častěji.

Současná technika nám dovoluje číst genetickou informaci z biologických vzorků. Tomuto procesu se říká sekvenování DNA. Dalším zdrojem sekvencí jsou veřejně dostupné databáze, ve kterých biologové sdílejí nasekvenované DNA. Příklad takové databáze je [7]. Každá sekvence uložená v databázi má svoji anotaci. Anotace je popis sekvence. Může obsahovat následující informace: unikátní identifikace, organismus, o jakou část genomu se jedná, autor sekvenování, atd.

Definice 1 (Sekvence RNA) *Nechť Σ je abeceda $\{A, C, G, U\}$. Pak slovo $W \in \Sigma^n$ nad touto abecedou je sekvence RNA délky n nukleotidů (bází).*

1.2 Alignment sekvencí a profil sekvencí



Obrázek 1.2: Příklad sekvenčního alignmentu tří krátkých sekvencí orchidejí. Hvězdičky na prvním řádku označují místa, ve kterých jsou všechny sekvence stejné. Druhý až čtvrtý řádek obsahuje vlastní alignmentované sekvence. Pokud jsou stejné písmena bází různých sekvencí pod sebou, znamená to, že si tyto báze v sekvencích odpovídají. Pokud jsou písmena pod sebou ale různé, znamená to, že v těchto místech nastala substituce. Znaky '-' značí, že zde daná sekvence nemá odpovídající báze, nejspíš došlo k deleci nebo inserci. Sloupcový graf v nejspodnější části alignmentu ukazuje jako moc jsou jednotlivé části sekvencí konzervované.

Sekvenční alignment:

sekvenceA	-	A	G	G	C	T	A	T	C	A	C	C	T	G
sekvenceB	T	A	G	–	C	T	A	C	C	A	-	-	-	G
sekvenceC	C	A	G	–	C	T	A	C	C	A	-	-	-	G
sekvenceD	C	A	G	–	C	T	A	T	C	A	C	–	G	G
sekvenceE	C	A	G	–	C	T	A	T	C	G	C	–	G	G

Odpovídající sekvenční profil:

báze A		1					1			0.8				
báze C	0.6				1			0.4	1		0.6	0.2		
báze G			1	0.2						0.2			0.4	1
báze T	0.2					1		0.6					0.2	
-	0.2			0.8							0.4	0.8	0.4	

Tabulka 1.2: Převod sekvenčního alignmentu na profil. Například v alignmentu je na první pozici (druhý sloupec tabulky) jednou mezera -, jednou T a třikrát C. V profilu jsou pak na první pozici (opět druhý sloupec) odpovídající pravděpodobnosti (pro C pravděpodobnost 0.6, pro T pravděpodobnost 0.2 a pro mezeru taky pravděpodobnost 0.2.). Pokud by alignment obsahoval pouze jednu sekvenci, jeho profil by v každém sloupci obsahoval jednu jedničku.

Máme-li dvě a více sekvencí, často chceme zjistit, jestli a jak moc jsou si podobné. K tomuto účelu se používá tzv. sekvenční alignment. Je to vzájemné přiřazení odpovídajících si pozic v sekvencích, v optimálním případě co "nejlepší" přiřazení[3]. Z alignmentu je vidět o jaké mutace se alignmentované sekvence liší (substituce[4], delece[5] a inserce[6]). Viz též obrázek 1.2. Sekvenční profil je pak alignment několika sekvencí, který je chápán jako jedna zobecněná sekvence. To znamená, že místo samotných bází A, C, G, U jsou v sekvenci pravděpodobnosti výskytu jednotlivých bází. Viz tabulka 1.2 - převod alignmentu na profil. Sekvenční alignment umí počítat například program ClustalX (viz kapitola 8.1). ClustalX umí též generovat sekvenční profil.

[3]Co je "nejlepší" přiřazení se v různých algoritmech liší (dokonce i při různém nastavení parametrů). Obecně jde o to, aby co nejmíň bází v sekvencích nemělo přiřazení.

[4]Substituce je typ mutace, při které dojde k nahrazení kusu sekvence jinou sekvencí.

[5]Delece je typ mutace, při které dojde k vypuštění kusu sekvence.

[6]Inserce je typ mutace, při které se do původní sekvence vloží kusu jiné sekvence.

1.3 Struktury RNA

Strukturu RNA můžeme chápat podle stupně zjednodušení ve třech úrovních: primární, sekundární a terciální.

Primární struktura je samotná sekvence tvořená A,C,G nebo U (viz definice 1).

Před vysvětlením pojmu sekundární struktura je třeba nejdřív definovat bazické páry. Některé typy nukleotidů se mají tendenci spojovat i jinou než kovalentní vazbou, která drží pohromadě primární strukturu (viz sekce 1.1). Vodíkovými vazbami se spojují nuleotidy A-U, C-G a G-U. Tím vytvářejí bazické páry. Také se říká, že adenin je komplementární báze k uracilu (v DNA k tyminu) a cytosin je komplementární ke guaninu. Pro molekulu RNA je energeticky výhodnější spojovat své vlastní nukleotidy do bazických párů. Ve výsledku se pak spojují celé inverzní a komplementární podsekvence jedné molekuly. Tím se vytváří sekundární struktura RNA. Více o tvorbě a predikci sekundární struktury RNA bude uvedeno v sekci 1.5. Sekundární strukturu nemá pouze RNA (u DNA je to např. dvoušroubovice). Na rozdíl od terciální struktury tvoří sekundární struktura pouze dvourozměrné útvary. Dále ale budeme v textu sekundární strukturou myslet právě sekundární strukturu RNA, pokud nebude uvedeno jinak.

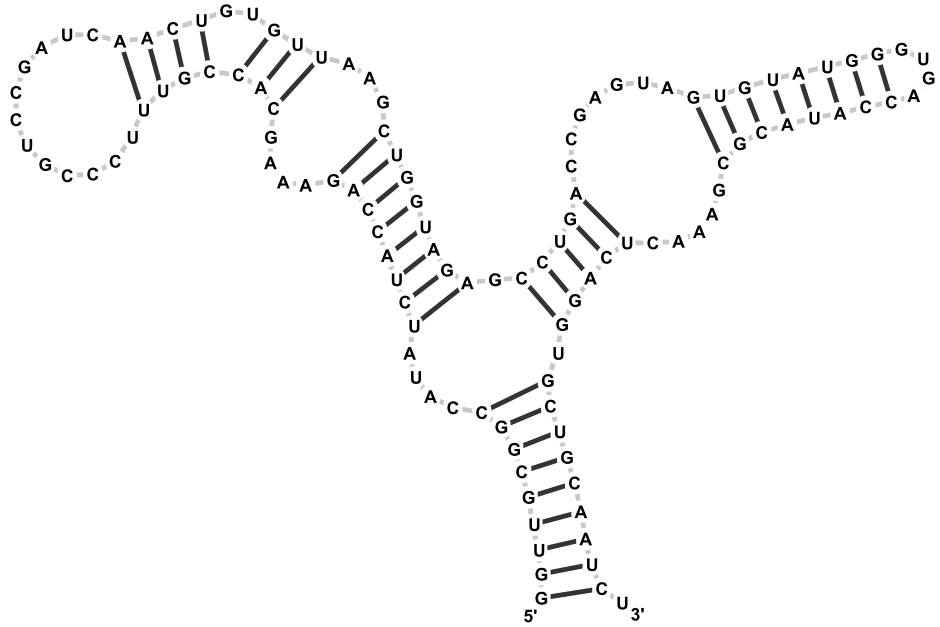
Terciální struktura je pak prostorové uspořádání sekundární struktury. Terciální struktura molekuly RNA je model, který má nejblíže k realitě.

1.4 Vyjádření sekundární struktury

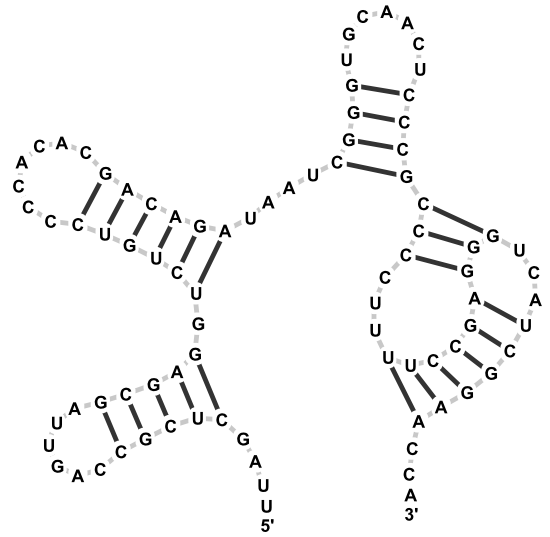
Definice 2 (Sekundární struktura na sekvenci RNA) *Nechť W je sekvence o délce n podle definice 1. Nechť je každé bázi z W přiřazen index z množiny $W_i = \{1, 2, ..., n\}$ podle pořadí v sekvenci. Pak sekundární strukturou S_{RNA} na sekvenci W nazveme binární relaci $S_{RNA} \subseteq W_i^2$, kde $\forall(i,j) \in S_{RNA}$ a $\forall(k,l) \in S_{RNA}$ platí:*

1. $i < j$
2. $i = k \Leftrightarrow j = l$
3. $k < j \Rightarrow i < k < l < j$

Druhá podmínka z definice říká, že žádná báze nemůže být v relaci bazických párů více než s jednou jinou bází. Třetí podmínka zakazuje existenci pseudouzlů (obrázek 1.4). To je sice další zjednodušení sekundárních struktur, ale



Obrázek 1.3: Příklad sekundární struktury. Tmavou čárkou jsou vyznačené spojení mezi bazickými páry. Světlou čárkou jsou spojené sousední báze v sekvenci. Jednotlivé části struktury jsou pojmenovány. Zdvojený úsek se nazývá stonek (stem). Ostatní části jsou cykly (loops). Například uprostřed je 3-cyklus, na konci stonku bývá 1-cyklus, kterému se též říká sponka (hairpin). Obrázek struktury byl vygenerován programem jViz.Rna [8].



Obrázek 1.4: Příklad sekundární struktury obsahující pseudouzly (pseudoknots). Pseudouzly jsou spojení dvou různých cyklů bazickými páry. Pseudouzel v zobrazené struktuře tvoří právě tři po sobě jdoucí bazické páry C-G. Pseudouzly nedovolují reprezentovat strukturu rekurzivně pomocí stromu. Obrázek struktury byl vygenerován programem jViz.Rna [8].

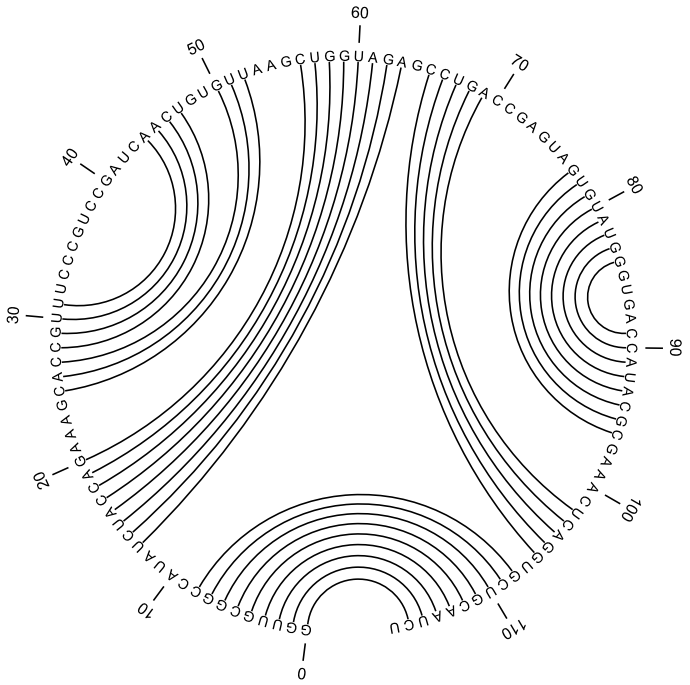
je nezbytné pro algoritmus predikce sekundární struktury založený na dynamickém programování. Další výhoda je, že lze strukturu bez pseudouzlů jednoduše uložit do stromové struktury (Viz postup v kapitole 4).

Obrázky 1.3, 1.5 a 1.6 zobrazují různými způsoby stejnou sekundární strukturu malé části RNA sekvence kvasinek (Saccharomyces cerevisiae)[7]. Soubor ve formátu fasta[8] této sekvence a její sekundární struktury vypadá takto:

```
>X67579.1 S.cerevisiae 5S ribosomal RNA.
GGTTGCGGCCATATCTACCAGAAAGCACCGTTTCCCGTCCGATCAACTGTGTTAAGCTG
GTAGAGCCTGACCGAGTAGTGTATGGGTGACCATACGCGAAACTCAGGTGCTGCAATCT
(((((((((....(((((((((..(((((.(((((.....)).)))).)))))...)))
)))))).....((((((.(((((((((....)))))))))...)))).)))))))))))).
```

[7] O zobrazování sekundárních struktur pojednává též kapitola 5

[8] Fasta je standardní formát textového souboru pro uložení sekvence nukleových kyselin nebo proteinové sekvence. Specifikace formátu fasta je zde [18].



Obrázek 1.5: Sekundární struktura z obrázku 1.3, ale zobrazena jinou metodou (tzv. Circular Feynman). Zde je sekvence nakreslena do kružnice a jednotlivé bazické páry jsou spojeny křikou napříč kruhem. Obrázek struktury byl vygenerován programem jViz.Rna [8].

První řádek specifikuje o jakou sekvenci se jedná včetně jejího jednoznačného identifikátoru "X67579.1". Na druhém a třetím řádku je vlastní sekvence. Poslední dva řádky obsahují sekundární strukturu ve formě závorkové notace. Závorková notace je řetězec složený ze znaků '.', '(' a ')'. Znak závorkové notace na k-tém místě odpovídá k-té bázi v sekvenci. Znak '.' znamená, že odpovídající báze netvoří bazický pár. Navzájem si odpovídající pár závorek '(' a ')' pak charakterizuje bazický pár v sekvenci. Závorková notace v této formě nedovoluje pseudouzly. To nám ale nevadí, protože ve zbytku práce budeme pracovat téměř vždy se sekundárními strukturami bez pseudouzlů, tedy dle definice 2.

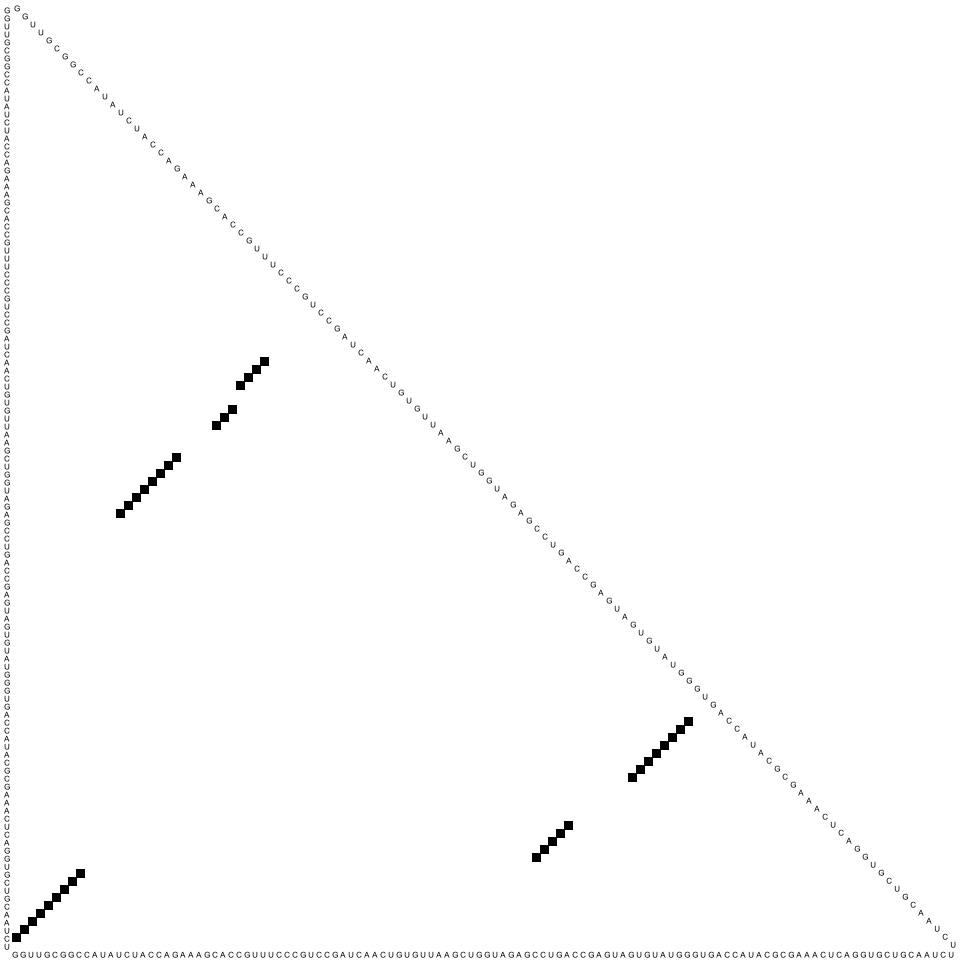
1.5 Predikce sekundárních struktur

Jak bylo uvedeno v sekci 1.3, řetězec RNA se skládá sám do sebe tak, že se spojují jeho inverzně-komplementární podsekvence. Nabízí se tedy otázky: Šlo by algoritmicky predikovat sekundární strukturu? Je možné použít na tuto úlohu algoritmus, který vyhledá inverzně-komplementární podsekvence a posléze je spojí pomocí bazických párů?

Predikování sekundárních struktur RNA je již jedna z typických úloh bioinformatiky. Pro řešení existuje několik metod. Jednotlivé algoritmy se liší výpočetní náročností a kvalitou řešení. Většinou se spojují právě inverzně-komplementární podsekvence. Existující algoritmy se dají rozdělit na ty, které hledají sekundární struktury s pseudouzly, a na ty které pseudouzly hledat neumějí. Jak jsme uvedli v sekci 1.4, my se budeme zabývat pouze strukturami bez pseudouzlů.

1.6 Shrnutí

V této kapitole jsme nastínili základní pojmy z genetiky a bioinformatiky, jejichž znalost je důležitá pro pochopení dalších částí této práce. Rozsáhlejší a obecnější popis genetiky pak najde čtenář v učebnicích nebo skriptech jako např. [12]. Ucelený přehled o bioinformatice je např. v knize [1].



Obrázek 1.6: Sekundární struktura z obrázku 1.3, ale zobrazena jinou metodou (tzv. dot plot). Primární struktura je zapsána do osy x a do osy y. Tmavé čtverečky vyjadřují, že jsou báze, odpovídající pozici čtverečku, spojeny do bazického páru. Sekvence po sobě jdoucích čtverečků je stonek. Obrázek struktury byl vygenerován programem jViz.Rna [8].

Kapitola 2

Úvod a motivace

V minulé kapitole jsme definovali vybrané části bioinfomatiky a genetiky. S těmito znalostmi již můžeme čtenáře uvést do problémů, kterými se v této práci zabýváme.

Cílem práce bylo vytvořit program pro dávkové zpracování RNA sekvencí a struktur. Program byl inspirován problémy řešenými RNDr. Helenou Štorchovou, CSc z Ústavu experimentální botaniky, laboratoře morfogeneze rostlin [6]. Zároveň je ale dostatečně obecný pro řešení jiných úloh. Program jsme pojmenovali RNAcut. Je k dispozici na přiloženém CD. Program má ulehčit práci biologům. RNAcut pracuje se seznamem podobných sekvencí. Sekvence mohou být do seznamu přidávány z různých zdrojů. RNAcut zkrátí jednotlivé surové sekvence podle definovaných kritérií, automaticky vyseparuje části sekvencí, které jsou pro biologa zajímavé, které právě studuje. Tím se s minimem námahy odstraní nezajímavé části sekvencí a můžou se tak podstatně zmenšit délky sekvencí ve studovaném souboru, čímž se zmenší velikost vstupních dat pro další zpracování. RNAcut umí ořezávat sekvence na základě jejich primární a sekundární struktury.

V bioinformatice nelze používat příliš exaktní algoritmy. Všechna vstupní data jsou do určité míry zatížena chybou. Proto jsme museli vymyslet způsob, jak definovat začátek a konec ořezání každé sekvence tak, aby řešení nebylo příliš postiženo chybami na vstupu.

V kapitole 3 se budeme zabývat ořezáním sekvencí podmíněné primární strukturou (tzv. RawCut). RawCut slouží hlavně ke zmenšení délky sekvencí před dalším stupněm ořezání (tzv. FineCut viz kapitola 4). FineCut je již ořezání podmíněné i sekundární strukturou. Počítá se až na ořezaných sekvencích po RawCut. Díky předstupni RawCut trvá výpočet FineCut výrazně

kratší dobu. V 5. kapitole rozebereme způsob zobrazování sekundární struktury pomocí pružinového modelu. Tento algoritmus je často používaný pro výpočet hezkého nakreslení složitých grafů. Rozebereme výhody a nevýhody jeho nasazení na problém kreslení sekundárních struktur a zmíníme také některé další metody. Kapitola 6 obsahuje uživatelskou dokumentaci k programu RNAcut.

Kapitola 3

Algoritmus pro hrubé ořezání sekvencí (RawCut)

Veřejné databáze genotypů obsahují obecně k jednomu organismu nejrůznější sekvence. Do veřejné databáze může kdokoliv přidávat své nasekvenované části genotypu. Tyto sekvence jsou sice anotovány, ale obecně jsou to libovolné části genotypu, které se značně liší svoji délkou. Biolog většinou potřebuje pracovat pouze s určitou částí genotypu, například s konkrétním genem nebo s konkrétní mezigenovou oblastí (jako třeba trnH-psbA v práci [6]). Proto je potřeba z každé sekvence v množině různorodých sekvencí vyextrahovat pouze tu část, kterou biolog právě studuje. Například databáze NCBI [7] při hledání výrazu "trnh psba angiosperms" najde v současnosti 5227 sekvencí. Délky jednotlivých nalezených sekvencí a informace v nich uložené se velmi liší (od nejkratších cca. 180 bází po nejdelších cca. 1980 bází). V této kapitole si popíšeme řešení problému extrahování.

Pojmem RawCut jsme označili hrubé ořezání, které pracuje pouze s primární strukturou sekvence, tedy pouze na úrovni lineárních sekvencí bází. Ořezání by mělo na vstupu dostat množinu sekvencí a jakási pravidla. Cílem je podle definovaných pravidel vyříznout z každé sekvence zajímavou část a zahodit nepotřebný zbytek. V našem případě jsou pravidla pro ořezání reprezentována počátečním a koncovým profilem (viz formálnější popis algoritmu níže). Tím se všechny sekvence zkrátí, některé až mnohonásobně. V následujícím textu formálně popíšeme konkrétní algoritmus, který jsme pro řešení tohoto problému vyvinuli.

3.1 Vstup

Seznam sekvencí Množina různorodých sekvencí, většinou výsledek vyhledávání v databázi genomů.

Profile cut on begin Sekvenční profil (viz sekce 1.2) pro specifikaci začátku ořezávané části.

Profile cut on end Profil sekvencí pro specifikaci konce ořezávané části.

3.2 Výstup

Seznam sekvencí Původní vstupní sekvence s informací kde se má provést jejich ořezání. Tedy pro každou sekvenci je na výstupu pozice začátku a pozice konce ořezávání.

3.3 Popis algoritmu

Provede se sekvenční alignment každé sekvence seznamu s oběma profily, tedy zjistí se pozice počátečního i koncového profilu vůči každé sekvenci seznamu. Výsledné oříznutí sekvencí je pak vždy mezi koncem počátečního profilu a začátkem koncového profilu (viz obrázek 3.1). Pokud platí jedna z podmínek:

- Kvalita alignmentu některé vstupní sekvence s některým vstupním profilem je příliš malá.
- Počáteční profil se naalignmentuje až za koncový profil.

pak zpracovávaná sekvence nejspíš neobsahuje požadovanou část, se kterou chce biolog dále pracovat. Pro úplnost dodejme, že pokud nejlepší alignment počátečního resp. koncového profilu není jednoznačný, pak se veme nejlevější resp. nejpravější nejlepší alignment[1].

[1]Na reálných biologických datech tento případ nastane jen s velmi malou praděpodobností.

Obrázek 3.1: Schéma fungování hrubého ořezání

Kapitola 4

Strukturální konsenzus a jemné ořezání sekvencí (FineCut)

V této kapitole popíšeme další stupeň dávkového ořezání sekvencí. Představme si situaci, kdy se chce biolog zabývat studiem nějaké konkrétní části sekundární struktury. Chce třeba studovat konkrétní část sekundární struktury[1] od různých taxonů a jak se mezi taxony jedna část struktury liší (například v [6]). V tom případě musí ručně procházet sekundární struktury všech sekvencí. V každé struktuře najít a vyseparovat tu část, která ho právě zajímá. Tahle práce je mechanická a velmi zdlouhavá. Navíc může biolog ručním zpracováním zanést do svého souboru chyby. Jak by se daly tyto podstruktury vyseparovávat automaticky? V této kapitole se pokusíme odpovědět na položenou otázku.

4.1 Hlavní myšlenka FineCut

Pojmem FineCut budeme označovat ořezání podmíněné sekundární strukturou, kterým se budeme zabývat v této části práce. Stejně jako v případě RawCut máme na vstupu množinu sekvencí. Každá vstupní sekvence navíc obsahuje svoji sekundární strukturu. Opět se předpokládá, že vstupní sek-

[1] Například konkrétní stonek napojený na 2-cyklus, který je napojený na stonek a konečně na sponku. Představme si, že biolog má množinu podobných sekundárních struktur, napřílkad můžou vypadat podobně jako jedna struktura na obrázku 4.1. Biologa ovšem zajímá pouze menší část z každé z těchto struktur. Například v obrázku 4.1 označená část v pravo dole. Jelikož jsou si všechny struktury v biologově množině podobné, obsahují také podobnou menší část, kterou chce biolog studovat.

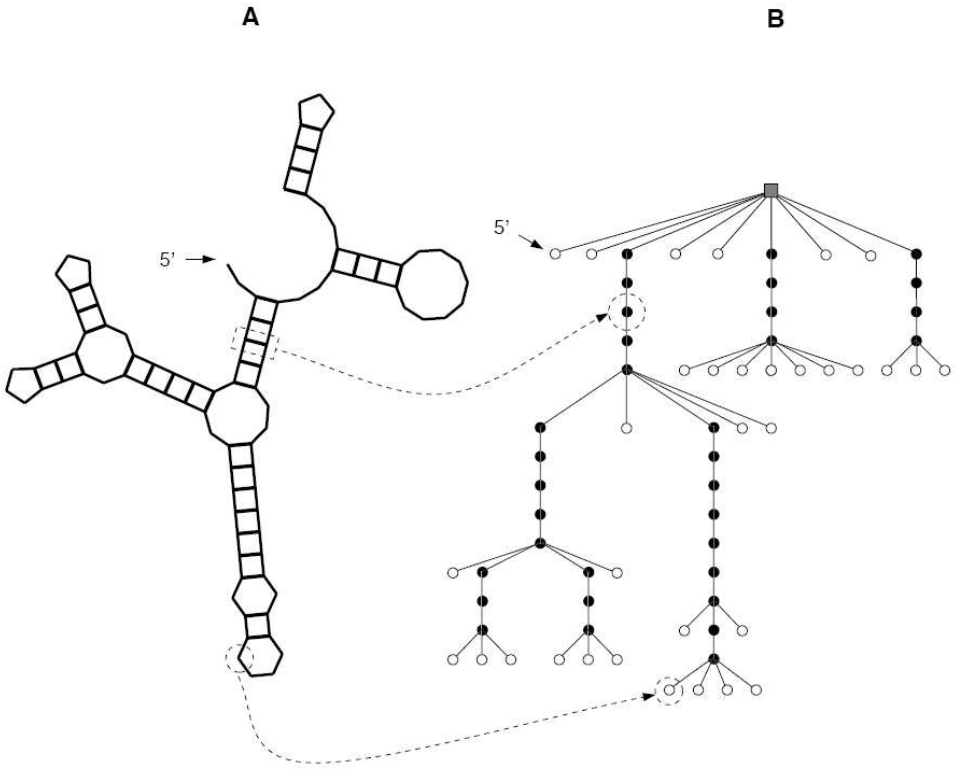
Obrázek 4.1: Příklad sekundární struktury s označenou částí.

vence jsou si v jistém smyslu podobné. FineCut pak mezi všemi vstupními sekundárními strukturami najde podobnosti a vygeneruje tzv. strukturální konsenzus. Tento konsenzus je zobecnění všech vstupních sekundárních struktur. Biolog pak v konsenzu jednou označí oblast struktury, která ho zajímá. Tato oblast je zpětně namapována do jednotlivých vstupních struktur a podle toho se na každé sekvenci provede ořezání. Takhle jsme neformálně popsali algoritmus FineCut. Ve zbytku kapitoly jej rozebereme podrobněji a formálněji.

4.2 Vhodná datová reprezentace sekundární struktury

Jak navrhnout správnou datovou strukturu pro uložení sekundární struktury? Existuje více způsobů. Nejjednodušší je ukládat si bazické páry jako dvojice. Je možné též reprezentovat sekundární strukturu závorkovou notací (viz kapitola 1.4). Výhoda závorkové notace je možnost uložení do textového souboru. Další možností je převést sekundární strukturu do stromu (viz obrázek 4.2). V následujícím odstavci rozebereme stromovou reprezentaci podrobně. Vhodnost použití konkrétní datové struktury závisí na způsobu práce se sekundární strukturou. My budeme téměř vždy používat výše popsanou stromovou reprezentaci sekundární struktury.

Ve stromové reprezentaci odpovídají bazické páry vnitřním (v obrázku 4.2 černým) uzlům. Báze, které nejsou s jinou bází v bazickém páru, tvoří listy stromu. Využijeme značení z definice 2. Pak $\forall(i,j) \in S_{RNA}$ a $\forall(k,l) \in S_{RNA}$ platí, že jestli $i < k$ a $l < j$ pak uzel odpovídající bazickému páru (k,l) je potomek uzlu odpovídající bazickému páru (i,j). Podobně pro listy. U vzniklého stromu záleží na pořadí synů. Jedná se tedy o uspořádané stromy. Do sekundární struktury se přidává fiktivní bazický pár (x,y). Báze x se přidá před první bázi sekvence a y se přidá za poslední bázi sekvence. Spojení fiktivních bází (x,y) tvoří kořen v sekundární struktuře vyjádřené stromem. Tenhle fiktivní bazický pár se do struktury přidává proto, aby z lesa vznikl strom (kořen zastřešující všechny uzly). Každý uzel stromu obsahuje ohodnocení (label) obsahující buď jeden nebo dva znaky. Ohodnocení vnitřních uzlů obsahuje dva znaky pro názvy bází bazického páru (AU, CG, UA, GC, GU nebo UG). Ohodnocení listů obsahuje jeden znak (A,U,C nebo G). Kořen (x,y) bude mít vždy ohodnocení FF. Pokud chceme sekundární strukturu oprostit od primární



Obrázek 4.2: Příklad zobrazení sekundární struktury reprezentované stromem, převzato z [2]

struktury, můžeme všechny uzly kromě kořene ohodnotit pomocí N.

4.3 Alignment dvou sekundárních struktur

Jak jsme již naznačili v sekci 4.1, potřebujeme vygenerovat zobecnění několika sekundárních struktur (tzv. strukturální konsenzus). Nejdřív musíme najít podobnosti mezi částmi jednotlivých vstupních struktur. Pro zjednodušení budeme v této sekci předpokládat, že vstupní sekvence a k nim příslušné sekundární struktury jsou pouze dvě. V sekci 4.4 rozšíříme metodu tak, aby pracovala s více vstupními strukturami. Problému hledání podobností mezi částmi dvou sekundárních struktur budeme říkat strukturální alignment, při stromové reprezentaci sekundární struktury můžeme také mluvit

o stromovém nebo lesovém alignmentu[2]. Strukturální alignment je zobecnění klasického alignmentu sekvencí. Opět je to relace mezi bázemi dvou sekvencí. Tahle relace ale narozdíl od klasického alignmentu respektuje sekundární strukturu obou sekvencí. Opět se řeší problém hledání maximálního (nejlepšího) alignmentu. Vstupní sekundární struktury převedeme do stromové reprezentace, která je uvedena v sekci 4.2. Problém hledání maximálního alignmentu sekundárních struktur tak převedeme na problém hledání maximálního alignmentu ohodnocených uspořádaných stromů. Nechť A je strom odpovídající první vstupní struktuře a B je strom odpovídající druhé vstupní struktuře. Potom alignment A a B je relace mezi uzly stromu A a uzly stromu B. Relace alignmentu musí zachovávat uspořádání i rodičovské vztahy. Tzn. že pokud a_1, a_2 jsou uzly stromu A a b_1, b_2 jsou uzly stromu B a pokud a_1 je v relaci alignmentu s b_1 a a_2 je v relaci alignmentu s b_2 a navíc pokud $a_1 < a_2$, pak musí taky $b_1 < b_2$, kde relace '<' značí uspořádání přímých potomků. Obdobně požadujeme zachování rodičovských vztahů. Tedy pokud a_1, a_2 jsou uzly stromu A a b_1, b_2 jsou uzly stromu B a pokud a_1 je v relaci alignmentu s b_1 a a_2 je v relaci alignmentu s b_2 a navíc pokud a_1 je potomek a_2 pak musí být b_1 potomkem b_2.

Nechť máme na vstupu dvě sekundární struktury vyjádřené pomocí dvou orientovaných ohodnocených stromů `structure_treeA`, `structure_treeB`. Zdrojový kód programu RNAcut[3] v jazyce Python detailně popisuje algoritmus pro hledání maximálního stromového alignmentu. Třída `StructureConsensusDynamicProgramingAlg2` dostane vstup v konstruktoru. Po vytvoření instance třídy se výpočet spustí metodou *go()*. Tato metoda také vrátí výsledek. Výstup algoritmu je spojení vstupních stromů. Ty uzly, které jsou v relaci alignmentu se sloučí do jednoho. Algoritmus pracuje na principu dvou dynamických programování[4]. Vyplňování M matice prvního dynamického programování charakterizují rovnice 4.1, 4.2 a 4.3.

$$M[u, v] = max\_dissimilarity \tag{4.1}$$

pokud je ohodnocení uzlů u a v různé a zároveň u nebo v je list,

$$M[u, v] = 2 \tag{4.2}$$

[2]O lesovém alignmentu píše již Hochsmann ve své práci [3]. My jsme se jeho prací nechali inspirovat, avšak algoritmy popsané dále v této práci a implementované v RNAcut nejsou stejné jako používá Hochsmann.

[3]Konkrétně soubor algorithms/saga.py třída StructureConsensusDynamicProgramingAlg2.

[4]Tento algoritmus je odvozen od algoritmu pro hledání maximálních shodných podstromů v práci [14] v kapitole "Subtree Matching"

pokud je ohodnocení stejné a zároveň u nebo v je list,

$$\begin{aligned} M[u,v] &= \max(A(u,v), B(u,v), C(u,v)) \qquad (4.3)\\ A(u,v) &= 2 + \max_{\phi \in map(u,v)} \left(\sum_{u_i \in sons(u)} M[u_i, \phi(u_i)] \right) \\ B(u,v) &= \max_{u_i \in sons(u)} (M[u_i, v] - jump\_penalty(u, u_i)) \\ C(u,v) &= \max_{v_i \in sons(v)} (M[u, v_i] - jump\_penalty(v, v_i)) \\ jump\_penalty(u,u_i) &= size\_of\_subtree(u) - size\_of\_subtree(u_i) \end{aligned}$$

v ostatních případech.

Konstanta *max_dissimilarity* je dostatečně velké záporné číslo. Tím je zajištěno, že se ve výsledné relaci alignmentu neobjeví (u, v) pokud je ohodnocení těchto uzlů různé. Pokud je u nebo v list a navíc jejich ohodnocení je stejné, pak nelze nic jiného než vložit do relace alignmentu (u, v). Kvalita této operace je pak vždy 2 (viz rovnice 4.2). Pokud jsou ohodnocení u, v stejné a oba uzly jsou vnitřní pak se hledá řešení přes tři volby. Tyto volby jsou vyjádřeny v rovnicích 4.3, funkcemi A, B a C. Přičemž se vybírá vždy nejlepší volba ($\max(A(u,v), B(u,v), C(u,v))$). Funkce $A(u,v)$ vrací kvalitu alignmentu, pokud by se měli spojit uzly u a v. Hledá se tedy nejlepší alignment mezi přímými potomky obou uzlů u a v. Tento alignment přímých potomků je vyjádřen prostou funkcí $\phi : sons(u) \rightarrow sons(v)$. Funkce $B(u,v)$ reprezentuje situaci, kdy je uzel u přeskočen a uzel v se spojí s některým potomkem uzlu u. Jako návratovou hodnotu vrací funkce B kvalitu takového řešení. Funkce $\max_{u_i \in sons(u)}$ rozhoduje, se kterým přímým potomkem uzlu u se spojí uzel v. Funkce $jump\_penalty(u, u_i)$ vrací celé číslo, které vyjadřuje počet uzlů, které nebudou moct být ve výsledné relaci alignmentu pokud je u přeskočen (jsou to velikosti podstromů všech bratrů uzlu u_i). Funkce $C(u,v)$ vyjadřuje symetricky to samé co funkce $B(u,v)$ (tentokrát se přeskakuje uzel v). Principem dynamického programování je, že se využívá již hotových parciálních řešení. To je vidět v rovnici 4.3. Pro výpočet $M[u,v]$ se přistupuje k dalším hodnotám matice M. Konkrétně k hodnotám na indexech z množiny dvojic $\{(sons(u) \cup \{u\}, sons(v) \cup \{v\})\} \setminus \{(u,v)\}$. Abychom zaručili, že jsou hodnoty na těchto indexech v matici M již spočítané, musíme matici vyplňovat ve správném pořadí. Nám stačí, aby se nikdy nepočítala hodnota $M[u,v]$ před tím, než jsou spočítány hodnoty pro všechny potomky u i v. Jako poslední se tedy bude počítat hodnota $M[r_1, r_2]$, kde r_1 a r_2

jsou kořeny prvního a druhého vstupního stromu. Pak kvalita nalezeného nejlepšího alignmentu bude zapsána v matici M právě na indexech r_1 a r_2.

Druhé dynamické programování[5] je použito pro výpočet $\max_{\phi \in map(u,v)}$ z rovnice 4.3 ve vzorci pro $A(u, v)$. Má za úkol provést klasický alignment přímých potomků dvou uzlů u a v a efektivně najít nejlepší mapování (alignment) ϕ. Tedy v jednom kroku dynamického programování použijeme další dynamické programování.

Probrali jsme způsob jakým se vyplňuje tabulka pro dynamické programování. Jako výsledek máme zatím pouze informaci o kvalitě nalezeného nejlepšího alignmentu - hodnota $M[r_1, r_2]$. Jistě bychom chtěli spočítat i konkrétní relaci pro tento nejlepší alignment a vědět které uzly dvou vstupních stromů jsou spojené alignmentem. Abychom to mohli zjistit, musíme si při vyplňování tabulky M ukládat ke každému indexu další informace. Pro konstrukci alignmentu je třeba vědět, které hodnoty byly u funkcí max() vybrány jako maximální, to znamená znát volbu funkce $A(u, v)$, $B(u, v)$ nebo $C(u, v)$. V případě volby $A(u, v)$ si musíme navíc uložit alignment přímých potomků ϕ. Pro volbu $B(u, v)$ resp. $C(u, v)$ musíme uložit u_i resp. v_i. Pokud máme všechny tyto informace uloženy v M, můžeme nyní popsat jak z této matice zkonstruovat[6] nejlepší alignment s kvalitou $M[r_1, r_2]$. Uvádíme rekurzivní řešení v pseudokódu.

```
function find_alignment(M, u, v): alignment_relation;
begin
    result = empty_relation();
    if u.label != v.label // různé ohodnocení
        return empty_relation();
    if M[u,v].choice == leaf // pokud u nebo v je list
        result.add((u,v)); // přidej do výsledné relace (u,v)
    if M[u,v].choice == A  // alignment přímých potomků
    begin
        foreach (e,f) in M[u,v].fi
            result.add((e,f));
            result = union(result, find_alignment(M,e,f));
        result.add((u,v)); // přidej do výsledné relace (u,v)
    end;
```

[5]Ve zdrojovém kódu RNAcut je druhé dynamické programování v metodě StructureConsensusDynamicProgramingAlg2::get_best_children_alignment_relation()

[6]Ve zdrojovém kódu RNAcut je konstrukce alignmetnu v metodě StructureConsensusDynamicProgramingAlg2::trace_back()

```
    if M[u,v].choice == B // přeskok uzlu u
    begin
        result = union(
            result,
            {(M[u,v].u_i,v)}, // přidej do výsledné relace (u_i,v)
            find_alignment(M,M[u,v].u_i,v)
        );
    end;
    if M[u,v].choice == C // přeskok uzlu v
    begin
        result = union(
            result,
            {(u,M[u,v].v_i)}, // přidej do výsledné relace (u,v_i)
            find_alignment(M,u,M[u,v].v_i)
        );
    end;
    return result;
end;
```

Rekurzi spustíme v kořenech voláním `find_alignment(`M`,`r_1`,`r_2`)`. Návratová hodnota této pseudofunkce je pak relace pro nejlepší alignment. Funkce se větví do pěti částí. Pokud je ohodnocení (labels) různé, vrátí prázdnou relaci. Pokud `u` nebo `v` jsou listy vrátí, relaci obsahující pouze vztah (u, v). Pokud v rovnici 4.3 bylo vybráno jako maximum hodnota funkce $A(u, v)$, pak funkce `find_alignment()` vrátí relaci obsahující (u, v), alignment přímých potomků ϕ a navíc výsledky rekurentních volání pro každou dvojici z alignmentu přímých potomků. Pokud v rovnici 4.3 bylo vybráno jako maximum hodnota funkce $B(u, v)$, pak funkce `find_alignment()` vrátí relaci obsahující (u_i, v) a návratovou hodnotu rekurentního volání na (u_i, v). Obdobně pokud v rovnici 4.3 bylo vybráno jako maximum hodnota funkce $C(u, v)$.

Implementaci celého algoritmu v jazyce Python najde čtenář ve zdrojovém kódu aplikace RNAcut, konkrétně soubor algorithms/saga.py třída `StructureConsensusDynamicProgramingAlg2`.

4.4 Mnohonásobný strukturální alignment

V minulé sekci jsme vyčerpávajícím způsobem popsali algoritmus pro alignment dvou sekundárních struktur. Nyní můžeme zobecnit problém na n vstupních sekundárních struktur resp. uspořádaných ohodnocených stromů. Budeme ho nazývat mnohonásobný strukturální alignment (anglicky multiple structural alinment). Vstupní sekundární struktury budeme opět reprezentovat pomocí uspořádaných ohodnocených stromů. Výstupní relace bude tentokrát trochu netradiční. Potřebujeme vyjádřit vztahy mezi uzly jednotlivých vstupních stromů. Relaci musíme formálně definovat takto:

Definice 3 (Relace mnohonásobného stromového alignmentu) *Nechť $T_1, T_2, \ldots T_n$ jsou množiny uzlů, které obsahují stromy $1, 2 \ldots n$. Pak definujeme relaci pro mnohonásobný alignment uspořádaných ohodnocených stromů jako následující strukturu disjunktních podmnožin:*

$$\Phi \subseteq \mathbb{P}\left(\bigcup_{i=1\ldots n} T_i\right)$$

Kde navíc pro každou množinu z Φ platí, že obsahuje nejvýše jeden uzel z každého stromu. Pak dva uzly jsou v relaci strukturálního alignmentu pokud náleží do stejné množiny v systému podmnožin Φ. Opět musí platit, že relace alignmentu musí zachovávat uspořádání i rodičovské vztahy ve stromu, stejně jako v sekci 4.3.

Algoritmus pro hledání nejlepšího alignmentu n uspořádaných ohodnocených stromů je zobecněním algoritmu, který jsme popsali v sekci 4.3. Dynamické programování se musí zobecnit pro n stromů. Tabulka hlavního dynamického programování bude n-rozměrná. V každém kroku hlavního dynamického programování se bude počítat vnitřní dynamické programování (alignment přímých potomků). Tabulka vnitřního dynamického programování může mít opět až n rozměrů. Paměťová a časová náročnost tohoto algoritmu je opravdu veliká, proto je toto řešení pro větší n nepoužitelné. Dopodrobna se jím tedy nebudeme zabývat. Je implementován ve zdrojovém kódu RNAcut v souboru algorithms/saga.py, konkrétně ve třídě `StructureConsensusDynamicProgramingAlgN`. Teoreticky by šlo vylepšit časovou a prostorovou složitost podobně jako to udělali Gupta, Kececioglu a Schäffer ve své práci [15] a jejich metody, původně zamýšlené pro sekvenční mnohonásobný alignment, přizpůsobit pro mnohonásobný strukturální alignment. Tuhle variantu jsme však zatím neimplementovali.

4.5 Heuristika v mnohonásobném strukturálním alignmentu

V této sekci se budeme zabývat algoritmy, které jsou méně náročné na paměť a čas. Budeme se však muset spokojit se suboptimálním řešením. Pro mnohonásobný strukturální alignment by se dali použít téměř všechny heuristiky z klasického sekvenčního alignmentu, které by se vhodně upravily. Například metody založené na genetických algoritmech, simulovaném žíhání, progresivní metody atd. My se v této práci budeme zabývat pouze progresivními metodami. Nejdříve musíme trochu upravit algoritmus pro výpočet alignmentu dvou sekundárních struktur ze sekce 4.3. Chceme docílit toho, abychom jako vstup mohli použít profil sekundární struktury, místo obyčejné sekundární struktury. Tedy abychom mohli výstup algoritmu ze sekce 4.3 použít jako vstup toho samého algoritmu. Profil sekundární struktury je zobecnění obyčejné sekundární struktury, podobně jako profil sekvencí je zobecnění sekvence. Profil sekundární struktury, reprezentovaný stromem, je opět uspořádaný ohodnocený strom, jehož všechny vrcholy navíc obsahují váhu (reálné číslo). Profil sekundární struktury je vlastně alignment sekundární struktury. Obyčejná sekundární struktura se převede na profil tím, že se všem jejím uzlům přiřadí váha 1. Z alignmentu vznikne profil tak, že se vrcholy, které jsou spolu v relaci alignmentu spojí v jeden a jejich váhy se sečtou. Algoritmu ze sekce 4.3 jsme trochu změnili formát vstupu. Ještě musíme následujícím způsobem změnit rovnici 4.2:

$$M[u,v] = w(u) + w(v) \tag{4.4}$$

a rovnici 4.3:

$$\begin{aligned}
M[u,v] &= \max(A(u,v), B(u,v), C(u,v)) && (4.5)\\
A(u,v) &= w(u) + w(v) + \max_{\phi \in map(u,v)} \left(\sum_{u_i \in sons(u)} M[u_i, \phi(u_i)] \right)\\
B(u,v) &= \max_{u_i \in sons(u)} (M[u_i, v] - jump_penalty(u, u_i))\\
C(u,v) &= \max_{v_i \in sons(v)} (M[u, v_i] - jump_penalty(v, v_i))\\
jump_penalty(u,u_i) &= weight_of_subtree(u) - weight_of_subtree(u_i)
\end{aligned}$$

Kde $w(u)$ je váha uzlu u. Poslední změna je, že algoritmus nebude vracet jako výstup přímo relaci alignmentu, ale dva vstupní profily podle vypočítaného

alignmentu spojí do jednoho profilu. Pojmenujme si tento algoritmus jako *PairwiseStructuralProfileAlignment*. Pak funkce $PSPA(A, B)$ vrací profil, který vznikl spojením profilů A a B algoritmem *PairwiseStructuralProfileAlignment*.

Definujme si ještě funkci, která charakterizuje podobnost dvou profilů.

Definice 4 (Podobnost dvou strukturálních profilů) *Nechť A a B jsou porovnávané profily. Pak funkce vyjadřující podobnost profilu A a B má následující předpis:*

$$\lambda(A, B) = \frac{M[root(A), root(B)]}{\sum_{u \in N(PSPA(A,B))} w(u)},$$

kde funkce $N(A)$ vrací množinu všech uzlů profilu A, funkce $root(A)$ vrací kořenový uzel profilu A a M je matice z algoritmu PairwiseStructuralProfileAlignment.

Definice 4 říká, že podobnost profilů A a B se spočítá jako podíl kvality alignmentu A,B a váhy profilu, který vznikne sloučením A a B. Pro výpočet $\lambda(A, B)$ je tedy potřeba spočítat $PSPA(A, B)$. Například $\lambda(A, B) = 1$ znamená, že A a B jsou identické, $\lambda(A, B) = 0.89$ znamená, že A a B jsou z 89% identické.

Náš upravený algoritmus *PairwiseStructuralProfileAlignment* nyní může na vstup dostat výstup ze svého předchozího běhu. Můžeme tedy vystavět mnohonásobný alignment pomocí alignmentu dvou profilů. Na začátku všechny sekundární struktury převedeme na profily (váha každého uzlu bude 1). Tyto profily budeme postupně po dvou slučovat, dokud nám nakonec nezůstane jeden jediný výsledný profil. Pokud si pří slučování profilů držíme reference na uzly v původních vstupních strukturách, jsme schopni z výsledného profilu zrekonstruovat relaci mnohonásobného strukturálního alignmentu. Obecně jsme si popsali progresivní algoritmus pro mnohonásobný strukturální alignment. Opět čerpáme inspiraci z algoritmů pro mnohonásobný sekvenční alignment. Definujme pořadí ve kterém se budou slučovat jednotlivé profily. Popisované metodě slučování se v sekvenčním alignmentu říka ClustalW [19]. Nejdříve se vytvoří matice podobnosti. Například nechť A, B, C, D, E jsou vstupní sekundární struktury. Ne-

chť je třeba matice podobnosti pro tyto vstupní struktury následující:

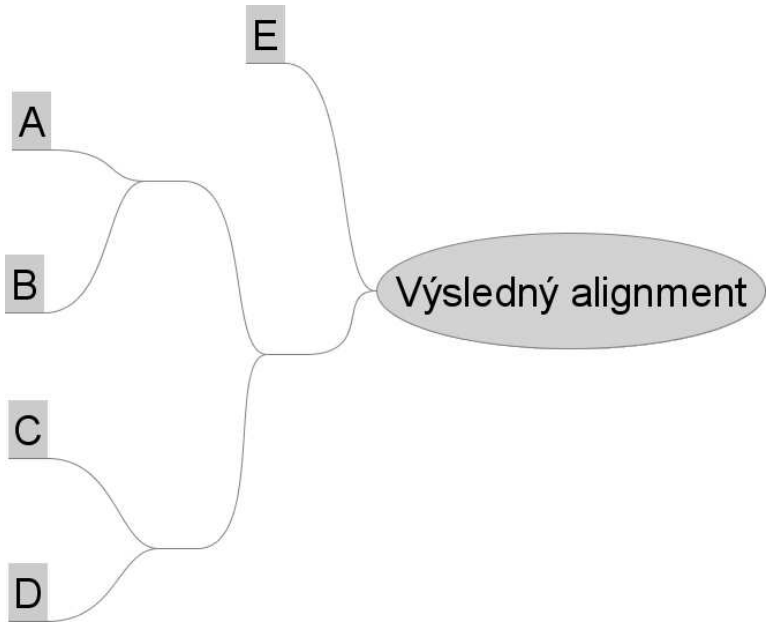
$$\lambda(A,B,C,D,E) = \begin{pmatrix} - & & & & \\ 0.95 & - & & & \\ 0.82 & 0.77 & - & & \\ 0.70 & 0.87 & 0.92 & - & \\ 0.50 & 0.45 & 0.48 & 0.52 & - \end{pmatrix},$$

kde první řádek (resp. sloupec) odpovídá struktuře A až poslední řádek (resp. sloupec) odpovídá struktuře E. Hodnoty v matici vyjadřují podobnost struktur, odpovídající danému řádku a sloupci. Například hodnota na pozici $[3,4]$ odpovídá podobnosti struktury C a D, čili hodnota funkce $\lambda(C,D)$ podle definice4. Matice je symetrická a na diagonále má vždy jedničky. Metoda ClustalW dále vytvoří strom alignmentu, podle kterého se spojují jednotlivé profily. Strom alignmentu se tvoří metodou neighbor-joining, tzn. že se jednotlivé profily po jednom rozdělí do clusterů a v každém kroku se spojí vždy dva nejbližší clustery. Při spojení clusterů se vždy zmenší matice o jeden řádek a sloupec. Ze dvou řádků (resp. sloupců) se vytvoří jeden, do kterého se dosadí vždy větší ze dvou původních hodnot. Pro naši matici $\lambda(A,B,C,D,E)$ je strom vytvořený na obrázku 4.3. Strom alignmentu vyjadřuje evoluční vztahy mezi organismy, jejichž alignment provádíme (jde pouze o hrubý odhad). Více o evolučních vztazích a tvorbě fylogenetických stromů najde čtenář třeba v [16].

V této sekci jsme tedy probrali algoritmy, které generují pouze suboptimální řešení. Jejich paměťová a časová náročnost je však mnohem menší, něž algoritmy, které najdou optimální řešení.

4.6 Ořezání podle strukturálního konsenzu

V předchozích sekcích jsme podrobně popsali, jak najít strukturální alignment množiny sekundárních struktur. To znamená, že dokážeme najít vztahy mezi uzly jednotlivých stromů vstupních struktur, respektive dokážeme spočítat strukturální profil všech vstupních sekundárních struktur. Jak jsme již naznačili v sekci 4.1, potřebujeme vygenerovat zobecnění několika sekundárních struktur (tzv. strukturální konsenzus). Strukturální profil již téměř jakýmsi strukturálním konsenzem je. Každý uzel profilu má svoji váhu. Hodnota $w(u)$ značí kolik uzlů ze vstupních sekundárních struktur bylo alignmentem spojeno do uzlu u, čili jak moc je daná část sekundární struk-



Obrázek 4.3: Strom alignmentu

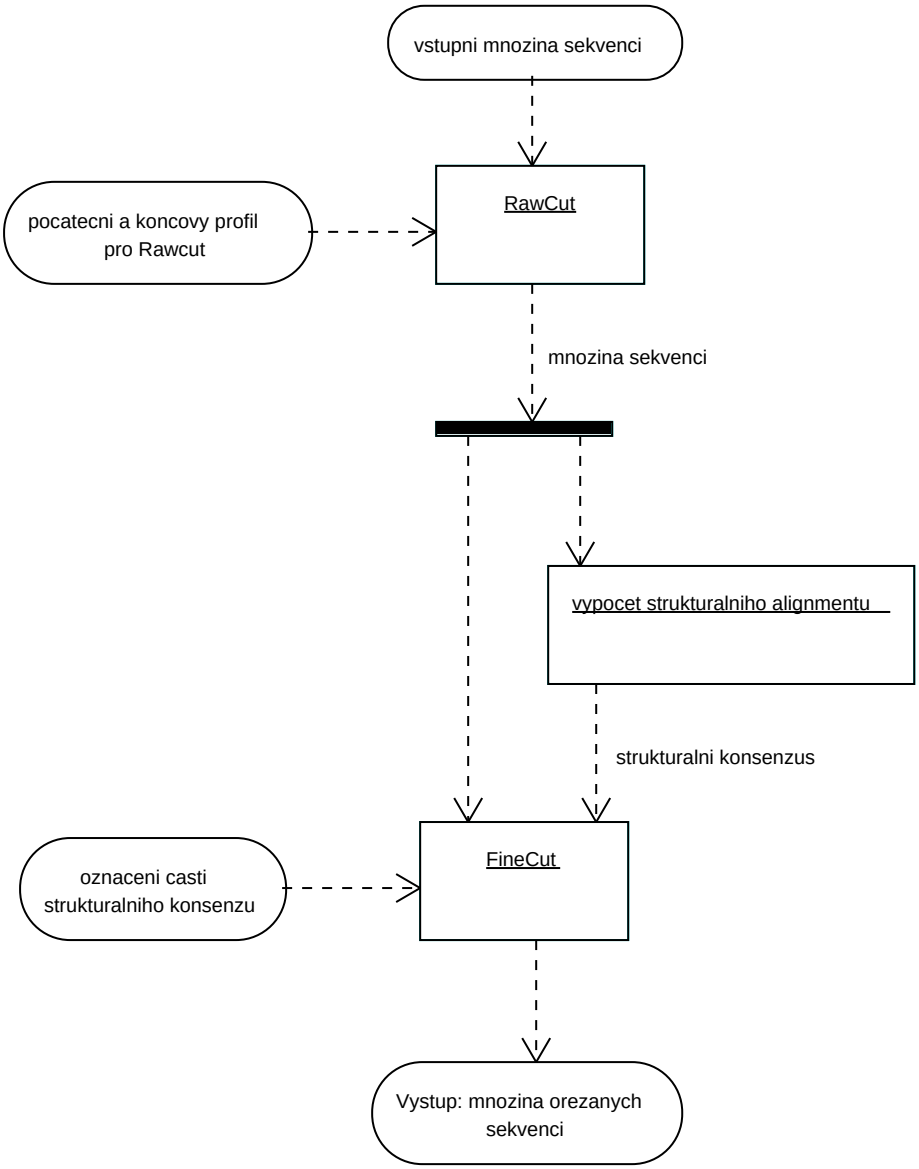
tury konzervovaná (neměnná) napříč všemi vstupními strukturami. Profil můžeme podle požadované obecnosti prořezat. Můžeme třeba vymazat všechny uzly, které mají váhu menší než parametr *min_weight* a tím získat strukturu obsahující pouze konzervované části.

Strukturální konsenzus jsme potřebovali v metodě ořezání FineCut. Zde se ve strukturálním konsenzu označí oblast, která je zajímavá a má být ve vstupních sekvencích vyseparována. Po označení je tato oblast zpětně namapována a promítnuta do jednotlivých vstupních struktur. Podle toho se na každé sekvenci provede ořezání.

4.7 Shrnutí

V této kapitole jsme popsali druhý stupeň dávkového ořezání sekvencí RNA. Jednalo se o metodu podmíněnou sekundární strukturou, kterou jsme nazvali FineCut. Metoda FineCut automaticky vyseparovává konkrétní podstruktury z množiny několika větších sekundárních struktur. Abychom mohli dostatečně popsat tuto metodu, museli jsme definovat mnoho dalších pojmů.

Mezi ně například patřil strukturální alignment sekvencí. Ten jsme zobecnili na strukturální alignment profilů a na něm vybudovali mnohonásobný strukturální alignment. Teprve pak jsme mohli popsat vlastní ořezání FineCut. Tok dat v celé proceduře ořezávání charakterizuje diagram na obrázku 4.4



Obrázek 4.4: Data flow diagram ořezání - Kulaté rámečky ze kterých vystupuje šipka jsou vstupy. Kulaté rámečky do kterých vstupuje šipka jsou výstupy. Obdélníky jsou operace. Přerušované čáry symbolizují tok dat.

Kapitola 5

Zobrazení sekundární struktury RNA

V sekci 1.4 jsme naznačily několik způsobů, jak zobrazit sekundární strukturu. Konkrétně pomocí závorkové notace, vlastním zapojením bází a bazických párů (obrázek 1.3), cirkulárním zobrazením (obrázek 1.5) nebo metodou dot plot (obrázek 1.6). Algoritmicky zajímavá úloha je nakreslit sekundární strukturu pomocí zapojení bází a bazických párů (jako na obrázku 1.3). Na vstupu je tedy jedna sekundární struktura[1]. Cílem je "hezky" nakreslit vstupní sekundární strukturu do roviny. "Hezky" znamená, že se budou co nejméně křížit hrany a taky že hrany nebudou příliš dlouhé. Pro řešení této úlohy existuje několik přístupů. Programy buď přímo biologovi umožňují editovat výsledné nakreslení nebo se snaží sekundární strukturu nakreslit automaticky. Automatizovaný přístup se dále dělí. Existují deterministické algoritmy (jako například naview [17]) nebo stochastické metody (například se na sekundární strukturu pohlíží jako na graf, který je potřeba nakreslit hezky do roviny a použije se pružinový model pro kresleni grafů [4]). Pro náš program RNAcut jsme potřebovali kreslit nejen sekundární struktury ale též strukturální profily[2]. Nechali jsme se tedy inspirovat programem RNA.jViz [5] a implementovali jsme do RNAcut pružinový model (force-based algorithm). Výhody tohoto algoritmu jsou, že na rozdíl od ostatních umí kreslit i složitější konstrukce, než jsou sekundární struktury (například strukturální konsenzus). Nevýhoda tohoto algoritmu je jeho

[1] V této kapitole budeme chápat sekundární strukturu jako graf. Jednotlivé báze jsou vrcholy grafu. Sousední báze v sekvenci a bazické páry tvoří hrany grafu.

[2] Popisované v kapitole 4.

nedeterminismus a relativně velká časová složitost. Algoritmus pružinového modelu počítá simulaci, kde na vrcholy grafu působí podle zapojení různé síly. Dva vrcholy se přitahují pokud mezi nimi existuje hrana a pokud jsou od sebe daleko. Navíc se všechny dvojice vrcholů grafu velmi malou silou odpuzují. Na začátku simulace jsou v klasickém force-based algorithm vrcholy náhodně umístěny do roviny. My jsme však podle vzoru RNA.jViz poskládali na začátku simulace vrcholy (jednotlivé báze) do kružnice v pořadí podle primární struktury[3], což se ukazuje být výhodnější než náhodné umístění. Nejtěžší při implementaci tohoto algoritmu bylo správně nastavit konstanty, charakterizující velikost sil působících na jednotlivé vrcholy. Funkce $f_p : \Re \rightarrow \Re$ charakterizuje velikosti síly, která působí na dva vrcholy spojené hranou. Velikost síly závisí na vzdálenosti l těchto vrcholů. Logicky se zdá, že aby algoritmus fungoval, nesmí být funkce f lineární, protože pokud by f lineární byla, způsobilo by to, že vrcholy které jsou daleko od sebe (třeba na začátku simulace) se k sobě začnou přibližovat příliš rychle a jejich vzájemný pohyb se zastaví až se přeletí, tím začne působit síla v opačném směru a celý efekt se opakuje. Ovšem nakonec se pro funkci f jako nejúčinnější ukázal předpis:

$$f(l) = w \cdot (l - l_{opt})/2 \tag{5.1}$$

Kde w je váha spojení, l_{opt} je parametr pro optimální vzdálenost (použili jsme konstantu 0.8). To že je funkce f lineární nejspíš nevadí protože v simulaci působí malá odpudivá síla mezi všemi ostatními uzly, která vyrovnává nechtěné zrychlení příliš vzdálených uzlů. K tomu je ale potřeba zvolit správný poloměr kružnice, do které se sekundární struktura vykreslí na začátku simulace. My jsme zvolili

$$R = \frac{n}{4\pi} \tag{5.2}$$

Samozřejmě lze všechny použité konstanty vynásobit libovolným reálným číslem a efekt simulace by měl být stejný. V rovnici 5.1 jsme použili parametr w (váha spojení). Tím se nám podařilo dát při vykreslování větší prioritu více konzervovaným částem sekundárních struktur ve strukturálním konsenzu. K tvarům silových funkcí a poloměru kružnice jsme dospěli experimentálně. Například pokud byly síly moc malé, obrázek se ustaloval velmi pomalu a hezké nakreslení trvalo příliš dlouho. Na druhou stranu pokud byly síly příliš velké, přibližující uzly se přeletěli. Příklad nakreslení je na obrázku 6.5.

[3]Jako na obrázku 1.5

Kapitola 6

Uživatelská dokumentace programu RNAcut

Součástí této práce je program RNAcut. Nejnovější verze má číslo 0.4. V této kapitole si rozebereme ovládání programu z pohledu uživatele. Funkčnost RNAcut lze rozdělit do následujících celků (seřazeno chronologicky, jak by se měly kroky za sebou provádět):

1. Načtení sekvencí RNA z různých zdrojů (vytvoření seznamu sekvencí).
2. Hrubé ořezání všech sekvencí v seznamu podle zadaných sekvenčních profilů (RawCut).
3. Vytvoření strukturálního konsezu.
4. Jemné ořezání všech sekvencí v seznamu – vybráním konkrétní části strukturálního konsenzu (FineCut).
5. Reprezentace výsledků.
6. Uložení/export ořezaných sekvencí.

Hlavní důraz byl kladen na jednoduchost ovládání (aby jej mohli používat biologové). Má tedy grafické uživatelské rozhraní.

6.1 Instalace

Aktuální verzi programu je možné stáhnout na webové adrese http://www.matfyz.cz/rnacut/. Verze 0.4 je nahrána na přiloženém

CD. Jsou připraveny dvě distribuce programu. Binární distribuce pro operační systém Win32[1] a platformě nezávislá distribuce zdrojových textů.

Binární distribuce nemá žádné prerekvizity. Stačí rozbalit zip archiv do libovolného adresáře a spustit soubor 'gui-application.exe'.

Distribuce zdrojových textů vyžaduje pro svůj běh nainstalovaný Python 2.5 a několik jeho knihoven (včetně jejich prerekvizit). Jsou uvedeny v následujícím seznamu prerekvizit:

- Biopython verze 1.43 a vyšší
- Piddle verze 1.0.15 a vyšší
- PIL verze 1.1.6 a vyšší
- Numeric verze 24.2 a vyšší
- wxPython verze 2.6 a vyšší

Dále je nutné do adresáře 'external' zkopírovat programy ClustalW a RNAfold přeložené pro vaši platformu. Tyto programy fungují jako externí moduly pro skládání sekundárních RNA struktur resp. pro aligning primárních struktur.

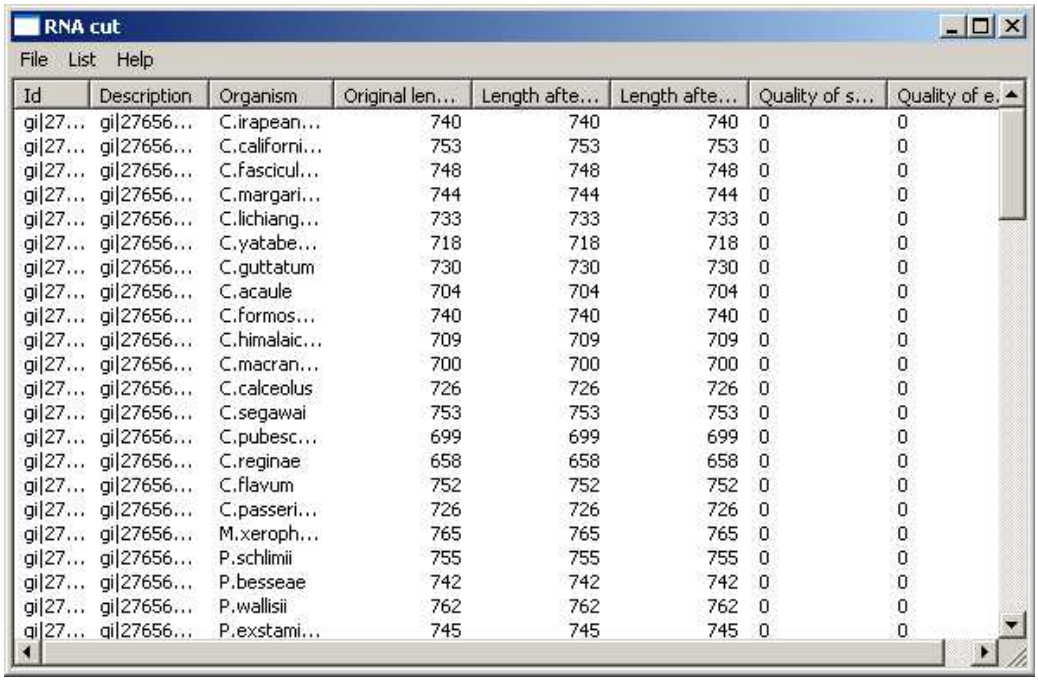
6.2 Spuštění aplikace

Po spuštění aplikace (gui-application.py nebo gui-application.exe) se otevře hlavní okno programu. Toto okno obsahuje menu a prázdný seznam sekvencí (viz též obrázek 6.1).

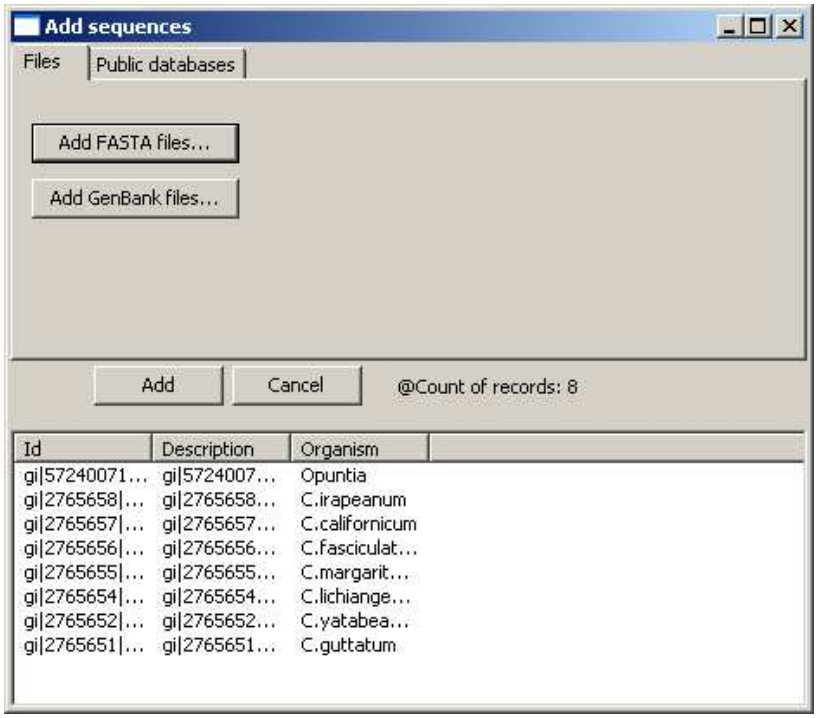
6.3 Přidávání sekvencí do seznamu v hlavním okně

Po spuštění programu se zobrazí hlavní okno programu s prázdným seznamem sekvencí. K zobrazení formuláře pro přidání sekvencí slouží klávesová zkratka '+' nebo menu 'List ⇒ Add sequences...'. Formulář 'Add sequences' má dvě záložky: 'Files' a 'Public databases', které určují typ zdroje přidávaných sekvencí.

[1] Win32 jsou operační systémy Windows 98, Windows 2000, Windows XP atd.



Obrázek 6.1: Hlavní okno programu s neprázdným seznamem sekvencí



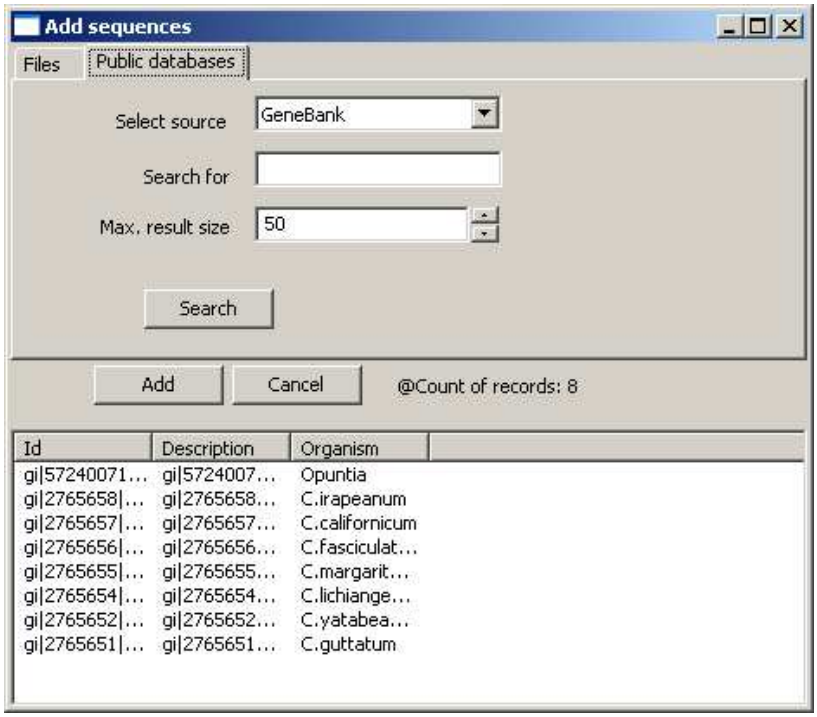
Obrázek 6.2: Formulář pro přidávání sekvencí - záložka 'Files'

Záložka 'Files' obsahuje tlačítka pro přidání FASTA a GeneBank souboru (viz obrázek 6.2). Po stisknutí jednoho z tlačítek se objeví standardní dialog pro výběr souboru příslušného typu. Po potvrzení výběru nějakého souboru se z něj načtou všechny sekvence a nahrají se do pomocného seznamu ve spodní části formuláře 'Add sequences'.

V druhé záložce 'Public databases' je možné získat sekvence z veřejných zdrojů na internetu (viz obrázek 6.3). Záložka obsahuje pole 'Select source', volbu databáze[2]. Pole 'Search for', do kterého se zadává výraz pro hledání v databázi. Poslední pole 'Max. result size' nastaví maximální počet sekvencí ve výsledku hledání. Tlačítkem 'Search' se spouští hledání a načítání sekvencí z internetu. Výsledek hledání se opět zobrazí v seznamu ve spodní části formuláře 'Add sequences'.

Nyní už seznam ve spodní části formuláře 'Add sequences' obsahuje

[2]Zatím je na výběr pouze databáze GeneBank, viz webové rozhraní této databáze [7], část Genome.



Obrázek 6.3: Formulář pro přidávání sekvencí - záložka 'Public databases'

Obrázek 6.4: Okno pro zadání hrubého ořezání

nějaké sekvence (buď výsledky hledání výrazu v databázi nebo sekvence načtené ze souboru). Tlačítko 'Add' přidá tento seznam do seznamu sekvencí v hlavním okně programu. Tlačítko 'Cancel' zavře formulář 'Add sequences' bez přidání jakékoliv sekvence.

6.4 Mazání sekvencí ze seznamu v hlavním okně

Nejprve označme sekvence v seznamu, které chceme smazat (vybráním myší se stisknutým shift nebo ctrl). Vlastní smazání se provede buď kontextovým menu (pravé tlačítko myši na seznamu) nebo klávesou Delete. Další možností, jak smazat sekvence, je menu 'List ⇒ Clear', které způsobí vymazání celého seznamu.

6.5 Hrubé ořezání sekvencí (RawCut)

Teorii k ořezání sekvencí metodou RawCut jsme probraly v kapitole 3. V hlavním okně vyberete menu 'List ⇒ Raw Cut'. Zobrazí se jednoduchý dialog, ve kterém si zvolte dva soubory typu '*.aln'. [3]

První z nich 'Profile cut on begin' slouží ke specifikaci nového začátku

[3]Soubory typu '*.aln' generuje programe ClustalX. Tyto soubory obsahují profil sekvencí. Na vstupu ClustalX se zada množina sekvencí. ClustalX spočítá alignment a vysledek uloží do souboru typu '*.aln'.

všech sekvencí a druhý 'Profile cut on end' naopak specifikuje nový konec sekvencí. Po vybrání souborů s profily a stisknutí tlačítka OK se provede RawCut. Zobrazí se progress bar a začne probíhat výpočet ořezání. Výpočet je možné kdykoliv přerušit tlačítkem Cancel. Po přerušení výpočtu nebude žádná sekvence ořezána. Pokud výpočet doběhne do konce, změní se v seznamu sekvencí v hlavním okně hodnoty sloupců 'Length after RawCut', 'Quality of start profile alignment' a 'Quality of end profile alignment'. Hodnoty těchto sloupců jsou popsány v sekci 6.9.

6.6 Zobrazení a výpočet strukturálního konsenzu

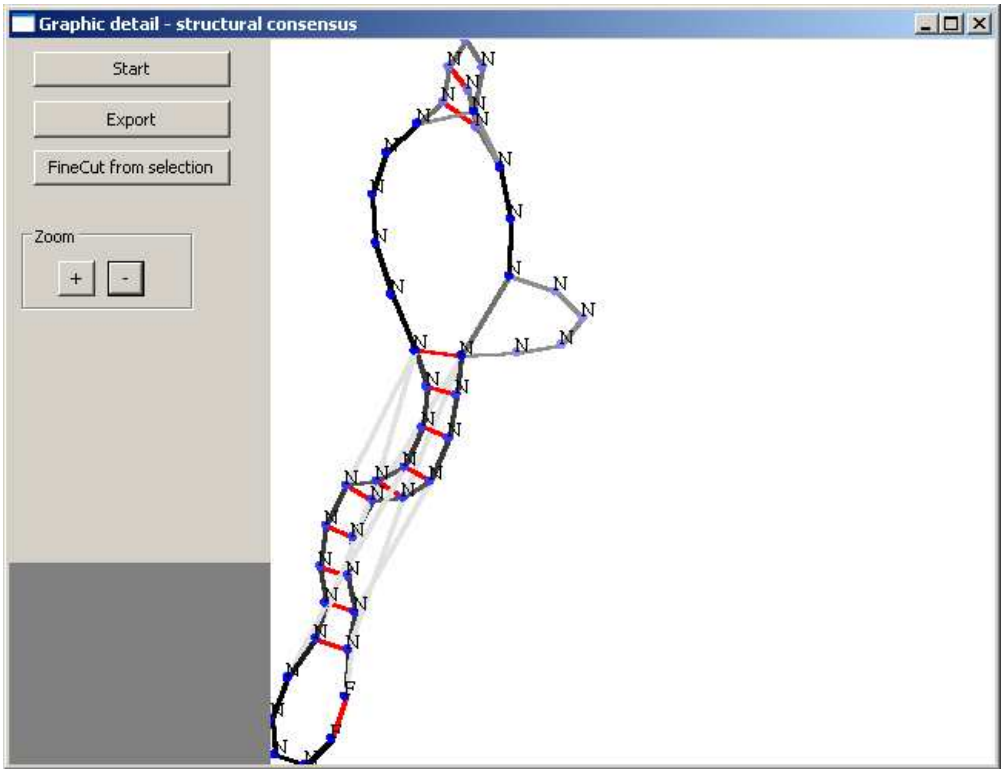
Strukturální konsenzus je jakýsi alignment sekundárních struktur různých sekvencí RNA. Více viz kapitola 4.

Jako vstup vezmeme struktury, které jsou automaticky poskládány pomocí programu RNAfold z aktuálních ořezaných sekvencí v hlavním okně. Pokud seznam sekvencí v hlavním okně aplikace obsahuje více jak jednu sekvenci, pak je možné spustit výpočet strukturálního konsenzu. Po kliknutí na menu 'List ⇒ Structural consensus' se aplikace zeptá uživatele, jestli chce pracovat se strukturálním konsenzem závislým na primární struktuře nebo ne. Po uživatelovi odpovědi se nastartuje výpočet a zobrazí se progress bar, ze kterého může uživatel vidět průběh výpočtu. Pokud uživatel nestiskne tlačítko Cancel před dokončení výpočtu, otevře se nové okno, kde je graficky znázorněná struktura konsenzu. Jednotlivé báze jsou nakresleny do kružnice jako modré body. Bazické páry jsou vyznačeny červenou čárou. Sousední báze v sekvenci jsou spojeny šedou čárou[4]. Tlačitkem 'start' se začne obrázek konsenzu skládat do přirozenější podoby[5]. Po stisknutí 'start' se tlačítko přejmenuje na 'pauze'. To pak zastaví skládání. Viz též obrázek 6.5. Pro lepší orientaci obsahuje struktura fiktivní bazický pár označený symboly 'F'. První báze fiktivního páru je umístěna v sekvenci před všechny báze všech sekvencí a druhá báze je umístěna za všechny báze všech sekvencí.

Po kliknutí na tlačítko 'Export...' se zobrazí standardní dialog pro výběr místa, kam se má exportovaný soubor uložit. Po potvrzení se tam taky uloží. Program aktuálně podporuje export grafiky ve formátech png a pdf. Dále je

[4]Čím častěji se jednotlivé části v sekvencích vyskytují, tím tmavěji šedou jsou nakresleny.

[5]Viz pružinový model pro kreslení strukturálního konsenzu, kapitola 5



Obrázek 6.5: Okno se strukturálním konsenzem

možné exportovat konsenzus jako textový soubor, který obsahuje alignment závorkové notace struktur všech sekvencí.

6.7 Jemné ořezání sekvencí (FineCut)

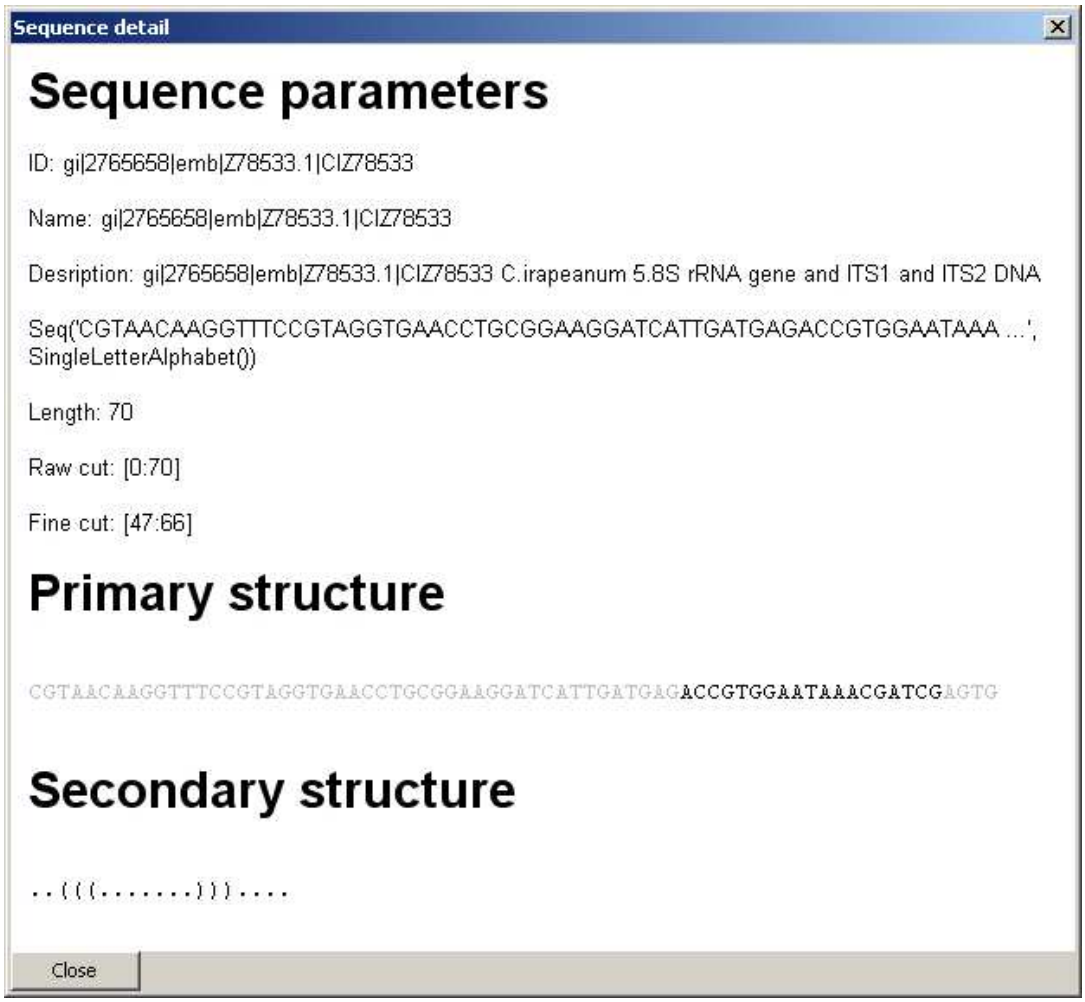
Další možnost ořezání sekvencí seznamu je po vygenerování strukturálního konsenzu. V okně s grafickým znázorněním konsenzu lze v obrázku táhnutím myši označit část konsenzu. Po stisknutí tlačítka 'FineCut from selection' se všechny sekvence hlavního seznamu ořežou tak, že výsledné sekvence odpovídají vybrané oblasti konsenzu.

6.8 Detail sekvence

V seznamu v hlavním okně aplikace můžeme chtít zobrazit detail pouze jedné sekvence. Tento detail se zobrazí po dvojitém kliknutí na jednu sekvenci nebo po vybrání položky 'Show sequence detail' v kontextovém menu. Okno detailu sekvence obsahuje pouze textové pole a tlačítko 'Close' (viz obrázek 6.6). Z textového pole je možné vykopírovavát části pomocí klávesové zkratky CTRL+C. V detailu jedné sekvence se o ní zobrazí všechny dostupné informace (záleží tedy na formátu a množství vstupních dat). Důležité jsou především informace o sekvenci jako její identifikátor, popis, původ, specifikace organismu. Dále můžeme zjistit délku původní sekvence. Relativně vůči původní sekvenci jsou určeny intervaly jednotlivých druhů ořezání (více v sekcích 6.5 a 6.7). Formát intervalů ořezání je '[A,B]', kde A je číslo určující počátek intervalu a B číslo určující konec intervalu (čísla udávají vzdálenost od počátku původní sekvence). Pokud sekvence nebyla ořezána, pak A je nula a B je délka sekvence. Sekce detailu 'Primary structure' obsahuje primární vlastní sekvenci, ve které jsou barevně znázorněné intervaly jednotlivých ořezání. V poslední části 'Secondary structure' se nachází závorková notace sekundární struktury, která je ořezána podle nejstriktnějšího ořezání.

6.9 Hlavní seznam sekvencí

Hlavní seznam sekvencí zobrazuje aktuálně zpracovávané sekvence v několika sloupcích (viz obrázek 6.1). Následuje popis jednotlivých slouců.



Obrázek 6.6: Okno s detailem sekvence

Id Jednoznačné identifikátory sekvence v jednotlivých databázích.

Description Popis sekvence načtený z její anotace.

Organism Organismus, jehož část genomu je uložena v sekvenci. Tato informace je načtena z anotace sekvence.

Original length Délka původní sekvence.

Length after RawCut Délka zkrácené sekvence (metodou RawCut).

Length after FineCut Délka zkrácené sekvence (metodou FineCut).

Quality of start profile alignment Kvalita alignmentu počátečního profilu v metodě RawCut. Je to číslo od nuly do sta. Vyjadřuje kolik procent v alignmentu tvoří nemezery (počáteční a koncové mezery se nepočítají). Pro hodnotu 100 tedy alignment neobsahuje žádné mezery. Specialně hodnota 0 znamená, že se RawCut jestě neprováděl nebo že se počáteční profil naalignmentoval za koncový profil. Pokud je hodnota v tomto sloupci příliš nízká nebo nula, pak nejspíš daná sekvence neobsahuje část, kterou chce uživatel vyseparovávat. Takovéto sekvence by měl uživatel po provedení RawCut odstranit nebo zpracovat ručně.

Quality of end profile alignment Kvalita alignmentu koncového profilu (podobně jako předchozí sloupec).

Po kliknutí levým tlačítkem myši na záhlaví některého ze sloupců seznamu se seznam setřídí podle abecedy nebo podle velikosti hodnot tohoto sloupce.

6.10 Export ořezaných sekvencí

Uložení ořezaných sekvencí do souboru je závěrečný krok. Seznam v hlavním okně, který je předzpracován pomocí hrubého a jemného ořezání můžete nyní exportovat z aplikace. V menu hlavního okna vyberte nabídku 'List ⇒ Export'. Zobrazí se standardní dialog pro vybrání souboru pro uložení. Po potvrzení se sekvence uloží do vybraného souboru. Zatím je pro export podporován pouze formát fasta. Do anotací jednotlivých sekvencí se přidá poznámka, že byly ořezáný. Uloží se vždy sekvence zkrácené nejstriktnějším ořezáním.

6.11 Příklad použití aplikace RNAcut

Zde uvádíme příklad nejtypičtějšího použití aplikace RNAcut (samozdřejmě se v konkrétních použitích budou lišit použité organismy nebo místa v genomu). K tomuto příkladu je vytvořen v distribucích adresář s testovacími soubory. Umístění tohoto adresáře je vůči kořenovému adresáři uživatelovi distribuce následující: 'testing_data/example-Opuntia-rpl16/'. Tento adresář obsahuje soubory se sekvencema rostliny opuntia konkrétně její gen rpl16. Soubor 'downloaded.fasta' obsahuje výsledky, které v současné době vrací databáze genebank na dotaz 'Opuntia AND rpl16'. Soubor 'raw-cut-begin.aln' resp. 'raw-cut-end.aln' obsahuje profil sekvence, který definuje počátek resp. konec pro ořezání metodou raw cut. Následuje bodový popis postupu.

1. **Spusťte aplikaci.** Objeví se okno s prázdným seznamem sekvencí.

2. **Načtěte sekvence RNA.** Do seznamu v hlavním okně přidejte sekvence buď ze souboru 'downloaded.fasta' nebo dotazem do GeneBank 'Opuntia AND rpl16' (Obě varianty by meli být totožné). Viz sekce 6.3.

3. **Hrubé ořezání RawCut.** Jako počáteční a koncový profil vyberte soubory 'raw-cut-begin.aln' a 'raw-cut-end.aln'. Spusťte RawCut. Po skončení výpočtu si zobrazte detail některé sekvence. Sekvence by měla být ořezaná (je to vyznačené tmavší barvou v primární struktuře). Viz sekce 6.5.

4. **Vytvoření strukturálního konsezu a FineCut.** Nechte si vygenerovat strukturální konsenzus nezávislý na primární struktuře, označte v něm část struktury a spusťte FineCut. Znovu si zobrazte detail některé sekvence. Nyní byste měli vidět barevně odlišené dvě úrovně ořezání (RawCut a FineCut).

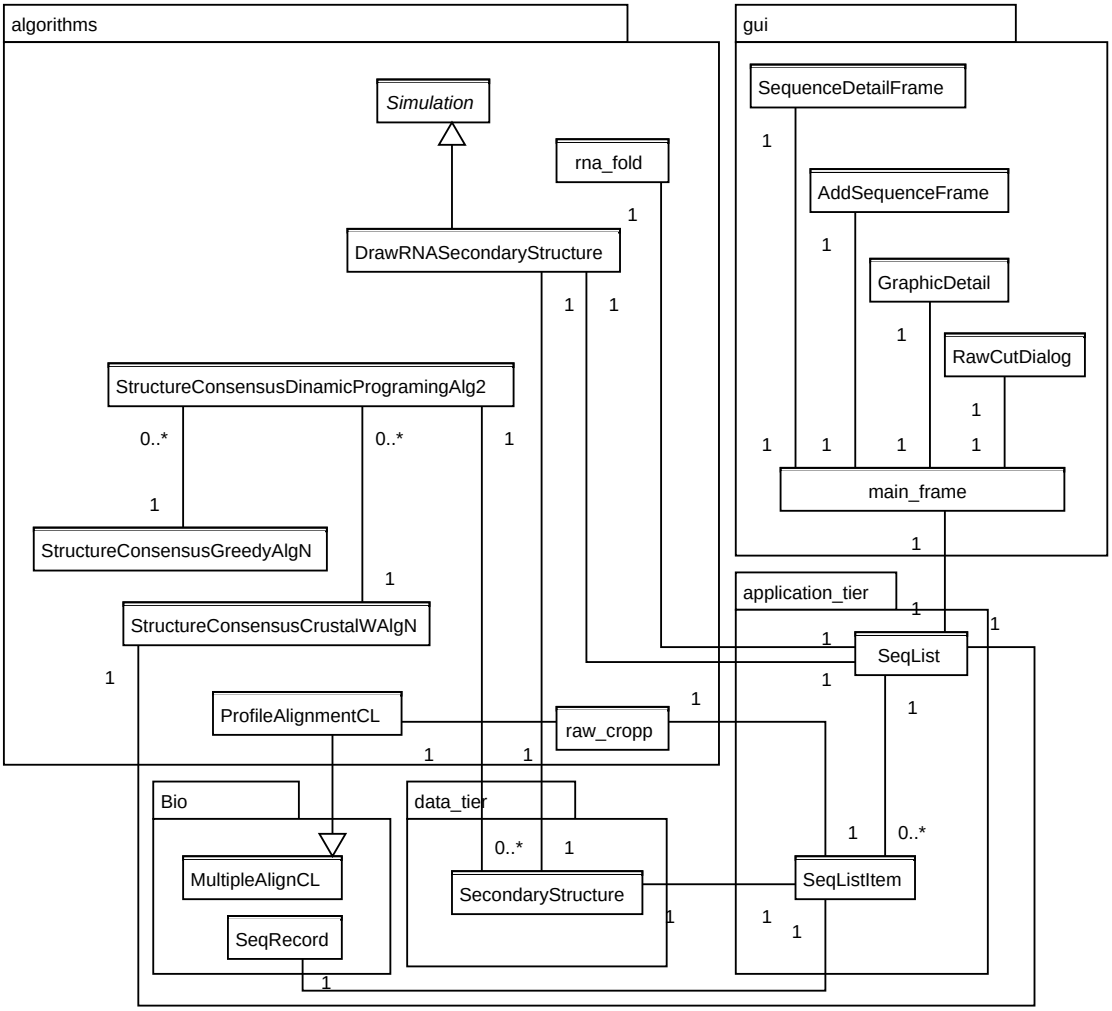
5. **Export ořezaných sekvencí.** Uložte si výsledek do souboru. Viz sekce 6.10

Uživatel může některé kroky přeskakovat, ale neměl by se v uvedeném pořadí vracet zpět. Některé kroky zpět jsou zakázány. Uživatel dostane hlášku: 'It is not a correct order of operation!'. Menu 'List ⇒ Clear' smaže celý seznam sekvencí a vrátí uživatele za krok 1.

Kapitola 7

Poznámky ke zdrojovému kódu

RNAcut je naprogramovaný v jazyku Python, kód je tedy snadno přenositelný na jiné platformy. Větší důraz než na přenositelnost byl kladen na jednoduchost ovládání, proto bylo použito grafické uživatelské rozhraní wxPython. Jazyk uživatelského rozhraní i komentářů v kódu je angličtina. Obrázek 7.1 obsahuje schéma celého programu RNAcut. Ukazuje jednotlivé třídy a závislosti mezi nimi. Celá aplikace se dělí do několika vrstev, respektive balíčků (konkrétně Bio, data_tier, algorithms, application_tier a gui). Bio reprezentuje knihovnu BioPython. Ve schematu je zobrazena pouze využívaná funkcionalita, BioPython samozřejmě umí mnohem víc. My využíváme sekvenční alignment a datovou strukturu sekvence (včetně načtení ze souboru nebo databaze a následné uložení do souboru). Ostatní balíčky jsou již naše. Balíček data_tier obsahuje definici datové struktury pro sekundární strukturu RNA. Balíček algorithms obsahuje naimplementované jednotlivé algoritmy, popisované v této práci. Konkrétně jde o RawCut (třída raw_cropp), mnohonásobný strukturální alignment (třída StructureConsensusCrustalWAlgN) a algoritmus pro nakreslení sekundární struktury (třída DrawRNASecondaryStructure). Balíček application_tier obsahuje aplikační logiku, která je nezávislá na uživatelském rozhraní. Poslední balíček gui je grafické uživatelské rozhraní, které používa knihovnu wxPython. Celá aplikace je programována v duchu třívrstvé architektury. Balíčky data_tier a Bio tvoří první vrstvu, která přistupuje k datům. Druhá vrstva (balíčky algorithms a application_tier) má na starosti vlastní logické fungování aplikace. Poslední vrstva (balíček gui) je vrstva uživatelského rozhraní. Díky použité třívrstvé architektuře bude moct být aplikace jednoduše rozšířena.



Obrázek 7.1: Class diagram programu RNAcut

7.1 Kompilace

K vývojovému prostředí Python existuje modul py2exe pro konverzi pythonovských skriptů do spustitelné aplikace pro MS Windows. Při instalaci této aplikace již není potřeba na uživatelův počítač instalovat prerekvizity, včetně pythonu. Při konverzi funkčního RNAcut se však vyskytla celá řada problémů. Většinou se jednalo o problémy zapříčiněné použitím některých knihoven, které nejsou zcela připraveny na kompilaci do *.exe souboru. Konkrétně nešlo kompilovat aplikaci obsahující zároveň balíčky Numpy a Numeric (prerekvizita knihovny BioPython). Dále byl problém v podmíněném importování konfigurací databáze v knihovně BioPython v balíčku GeneBank. Do výsledné zkompilované aplikace je potřeba přidat adresář s fonty 'pilfonts'. Ten leží v knihovně piddle. Problémy s kompilací se daly vyřešit lehčím zásahem do kódů těchto problematických knihoven, za pomoci návodů z internetových diskuzí. Přesto aplikace občas generuje do log souboru hlášky typu DeprecationWarning. Jinak funguje exe aplikace stejně jako aplikace ze zdrojových kódů. Konfigurační soubor kompilace 'setup.py' pro py2exe jsme umístily do kořenového adresáře v distribuci zdrojových kódů.

Kapitola 8

Příbuzné projekty

V této kapitole uvedeme přehled bioinformatických aplikací, které jsou svojí funkcionalitou příbuzné programu RNAcut. Jedná se o programy pro sekvenční alignment (ClustalX) a zobrazení sekundárních struktur (jViz.Rna, Rnamovies a RNAplot).

8.1 ClustalX a ClustalW

Aplikace ClustalX [10] je oblíbená pomůcka pro počítání klasického sekvenčního alignmentu. Má dvě uživatelské rozhraní. Jedno rozhraní pro příkazovou řádku (ClustalW) a jedno rozhraní okenní aplikace (ClustalX). Pomocí externího volání ClustalW je v knihovně BioPython implementován alignment sekvencí. Program umí spočítat sekvenční alignment pro množinu sekvencí nebo pro dva profily. Výsledky ukládá do souboru s příponou '*.aln'. Tyto soubory používáme jako vstup do programu RNAcut, konkrétně v části RawCut.

8.2 Vienna RNA package

Vienna [9] je soubor aplikací pro práci se sekundárními strukturami. Obsahuje například program pro predikci sekundárních struktur RNA (RNAfold) a program pro kreslení sekundárních struktur (RNAplot).

8.3 jViz.Rna

jViz.Rna [8] je aplikace napsaná v Javě. Umí kreslit sekundární struktury pomocí pružinového modelu. Právě tímto programem jsme se nechali inspirovat při návrhu programu RNAcut, konkrétně jsme tento algoritmus použili pro kreslení strukturálního konsenzu. Bohužel není možné využít tuto aplikaci jako externí modul v RNAcut, protože umí kreslit pouze jednu samostatnou sekundární strukturu. My jsme ale potřebovali nakreslit celý konsenzus několika sekundárních struktur. Celý algoritmus jsme tedy museli v přizpůsobené podobě implementovat sami.

8.4 Rnamovies

Rnamovies [11] je aplikace napsaná v Javě. Umí vytvořit animaci přechodu mezi různými sekundárními strukturami jedné sekvence. Z animací je vidět které bazické páry při přechodu mezi strukturami vznikají a které naopak zanikají. Bohužel Rnamovies neumí tvořit přechody mezi různými strukturami různých sekvencí, takže je tento program pro náš projekt nepoužitelný. Možná se tímto způsobem zobrazení několika struktur budeme inspirovat při dalším rozšiřování RNAcutu.

Kapitola 9

Závěr

Vytvořili jsme program RNAcut pro dávkové zpracování sekvencí RNA a popsali jsme použité metody a algoritmy. Stanovili jsme postupy pro automatické hledání podobností mezi sekvencemi RNA, nejen na základě sekvence bází tvořící RNA, ale též na základě jejich sekundárních struktur. Problém hledání podobností a vztahů mezi částma jednotlivých sekundárních struktur jsme popsali jako strukturální alignment. Problém je komplikovaný a řešení musí umět pracovat s biologickými daty, které jsou zatížené chybami. Optimální řešní mnohonásobného strukturálního alignmentu má příliš vysokou časovou a prostorovou složitost. Navrhli jsme tedy suboptimální řešení, jehož časová a prostorová složitost je rozumná. Program byl uvolněn pro použití biologům. Budeme očekávat reakce na jejichž základě bedeme implementovat další vylepšení programu. Doufáme, že naše aplikace usnadní biologům zpracování sekvencí RNA.

Literatura

[1] Neil C. Jones, Pavel A. Pevzner: *An Introduction to Bioinformatics Algorithms*, The MIT Press, 2004.

[2] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, L. Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster: *Fast folding and comparison of rna secondary structures*, Momathefte für Chemie, 125(1):167–188 (1994).

[3] Matthias Hochsmann: *The Tree Alignment Model: Algorithms, Implementations and Applications for the Analysis of RNA Secondary Structures*, PhD thesis, Technische Fakultät der Universität Bielefeld (2005).

[4] P. Eades: *A heuristic for graph drawing*, Congressus Nutnerantiunt, 42, 149-160, 1984

[5] Kay C. Wiese, Edward Glen: *jViz.Rna - An Interactive Graphical Tool for Visualizing RNA Secondary Structure Including Pseudoknots*, Proceedings of the 19th International Symposium on Computer Based Medical Systems (IEEE/CBMS-2006), June 2006, pp. 659-664.

[6] Helena Štorchová, Matthew S. Olson: *The architecture of the chloroplast trnH-psbA non-coding region in angiosperms*, American Journal of Botany. 2007

[7] *NCBI public Genome database*,
`http://www.ncbi.nlm.nih.gov/sites/entrez?db=Genome`

[8] *jViz.Rna program homepage*,
`http://jviz.research.iat.sfu.ca/`

[9] *Vienna RNA package homepage*,
http://www.tbi.univie.ac.at/~ivo/RNA/

[10] *ClustalX program homepage*,
http://bips.u-strasbg.fr/fr/Documentation/ClustalX/

[11] *Rnamovies program homepage*,
http://bibiserv.techfak.uni-bielefeld.de/rnamovies/

[12] E. Kočárek: *Genetika*, nakl.: Scientia, 1. vyd., 2005

[13] *University of Texas Medical Branch - Cell Biology Graduate Program Lecture*,
http://cellbio.utmb.edu/cellbio/

[14] L. M. C. Meireles, T. Akutsu : *A Gibbs Sampling Approach to Detection of Tree Motifs*, Genome Informatics 16(1): 34-43 (2005)

[15] S. Gupta, J. Kececioglu, A. Schäffer: *Improving the practical space and time efficiency of the shortest-paths Approach to sum-of-pairs multiple sequence alignment*, J Comput. Biol 1995; 2:459-472

[16] Chinh Hoang: *Methods for constructing phylogenetic tree*,
http://darwin.nmsu.edu/~molb470/fall2005/projects/hoang/methods.html

[17] R. E. Bruccoleri, G. Heinrich: *An improved algorithm for nucleic acid secondary structure display* , CABIOS, 4:, 167–173. (1988)

[18] *NCBI FASTA format description*,
http://www.ncbi.nlm.nih.gov/blast/fasta.shtml

[19] J. D. Thompson, D. G. Higgins, T. J. Gibson: *CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice*, Nucleic Acids Research, 1994, Vol. 22, No. 22 4673-4680