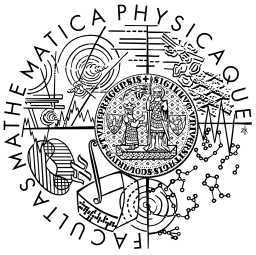


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Bc. Rastislav Galvánek

Predikce terciární struktury RNA s využitím více vzorů

Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce:	RNDr. David Hoksza, Ph.D
Studijní program:	Informatika
Studijní obor:	IUI

Praha 2019

Prohlašuji, že jsem tuto diplomovou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Praze dne ..... Podpis autora

Výpočtové zdroje boli poskytnuté Ministerstvom školstva, mládeže a športu Českej Republiky v projekte CESNET (Project No. LM2015042) a CERIT-Scientific Cloud (Project No. LM2015085) spadajúce do programu Projects of Large Research, Development and Innovations Infrastructures.

Název práce: Predikce terciární struktury RNA s využitím více vzorů

Autor: Bc. Rastislav Galvánek

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: RNDr. David Hoksza, Ph.D, Katedra softwarového inženýrství

Abstrakt: V tejto práci ďalej rozvíjame algoritmus homológnej predikcie terciárnej štruktúry RNA, ktorý bol navrhnutý a naimplementovaný v rámci mojej bakalárskej práce. Venujeme sa väčšej automatizácii implementácie algoritmu a jeho jednoduchšiemu použitiu tak, aby bol schopný predikovať RNA štruktúru molekuly bez manuálnych zásahov, len na základe sekvencie cieľovej štruktúry. Algoritmus ďalej rozširujeme o homológnu predikciu sekundárnej štruktúry a možnosť použitia viacerých template štruktúr, čo by malo viesť k zmenšeniu prehľadávaného priestoru pri predikcii nekonzervovaných úsekov predikovanej štruktúry, a tým zvýšiť celkovú presnosť predikcie.

Klíčová slova: predikcia RNA template komparatívne modelovnie

Title: RNA tertiary structure prediction using multiple templates

Author: Bc. Rastislav Galvánek

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. David Hoksza, Ph.D, Department of Software Engineering

Abstract: In this thesis we will further develop the algorithm of homologous prediction of tertiary RNA structure. The algorithm was originally created and implemented in my bachelor thesis. We will focus on further automatization of algorithm implementation and we are going to make it easier to use. The user will be able to predict tertiary structure of RNA based only on target structure sequence. The algorithm will be also extended to use multiple template structures for prediction and it will be able to firstly predict the secondary structure of the target molecule. Both of these modifications should lead to more precise prediction by restricting the search space and reducing the size of unconserved regions of the predicted structure.

Keywords: prediction RNA template comparative modeling

Obsah

Úvod

Štruktúry RNA hrajú významnú úlohu napríklad v syntéze proteínov, alebo v prenose a uchovávaní genetickej informácie organizmov. Znalosť ich 3D štruktúry je kľúčova pri porozumení ich funkcie. Na rozdiel od získania sekvencie RNA, čo je pomerne jednoduché a v dnešnej dobe pomerne lacno zvládnuteľné, je získanie terciárnej štruktúry RNA stále pomalý a nákladný proces. Z toho vyplýva, že rozdiel medzi počtom známych RNA sekvencií a množstvom známych terciárnych RNA štruktúr sa neustále zväčšuje, čo vytvára priestor pre oblasť výpočetnej predikcie terciárnych štruktúr.

Algoritmy predikcie môžeme rozdeliť na dve skupiny podľa prístupu k predikcii. v Prvej skupine sú algoritmy ab initio, ktoré predikujú štruktúru priamo zo sekvencie na základe biologických, chemických a fyzikálnych vlastností. Druhá skupina je tvorená homológnymi algoritmami, ktoré využívajú dôsledky evolúcie, kedy platí, že molekuly s podobnými úsekmi v sekvenciách majú aj podobnú štruktúru. Takéto úseky nazývame konzervované.

V bakalárskej práci sme predstavili metódu homológneho modelovania RNA - TROOPER, ktorú budeme v tejto práci ďalej rozvíjať. Upravíme implementáciu tak, aby bola samotná predikcia užívateľsky jednoduchšia vďaka väčšej miere automatizácie. Následne naše výsledky porovnáme s nástrojom ModeRNA založenom tiež na princípe komparatívneho modelovania RNA štruktúr. Ďalej sa budeme zaoberať rozšírením algortimu o dva moduly. Prvý modul rozširuje predikciu terciárnej štruktúry o krok napredikovania sekundárnej štruktúry, ktorá je následne použitá pri de novo predikcií nekonzervovaných úsekov, čím by sa mal zmenšiť prehľadávaný priestor pri modelovaní. Druhý modul dopĺňa predikciu o možnosť použitia separátnej template štruktúry pre každý dlhý nekonzervovaný úsek v štruktúre získanej z primárnej template štruktúry.

1. RNA štruktúra

V tejto práci sa venujeme automatickej predikcii priestorovej štruktúry ribonukleovej kyseliny, preto sa budeme v prvej kapitole zaoberať jej významom z biologického hľadiska. Ďalej preberieme možnosti, akým spôsobom a aké podrobné informácie o štruktúre RNA dokáže súčasná veda získať experimentálnym spôsobom a aká je motivácia pre počítačovú predikciu RNA. Nakoniec uvedieme možnosti, ako RNA štruktúru reprezentovať vo formáte textových súborov a aké informácie o štruktúre jednotlivé typy súborov uchovávajú.

1.1 RNA

Ribonukleová kyselina slúži na prenos a uchovávanie genetickej informácie vo všetkých živých organizmoch alebo moduláciu génovej expresie. Najznámejšia je jej úloha v Centrálnej dogme molekulárnej biológie Crick (1970), kde slúži pri syntéze proteínov z DNA na prenášanie genetickej informácie.

RNA je rovnako ako DNA tvorená štyrmi typmi nukleotidov (báz), pričom jedna báza je tvorená práve jednou molekulou. Typy nukleotidov sú adenín (A), guanín (G), cytozín (C) a uracil (U). Narozdiel od RNA sa v DNA namiesto uracilu vyskytuje báza tymín (T). Jednotlivé nukleotidy sú chemicky naviazané na cukor - ribózu, ktorý ich spája do vlákna (v prípade DNA sa jedná o deoxyribózu) , pričom nukleotidy sú vo vlákne sekvenčne usporiadané. Dĺžka vlákna môže byť v závislosti na type RNA od niekoľkých jednotiek až po tisíce nukleotidov. Pre DNA následne platí, že sa vodíkovými väzbami spájajú dva komplementárne reťazce do špirály, čo čiastočne určuje pravidelný tvar molekuly v priestore. RNA sa však vyskytuje hlavne v jednovláknovej forme, pričom sa vlákno spája vodíkovými väzbami samo so sebou, a to na rôznych miestach, čo prináša veľkú variabilitu v tvare molekuly. Platí, že tromi vodíkovými väzbami sa navájom viažu nukleotidy cytozín a guanín, a dvomi väzbami nukleotidy adenín a uracil (prípadne tymín v DNA).

1.2 Druhy a funkcie RNA

Okrem prenosu genetickej informácie pri syntéze proteínov, ktorý pozostáva z replikácie DNA, transkripcie DNA do RNA a nakoniec translácie z RNA do samotnej primárnej štruktúry proteínu (pričom sa využívajú rôzne typy RNA), zastáva RNA aj iné funkcie. Slúži napríklad na uchovávanie genetickej informácie niektorých jednoduchých organizmov ako sú vírusy. Tie môžu na uchovanie genetickej informácie používať jednovláknovú RNA, dvojvláknovú RNA a v prípade retrovírusov špeciálny typ RNA, ktorý je schopný prepisovať genetickú informáciu z RNA do DNA procesom reverznej transkriptázy a vložiť tak svoju genetickú informáciu do genomu napadnutej bunky Alberts B (2002). Medzi tieto vírusy patrí napríklad známy vírus HIV. Krupovic a kol. (2018)

Prehľad niektorých typov RNA:

- proteín kódujúca (2%)

  - mediátorová RNA (mRNA)
- proteín nekódujúca (98%)
  - ribozomálna RNA (rRNA)
  - prenosová RNA (tRNA)
  - funkcionálna RNA (fRNA)
  - mikro RNA (miRNA)
  - malá interferujúca (small interfering) RNA (siRNA)
  - jadrová (nuclear) RNA (snRNA)
  - jadierková (nucleolar) RNA (snoRNA)
  - vírusová RNA (vRNA)
  - dlhá nekódujúca RNA (lncRNA)
  - ďalšie...

Messenger RNA (mRNA) vzniká pri prepise (transkripcii) DNA v jadre bunky. Najprv je vytvorená pre-mRNA, ktorá obsahuje aj nekódujúce úseky. V ďalšom kroku sa z nej ešte v jadre bunky procesom nazývanzým splicing odstránia intróny (nekódujúce úseky) za pomoci snRNA. snRNA rozpoznáva sekvenciu báz AGGU označujúcu prechod medzi intrónom a exónom. Následne mRNA putuje cez póry v jadrovej membráne von do cytoplazmy, kde sa naviaže na ribozóm. Alberts B (2002)

Ribozóm obsahuje ribozomálnu RNA (rRNA), ktorá sa zúčastňuje translácie (prekladu) mRNA do primárnej skevencie kódovanej bielkoviny. Okrem toho je to typicky najčastejšie sa vyskytujúca RNA v bunke, pričom jej dĺžka môže byť až niekoľko tisíc nukleotidov.

Prenosová tRNA sa nachádza v cytoplazme bunky a jej funkcia spočíva v dopravení správnej aminokyseliny do procesu translácie. Každá aminokyselina má svojú vlastnú špecifickú tRNA, na ktorú je naviazaná aminoacyl-tRNA syntetázou, a následne dopravená na miesto syntézy.

Niektoré typy RNA plnia regulačnú funkciu. Napríklad mikro RNA (miRNA) zabraňuje procesu translácie mRNA tým, že sa na ňu naviaže a zabráni jej spojeniu s ribozómom. snoRNA zas hrá úlohu pri modifikácii ostatných typov RNA - hlavne rRNA, tRNA a snRNA.

Sekvencie lncRNA mávajú často úlohu v transkripčnej regulácii. Jeden zo známych zástupcov je gén XIST, ktorý sa uplatňuje pri procese inaktivácie chromozómu X. Rinn (2012)

1.3 Reprezentácia a práca s RNA

Fyzicky je RNA v bunke vlastne len mnoho atómov vodíka, kyslíka, uhlíka, dusíka a fosforu usporiadaných v priestore vďaka chemickým a fyzikálnym interakciám a vlastnostiam atómov. Pre to, aby sme ich mohli spracovávať algoritmicky, potrebujeme vhodnú reprezentáciu štruktúry. Typ reprezentácie závisí od toho, aké informácie o štruktúre chceme mať k dispozícii a takisto aké informácie sme schopní získať. Principiálne môžeme rozdeliť reprezentáciu RNA štruktúr na nasledujúce štyri úrovne Obrázok 1.1:

- Primárna
- Sekundárna
- Terciárna
- Kvartérna

Primárna štruktúra je najviac zjednodušená reprezentácia RNA daná sekvenčným usporiadaním nukleotidov vo vlákne. Určuje len poradie a typ jednotlivých nukleotidov v RNA štruktúre a ďalej ju budeme v práci tiež označovať ako sekvenciu. V počítači ju reprezentujeme ako textový súbor typu fasta, ktorý má v prvom riadku identifikáciu štruktúry (názov, chain) a v ďalších riadkoch sú to len písmena A, G, C, U určujúce presné poradie nukleotidov. V prípade, že typ nukleotidu na niektorej pozícii je neznámy, používame písmeno X alebo N. Primárna sekvencia RNA takisto slúži ako jeden zo vstupov pre náš prediktor a definuje sekvenciu štruktúry, ktorú cheme získať.

Sekundárna štruktúra popisuje vodíkové väzby medzi jednotlivými nukleotidmi vo vlákne RNA. Dva nukleoidy, ktoré sú spojené vodíkovou väzbou, označujeme ako base pair. Vďaka spájaniu jednotlivých nukleotidov vieme v sekundárnej štruktúre pozorovať rôzne podštruktúry, ako napríklad helix, loop, pseudoknot, hairpin loop, internal loop, branch loop, stem a ďalšie Obrázok 1.2. Sekundárnu štruktúru budeme v tejto práci používať na pomoc pri predikcii terciárnej štruktúry, nakoľko nám dáva informáciu o nukleotidoch, ktoré sú spojené vodíkovou väzbou, a teda sa nachádzajú blízko pri sebe. Sekundárnu štruktúru molekuly budeme reprezentovať ako textový súbor, kde bodka značí, že nukleotid danej pozície nie je viazaný žiadnou väzbou, base pair spojený vodíkovou väzbou je značený ako valídne uzátvorkovanie jednoduchými zátvorkami a pseudoknot býva reprezentovaný hranatými zátvorkami. Sekundárnu štruktúru je možné nakresliť do dvojdimenzionálneho euklidovského priestoru a toto nakreslenie graficky znázorňuje prepojenia nukleotidov vodíkovými väzbami.

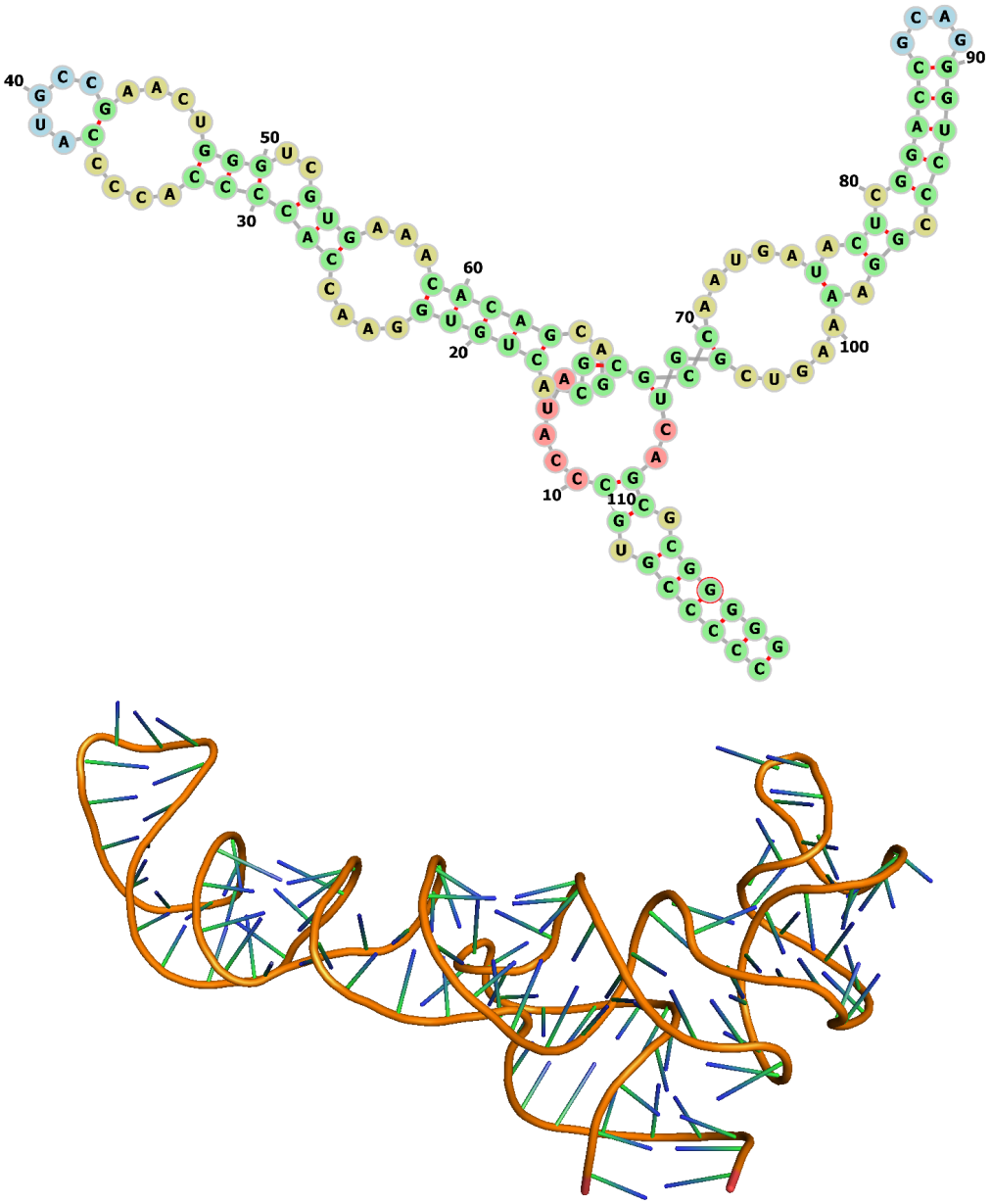
Terciárna štruktúra koordinátmi popisuje presné rozloženie jednotlivých atómov v trojdimenzionálnom euklidovskom priestore. V našej práci je hlavným cieľom tieto koordináty určiť za predpokladu znalosti primárnej sekvencie a terciárnych štruktúr ďalších RNA molekúl, ktoré sa pokúšame pri predikcii použiť ako vzory.

Kvartérna štruktúra RNA popisuje vzťahy medzi celými molekulami RNA - napríklad interakcie medzi jednotlivými molekulami RNA v ribozómoch Noller (1984) a taktiež vzťahy medzi RNA a molekulami bielkovín.

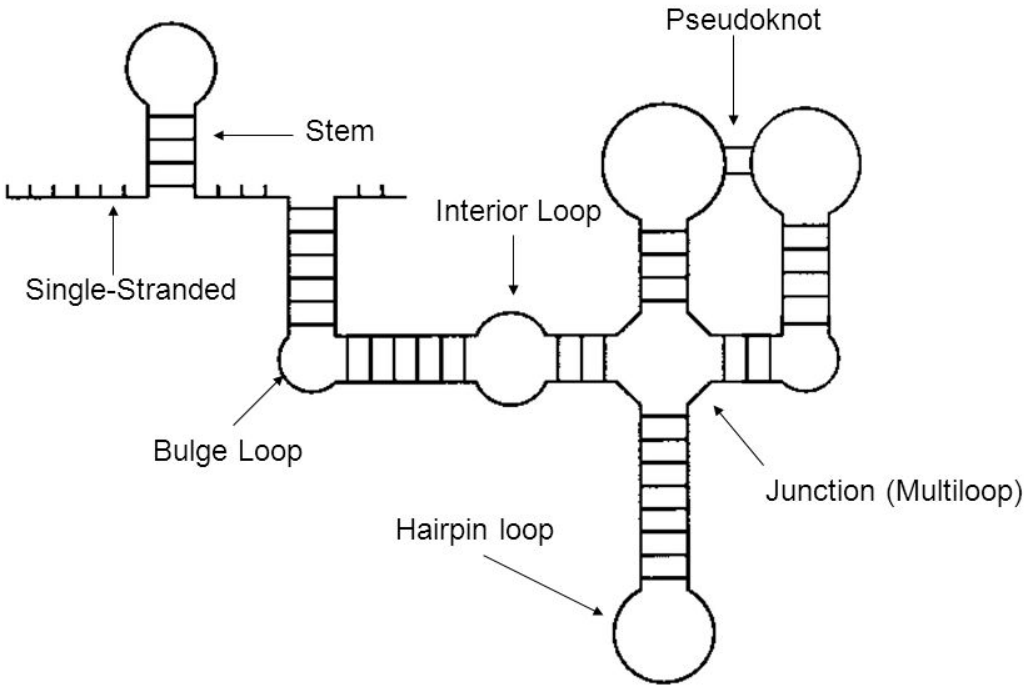
1.4 Význam a získavanie terciárnej štruktúry

Štruktúra RNA priamo súvisí s funkciou, ktorú vykonáva. Tvar štruktúry Obrázok 1.3 určuje, ktoré enzýmy sa na ňu dokážu pripájať, prípadne ju modifikovať, a s ktorými bielkovinami a nukleovými kyselinami sa dokáže viazať. Bolo preukázané, že väčšina častí RNA štruktúry, ktoré sú schopné sa viazať s inými molekulami, nie sú súčasťou žiadneho base pair-u, a teda sú v sekundárnej štruktúre označené ako nespárované Schudoma C. (2010). Zmena terciárnej štruktúry, ktorá vedie ku strate pôvodnej funkcie molekuly, sa nazýva denaturácia.

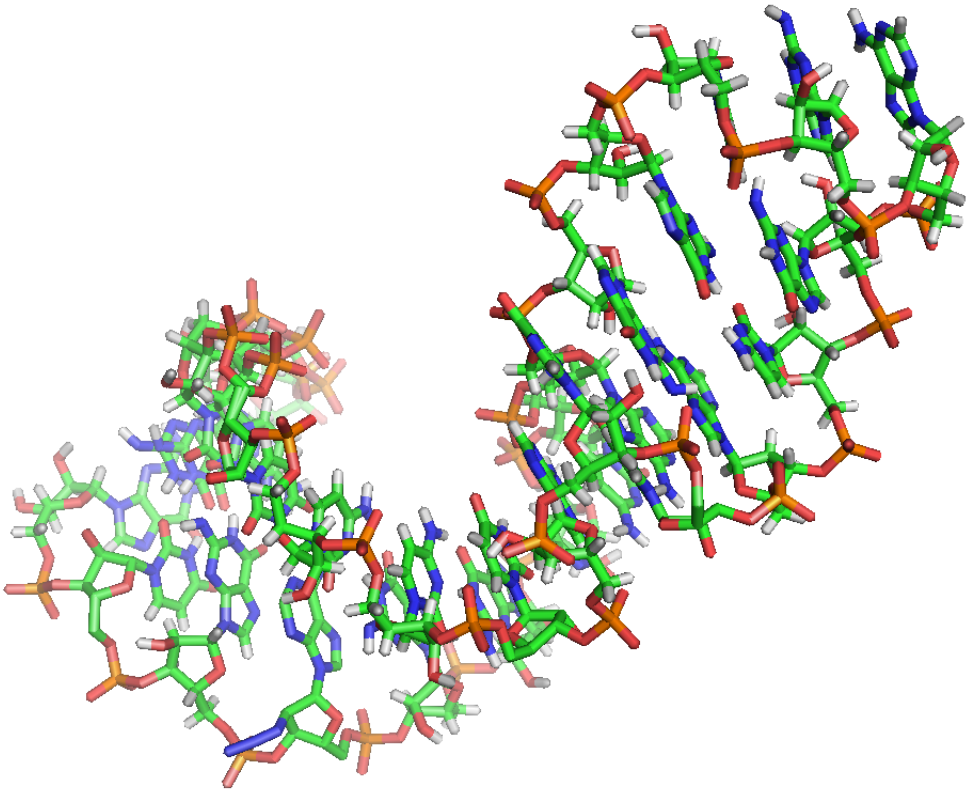
```
>3FWO:B|PDBID|CHAIN|SEQUENCE
CCCCCGUGCCCAUAGCACUGUGGAACCACCCCACCCCAUGCCGAACUGGGUCGUGAAACA
CAGCAGCGCCAAUGAUACUCGGACCGCAGGGUCCCGGAAAAGUCGGUCAGCGCGGGGG
```



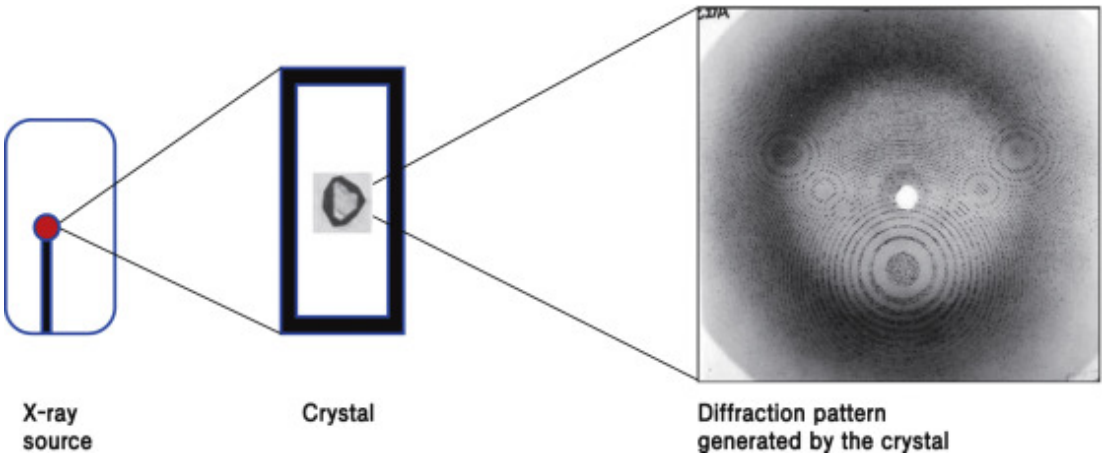
Obrázek 1.1: Na obrázku vidíme zhora-dole tri reprezentácie molekuly 1FWO_B: primárna sekvenciu, sekundárna štruktúra a terciárna štruktúra.



Obrázek 1.2: Príklad nakreslenia sekundárnej štruktúre RNA. Eddy (2004)



Obrázek 1.3: Príklad terciárnej štruktúry RNA nachádzajúcej sa v baktérii Escherichia coli.



Obrázek 1.4: Princíp rentgenovej kryštalografie. Ryu (2017)

Bolo vyvinutých viacero experimentálnych metód, pomocou ktorých môžeme získať terciárnu štruktúru RNA Felden (2007):

- metódy s vysokou presnosťou
  - X-ray crystallography
  - Cryo-electron microscopy
  - Nuclear Magnetic Resonance (NMR) spectroscopy
- metódy s nižšou presnosťou
  - Mass spectrometry
  - Chemical probing
  - Thermal denaturation
  - RNA engineering

X-ray crystallography (rentgenová kryštalografia) funguje principiálne tak, že sa molekula najprv zkryštalyzuje a následne sa nasvieti rentgenovým lúčom. Z kryštálu je lúč odrazený a pritom rozdelený na viacero lúčov. Zmeraním uhlov odrazu a intenzity odrazených lúčov je následne možné určiť pozície jednotlivých atómov v molekule. Momentálne je to jedna z najpoužívanejších metód získavania mnohých makromolekulárnych štruktúr. Rozlíšenie získanej štruktúry sa pohybuje okolo 2.0 Å. Obrázok 1.4

Cryo-electron microscopy metóda využíva zmrazenie molekuly v substancii, ktorá je následne pozorovaná elektrónovým mikroskopom. Princíp tejto metódy je známy približne od roku 1970, ale až donedávna nebolo možné pomocou nej získať tak presné vysledky ako pomocou rentgenovej kryštalografie. Na druhej strane, dĺžka skúmanej štruktúry nie je pri tejto metóde tak limitujúcim faktorom. V roku 2017 bola udelená Nobelova cena za chémiu J. Dubochetovi, J. Frankovi a R. Hendersonovi za vyvinutie metódy, ktorou sa dá získať atómová štruktúra molekuly s vysokým rozlíšením.

Metóda Nuclear Magnetic Resonance je založená na pôsobení statického magnetického poľa na jadrá atómov v molekule. Je vhodná hlavne na získavanie kratších štruktúr.

Experimentálne prístupy sa od seba navzájom líšia presnosťou výsledku, dĺžkou štruktúry, s ktorou sú schopné pracovať, ale ich hlavnou nevýhodou je, že sú stále časovo náročné a drahé. Pretože získavanie primárnej štruktúry RNA a proteínov je oveľa ľahšia úloha, začali byť skúmané aj možnosti, ako predikovať sekundárnu a terciárnu štruktúru algoritmicky, čomu sa budeme v našej práci venovať.

2. Metódy výpočetnej predikcie

Cieľom výpočetnej predikcie RNA štruktúry je dokázať algoritmicky modelovať terciárnu alebo sekundárnu štruktúru na základe znalosti primárnej sekvencie RNA molekuly. Pri takejto predikcii je dôležité, aby sme dostali čo najpresnejší výsledok v porovnaní s experimentálnymi metódami, a zároveň aby výpočtové nároky a čas boli výrazne nižšie, než v prípade experimentálnej rezolúcie štruktúry. Aby malo zmysel sa pokúšať o predikciu štruktúry zo sekvencie potrebujeme vedieť, že terciárna a teda aj sekundárna štruktúra je do veľkej miery jednoznačne určená štruktúrou primárnou.

Túto otázku môžeme zodpovedať vďaka znalostiam zo skladania (foldingu) proteínov, ktorých výskumu sa venovalo viacej úsilia. Platí, že skladanie bielkovín a RNA prebieha veľmi podobne, a preto poznatky o štruktúrach a sekvenciách bielkovín môžeme použiť aj pri RNA. Moore (1999)

Existujú dve hlavné pozorovania, ktoré nám umožnujú štruktúry makromolekúl modelovať Jenny Gu (2009):

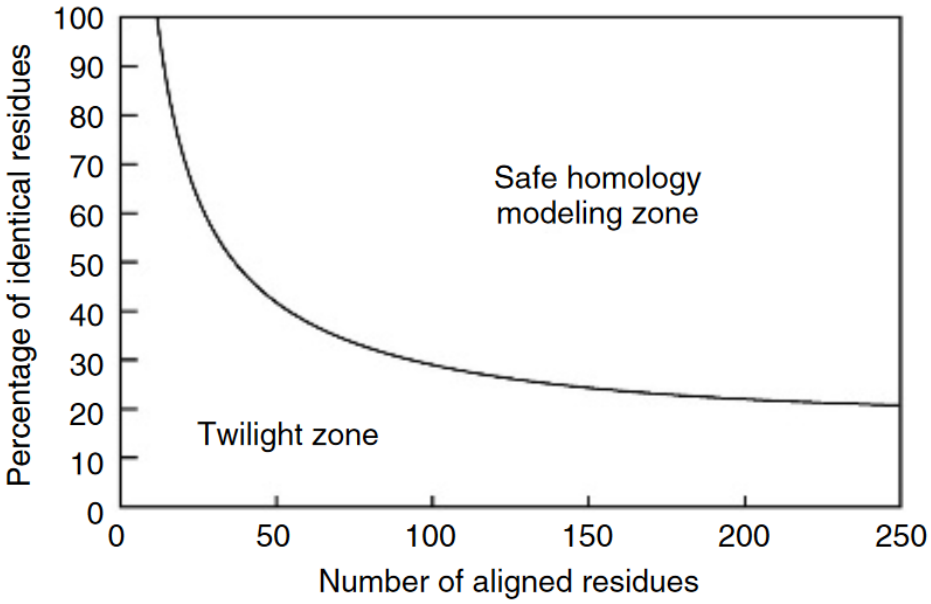
- Štruktúra proteínu je unikátne určená sekvenciou aminokyselín.
- Štruktúra sa zachováva aj pri určitých zmenách v sekvencii, a teda platí, že napriek odlišnosti v sekvenciách sú štruktúry veľmi podobné. Je to spôsobené tým, že počas evolúcie štruktúra stále plnila podobnú úlohu, a preto sa jej tvar nemenil aj napriek mutáciám sekvencie. Vďaka rozširujúcej sa databáze makromolekúl (Protein Data Bank) boli empiricky získané vzťahy definujúce, aká by mala byť podobnosť rovnako dlhých sekvencií, aby sme mohli predpokladať, že aj ich štruktúry sú podobné. Tento vzťah zobrazuje obrázok Obrázok 2.1.

2.1 Ab initio predikcia

Pri ab initio predikcii štruktúry vychádzame iba z primárnej sekvencie a chemicko-fyzikálnych vlastností, vďaka ktorým sa v reálnom svete štruktúra skladá do stabilného tvaru. Algoritmus postupne vytvára kandidátske štruktúry tak, že sa snaží minimalizovať funkciu predstavujúcu voľnú energiu (energia, ktorá je ľahko dostupná v systéme). Algoritmus následne z takto vygenerovaných kandidátov musí vybrať najprirodzenejšiu štruktúru. Najväčším problémom tohoto prístupu je mnoho lokálnych miním vo funkcii predstavujúcej voľnú energiu, a preto aj výpočetná zložitosť.

Tieto komplikácie sa dajú čiastočne riešiť viacerými spôsobmi. Jedna cesta je zvýšiť výpočetný výkon - použitie superpočítača, alebo distribuovať výpočet na mnoho výpočetných staníc. Ďalšia je pokus o zmenšenie vyhľadávacieho priestoru a efektívnejšie vyhľadávať kandidátske štruktúry. Jedna metóda je označovaná ako coarse-grained reprezentácie, kde nie sú reprezentované všetky atómy. Využívajú sa taktiež heuristické a pravdepodobnostné metódy na zmenšienie prehľadávaného priestoru.

Stále však platí, že ab initio predikcia je pre dlhšie štruktúry nepoužiteľná. Napriek tomu, že súšasný state-of-the art umožňuje predikovať krátke štruktúry



Obrázek 2.1: Vzťah dĺžky štruktúr a percentuálneho pomeru identických residuí v sekvenciách určujúce predpoklad, že štruktúry takýchto sekvencií sú podobné. Jenny Gu (2009)

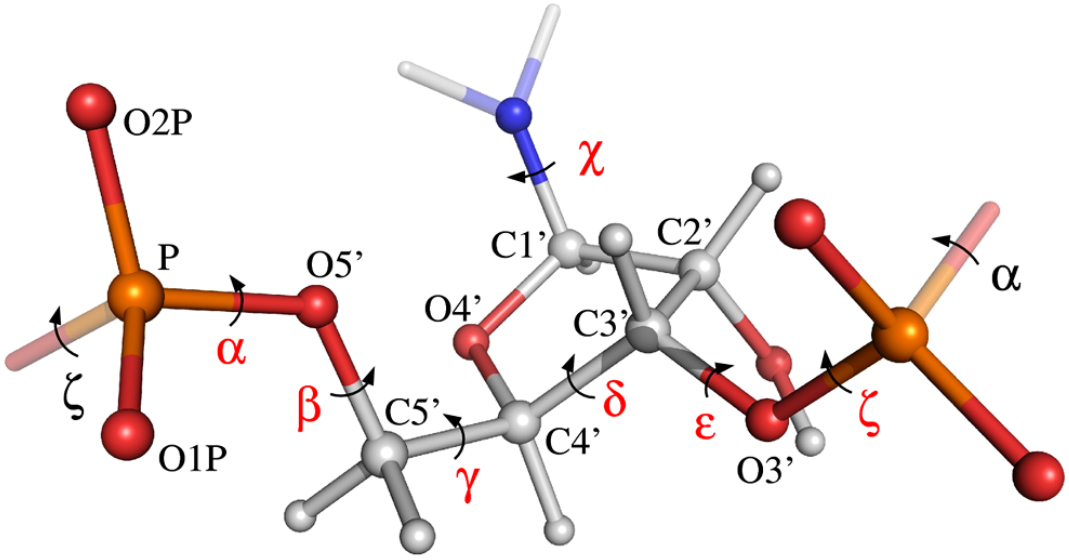
celkom presne, s rastúcou dĺžkou sekvencie neúmerne rastie výpočetná náročnosť. Ako príklad uvedieme pokus predikovať štruktúru dlhú 112 nukleotidov, pričom výsledná štruktúra sa líšila od experimentálne získanej len minimálne, výpočet však stál viac ako 100 000 hodín CPU. Qian a kol. (2007)

2.2 Knowledge based de novo predikcia

Princíp tohoto typu predikcie je veľmi podobný ako ten v ab initio metóde, ale namiesto samplovania možných usporiadaní atómov používa knižnicu krátkych úsekov štruktúry (väčšinou dĺžky 2-5 nukleotidov). Algoritmus následne vytvára kandidátske štruktúry tým, že kombinuje jednotlivé krátke fragmenty štruktúr z knižnice do kandidátskych štruktúr, a takisto minimalizuje voľnú energiu modelu. Výhodou je hlavne zrýchlené generovanie kandidátskych štruktúr oproti ab iniio predikcii. Aj tak je však predikovanie dlhých štruktúr príliš pomalé. Mnohé nástroje preto umožňujú vložiť sekundárnu štruktúru predikovanej sekvencie, a tak zmenšiť prehľadávaný priestor.

Ďalší spôsob ako znížiť prehľadávaný priestor, je použitie šikovnejšej reprezentácie štruktúry. V prípade, že atómy reprezentujeme súradnicami v trojdimenzionálnom priestore, ich síce vieme dobre zobraziť, ale takáto reprezentácia má 3*počet_átomov stupňov voľnosti. Výhodnejšie je štruktúru reprezentovať napríklad pomocou reprezentácie uhlov medzi nukleotidmi. Obrázok 2.2

Nástroj FARFAR Das R. (2010), ktorý používame v našej predikcii, patrí tiež medzi knowledge based modelovacie metódy.



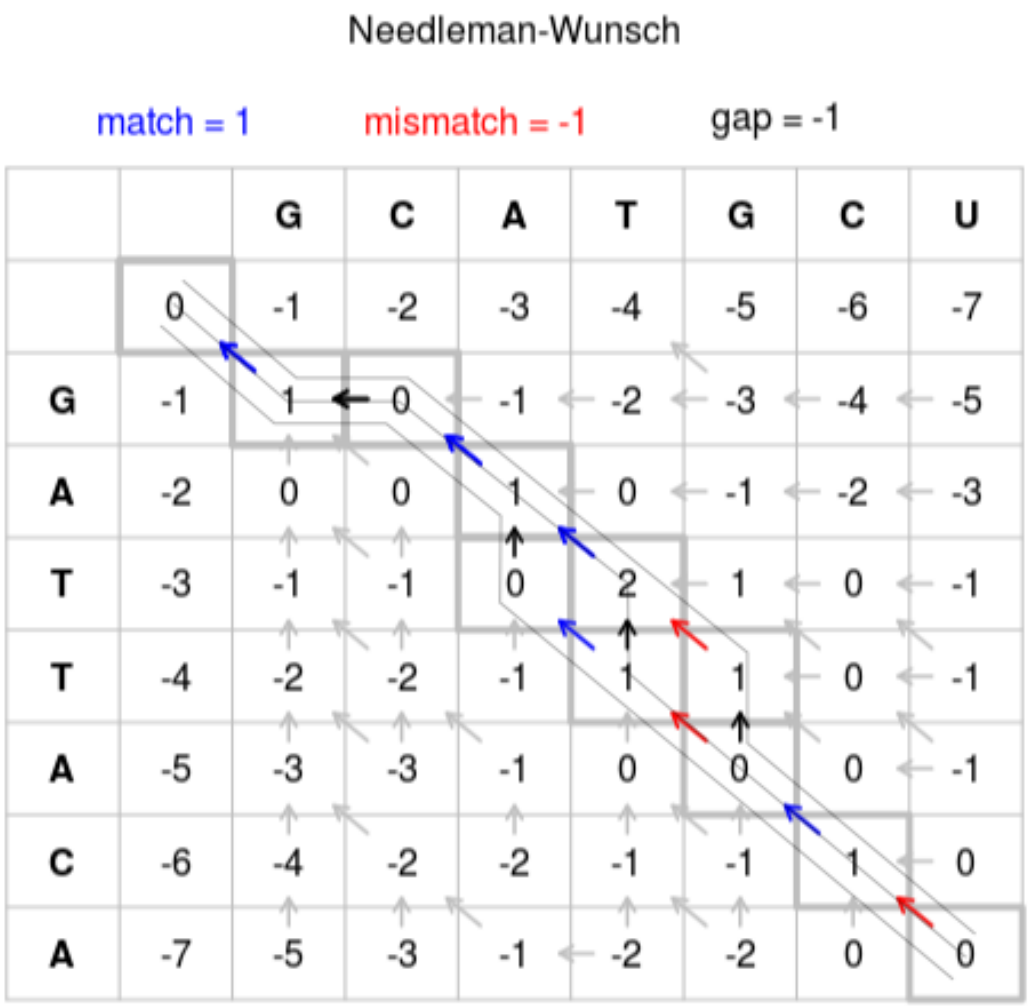
Obrázek 2.2: Reprezentácia RNA fragmentu pomocou siedmych uhlov. Frellsen a kol. (2009)

2.3 Alignment sekvencií

Zarovnanie dvoch sekvencií slúži na získanie informácie o tom, ktoré časti sekvencii sú nejako evolučne, štrukturálne, alebo funkčne príbuzné. Existuje viacero druhov algoritmov zarovnania - napríklad jednoduchý dot plot vhodný na jednoduchú vizualizáciu zarovnania Gibbs (1970), kombinácia heuristických metód a dynamického programovania ako algoritmy FASTA a BLAST určené na čo najrýchlejšie porovnanie sekvencie s rozsiahlou databázou ďalších sekvencií, alebo metódy počítajúce najlepšie zarovnanie určené skórovacím systémom za pomoci dynamického programovania.

V tejto práci budeme využívať semiglobálne zarovnanie pomocou modifikovaného algoritmu Needleman–Wunsch Needleman S. B. (1970) implementovaného v programe EMBOSS Rice P. a A. (2000). Ako vstup algoritmus dostáva dve sekvencie dĺžiek m a n, ktoré chceme zarovnať, a hodnoty parametrov gap open (penalizácia v skóre za otvorenie medzery v zarovnaní) a gap extend (penalizácia v skóre za predĺženie medzery v zarovnaní). Algoritmus následne za pomoci dynamického programovania Obrázok 2.3 vypočíta zarovnanie s najnižším skóre v čase aj priestore $O(nm)$. Výstupom algoritmu sú optimálne zarovnané sekvencie a príslušné najlepšie dosiahnuteľné skóre zarovnania. V zarovnaní na určitej pozícii môžu nastať tri prípady, a to zarovnanie dvoch rovnakých reziduí (match), zarovnanie dvoch odlišných reziduí (mismatch), a nakoniec zarovnanie rezidua na medzeru (gap) vloženú do druhej sekvencie. Nami používaná implementácia algoritmu nepenalizuje za medzery v zarovnaní nachádzajúce sa na začiatku alebo na konci zarovnania, preto je možné ňou zmysluplne zarovnať krátku štruktúru na časť oveľa dlhšej štruktúry. Zarovnanie s takýmito vlastnosťami zaraďujeme medzi hybridné zarovnávacie metódy a označujeme ako semiglobálne Brudno (2003).

Okrem globálneho poznáme aj presné lokálne zarovnanie vyriešené algoritmom Smith–Waterman Smith T. F. (1981). Tento algoritmus pracuje taktiež na princípe dynamického programovania a vyhľadáva zarovnanie dvoch subsekven-



Obrázek 2.3: Needleman–Wunsch algorithm (2014) Wikipedia dostupné na `https://en.wikipedia.org/wiki/Needleman\T1\textendashWunsch_algorithm` 27.05.2019. Príklad jedného z troch najlepších zarovnaní dvoch sekvencií:
GCATG-CU
G-ATTACA

cií s najlepím skóre. Používa sa na nájdenie podobných regiónov medzi dvomi sekvenciami.

2.4 Homológne modelovanie

Tvrdenie zo začiatku kapitoly, ktoré hovorí, že štruktúra si zachováva podobný tvar aj napriek tomu, že jej sekvencia postupne mutuje, umožňuje zmysluplne predikovať štruktúru na základe vzoru. Homológne modelovanie používa na modelovanie neznámej štruktúry zo sekvencie ešte jednu vzorovú molekulu (template), ktorej štruktúra je známa, teda získaná za pomoci nejakej experimentálnej metódy. Predikovanú molekulu zvykneme nazývať cieľ (target).

Homológne modelovanie pozostáva z nasledujúcich krokov:

1. Určenie vhodnej template štruktúry.
2. Zarovnanie sekvencií a získanie konzervovaných úsekov.
3. Dopredikovanie nekonzervovaných úsekov.
4. Optimalizácia napredikovaného modelu.

Prvým krokom je teda určenie vhodnej template štruktúry, pomocou ktorej budeme predikovať target štruktúru. Typicky budeme hľadať molekulu s čo najpodobnejšou sekvenciou, aby sa neskôr minimalizoval počet a veľkosť nekonzervovaných úsekov.

Druhým krokom je globálne zarovnanie oboch sekvencií a získanie konzervovaných úsekov, teda úsekov, v ktorých by mali byť obe štruktúry veľmi podobné. Konzervované úseky môžu byť po nejakých úpravách prenesené do cieľovej štruktúry. Z princípu vyplýva, že čím podobnejšie sekvencie budú mať target a template štruktúry, tým viac konzervovaných úsekov bude existovať a tým jednoduchšia a presnejšia by mala predikcia byť.

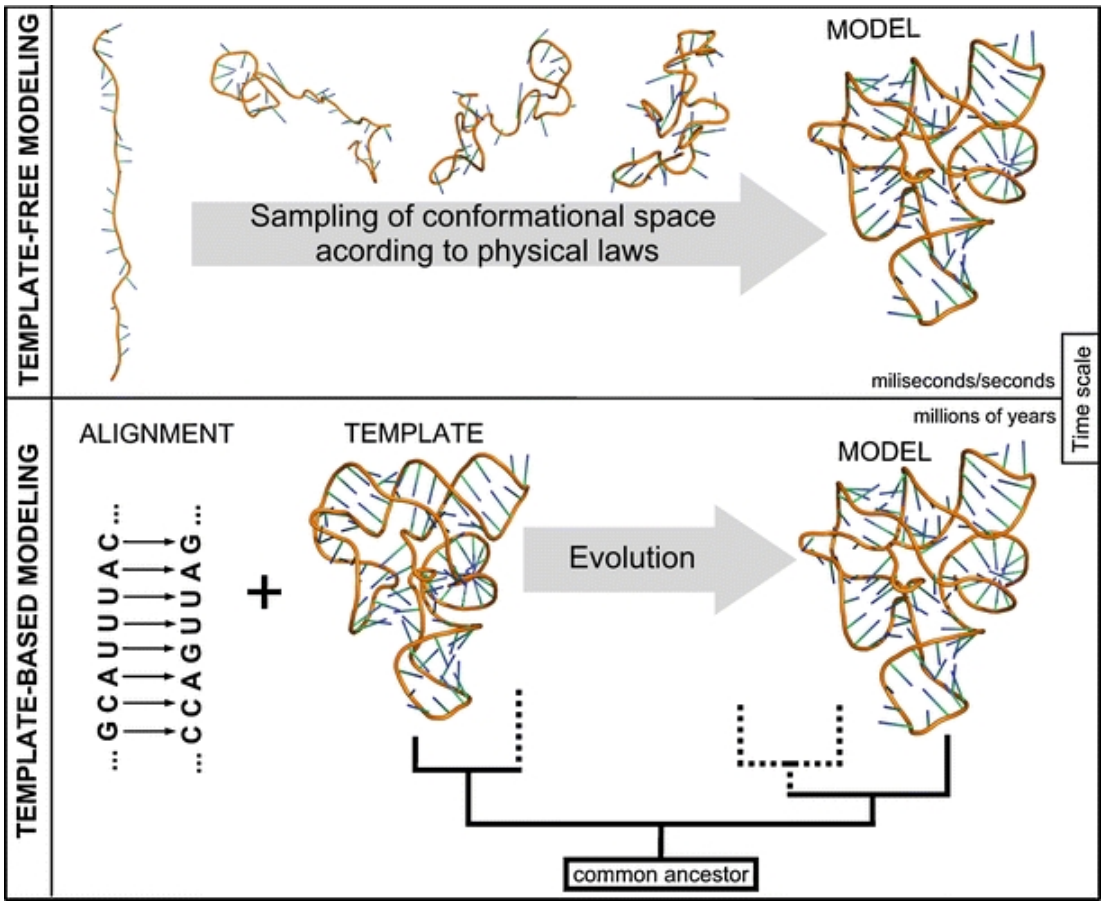
V treťom kroku musia byť dopredikované nekonzervované (chýbajúce úseky) cieľovej štruktúry. Existuje viacero prístupov. Jedným z nich je knižnica fragmentov, kde sa do chýbajúcej medzery v cieľovej štruktúre snažíme vhodne umiestniť fragment štruktúry z knižnice, ďalším je napríklad dopredikovanie medzery ab initio alebo de novo algoritmami.

Takto hotový model sa nakoniec môže optimalizovať použitím algoritmu na minimalizovanie voľnej energie, alebo sa môžu vyriešiť kolízie medzi jednotlivými nukleotidmi.

Hlavnou výhodou homológneho modelovania je, že je možné ho použiť na dlhé štruktúry. Problémom môže byť výber správnej template štruktúry a dopredikovanie nekonzervovaných úsekov. Obrázok 2.4

2.5 Prehľad existujúcixh nástrojov

Vďaka tomu, že počet dostupných primárnych sekvencií stále rastie rýchlejšie, ako počet experimentálne zistených terciárnych štruktúr, vzniklo mnoho nástrojov na predikciu terciárnej štruktúry RNA. V tabuľke uvádzame prehľad a základné princípy fungovania najpoužívanejších aktuálne dostupných algoritmov



Obrázek 2.4: Porovnanie princípu de novo a template based predikcie Rother K (2011)

na predikciu RNA štruktúr Tabuľka 2.1. V nasledujúcich dvoch sekciách bližšie predstavíme algoritmy ModeRNA a FARFAR, ktoré budeme používať v našej práci.

2.6 ModeRNA

ModeRNA je algoritmus komparatívneho modelovania RNA. Je dostupný ako ModeRNA server a ponúka službu kompletnej predikcie submitovanej sekvencie (nájdenie vhodného template, zarovnanie sekvencií a vytvorenie modelu terciárnej štruktúry). Okrem toho je možné stiahnuť jej zdrojové kódy (Python) a nainštalovať a používať ju lokálne pre automatické hromadné spracovanie.

Ako vstup ModeRNA požaduje zarovnanie template a target sekvencií spolu so štruktúrou template molekuly. Dodané zarovnanie ModeRNA nijako nemodifikuje a od jeho kvality a zvoleného template záleží výsledná presnosť predikcie. Zjednodušený algoritmus, ktorým ModeRNA predikuje štruktúru Rother K (2011):

- Skopírovanie zarovnaných nukleotidov.
- Substitúcia nukleotidov, ktoré boli zarovnané na iný nukleotid.
- Modelovanie indelov (označujeme tak miesto vloženia alebo zmazania nukleotidu v sekvencii) vložením fragmentov štruktúr z knižnice obsahujúcej 131 316 fragmentov dĺžky 2-19 nukleotidov. ModeRNA najprv rýchlym filtrovaním podľa vzdialeností prekrývajúcich sa atómov fragmentu a template-u vyberie 50 najvhodnejších kandidátov, pokúsi sa ich vložiť do medzery a pre každého kandidáta spočíta skóre. Následne vyberie jediného s najlepším skóre a vloží ho do medzery.
- V prípade, že predikovaná štruktúra (backbone) nie je spojitá, ModeRNA sa ju pokúsi opraviť.

Názov	Info	Referencia
MacroMolecule Builder	komparatívny modeling RNA	Flores a kol. (2011)
ModeRNA	komparatívna predikcia s knižnicou databázových fragmentov na predikovanie medzier	Rother a kol. (2011)
SimRNA	corase-grained model s Monte Carlo samplingom štruktúr	Boniecki a kol. (2015)
FARFAR	knowledge based de novo prediktor	Das R. (2010)
RNAComposer	knowledge-based automatizovaná predikcia štruktúry RNA s využitím sekundárnej štruktúry	Biesiada a kol. (2016)
iFoldRNA	de novo predikcia RNA založená na corase-grained model	Sharma a kol. (2008)

Tabulka 2.1: Prehľad najčastejšie používaných programov určených na predikciu RNA s informáciou o type použitého algoritmu.

2.7 FARFAR

FARFAR je algoritmus de novo predikcie RNA implementovaný spolu s ďalšími bioinformatickými algoritmami a nástrojmi v balíčku Rosetta. My ho používame na predikovanie krátkych nekonzervovaných úsekov v našom algoritme.

Nástroj je schopný predikovať štruktúru iba z primárnej sekvencie.

Algoritmus pracuje tak, že nedeterministicky generuje kandidátske štruktúry, z ktorých vyberie tú s najnižšou voľnou energiou. Keďže sa jedná o algoritmus typu Monte Carlo, dve rôzne spustenia algoritmu môžu generovať rôzne výsledky a platí, že čím viac kandidátskych štruktúr vygenerujeme, tým viac máme šancí na vygenerovanie čo najlepšej štruktúry.

Z pohľadu výkonnosti platí, že čím viac nukleotidov predikujeme (včetne tých pevne daných) tým predikcia dlhšie trvá. Pre presnejšiu predstavu sme urobili porovnanie, pričom sme predikovali nekonzervovaný úsek dlhý 9 nukleotidov. Pri zahrnutí zvyšných 129 konzervovaných nukleotidov do predikcie trvalo vygenerovanie jednej štruktúry približne 13 minút. Pri totožných podmienkach a parametroch s jedinou zmenou, a to, že sme do predikcie vybrali len 41 okolitých konzervovaných nukleotidov, trvala predikcia jednej kandidátskej štruktúry v priemere menej ako 6 minút.

Očakávaná presnosť predikcie je priamo úmerná dĺžke neznámeho predikovaného úseku. Autori algoritmu uvádzajú, že pri predikcií štruktúr dĺžky 6 až 13 nukleotidov je priemerná RMSD menšia ako 2 Å. Pri štruktúrach dlhých 13-23 nukleotidov predstavovala priemerná RMSD už 6,5 Å.

Je však takisto možné dať mu na vstup pdb súbor s koordinátami niektorých nukleotidov a zakázať mu tieto nukleotidy modifikovať. Takisto je možné mu dodať pdb súbor s nukleotidmi a dovoliť mu, aby ho algoritmus bral ako fixovaný kus štruktúry, ktorým môže ľubovoľne pohybovať oproti zvyšku štruktúry. Posledná pre nás využiteľná možnosť je dodať algoritmu sekundárnu štruktúru target molekuly. Touto štruktúrou sa potom algoritmus pri predikcii riadi a zmenšuje sa tak prehľadávaný priestor.

3. Algoritmus Trooper

V tejto práci naväzujeme a ďalej vylepšujeme algoritmus Trooper, ktorý bol vytvorený v rámci mojej bakalárskej práce na Matematicko-Fyzikálnej fakulte Galvánek (2016). Preto v tejto kapitole uvedieme princípy fungovania a stav implementácie algoritmu tak, ako bol popísaný v bakalárskej práci. Na rozdiel od bakalárskej práce rozoberieme aj časovú zložitosť algoritmu. Jedná sa o algoritmus založený na princípe homológneho modelovania, čo znamená, že predikujeme terciárnu RNA štruktúru na základe primárnej sekvencie molekuly označovanej ako target a známej terciárnej štruktúry a sekvencie inej RNA molekuly označovanej ako template.

3.1 Kostra algoritmu

V nasledujúcom zozname uvázdame postupnosť hlavných krokov algoritmu.

1. Predpríprava a validácia vstupných súborov: template sekvencia, target sekvencia a štruktúra.
2. Alignemnt: Zarovnanie target a template sekvencií.
3. Sliding window: Algoritmus posuvného okienka na zarovnaní.
4. Treating indels: Vyriešenie medzier v zarovnaní.
5. Kopírovanie a mapovanie konzervovaných nukleotidov z target štruktúry do predikovanej template štruktúry.
6. Vyčlenenie predikcie príliš dlhých medzier v target štruktúre.
7. Príprava vstupu pre FARFAR.
8. Predikcia nekonzervovaných úsekov pomocou algoritmu FARFAR.
9. Zloženie predikovaných úsekov a dlhých medzier do finálnej štruktúry.

3.2 Popis algoritmu

Ako vstup algoritmus dostane target sekvenciu a template sekvenciu aj štruktúru. Na výstupe očakávame terciárnu štruktúru target molekuly RNA.

Ako prvý krok algoritmus skontroluje, či sú sekvencia vo fasta súbore a štruktúra v pdb súbore rovnako indexované. Nukleotidy v pdb súbore sú očíslované, ale vo fasta súbore číslo nukleotidu odpovedá jeho pozícii v súbore. Kontrolujeme to prechodom cez pdb súbor tak, že indexom nukleotidu z pdb zaindexujeme do fasta súboru a typ nukleotidu musí byť v oboch súboroch na tejto pozícii zhodný. V prípade, že zhodný nie je, skúšame ešte posunúť fasta sekvenciu pridaním dummy nukleotidov na začiatok sekvencie (pre prípad, že by začiatok sekvencie v súbore chýbal). Ak sa nám nepodarí ani takýmto spôsobom dosiahnuť, aby sa typy nukleotidov v rovnakých indexoch zhodovali, označíme target za nevhodný pre predikciu a algoritmus končí neúspechom.

V druhom kroku urobíme globálne zarovnanie (alignment) target a template sekvencií v programe Emboss Needle. Na vytvorené zarovnanie použijeme algoritmus posuvného okienka (sliding window) a pre každú pozíciu určíme percentuálnu mieru okolitých úspešne zarovnaných nukleotidov spadajúcich do okienka. V prípade, že získaná hodnota je vyššia ako parametrom určená hranica, označíme príslušnú pozíciu v zarovnaní ako konzervovanú.

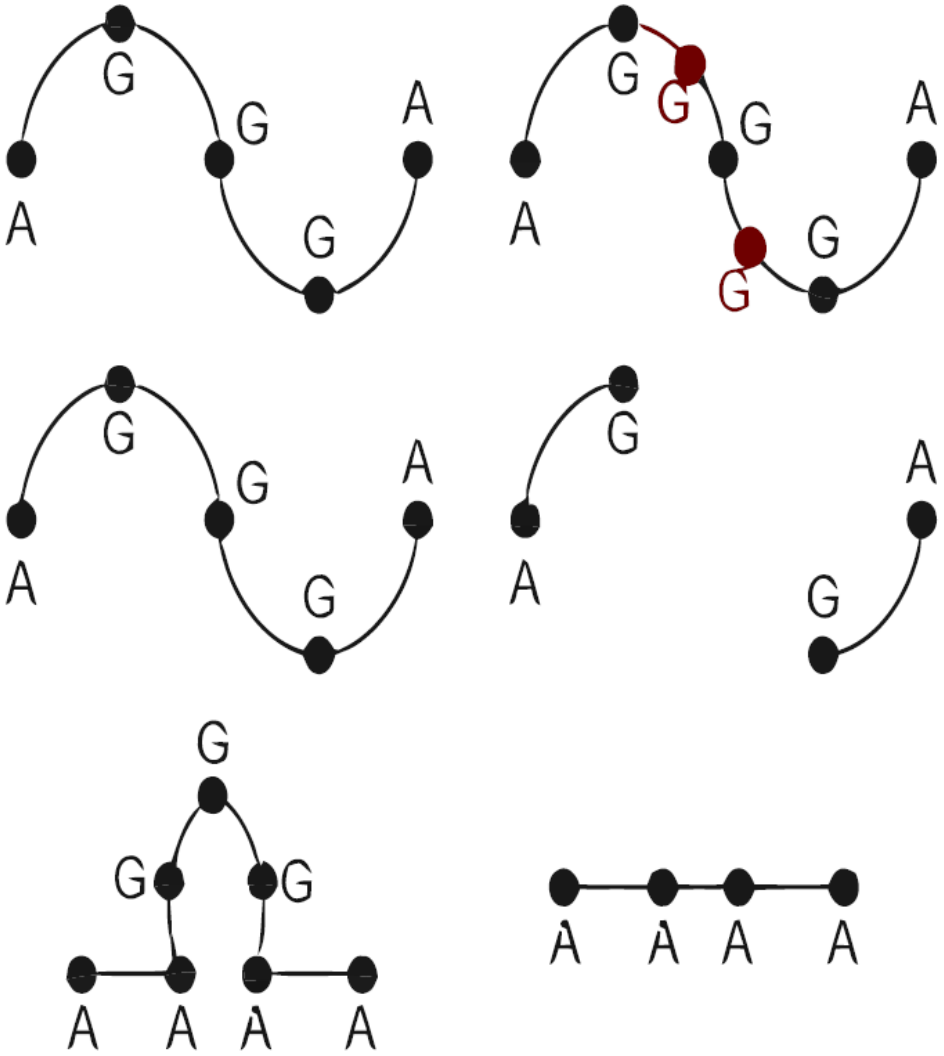
V treťom kroku sa zaoberáme medzerami (indels), ktoré vznikli v terget alebo template sekvencii pri zarovnaní. Inak povedané, do oboch sekvencií mohol algoritmus zarovnania ľubovoľne vložiť medzery alebo na seba zarovnať nezhodujúce sa nukleotidy tak, aby získal zarovnanie s čo najlepším skóre Tabuľka 3.1. To znamená, že medzery v inak konzervovanom úseku template sekvencie by vo výsledku nenechali miesto na doplnenie nukleotidov z target sekvencie zarovnaných oproti týmto medzerám z template sekvencie. Naopak, medzery v target sekvencii zarovnané oproti nukleotidom v template sekvencii v inak konzervovanom úseku by mohli spôsobiť medzeru v predikovanej štruktúre, nakoľko by sme z fragmentu konzervovanej štruktúry len odmazali nejaké nukleotidy a ničím ich nedoplnili. Obrázok 3.1 Oba tieto problémy riešime tak, že nukleotidy v určitom okolí takýchto úsekov označíme za nekonzervované a budú dopredikované algoritmom FARFAR. Taktiež označíme za nekonzervované tie nukleotidy, ktoré boli zarovnané na nezhodujúci sa typ nukleotidu.

V štvrtom kroku skopírujeme konzervované časti template štruktúry do predikovanej target štruktúry. Vzhľadom na to, že v zarovnaní môžu byť rôzne vložené medzery do target aj template sekvencie, musíme premapovať indexy nukleotidov z template štruktúry tak, aby odpovedali nukleotidom, na ktorých miesto sú vložené v target sekvencii. Toto urobíme jednoducho vďaka informáciám zo zarovnania. Takto získame target štruktúru s medzerami, ktoré potrebujeme dopredikovať.

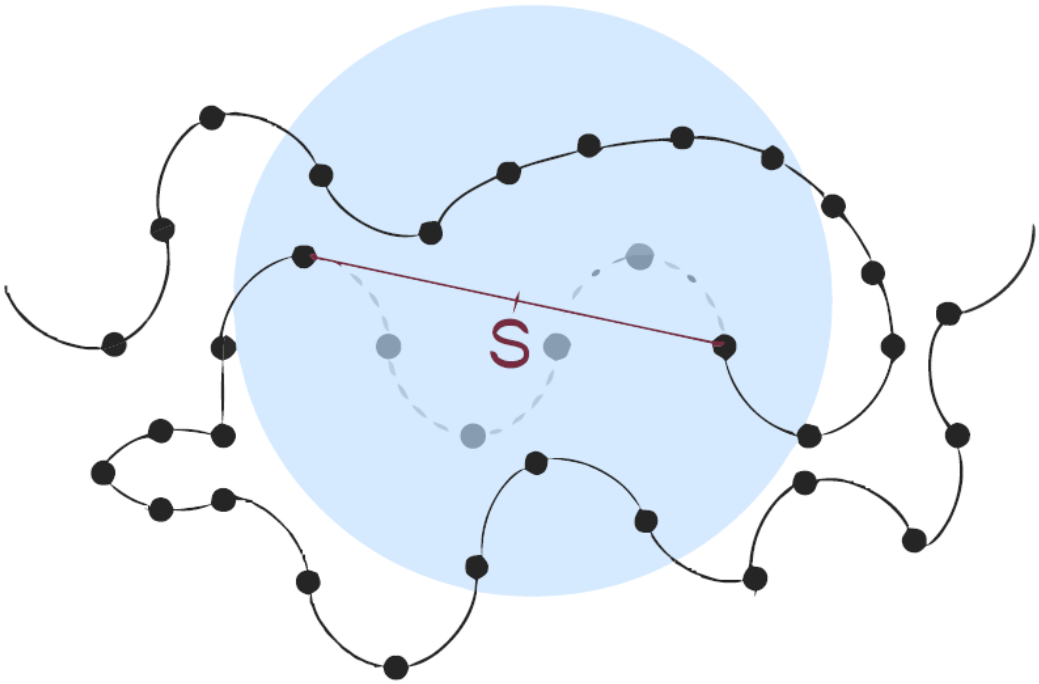
V piatom kroku identifikujeme dlhé nekonzervované úseky a vyčleníme ich následnu predikciu do samostatných behov algoritmu FARFAR. Prvým dôvodom je, že takto sa môže FARFAR zamerať iba na predikciu dlhého úseku, a tým znížime celkovú výpočetnú náročnosť. Takisto môžeme zmeniť jeho parametry, ako napríklad zvýšiť počet samplovaných modelov, prípade zvýšiť celkový čas predikcie. Ďalším dôvodom, prečo dopredikovanie nekonzervovaných úsekov takto delíme je, že algoritmus FARFAR sa nedokáže dobre vysporiadať s predikciami príliš dlhých štruktúr, aj keď je časť nukleotidov pevne daná. Z tohto dôvodu rozdeľujeme dlhé štruktúry na úseky dĺžky 300 nukleotidov na základe ich poradia v sekvencii. Takéto delenie spôsobuje ďalší problém - nukleotidy, ktoré sú od seba vzdialené v sekvencii môžu byť blízko pri sebe v terciárnej štruktúre. Naše riešenie teda vyberie tieto dlhé nekonzervované úseky spolu s okolitými nukleotidmi, ktoré ležia v guli so stredom určeným úsečkou spájajúcou posledný konzervovaný nukleotid

Sekvencia	konzervované	nekonzervované	gap	gap
tamplate	G	A	-	U
target	G	G	C	-

Tabulka 3.1: Prehľad štyroch situácií, ktoré môžu nastať na každej pozícii v zarovnaní dvoch sekvencií.



Obrázek 3.1: Problémy, ktoré môžu nasať v štruktúre pri vložení medzier do target alebo template časti zarovnania. Prvý riadok zobrazuje komplikácie pri pokuse vložiť nukleotidy (znázornene červenou farbou) do celistvej štruktúry (teda v zarovnaní boli pridané medzery do target sekvencie). Druhý riadok ukazuje opačný problém, a to vynechanie dvoch nukleotidov a roztrhnutie štruktúry (zodpovedá to vložením medzier do target sekvencie). Tretí riadok odpovedá rovnakej situácii ako druhý, ale odstránenie nukleotidov zo štruktúry nespôsobuje problém, pretože odstránené nukleotidy tvorili loop, ktorý môžeme bez problémov odobrať.



Obrázek 3.2: Schématické nakreslenie sféry so stredom v bode S, ktorý je stredom úsečky spájajúcej dva krajné konzervované nukleotidy medzery v štruktúre. Všetky nukleotidy, ktoré padnú do bledomodrej gule, budú použíte pri predikcii daného úseku.

pred nekonzervovaným úsekom s prvým konzervovaným nukleotidom za nekonzervovaným úsekom vzhľadom na ich poradie v sekvencii. Polomer tejto gule je určený empiricky ako 0,75-násobok dĺžky úsečky, kedy by guľa mala obsiahnuť všetky relevantné nukleotidy. Obrázok 3.2

V šiestom kroku pripravíme vstupné dáta pre algoritmus FARFAR. To zahŕňa prípadné rozdelenie na úseky po 300 nukleotidov spomenuté v predchádzajúcom odstavci a prepísanie informácií o tom, ktoré nukleotidy sú pevne dané a ktoré treba dopredikovať do vstupného súboru. Takisto tu určíme parametre pre jednotlivé predikcie, ako napríklad počet vygenerovaných štruktúr.

V siedmom kroku všetky takto pripravené časti predikcie spustíme a počkáme na výsledok. Toto je najpomalšia časť algoritmu, kedy FARFAR potrebuje čas minimálne pár hodín až niekoľko desiatok hodín, aby dokázal predikovať dlhšie nekonzervované úseky. Tie sú napriek tomu najväčšou slabinou nášho algoritmu, pretože podľa výsledkov získaných v bakalárskej práci platí, že čím dlhšie nekonzervované úseky sa nachádzajú v predikovaných štruktúrach tým je presnosť predikcie nižšia.

V poslednom kroku najprv konvertujeme úseky z internej reprezentácie FARFARu do klasických pdb súborov a tie spojíme do výsledku.

3.3 Používané typy súborov

V algoritme opakovane pracujeme s určitými typmi textových súborov. Sú to súbory s príponami fasta *(uloženie sekvencie)*, secstr *(uloženie sekundárnej*

4MGN_B.fasta

```
>4mgn:B|PDBID|CHAIN|SEQUENCE
GCGGAAGUAGUUCAGUGGUAGAACACCACCUUGCCAAGGUGGGGGGUCGCGGGUUCGAGUCCCGUCUUCCGCU
```

4MGN_B.secstr

```
GCGGAAGUAGUUCAGUGGUAGAACACCACCUUGCCAAGGUGGGGGGUCGCGGGUUCGAGUCCCGUCUUCCGCU
(((((((..((((....{..)))).(((((..]]]..)))))).....(((((..}....)))))))))))).
```

4mgn.pdb

```
ATOM     42  C2'   G A  11       5.725 -27.941   2.484  1.00169.89           C
ATOM     43  O2'   G A  11       6.056 -27.404   1.212  1.00153.29           O
ATOM     44  C1'   G A  11       6.859 -28.865   2.926  1.00172.17           C
ATOM     45  N9    G A  11       6.365 -29.978   3.759  1.00162.95           N
ATOM     46  C8    G A  11       6.462 -30.105   5.123  1.00173.38           C
ATOM     47  N7    G A  11       5.922 -31.205   5.571  1.00177.55           N
ATOM     48  C5    G A  11       5.437 -31.841   4.436  1.00167.56           C
ATOM     49  C6    G A  11       4.751 -33.076   4.296  1.00164.57           C
ATOM     50  O6    G A  11       4.425 -33.879   5.179  1.00160.13           O
ATOM     51  N1    G A  11       4.441 -33.345   2.967  1.00161.51           N
ATOM     52  C2    G A  11       4.751 -32.530   1.906  1.00160.80           C
ATOM     53  N2    G A  11       4.366 -32.965   0.697  1.00157.05           N
ATOM     54  N3    G A  11       5.391 -31.376   2.023  1.00154.75           N
ATOM     55  C4    G A  11       5.702 -31.096   3.307  1.00161.46           C
ATOM     56  P     A A  12       3.619 -25.383   3.517  1.00163.33           P
ATOM     57  OP1   A A  12       3.305 -23.975   3.166  1.00180.95           O
ATOM     58  OP2   A A  12       3.353 -25.873   4.895  1.00120.42           O
ATOM     59  O5'   A A  12       2.845 -26.334   2.502  1.00133.91           O
ATOM     60  C5'   A A  12       2.978 -26.155   1.100  1.00114.52           C
```

Obrázek 3.3: Na obrázku vidíme zhora nadol príklady súborov fasta, secstr a pdb pre molekulu 4mgn.

štruktúry), pdb *(uloženie terciárnej štruktúry)* a aln *(uloženie zarovnania dvoch sekvencii)* Obrázok 3.3.

Súbory typu fasta slúžia na ukladanie sekvencií. Pozostávajú z dvoch riadkov, v prvom je identifikátor sekvencie pozostávajúci z jej názvu a chain-u (jedna sekvencia máva často viacero chains) a v ďalších riadkoch sú za sebou zoradené jednotlivé nukleotidy *A, C, G, U*. V prípade, že je nejaký nukleotid v rade neznámy, bežne sa namiesto jeho typu úvádza písmeno *N* alebo *X*.

Súbory secstr nám slúžia na ukladanie informácií o sekundárnej štruktúre molekuly. Sú tvorené kombináciou rôznych typov zátvoriek, ktoré popisujú sekundárnu štruktúru tak, že medzi nukleotidmi odpovedajúcimi zátvorkám existuje chemická väzba - takéto dva nukleotidy sa tiež nazývajú base pair. Z toho vyplýva, že sa v terciárnej štruktúre budú nachádzať blízko pri sebe. Bodka v sekundárnej štruktúre znamená, že nukleotid netvorí base pair so žiadnym ďalším nukleotidom. Rôzne typy zátvoriek ako *[] , {}, <>* reprezentujú pseudouzly Ramlan a Zauner (2008).

Súbory typu pdb uchovávajú okrem iného informácie o jednotlivých atómoch molekuly. V každom riadku sú uložené informácie o presných koordinátoch atómu v 3D priestore, typ atómu vrámci nukleotidu, chain do ktorej atóm patrí a index nukleotidu, ktorému patrí. PDB súbory často nie sú kompletné, chýbajú v nich atómy alebo celé nukleotidy. Takisto sa stáva, že indexy nukleotidov v pdb súboroch a fasta súboroch nesúhlasia. Táto nekonzistentnosť medzi súbormi, alebo chýbajúce časti pdb súborov sťažujú ich algoritmické spracovanie.

Súbory s príponou aln označujú výstup zarovnania dvoch sekvencií z programu EMBOSS Needle. Súbor obsahuje presné zarovnanie sekvenciií, skóre zarovnania, percentuálny pomer medzier v zarovnaní (gaps) a percentuálny pomer korektne

zarovnaných nukleotidov označený ako similarity.

3.4 Popis implementácie

Algoritmus bol implementovaný prevažne v programovacom jazyku Python 2.7. s využitím knižnice BioPython, ktorá zjednodušuje prácu so štandardnými súbormi používanými v bioinformatike, ako napríklad pdb a fasta. Okrem toho sme použili bash scripty na manipuláciu so súbormi a spustenie predikcie vo FARFAR.

Algoritmus bol rozdelený na tri časti. Prvá pozostávala zo spustenia predikcie dopredu pripravenej target-template dvojice molekúl na lokálnom PC s operačným systémom Windows. To obsiahlo algoritmus po siedmy krok, teda bola pripravená target štruktúra do stavu, kedy treba dopredikovať nekonzervované úseky algoritmom FARFAR. Následne boli takto pripravené vstupy pre FARFAR skopírované na servery organizácie Metacentrum používajúce operačný systém unixového typu s dávkovým spracovaním úloh. Vďaka tomu, že predikcie nekonzervovaných úsekov v rámci jednej molekuly sú na sebe nezávislé a tiež predikcie molekul medzi sebou sú nezávislé, môžeme predikcie nekonzervovaných úsekov algoritmom FARFAR paralelizovať. To je veľmi dôležité, pretože de novo predikcia nekonzervovaných úsekov bola najdlhšie trvajúca časť algoritmu a predikcia jednej štruktúry mohla obsahovať niekoľko takýchto de novo predikcií. Po skončení FARFAR predikcií nekonzervovaných úsekov boli výsledky skopírované späť na lokálny PC a tam boli v treťom kroku vyhodnotené výsledky.

3.5 Pseudokód

V tejto sekcii uvádzame pseudokód algoritmu Trooper v stave po bakalárskej práci. Premenné začínajú malým písmenom, názvy metód veľkým písmenom. Pseudokód ignoruje prácu s datami a znaky *[]* za premennou znamenajú, že premenná nahrádza množinu pdb súborov. Pseudokód nám bude ďalej v práci slúžiť aj na to, aby sme sa vedeli na jendotlivé metódy v prípade potreby odkazovať.

```
1  Main(fastaTarget, fastaTemplate, pdbTemplate)
2  {
3      CheckTemplateMapping
4          (fastaTemplate, pdbTemplate)
5      alignment := Align
6          (fastaTarget, fastaTemplate)
7      alignment1 := UseSlidingWindow
8          (alignment)
9      alignment2 := ProcessGaps
10         (alignment1)
11     conservedParts := CopyConservedParts
12         (alignment2, pdbTemplate)
13     mappedConservedParts := MapConservedParts
14         (conservedParts, alignment2)
15     longParts[] := ProcessLongUnconservedParts
16         (mappedConservedParts)
```

```
17      shortParts[] := ProcessShortUnconservedParts
18          (mappedConservedParts)
19      predictedParts[] := PredictUnconservedParts
20          (longParts[], shortParts[])
21      finalModel := ConnectPredictedParts
22          (predictedParts)
23  }
24
25  CheckTemplateMapping(fasta, pdb)
26  {
27      foreach res in pdb
28      {
29              if (fasta[res[id]] != res[type])
30              ERROR: Input needs manual editing!
31              EXIT PROGRAM
32      }
33  }
34
35  Align(fastaTarget, fastaTemplate)
36  {
37      string alignment := CallEmbossAln
38          (fastaTarget, fastaTemplate)
39      return EditAlignmentFormat(alignment)
40  }
41
42  SlidingWindow(alignment, windowLength, minimalLimit)
43  {
44      hL = windowLength DIV 2
45      foreach nucleotide in alignment
46      {
47          check if in [nucleotide[id]-hL, nucleotide[id]+hL]
48          is less than minimalLimit conserved nucleotides
49          if so mark nucleotide as unconserved
50      }
51      return modifiedAln
52  }
53
54  ProcessGaps(alignment, cutoff)
55  {
56      foreach gap in alignment
57      {
58          alignment := mark "cutoff" nucleotides from
59              both sides of the gap as unconserved
60      }
61      return alignment
62  }
63
64  CopyConservedParts(alignment, pdb)
```

```
65 {
66    conservedParts = ""
67    foreach nucleotide in pdb
68    {
69       if (IsConserved(nucleotide, alignment))
70          conservedParts += nucleotide
71    }
72    return conservedParts
73 }
74
75 MapConservedParts(conservedParts, alignment)
76 {
77    map nucleotide id's from current state (nc with
78    id x corresponds to x-th nc in template fasta) to
79    state where nc id correctly corresponds to
80    nucleotide in target fasta
81    return modifiedConservedParts
82 }
83
84 ProcessLongUnconservedParts
85    (conservedParts, ncAvarageLength, minLengthGapLimit)
86 {
87    longParts = []
88    foreach gap in conservedParts
89    {
90       if length(gap) <= minLengthGapLimit
91          continue
92       startNc := conserved nucleotide before gap
93       endNc := conserved nucleotide after gap
94       ph="phosphor"
95       s := FindMiddle(startRes[ph], endRes[ph])
96       p := length(gap) * ncAvarageLength / 2
97       ncsInSphere := all nucleotides inside sphere(p, s)
98       preparedPart := PrepareCorectFormatForFARFAR
99          (ncsInSphere)
100      longParts[] += preparedPart
101   }
102   return longParts[]
103 }
104
105 ProcessShortUnconservedParts
106    (conservedParts, lengthOfSection, maxLengthGapLimit)
107 {
108    createdSections = []
109    createdSections[] = divide "conservedParts" into
110       sections with length of "lengthOfSection"
111    preparedSections = []
112    foreach section in createdSections[]
```

```
113     {
114         preparedSections[] += PrepareCorectFormatForFARFAR
115             (section, maxLengthGapLimit)
116     }
117     return preparedSections[]
118 }
119
120 PredictUnconservedParts(lgUnconsPts[], shUnconsPts[])
121 {
122     predictedParts = []
123     allParts := lgUnconsPts[] + shUnconsPts[]
124     foreach input in allParts
125     {
126         predictedParts[] += CallFARFAR(input)
127     }
128     return predictedParts[]
129 }
130
131 ConnectPredictedParts(predictedParts[])
132 {
133     connect all conserved and predicted nucleotides
134     into single file in case of duplicity
135     (they are almost the same) choose only one
136 }
```

3.6 Časová zložitosť

V tejto sekcii rozoberieme časovú zložitosť algoritmu ako reálnu, tak aj asymptotickú, ktorou sme sa v bakalárskej práci nezaoberali. Budeme na to potrebovať definovať premennú dĺžka molekuly l, kedy budeme predpokladať, že všetky molekuly s ktorými pracujeme majú takúto dĺžku.

Časová náročnosť celého algoritmu je závislá hlavne na nekonzevovaných úsekoch, ktoré treba predikovať. Časť predikcie po algoritmus FARFAR beží v rádoch sekúnd. Pre FARFAR sme okrem pár problémových predikcií používali obmedzenie predikcie časom 24 hodín, prípadne 100 štruktúr. To znamená, že FARFAR buď stihne do 24 hodín namodelovať 100 modelov štruktúr, alebo s modelovaním skončí predčasne po 24 hodinách a výsledkom bude počet modelov štruktúr, ktoré sa mu do 24 hodín podarilo namodelovať. Z namodelovaných štruktúr sa vyberie tá s najmenšou voľnou energiou. Počet a kvalita kandidátskych štruktúr, ktoré za tento čas algoritmus stihne vygenerovať, záleží na počte nekonzervovaných úsekov, ich dĺžke (problematické sú hlavne dlhé nekonzervované úseky) a dĺžke celej predikovanej štruktúry včetne konzervovaných nukleotidov.

Asymtoticky zložitosť rozoberieme podľa jednotlivých metód pseudokódu:

1. Main: hlavná metóda, ktorá len prevoláva ostatné metódy. *O(1)*
2. CheckTemplatetMapping: Sekvenčný prechod cez fasta a pdb súbory template molekuly: *O(l)*

3. Align: zarovnanie target a template sekvencie: $O(l^2)$

4. Sliding window: sekvenčný prechod zarovnaním $O(l)$

5. ProcessGaps: sekvenčný prechod zarovnaním $O(l)$

6. CopyConservedParts: sekvenčný prechod zarovnaním $O(l)$

7. MapConservedParts: Je možné to urobiť dvomi sekvenčnými prechodami v čase O(l), implementované to však je dosť nešikovne v čase $O(l^2)$

8. ProcessLongUnconservedParts: sekvenčný prechod target sekvenciou $O(l)$

9. ProcessShortUnconservedPairs: sekvenčný prechod target sekvenciou $O(l)$

10. PredictUnconservedPairs: volanie algoritmu FARFAR. Čas považujeme za konštantný, pretože beh obmedzujeme na 1 deň $O(1)$. Asimptotickú zložitosť nástroja sa nám nepodarilo dohľadať.

11. ConnectUnconservedPairs: sekvenčný prechod napredikovanými úsekmi $O(l)$

Celková asymptotická zložitosť nášho algoritmu (v prípade, že FARFAR obmedzíme na $O(1)$) je $O(l^2)$.

Keď sa pozrieme na reálny čas behu, najviac sa na ňom podieľa algoritmus FARFAR a to jedným dňom, pri štruktúrach dlhých 50-500 nukleotidov. Príprava predikcie a následne spojenie a vyhodnotenie výsledkov zaberie pre jednu štruktúru dlhú 50-500 nukleotidov približne 10 sekúnd. Tento čas sme namerali na počítači s procesorom Intel i5-6200Li 2,8GHz.

3.7 Hlavné problémy algoritmu a jeho implementácie

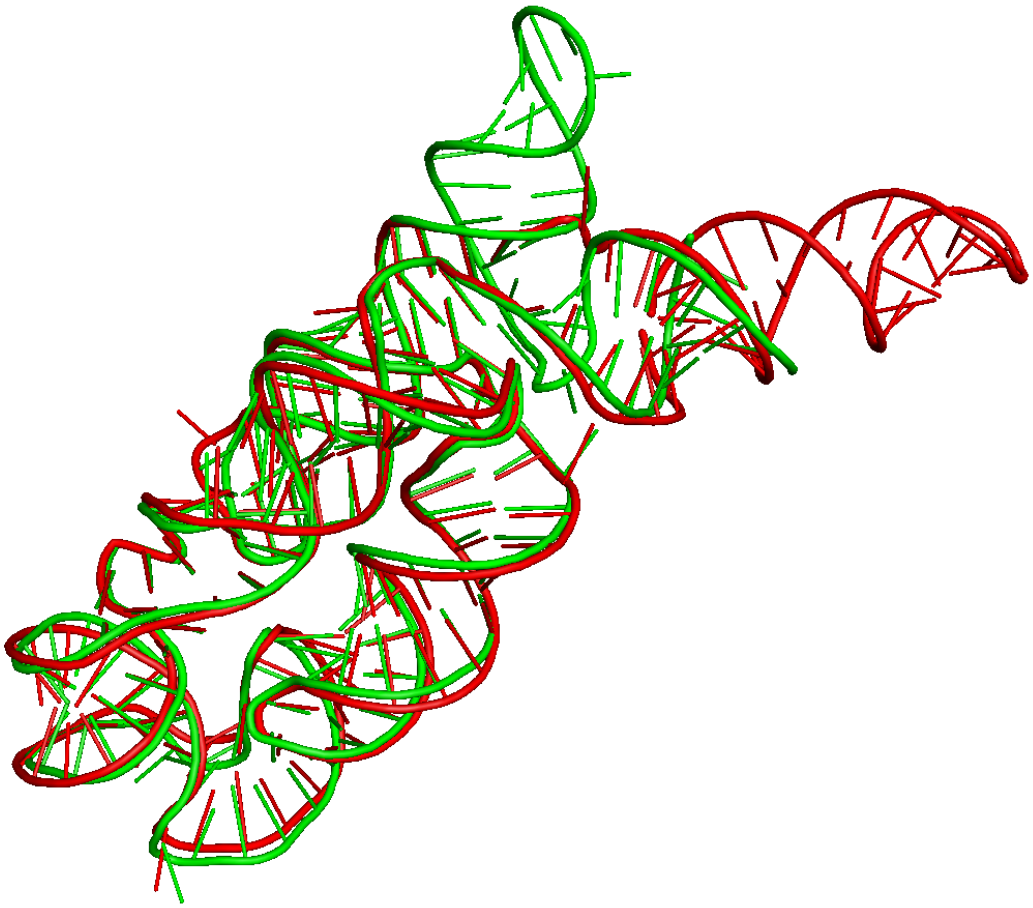
Najväčším problémom v našom algoritme, ktorý sme identifikovali na základe výsledkov bakalárskej práce, je predikcia dlhších nekonzervovaných úsekov Obrázok 3.4. Preto by sme potrebovali minimalizovať takéto úseky, prípadne pomôcť algoritmu FARFAR zmenšiť prehľadávaný priestror pri generovaní kandidátskych štruktúr. To by potom mohlo pomôcť rýchlosti predikcie kandidátskych štruktúr FARFAR-om a zlepšiť jeho presnosť. Takisto by sme chceli do väčšej miery automatizovať predikciu, a urobiť ju užívateľsky prívetivejšou.

Aby sme zlepšili problematické časti algoritmu a jeho implementácie navrhujeme niekoľko vylepšení ako algoritmu, tak aj implementácie. Tieto opatrenia popíšeme v nasledujúcich troch kapitolách a sú hlavným prínosom tejto práce. Tu uvádzame ich prehľad:

1. Algortimus predikcie

   (a) Pridanie predikcie sekundárnej štruktúry, ako medzikroku pri predikovaní terciárnej štruktúry.

   (b) Pridanie možnosti použiť viacero template molekúl pri predikcii jednej target molekuly.



Obrázek 3.4: Na obrázku vidíme zarovnanie experimenntálne získanej (3DIG) štruktúry a jej predikcie, napredikovanou našim algoritmom vytvoreným v bakalárskej práci. V pravej hornej časti obrázka vidíme, že algoritmu FARFAR sa nepodarilo správne napredikovať nekonzervovaný úsek.

   (c) Algoritmické vyhľadanie vhodnej template štruktúry.

2. Implementácia

   (a) Presunúť celú predikciu na Metacentrum.

   (b) Predikcia prebehne bez manuálnych zásahov do procesu.

   (c) Vysporiadanie sa s chybnými pdb a fasta súbormi.

3. Vyhodnotenie výsledkov

   (a) Porovnanie s nástrojom ModeRNA, ktorý tiež funguje na princípe komparatívneho modelovania.

   (b) Vyhodnotenie na všetkých dostupných kombináciach target-template párov, ktoré má zmysel predikovať, nakoľko v bakalárskej práci sme vybrali len niekoľko desiatok párov, na ktorých sme predikciu skúšali a vyhodnocovali výsledky.

4. Automatizácia predikcie a porovnanie s ModeRNA

V tejto kapitole sa budeme venovať ďalšej automatizácii algoritmu, zjednodušeniu predikcie a implementácií. Pripomenieme výsledky bakalárskej práce a predstavíme dáta na ktorých budeme implementované algoritmy testovať. Okrem toho sme náš algoritmus porovnáme s ďalším nástrojom na predikciu RNA štruktúr ModeRNA, ktorý funguje na podobnom princípe ako náš algoritmus, teda na princípe komparatívneho modelovania. Ďalej doplníme funkcionalitu, vďaka ktorej algoritmus nájde vhodnú template štruktúru pre zadaný target. Nakoniec sa pokúsime riešiť datové problémy, ktoré vznikajú vďaka nezhode fasta sekvencií a sekvencií extrahovaných z odpovedajúcich pdb štruktúr.

4.1 Dáta

Všetky testovacie dáta boli stiahnuté z webu ProteinDataBank `https://www.rcsb.org` Berman (2000). Jedná sa o fasta a pdb súbory RNA molekúl, ktoré majú experimentálne získanú terciárnu štruktúru. Celkový počet molekúl je 1158. Fasta súbory sme ešte následne rozdelili podľa identifikátora vlákien *(chain)* (jedna stiahnutá molekula môže obsahovať viacero štruktúr rozdelených do vlákien). Po tomto rozdelení fasta súborov sme získali 2081 sekvencií. Na tieto dáta budeme v práci ďalej odkazovať ako na *databázu štruktúr*.

Pre testovanie algoritmu sme používali všetky stiahnuté štruktúry sekvencie dĺžky 50 a viac nukleotidov. Najprv sme ich rozdelili do priehradok podľa dĺžky (50-100nt, 101-500nt, 500-viac nt). Štruktúry v priehradkách sme následne spárovali každú s každou ako target - template dvojice a tieto dvojice rozdelili do skupín podľa podobnosti a pomeru medzier vzhľadom na zarovnanie. Takto pripravené dáta sme používali pre testovanie algoritmov z tejto práce.

4.2 RMSD

V tejto sekcii si predstavíme mieru, ktorou budeme vyjadrovať presnosť získaných výsledkov. Budeme merať rozdiely medzi experimentálne získanou štruktúrou a nami napredikovanou štruktúrou. Sekvencia oboch štruktúr je rovnaká, na základe čoho je možné učiť vzájomne si odpovedajúce nukleotidy a tiež atómy. Mierou RMSD (root mean square deviation) vyjadrujeme priemerná vzdialenosť medzi dvojicami atómov v zarovnaných terciárnych štruktúrach a vypočíta sa nasledovne:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i^2}$$

, pričom δ je vzdialenosť medzi N pármi ekvivalentných atómov. Hodnota je vyjadrená v jednotke *Ångström (Å)*, ktorá je rovná $10^{-10}m$. Na zarovnanie a samotný výpočet RMSD používame funkciu align v programe PyMol Schrödinger, LLC (2015).

4.3 Implementácia a automatizácia

V tejto sekcii vysvetlíme, kde je implementácia algoritmu Trooper dostupná a ako sá dá nainštalovať, použiť a porovnáme v nej tiež zmeny v automatizácií oproti stavu z bakalárskej práce.

Pre implementáciu máme vytvorený repozitár na GitHub-e s adresou `https://github.com/galvaner/Trooper`. Repozitár obsahuje viacero vetví. Najaktuálnejšia je vetva *multiple_templates*, ktorá obsahuje aktuálny kód a dáta, ktoré sme používali. Okrem toho za zmienku stojí vetva *run_on_local_pc*, kde je pripravená testovacia predikcia štruktúry 4QLM_A.

Pre spustenie treba splniť nasledujúce software-ové požiadavky:

1. UNIX-ový operačný systém.
2. Nainštalovať software Rosetta `https://www.rosettacommons.org/software/license-and-download`.
3. Nainštalovať software Emboss `http://emboss.sourceforge.net/what/` (musí byť dostupný z príkazového riadku).
4. Nainštalovať balík VienaRNA `https://www.tbi.univie.ac.at/RNA/` pri použití predikcie sekundárnej štruktúry (musí byť dostupný z príkazového riadku).
5. Nainštalovať PyMOL (musí byť dostupný z príkazového riadku).
6. Nainštalovať Python v2.7 (musí byť dostupný z príkazového riadku).
7. Nainštalovať BioPython.

Pre spustenie treba dodať nasledujúce vstupné súbory:

1. Dodať súbory do priečinka pairs pomenované *60-75s30-45g_51_100l.txt*, pričom čísla a písmená môžu byť zmenené. Každý riadok súboru znamená jednu predikciu a skladá sa z názvov target a template štruktúr. Názvy sú oddelené medzerou a zahŕňajú identifikátor vlákna oddelený od mena štruktúry podtržítkom, napríklad: *4QLM_A.fasta 4QK9_A.fasta*.
2. Dodať potrebné pdb súbory do priečinka *pdbs*.
3. Dodať potrebné fasta súbory do priečinka *fastas*.
4. Dodať potrebné súbory obsahujúce sekundárne štruktúry do priečinka *secondary_structures*.

V pôvodnej implementácii bežala časť algoritmu na OS Windows a časť na UNIX-e. Medzi operačnými systémami bolo treba manuálne kopírovať súbory, čo je zbytočne zdĺhavé. Kvôli tomu sme algoritmus testovali iba na skupine vybraných dát. Upravili sme implementáciu tak, že dokáže bežať na jednom operačnom systéme a z hľadiska obsluhy sa predikcia skladá z nasledujúcich štyroch krokov (spustení štyroch skriptov):

1. Príprava predikcie pre FARFAR *prepare_rosetta_prediction.sh*.

2. Predikcia FARFAR *startFARFAR.sh*.

3. Konverzia a spojenie výsledkov FARFAR do jedného pdb súbora *extract_pdbs.sh*.

4. Vyhodnotenie predikcie *concat_pdbs.sh*.

Rozhodli sme sa teda presunúť beh celého algoritmu na unixový systém. Prečo sme sa rozhodli pre Unix a nie Windows? Hlavným dôvodom je možnosť paralelizácie predikcie FARFAR v Metacentre a taktiež fakt, že Rosetta nie je na Windowse podporovaná. Algoritmus bol upravený tak, že pre predikciu stačí dodať vstupné dáta a spustiť jeden skript na prípravu predikcie bodov 1-7 Sekcia 3.1 (teda až do spustenia FARFAR predikcie). Následne treba spustiť druhý skript, ktorý spustí predikciu FARFAR. Bolo by možné automaticky spustiť tento skript hneď po prvom, avšak v prípade, že by prvá časť skončila s chybami, máme možnosť ju prekontrolovať a nezahltiť tak metacentrum naplánovaním zbytočných taskov. Po dobehnutí predikcie nekonzervovaných úsekov je treba spustiť tretí skript, ktorý konvertuje výstupy FARFAR z internej reprezentácie do pdb súborov, a zároveň ich zmerguje do výslednej štruktúry. Nakoniec v prípade, že v priečinku s experimentálne získanými štruktúrami existuje predikovaná target štruktúra, môžeme spustiť štvrtý skript, ktorý s ňou porovná napredikovanú štruktúru pomocou programu PyMol a výsledok uloží.

V našom prediktore existuje mnoho konfigurovateľných parametrov, ktoré necheme vždy zadávať na vstupe a ani ich nechceme hardcodovať v rôznych scriptoch. Niektoré sa nachádzajú v shell scriptoch a niektoré v python scriptoch. Preto sme vytvorili konfiguračný súbor `Predictor/scripts/config.py`, ktorý obsahuje všetky relevantné parametre uložené prehľadne na jendom mieste Obrázok 4.1.

4.4 Výsledky pôvdného algoritmu

Pôvodne sme vďaka nedostatočnej automatizácií algoritmu predikovali iba niekoľko náhodne vybraných štruktúr a takto získané výsledky sme prezentovali v bakalárskej práci. Tieto výsledky tu uvedieme kvôli dvom pozorovaniam, ktoré z nich plynú a budú nás zaujímať pri výbere dát pre ďalšie testy v tejto práci. Prvé pozorovanie ukazuje, že predikovať sekvenciu na základe štruktúry s podobnosťou menšou ako 60% nie je vhodné, ako vidno aj v tabuľke výsledkovTabuľka 4.1. Naopak druhé pozorovanie ukazuje, že pri podobnosti sekvencii väčšej ako 60% sme schopní dosahovať presnosť predikcie pod 10Å. Tento výsledok je porovnateľný s algoritmom ModeRNA, ako ukážeme v nasledujúcej sekcii. Taktiež je to výrazné zlepšenie oproti de novo algoritmom, kde v článku Yunjie Zhao (2012) je uvádzaná priemerná RMSD používaných algoritmov pri predikcii štruktúr dĺžky 50-130 nukleotidov približne 20Å.

4.5 Porovnanie s ModeRNA

V bakalárskej práci sme algoritmus porovnávali s algortimom FARFAR, ktorý z princípu nedokáže predikovať príliš dlhé štruktúry *(autori uvádzajú, že FARFAR*

Trooper / Predictor / scripts / config.py

Edit file | Preview changes

```
1
2   configSlidingWindowSize=10
3   configSlidingWindowMinCoverage=5
4   configMinGabLengthToOwnPrediction=7
5   configDividePredictionIntoRegionsOfSize=300
6   configNumberOFStructuresGeneratedByFARFAR="1"
7   configPathToRosetta='/storage/praha1/home/galvaner/rosetta
    /rosetta_bin_linux_2015.38.58158_bundle/'
8   configPathToRNATools='/storage/praha1/home/galvaner/rosetta
    /rosetta_bin_linux_2015.38.58158_bundle/tools/rna_tools/bin/'
9   configWallTime='23:59:00'
10  predictSecondaryStructure=True
11  useMultipleTemplates=True
12  predictListOfTargets=True # if false predict target-template pairs from txt (do not find
    best templates for target) (old style)
13  singleTargetPredictionFromCmd=False  # read name of target fasta from command line and
    select template working only with predictListOfTargets
14  pathToCygwin='C:\\cygwin64\\bin\\bash' # only relevant if running on windows
```

Obrázek 4.1: Príklad konfiguračného súboru.

dosahuje vysokú presnosť len pri štruktúrach veľkosti do 12 nukleotidov). Teraz sme sa rozhodli porovnať náš algoritmus s komparatívnou predikciou ModeRNA. Pre porovnanie sme sa rozhodli napredikovať rovnaké sekvencie za pomoci rovnakých target štruktúr a porovnať výsledky z ModeRNA s výsledkami nášho algoritmu. Napísali sme skript `ModeRNA/moderRNA.py`, ktorý sa pokúša predikovať target - template páry poskytuté na vstupe. Predikcia v ModeRNA je výrazne rýchlejšia a vytvorenie jednej štruktúry sa väčšinou pohybuje v rádoch minút. Výsledky oboch algoritmov sme porovnali Tabuľka 4.2, pričom sme uvažovali iba predikcie, ktoré úspešne vytvorili oba algoritmy. V tomto prípade sme rozlišovali dĺžky štruktúr, a to 50-100 nukleotidov a 101-500 nukleotidov. Kritérium na podobnosť tempate štruktúry bolo pre všetky dĺžky rovnaké, a to 60-90%. Z výsledkov vyplýva, že ModeRNA bola úspešnejšia v predikovaní dlhších štruktúr,

Podobnosť (%)	Gap (%)	Priemer RMSD (Å)	Smerodatná odchýlka (Å)
30-45	30-45	32,05	6,09
45-60	30-45	32,30	4,78
60-75	0-15	11,88	7,81
60-75	15-30	9,63	2,33
60-75	30-45	8,80	7,20
75-90	0-15	6,02	4,37
75-90	15-30	6,93	4,45

Tabulka 4.1: Výsledky predikcie pre target template páry dĺžky 50-500 nukleotidov z bakalárskej práce.

pričom naopak náš algoritmus si viedol lepšie pri predikovaní štruktúr kratších. Na druhej strane, ModeRNA dokázala napredikovať celkovo viac štruktúr oproti nášmu algoritmu, pretože dokázala lepšie zvládnuť nedokonalé vstupné dáta. Tieto výsledky sme publikovali v článku Galvanek a kol. (2016).

4.6 Porovnanie predikcie dlhých štruktúr s ModeRNA

Algoritmy založené na princípe komparatívneho modelovania majú vďaka vysoko konzervovaným RNA štruktúram potenciál predikovať prakticky neobmedzene dlhé štruktúry, ak pre ne existuje dosť dobrá template štruktúra. Predikcia štruktúr dlhých niekoľko tisic nukleotidov samozrejme prináša problémy, ako napríklad dlhšie nekonzervované úseky a celkovo dlhší čas predikcie. Skúšali sme predikovať 4 rôzne target-template páry aj pomocou nášho algoritmu, aj pomocou ModeRNA algoritmu. Pravdepodobne kvôli tomu, že ModeRNA používa na vyplnenie nekonzervovaných úsekov knižnicu fragmentov, sa v takto dlhej štruktúre našiel nekonzervovaný úsek, ktorý ModeRNA nedokázala napredikovať. Náš algoritmus má síce s dlhými úsekmi problémy tiež, ale vďaka ich de novo predikcii dokázal napredikovať aj takýto úsek Tabuľka 4.3.

4.7 Automatické vyhľadanie tempalte štruktúry

V pôvodnej implementácii algoritmus očakáva pripravený súbor target a template dvojíc, ktoré bude predikovať. To je vhodné pre účely testovania algoritmu, kedy je našim cieľom zadať vzťah medzi target a template sekvenciou (dĺžka, podobnosť, počet medzier v zarovnaní). V prípade, že by sme našu implementáciu chceli poskytnúť užívateľom tak, aby mohli pomocou nej napredikovať neznámu štruktúru, očakávame, že užívateľ nebude vedieť, akú štruktúru je vhodné použiť ako target. Preto sme naimplementovali možnosť zadať na vstup iba target sek-

Dĺžka (nt)	ModeRNA RMSD (Å)	Trooper RMSD (Å)
50-100	5,77	8,34
101-500	8,25	4,29
50-500	6,21	7,65

Tabulka 4.2: Porovnanie nášho algoritmu s ModeRNA.

Target(len)	Template(len)	Sim(%)	Trooper	ModeRNA
3DG0(2904)	2O45(2880)	68	13,65	No fragments candidates.
4JI1(1522)	4V4Q(1542)	71,6	14,5	No fragments candidates.
4V6W(1995)	4V6X(1869)	68,3	32,7	Sequences do not match.
3DG5(1542)	3J2G(1533)	99,4	9,6	9,6

Tabulka 4.3: Porovnanie výsledkov predikcie vybraných dlhých štruktúr pomocou našeho algoritmu a pomocou ModeRNA.

venciu a náš algoritmus sám určí vhodnú sadu template sekvencií, podľa ktorých by bolo možné cieľovú sekvenciu predikovať čo najlepšie. Počet template sekvencií dodáme algoritmu ako parameter *returnFirstSuitableX* a uživateľovi to dáva teoretickú možnosť napredikovať jeden target podľa viacerých template štruktúr.

Pseudokód algoritmu vyhľadávajúceho template štruktúru:

```
1  SelectTemplates(targetSeq, returnFirstSuitableX)
2  {
3      foundSeqs = []
4      foreach fastaSeq in allSequences
5      {
6          if foundSeqs.Count() == returnFirstSuitable:
7           return foundSeqs
8          if fastaSeq not usable as template for targetSeq:
9             continue
10         align := CallEmbossAln(fastaSeq, targetSeq)
11         if fastaSeq is better than any of foundSeqs:
12            if fastaSeq not too similar to other foundSeqs:
13              foundSeq.Add(fastaSeq)
14     }
15     return foundSeqs
16 }
```

Algoritmus sme naimplementovali v metóde SelectTemplates a pozostáva z toho, že v cykle postupne globálne zarovnávame target sekvenciu so sekvenciou z databázy sekvencií (v našom prípade zložka s fasta súbormi). Následne zarovnanie validujeme viacerými validáciami. Najpv validujeme, či vôbec budeme môcť uvažovaný template použiť podľa toho, že neobsahuje príliš veľa neznámych nukleotidov (ak bol fasta súbor generovaný z pdb, znamená to chýbajúce nukleotidy v pdb). My sme určili minimálny limit 95% platných nukleotidov. Následne validujeme, podľa samotnej podobnosti target sekvencie a potencionálnej template sekvencie. Nami určený interval podobnosti je v tomto prípade 60% - 90% (minimum vychádza z pokusov z bakalárskej práce, maximum je nastavené empiricky tak, aby sa target a template dostatočne líšili). Nakoniec validujeme potencionálnu template sekvenciu medzi už vybranými template sekvenciami. S každou z nich môže mať podobnosť maximálne 90%. Ak zarovnanie prejde validáciou, štruktúru si označíme ako vhodnú pre predikciu a pokračujeme v cykle. Štruktúru taktiež považujeme za vhodnú ako template iba vtedy, ak žiadnu veľmi podobnú štruktúru už takto označenú nemáme (podobnosť maximálne 90%). Ukončovacie kritérium pre cyklus je buď prejdenie všetkých štruktúr a vrátenie požadovaného počtu najlepších, alebo nájdenie prvých *returnFirstSuitableX* štruktúr splňujúce parametre hľadania.

Takto môže užívateľ predikovať štruktúru podľa viacerých odlišných template štruktúr vo viacerých nezávislých predikciách, výsledky potom manuálne porovnať a vybrať ten najlepší. Nevýhodou vyhľadávania vhodných template štruktúr takýmto spôsobom je časová náročnosť, kedy vyhľadanie v našej databáze môže trvať niekoľko minút, čo je výrazné spomalenie. Takýto problém by bolo možné vyriešiť použitím rýchleho heuristického zarovnania typu BLAST alebo FASTA, no vzhľadom na trvanie predikcie FARFAR a predpokladu, že táto funkcionalita nebude využívaná na hromadnú predikciu štruktúr, sme to nepovažovali za

prioritu. Z pohľadu asymptotickej náročnosti platí všetko, čo sme o nej povedali v rozbore v tretej kapitole a okrem toho na začiatok pridáme zložitosť metódy SelectTemplates, ktorá je $O(nl^2)$, kde n označuje počet štruktúr v databáze a l označuje dĺžku štruktúry v databáze. Celková asymtotická zložitosť algoritmu sa teda zvýšila na $O(nl^2)$.

4.8 Úprava vstupných dát

V tejto sekcii vysvetlíme chyby objavujúce sa vo vstupných dátach a problémy pre algoritmus z toho plynúce. Následne budeme generovať fasta súbory z pdb súborov preto, aby sme vedeli určiť molekulu nepoužiteľnú pre Trooper a vyhli sa tak problémom, ktoré by nastali neskôr v algoritme počas predikcie.

Fasta a pdb stiahnutéi zo stránok ProteinDataBank sú experimentálne získane štruktúry, ktoré často nie sú kompletné, alebo obsahujú chyby. Problém pre náš algoritmus nastáva hlavne v prípade, kedy sekvencia fasta súboru nesedí so sekvenciou extrahovanou z pdb súboru. Na základe dát, ktorých modifikácia je popísaná v tejto sekcii, sme testovali posledné vylepšenie algoritmu, teda použitie viacerých template štruktúr. Generovanie modifikovaných dát je naimplementované v priečinku s relatívnou cestou Predictor/MyTools/CreateFastasFromPdb.

Principiálne problémy s rozdielnymi sekvenciami v našom algoritme nastávajú v troch konkrétnych prípadoch:

1. Pri hľadaní vhodnej template štruktúry pre predikciu (funkcionalita automatického vyhľadávanie template štruktúry).

2. Pri mapovaní temlate štruktúry na konzervované úseky plynúce zo zarovnania sekvencií. Bod 5

3. Pri hľadaní a mapovaní sekundárných template štruktúr.

Všetky tri situácie majú rovnakú príčinu a tou je, že algoritmus bude potrebovať namapovať template pdb štruktúru na template fasta sekvenciu. V prípade, že v pdb súbore chýba skupina nukleotidov a zároveň číslovanie nukleotidov oproti fasta súboru nesedí, môže existovať viacero možných mapovaní štruktúry na sekvenciu a my nie sme schopní určiť, ktoré mapovanie je správne.

V prípade že nastane takáto situácia, algoritmus nedokáže pokračovať v predikcii. Doteraz sme to riešili tak, že sme sa pokúsili template fasta sekvenciu posunúť pridaním dummy reziduí na jej začiatok (predpoklad, že z FASTA sekvencie chýba na začiatku pár reziduí a ďalej sa už bude poradie nukleotidov zhodovať s tým v pdb súbore) a v prípade, že sa nám niečo takéto nepodarilo nájsť, danú štruktúru sme ako template nepoužili.

Rozhodli sme sa skúsiť eliminovať tento problém vygenerovaním fasta sekvencie z pdb súborov. Sitáciu nám však komplikuje častá nekompletnosť pdb súborov, kedy sa pravidelne stáva, že začiatok, prípadne koniec štruktúry chýba. Chýbajúce úseky znova nahradzujeme dummy reziduami (písmeno X). Takéto štruktúry môžu byť následne bez problémov použité ako template alebo sekundárny template pri predikcii, pretože pridaný dummy nukleotid X sa nezarovná na target sekvenciu a bude neskôr dopredikovaný. Z princípu sekvencia obsahujúca takéto

dummy nukleotidy nemôže byť predikovaná, keďže nevieme, aký nukleotid máme na dané miesto vlastne predikovať.

Okrem problému s mapovaním štruktúry a sekvencie sme riešili aj duplicitu rôzne pomenovaných, ale pritom rovnakých štruktúr. Tomu sme sa v pôvodnej implementácii vyhli, pretože sme si dopredu testovacie dáta rozdelili podľa podobnosti zarovnaní ich sekvencií, a tie sme vzájomne spárovali. Teda ako template sme používali len dosť odlišné template sekvencie od target sekvencie, nakoľko predikovať sekvenciu pomocou inej so skoro 100% zhodou by bolo triviálne. V prípade, že hľadáme vhodný template, sa však chceme vyhnúť zbytočnému zarovnávaniu sekvencií, a preto si pri generovaní fasta súborov rovno podobné odfiltrujeme. Docielime to jednoducho tak, že pri vygenerovaní novej fasta sekvencie ju globálne zarovnáme na všetky už vygenerované fasta sekvencie, a ak podobnosť presiahne istú konštantnú hranicu (použíli sme 97,5%), tak takúto sekvenciu nepridáme medzi už vygenerované sekvencie.

5. Predikcia sekundárnej štruktúry

Na zlepšenie rýchlosti a presnosti predikovania nekonzervovaných úsekov sme ako prvé opatrenie navrhli riešiť najprv jednoduchší problém, a to predikciu sekundárnej target štruktúry, a pomocou nej neskôr predikovať terciárnu štruktúru nekonzervovaných úsekov. To by malo viesť k zmenšeniu prehľadávaného priestoru pri generovaní kandidátskych štruktúr algoritmom FARFAR. Principiálne teda najprv komparatívnym modelovaním napredikujeme sekundárnu target štruktúru, a tú potom poskytneme ako vstup algoritmu FARFAR, ktorý ju použije pri predikcii terciárnej target štruktúry. Tento postup a jeho výsledky sme popísali aj v článku Galvanek a Hoksza (2017).

5.1 Príprava dát

Pretože algoritmus doplníme o komparatívne predikovanie sekundárnej štruktúry, nebude na vstupe potrebovať len sekvenciu target molekuly a terciárnu štruktúru spolu so sekvenciu template molekuly, ale aj sekundárnu štruktúru template molekuly. Sekundárnu štruktúru očakávame v dot-bracket notácii, ktorú sme popísali v druhej kapitole. Sekundárnu štruktúru by bolo možné získať z terciárnej template štruktúry aj počas predikcie, avšak v prípade, kedy si užívateľ chce dodať vlastnú sekundárnu štruktúru, je preňho výhodnejšie, keď je sekundárna štruktúra dodaná ako ďalší vstupný parameter.

Získať dáta sme sa rozhodli ich určením z terciárnej štruktúry programom DSSR (Defining the Secondary Structures of RNA) zo software balíku x3DNA Lu a Olson (2003). Na hromadné extrahovanie štruktúr sme si potom vytvorili dva krátke scripty, ktoré len prejdú všetky pdb súbory v databáze, spustia DSSR a uložia výsledok. DSSR vytvára v sekundárnej štruktúre base-pairs vyznačené jednoduchými zátvorkami, ako aj pseudo uzly vyznačené hranatými zátvorkami Obrázok 5.2. Z dôvodov implementačných komplikácií a zladenia s algoritmom FARFAR sme sa rozhodli pracovať iba so sekundárnou štruktúrou označujúcou base-pairs, a preto pseudo uzly neberieme do úvahy.

5.2 Pseudokód

Pseudokód algoritmu predikujúceho sekundárnu štruktúru spolu s integráciou do pôvodného algoritmu, ktorý vychádza z pseudokódu algoritmu Trooper v kapitole 3 (Sekcia 3.5).

```
1 GGAUCUUCGGGGCAGGGUGAAAAUUCCCGACCGGUGGUAUAGUCCACGAAAGUACGCUUUGAUUUGGUGAAAUUCCAAAACCGACAGUAGAGUCUGGAUGAGAGAAGAUUC
2 (((((((.....(((.....))).[[(((((((....[[)))).((((..)))).((((.....)))))]])).(((....]]))).....)))))))
```

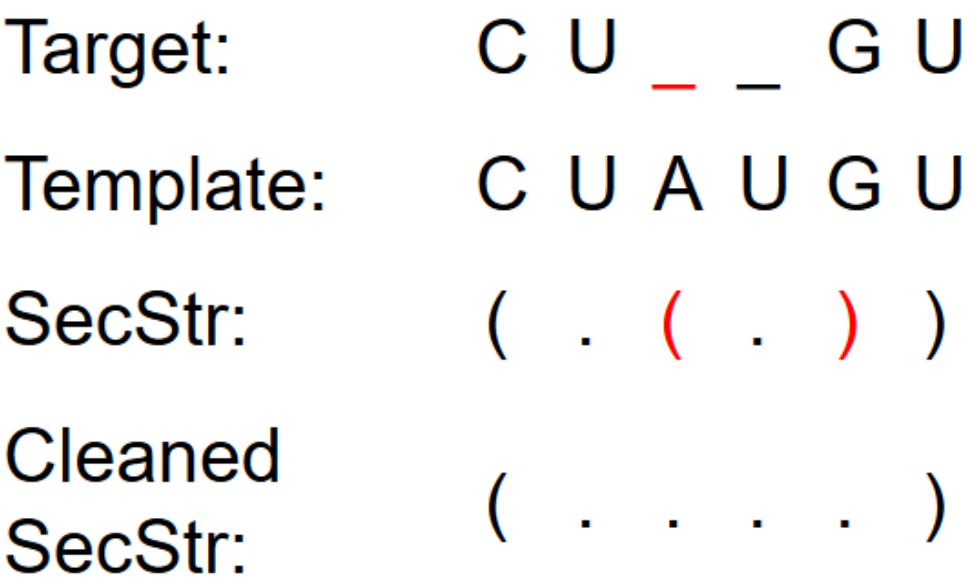
Obrázek 5.1: Príklad súboru sekundárnej štruktúry získanej progrmaom DSSR zo štruktúry terciárnej. V prvom riadku sú príslušné typy nukleotidov a v druhom riadku je samotná sekundárna štruktúra v dot-bracket reprezentácii.

```
Main(fastaTarget, fastaTemplate
        , secStrTemplate, pdbTemplate)
{
    CheckTemplateMapping
        (fastaTemplate,pdbTemplate)
    alignment := Align
        (fastaTarget, fastaTemplate)
    alignment1 := UseSlidingWindow
        (alignment)
    alignment2 := ProcessGaps
        (alignment1)
    conservedParts := CopyConservedParts
        (alignment2, pdbTemplate)
    mappedConservedParts := MapConservedParts
        (conservedParts, alignment2)
    conservedPartsSecStr := CopyConservedPartsSecStr
        (alignment2, secStrTemplate)
    cleanedConservedPartsSecStr:= CleanConservedPartsSecStr
        (conservedPartsSecStr, alignment2)
    mappedConservedPartsSecStr := MapConservedPartsSecStr
        (cleanedConservedPartsSecStr, alignment2)
    predictedSecStr := PredictUnconservedSecStr
        (mappedConservedPartsSecStr)
    longParts[] := ProcessLongUnconservedParts
        (mappedConservedParts, secStr)
    shortParts[] := ProcessShortUnconservedParts
        (mappedConservedParts, secStr)
    predictedParts[] := PredictUnconservedParts
        (longParts[], shortParts[])
    finalModel := ConnectPredictedParts
        (predictedParts)
}

CopyConservedPartsSecStr(alignment, secStr)
{
    conservedPartsSecStr = ""
    foreach nucleotide in secStr
    {
        if (IsConserved(nucleotide, alignment) and
            CorrespondingPairIsConserved)
           conservedParts += nucleotide
    }
    return conservedParts
}

CleanConservedPartsSecStr(conservedPartsSecStr, aln2)
{
    remove base-pairs from conservedPartsSecStr, which
```

```
49     were aligned to gaps in target structure
50 }
51
52 MapConservedPartsSecStru(conservedPartsSecStr, alignment)
53 {
54     map nucleotide id's from current state (nc with
55     id x corresponds to x-th nc in template fasta) to
56     state where nc id correctly corresponds to
57     nucleotide in target fasta
58     return modifiedConservedSecStrParts
59 }
60
61 PredictUnconservedSecStr(mappedConservedPartsSecStr)
62 {
63      input = Tuple(targetFasta, mappedConservedPartsSecStr)
64      predictedSecstr+= CallRNAFold(input)
65      return predictedSecstr
66 }
67
68 ProcessLongUnconservedParts
69     (conservedParts, ncAvrgLength, minLenGapLimit, secStr)
70 {
71     longParts = []
72     foreach gap in conservedParts
73     {
74         if length(gap) <= minLenGapLimit
75             continue
76         startNc := conserved nucleotide before gap
77         endNc := conserved nucleotide after gap
78         ph="phosphor"
79         s := FindMiddle(startRes[ph], endRes[ph])
80         p := length(gap) * ncAvrgLength / 2
81         ncsInSphere := all nucleotides inside sphere(p, s)
82         secStrInSphere := all secstr pairs(ncsInSphere)
83         ncsInSphere += add missing nucleotides from
84             secStrInSphere, when second nucleotide from
85             pair is missing in ncsInSphere
86         preparedPart := PrepareCorectFormatForFARFAR
87             (ncsInSphere, secStrInSphere)
88         longParts[] += preparedPart
89     }
90     return longParts[]
91 }
```



Obrázek 5.2: Príklad problému, ktorý môže nastať pri mapovaní sekundárnej štruktúry na zarovnanie. Z obrázku vidíme, že tretí a štvrtý nukleotid template štruktúry, sú zarovnané na medzery v target štruktúre. Sekundárna template štruktúra, ktorú potrebujeme namapovať na zarovnanú target štruktúru obsahuje base-pair tvorený tretím a piatym nukleotidom (vyznačené červenou farbou). Z dôvodu, že tretí nukleotid template štruktúry je zarovnaný na medzeru v target štruktúre, musíme celý base pair so sekundárnej štruktúry odstrániť.

5.3 Predikcia sekundárnej štruktúry

Sekundárnu štruktúru predikujeme rovnakým spôsobom ako štruktúru terciárnu, teda komparatívnym modelovaním. Ako vstup potrebuje algoritmus dostať sekvenciu target molekuly, sekvenciu, sekundárnu a terciárnu štruktúru template molekuly.

Používame rovnakú template štruktúru ako pre terciárnu predikciu, a preto sa algoritmus predikcie sekundárnej štruktúry zhoduje s algoritmom predikcie terciárnej štruktury až po získanie konzervovaných úsekov v zarovnaní (teda metódu *ProcessGaps*). Následne potrebujeme na zarovnanie namapovať sekundárnu template štruktúru. Tu musíme riešiť nasledujúcu situáciu: ak práve jeden nukleotid z base-pair z template sekvencie je zarovnaný na medzeru v target sekvencii a druhý je zarovnaný na nukleotid, musíme v sekundárnej template štruktúre preznačiť oba nukleotidy ako nespárované, inak by namapovaná sekundárna štruktúra bola neplatná Obrázok 5.2. Tiež to môžeme chápať tak, že sekundárnu štruktúru udržujeme stále správne uzátvorkovanú, čo znamená, že zátvorky rovnakého typu sa navzájom nekrížia, a máme stále rovnaký počet pravých aj ľavých zátvoriek v sekundárnej štruktúre.

Namapovaním sekundárnej štruktúry na alignment získame neúplnú sekundárnu štruktúru target molekuly. Túto následne predáme spolu s target sekvenciou nástroju RNAfold Gruber AR (2008), ktorý predikuje sekundárnu štruktúru vzhľadom na obmedzenia plynúce z dodanej sekundárnej štruktúry.

5.4 Integrácia do existujúceho algoritmu

Do konfiguračného súboru sme pridali nastavenie, ktoré určuje, či algoritmus pracuje aj so sekundárnou štruktúrou, alebo nie.

Z pohľadu začlenenia do algoritmu sa sekundárna štruktúra predikuje po kroku mapujúcom terciárnu template štruktúru na alignment, ako môžeme vidieť aj v pseudokóde. V ďalšom kroku, ktorý ošetruje dlhé medzery, už namapovanú sekundárnu štruktúru používame na to, aby sme k fragmentom terciárnej štruktúry slúžiacej ako template pre FARFAR vybrali aj odpovedajúce fragmenty sekundárnej štruktúry. Keďže FARFAR-u musíme dať validnú sekundárnu štruktúru v dot-bracket notácii, v prípade, ak je do predikcie dlhého nekonzervovaného úseku vybraný z base-pairu v sekundárnej štruktúre práve jeden nukleotid, musíme pridať aj druhý, aby sme zachovali validitu sekundárnej štruktúry. To isté rovnakým spôsobom riešime v ďalšom kroku, keď pripravujeme predikciu zvyšku štruktúry už bez dlhých nekonzervovaných úsekov.

Nakoniec sme upravili shell scripty tak, aby nakopírovali na správne miesta súbory so sekundárnymi štruktúrami, a v parametri predali príslušný template súbor so sekundárnou štruktúrou algoritmu FARFAR.

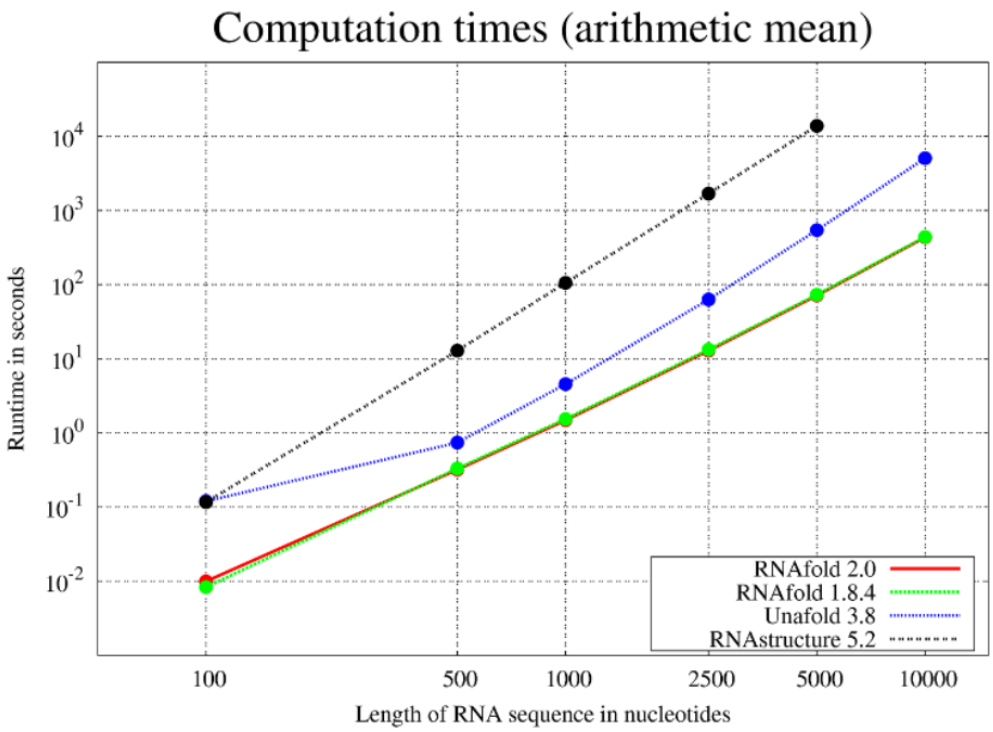
5.5 Časová zložitosť

V tejto sekcií popíšeme, ako sa zmenila časová zložitosť v porovnaní s pôvodným algoritmom Trooper. Principiálne do predikcie pribudli len metódy prípravy predikcie sekundárnej štruktúry a samotná jej predikcia. Metódy pripravujúce predikciu sekundárnej štruktúry majú časovú zložitosť zhodnú s podobnými metódami z prípravy predikcie terciárnej štruktúry. Jediná odlišná metóda je *PredictUnconservedSecStr*, ktorá volá program RNAfold. Pri ňom autori na stránkach uvádzajú iba analýzu trvania predikcie na rôzne dlhých štruktúrach Obrázok 5.3. Asymptotickú zložitosť teda presne určiť nevieme, ale ak bude trend rastu časovej náročnosti v závislosti na dĺžke predikovanej molekuly pokračovať podľa obrázku, mal by sa algoritmus zmestiť do $O(l^3)$, kde l označuje dĺžku predikovanej štruktúry. Z obrázku ďalej vidíme, že reálny čas potrebný na predikciu štruktúr dlhých 100-500 nukleotidov sa pohybuje do jednej sekundy. Zhrnúť to môžeme tak, že asymptotická časová náročnosť sa zhoršila z $O(l^2)$ na $O(l^3)$, ale reálna časová náročnosť pri predikcii štruktúr dlhých 50-500 nukleotridov sa zhoršila o pár jednotiek sekúnd.

5.6 Experiment a Výsledky

Za účelom overenia vplyvu pridania predikcie sekundárnej štruktury do pôvodného algoritmu sme urobili rovnaké testovanie nad rovnakými dátami, ako pri porovnávaní pôvodného algoritmu s ModeRNA Sekcia 4.5. Skúšali sme napredikovať všetky páry s podobnosťou medzi 60% - 90% a dĺžkou 50-100 a 101-500 nukleotidov najprv pôvodnou verziou algoritmu Trooper, verziou algoritmu vylepšenou o predikciu sekundárnej štruktúry a referenčnou predikciou ModeRNA.

Výsledky a porovnanie predikcií uvádzame v tabuľke Tabuľka 5.1. Z výsledkov vidíme, že pridanie predikcie sekundárnej štruktúry spôsobilo, že priemerná



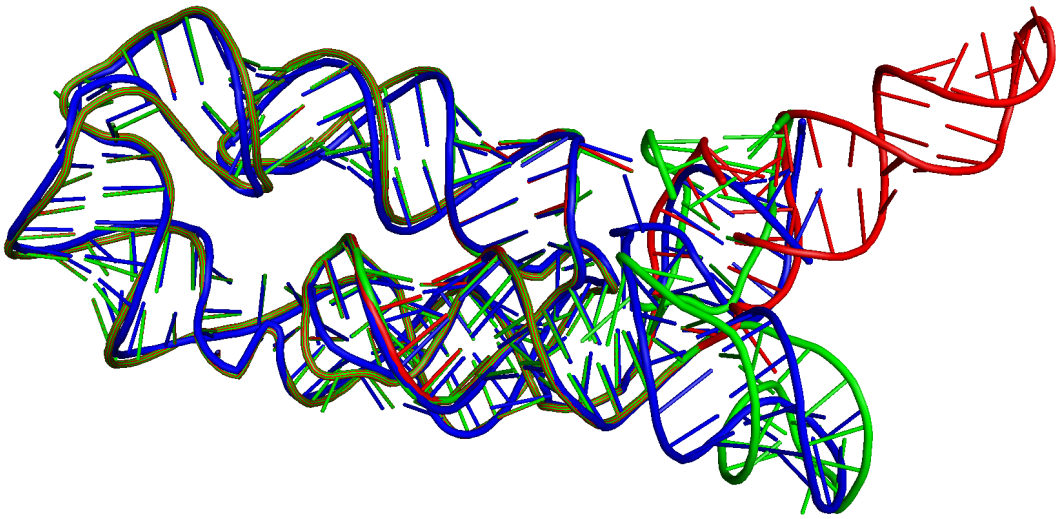
Obrázek 5.3: Výsledky analýzy dĺžky behu programu RNAfold, v závislosti na dĺžke sekvencie predikovanej sekundárnej štruktúry. Obrázok bol 13.07.2019 stiahnutý z `https://www.tbi.univie.ac.at/RNA/`.

RMSD pre štruktúry s dĺžkou medzi 50-100 nukleotidov na 413 úspešne napredikovaných párach sa zhoršila z 5,80Å na 8,23Å, čo približne zodpovedá výsledku 8,53Å, ktoré na predikcii rovnakých štruktúr dosiahla ModeRNA. Pri predikcii štruktúr dĺžky 101-500 nukleotidov a 98 úspešne napredikovaných párach sa naopak priemerné výsledky predikcie s pridaním predikcie sekundárnej štruktýry výrazne zlepšili a to z 6,91Å na 3,46Å, čo je o niečo lepšie než výsledky ModeRNA, ktorá pri rovnakých podmienkach dosiahla priemer 3,72Å.

Z výsledkov je teda zrejmé, že pridanie predikcie sekundárnej štruktúry a jej poskytnutie algoritmu FARFAR v niektorých prípadoch výsledky zlepšilo a v niektorých zhoršilo. Prečo boli zlepšené práve dlhšie štruktúry by sme mohli vysvetliť tak, že dlhšie štruktúry môžu obsahovať dlhšie nekonzervované úseky, ktorých predikcia je pre FARFAR náročnejšia. Príklad z obrázka Obrázok 5.4 práve podporuje takúto teóriu, kedy vidíme, že pôvodná predikcia FARFAR bez sekundárnej štruktúry napredikovala nekonzervovaný úsek zle, ale po pridaní sekun-

Veľkosť	Počet párov	Trooper bez SS	Trooper s SS	ModeRNA
50-100	413	5,80	8,23	8,53
101-500	98	6,91	3,46	3,72
50-500	511	5,95	7,31	7,61

Tabulka 5.1: Porovnanie priemernej RMSD predikcie RNA nástrojmi Trooper, Trooper s predikciou sekundárnej štruktúry a ModeRNA.



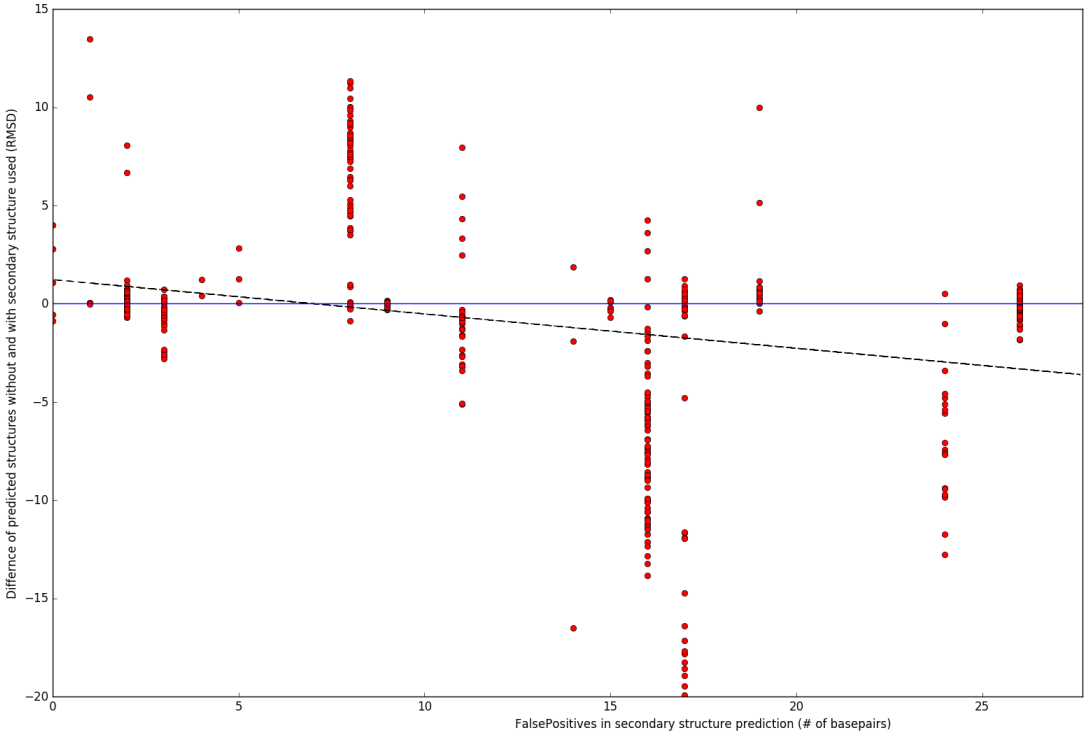
Obrázek 5.4: Modrá štruktúra je experimentálne získana štruktúra molekuly 3DIG:X s dĺžkou 175 nukleotidov. Červená štruktúra je predikovaná algoritmom Trooper bez použitia sekundárnej štruktúry s výslednou RMSD na úrovni 14,44Å. Zelená štruktúra je predikcia urobená algoritmom Trooper s výslednou RMSD 4,44Å. Pre obe predikcie bol použitý rovnaký template, a to štruktúra 3DOU:A.

dárnej štruktúry sa FARFAR-u podarilo úsek napredikovať oveľa presnejšie.

Ďalej sme ešte skúmali značné zhoršenie výsledkov v triede štruktúr veľkostí 50-100 nukleotidov z RMSD 5,8Å na 8,23Å. Zastávame hypotézu, že by to mohlo byť spôsobené nesprávnou sekundárnou štruktúrou dodanou algoritmu FARFAR, pretože inak boli podmienky oboch predikcií rovnaké, a teda by sa priemerné výsledky nemali od seba značne líšiť.

Správnosť predikovanej sekundárnej štruktúry vieme popísať a klasifikovať štyrma mierami: true positives (správne určený existujúci base-pair), true negatives (správne určený nespárovaný nukleotid), false positives (nesprávne určený base-pair) a nakoniec false negative (v štruktúre by mal existovať base-pair, ale nie je napredikovaný). Z týchto štyroch ukazateľov sú prvé dva pozitívne a druhé dva negatívne. Pritom ale FARFAR-u môže uškodiť iba prípad false negative, pretože ten mu indikuje, že má v predikcii spárovať (a teda umiestniť blízko seba) dva nukleotidy. Prípady s false negatives by predikciu FARFAR-u nemali zlepšiť, ale ani zhoršiť, pretože nekladú žiadnu podmienku na spárovanie nukleotidov.

Zamerali sme sa preto na prípady false positives v predikovaných sekundárnych štruktúrach a analyzovali sme koreláciu medzi zhoršením, prípadne zlepšením predikcie algoritmu Trooper bez sekundárnej štruktúry a algoritmu Trooper s predikciou sekundárnej štruktúry vzhľadom na počet false positives v napredikovanej sekundárnej štruktúre dodanej ako vstup algoritmu FARFAR. Výsledok je graficky znázornený na obrázku Obrázok 5.5 a podporuje našu hypotézu o tom, že čím viac vzrastal počet false positives v predikovanej sekundárnej štruktúre, tým bolo pravdepodobnejšie, že predikcia so sekundárnou štruktúrou sa oproti predikcii bez sekundárnej štruktúry zhorší. Preto si myslíme, že zhoršenie v niektorých



Obrázek 5.5: Závislosť medzi rozdielom výsledkov predikcie s a bez sekundárnej štruktúry a počtom false positive base-pairs v napredikovanej sekundárnej štruktúre. Červené body sú jednotlivé záznamy a čiarkovanou čiarou je zobrazená lineárna regresia týchto bodov.

preikciách bolo spôsobené hlavne zle napredikovanou sekundárnou štruktúrou.

6. Použitie viacerých template štruktúr pri predikcii

Ako druhé opatrenie, na zlepšenie presnosti predikovania nekonzervovaných úsekov sme navrhli použiť viacero template štruktúr na predikciu jednej target štruktúry tak, že ich použijeme spolu na vytvorenie jednej štruktúry konzervovaných úsekov. Očakávali sme pritom, že sa nám podarí zmenšiť počet a dĺžku nekonzervovaných úsekov v predikovanej štruktúre a tým výrazne zjednosušiť ich predikciu algoritmom FARFAR.

6.1 Výber sekundárnych template štruktúr

Hlavnou myšlienkou algoritmu je použiť viacero template štruktúr pre predikciu jednej target štruktúry. Tým môžeme dosiahnuť väčšiu percentuálnu mieru konzervovaných úsekov v štruktúre, čo by malo zjednodušiť následnu predikciu nekonzervovaných úsekov.

Existuje viacero spôsobov, ktorými je možné predikovať target molekulu pomocou viacerých template molekúl. Jeden prístup by mohol byť rozdeliť si target sekvenciu na regióny a pre predikciu každého regiónu použiť inú template štruktúru. My sme sa aj vzhľadom na jednoduchšiu implementáciu do už existujúceho algoritmu rozhodli postupovať tak, že používame jednu template štruktúru ako primárnu (hlavnú) a ďalšie template štruktúry, ako sekundárne (vedľajšie), ktoré sú použité na vyplnenie dlhých nekonzervovaných úsekov (to sú tie, ktoré v pôvodnom algoritme vyčleňujeme do samostatných predikcií v metóde *ProcessLongUnconservedParts*).

Potencionálne spôsoby ako nájsť vhodné sekundárne template štruktúry pre nekonzervované úseky sú:

1. Globálne zarovnanie potenciálnych sekundárnych temeplate molekúl s target molekulou.

2. Lokálne zarovnanie potenciálnych sekundárnych template molekúl s target molekulou.

3. Semiglobálne zarovnanie potenciálnych sekundárnych temeplate molekúl s target molekulou.

4. Najprv zarovnať heuristickým algoritmom (BLAST, FASTA) Stephen F. Altschul (1994) a následne z takto vybranej skupiny najlepších štruktúr vybrať tú najvhodnejšiu za pomoci jedného z troch hore uvedených spôsobov.

My sme najprv skúšali prvý a najjednoduchší postup, a to globálne zarovnávať potencionálne sekundárne template molekuly na target molekulu, pričom hľadáme takú sekundárnu template štruktúru, ktorá by dobre vyplnila nekonzervované úseky (teda by ich pokryla aspoň na 60%, ktoré sme na základe doterajšieho testovania určili ako dolnú hranicu pokrytia v zarovnaní). Pri globálnom zarovnaní sme však nenachádzali vhodné sekundárne template štruktúry, ktoré by

pokrývali nekonzervované úseky s aspoň 60%. Je to dané tým, že okrem toho, že v sekundárnej template štruktúre sa musí nachádzať vhodný konzervovaný úsek, musí byť aj na správnom mieste v sekvencii tak, aby bol globálne zarovnaný na miesto nekonzervovaného úseku v target sekvencii. Tento prístup sme nakoniec označili za nepoužiteľný.

Lokálne zarovnanie hľadá zarovnanie s najlepším skóre dvoch podúsekov z target aj template sekvencie. To znamená, že dĺžka zarovnaných úsekov je určená najvyšším skóre zarovnania oboch sekvencií (ak by bol do zarovnania pridaný alebo odobratý ľubovolný nukleotid z target alebo template sekvencie skóre zarovnania by sa zhoršilo). Vzhľadom na to, že my potrebujeme zarovnať celý nekonzervovaný úsek target sekvencie na ľubovoľný úsek template sevencie bolo by lokálne zarovnanie ťažko použiteľné.

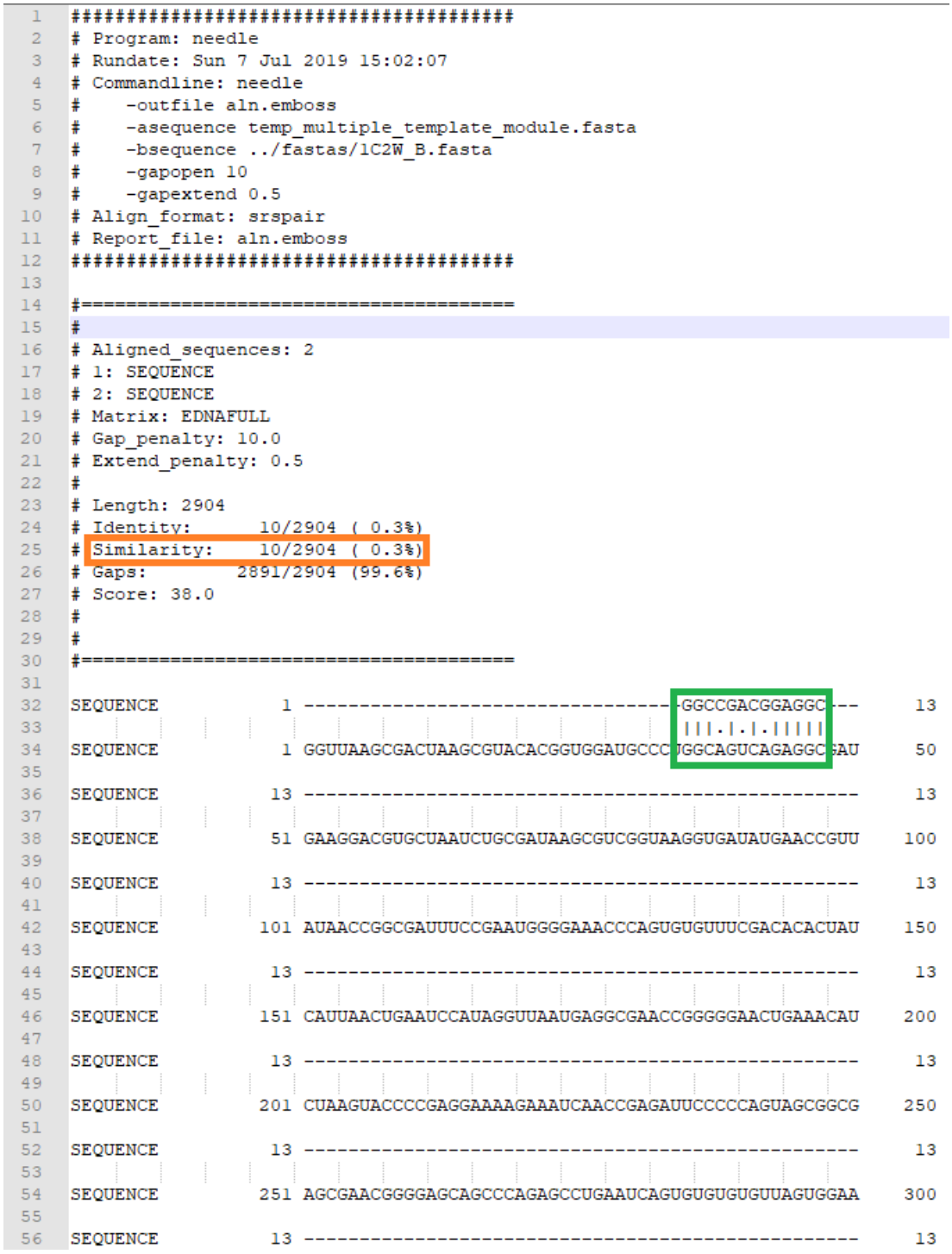
Semiglobálne zarovnanie je kombinácia globálneho a lokálneho zarovnania. Funguje tak, že kratšiu štruktúru zarovná na časť dlhšej štruktúry a nezarovnané úseky pred a po kratšej štruktúre sa do výsledného skóre zarovnania nezapočítavajú. Teda z globálnym zarovnaním má spoločné to, že kratšia sekvencia je zarovnaná celá na časť dlhšej. Z lokálnym zarovnaním má spoločné to, že pridaním alebo odobratím ďalšieho nukleotidu z dlhšej štruktúry do zarovnania by sme výsledné skóre len zhoršili. V našom algoritme ho budeme používať na vyľadanie optimálnych sekundárnych template molekúl. Postupovať budeme tak, že vyberieme nekonzervovaný úsek z target sekvencie a postupne ho budeme zarovnávať na rôzne potencionálne template sekvencie. Buď môžeme prejsť všetky dostupné sekvencie pre každý nekonzervovaný úsek a vybrať najlepšiu, alebo vyberieme prvú ktorá splní nejaké nami zadané požiadavky. My sme zvolili spôsob, kedy vyberieme prvú vhodnú template štruktúru, ktorú nájdeme.

Semiglobálne zarovnanie získavame rovnakým nástrojom ako globálne, teda pomocou programu Emboss.Needle Rice P. a A. (2000). Z výsledného zarovnania nás nezaujíma skóre zarovnania, ale percentuálny pomer správne zarovnaných nukleotidov oproti nesprávne zarovnaným nukleotidom, prípadne medzerám v zarovnaní (bez medzery na začiatku a na konci), čo označujeme ako podobnosť, alebo tiež similarity. Emboss ale do výstupného súboru zarovnania uvádza len globálnu podobnosť v kombinácií so semiglobálnym skóre Obrázok 6.1. To pre nás znamenalo, že sme museli naimplementovať parser, ktorý je založený na regulárnych výrazoch, vyberie zo zarovnania len zarovnanú časť a na základe toho určí percentuálnu podobnosť zarovnania.

6.2 Algoritmus

Kroky algoritmu používajúceho viacero sekundárnych template štruktúr:

1. Nájdeme dlhé nekonzervované úseky *IdentifyLongUnconservedRegions*.
2. Otagujeme a vyfiltrujeme potenciálne template štruktúry sekvenčným prechodom všetkých štruktúr, uložíme ich do poľa *FindPotentialTemplates*.
3. Pre každý nekonzervovaný úsek:
   (a) Upravíme nekonzervovaný úsek na rozšírený nekonzervovaný úsek (o úseky, ktorými budeme spájať s primárnou target štruktúrou) *ExtendGap*.



```
########################################
# Program: needle
# Rundate: Sun 7 Jul 2019 15:02:07
# Commandline: needle
#    -outfile aln.emboss
#    -asequence temp_multiple_template_module.fasta
#    -bsequence ../fastas/1C2W_B.fasta
#    -gapopen 10
#    -gapextend 0.5
# Align_format: srspair
# Report_file: aln.emboss
########################################

#=======================================
#
# Aligned_sequences: 2
# 1: SEQUENCE
# 2: SEQUENCE
# Matrix: EDNAFULL
# Gap_penalty: 10.0
# Extend_penalty: 0.5
#
# Length: 2904
# Identity:      10/2904 ( 0.3%)
# Similarity:    10/2904 ( 0.3%)
# Gaps:        2891/2904 (99.6%)
# Score: 38.0
#
#
#=======================================

SEQUENCE           1 ----------------------------------GGCCGACGGAGGC---     13
                                                       |||.|.|.|||||
SEQUENCE           1 GGUUAAGCGACUAAGCGUACACGGUGGAUGCCCUGGCAGUCAGAGGCGAU     50

SEQUENCE          13 -------------------------------------------------     13

SEQUENCE          51 GAAGGACGUGCUAAUCUGCGAUAAGCGUCGGUAAGGUGAUAUGAACCGUU    100

SEQUENCE          13 -------------------------------------------------     13

SEQUENCE         101 AUAACCGGCGAUUUCCGAAUGGGGAAACCCAGUGUGUUUCGACACACUAU    150

SEQUENCE          13 -------------------------------------------------     13

SEQUENCE         151 CAUUAACUGAAUCCAUAGGUUAAUGAGGCGAACCGGGGGAACUGAAACAU    200

SEQUENCE          13 -------------------------------------------------     13

SEQUENCE         201 CUAAGUACCCCGAGGAAAAGAAAUCAACCGAGAUUCCCCCAGUAGCGGCG    250

SEQUENCE          13 -------------------------------------------------     13

SEQUENCE         251 AGCGAACGGGGAGCAGCCCAGAGCCUGAAUCAGUGUGUGUGUUAGUGGAA    300

SEQUENCE          13 -------------------------------------------------     13
```

Obrázek 6.1: Príklad semiglobálneho zarovnania nekonzervovaného rozšíreného úseku dlhého 13 nukleotidov na sekvenciu 1C2W_B dlhú 2904 nukleotidov. Zeleným obdĺžnikom je ohraničené zarovnanie, ktorého podobnosť nás zaujíma a oranžovým obdĺžnikom je označená hodnota globálnej podobnosti vypočítanej programom Emboss.

(b) Pomocou semiglobálneho zarovnania nájdeme v indexovaných štruktúrach vhodnú sekundárnu štruktúru pre nekonzervovaný úsek *FindTemplateForGap*.

(c) Namapujeme vybraný fragment sekundárnej target štruktúry na príslušné miesto v template štruktúre (preindexovanie nukleotidov v sekundárnej template štruktúre) *MapTemplateToGap*.

(d) Vykonáme jeden z následujúcich dvoch krokov, podľa verzie algoritmu *IntegrateNewTemplateStr*:

   i. Dáme do superpozície štruktúru tvorenú konzervovanými úsekmi z primárnej template štruktúry a fragment zo sekundárnej template štruktúry.

   ii. Uložíme fragment sekundárnej template štruktúry do separátneho súboru, ktorý potom poskytneme algoritmu FARFAR.

Logika algoritmu je naimplementovaná v metóde *MultipleTemplateModule*. Superimposer použitý v metóde *IntegrateNewTemplateStr* bol pôvodne inšpirovaný skriptom od Anders S. Christensen-a dostupného na `https://gist.github.com/andersx/6354971` a výrazne upravený.

V prvom kroku sekvenčne prejdeme target štruktúru skladajúcu sa zatiaľ len z konzervovaných úsekov primárnej template štruktúry a vyhľadáme nekonzervované úseky dlhšie ako určitá hranica daná parametrom (momentálne nastavené na 7 nukleotirdov).

V druhom kroku si predpripravíme potenciálne template štruktúry tak, že odstránime tie, ktoré obsahujú viac ako 5% neplatných nukleotidov (znaky iné ako A,C,G,U). Hodnotu tohoto parametra sme zistili empiricky, počas ladenia algoritmu. Okrem toho si uložíme dĺžku každej sekvencie a v nasledujúcom kroku budeme na seba zarovnávať len dosť dlhé sekvencie vzhľadom k nekonzervovanému úseku (minimálne 0,8 násobok dĺžky úseku). Dôvody sú vyhnutie sa problémovým dátam a ušetrenie času pri zarovnávaní hlavne dlhších nekonzervovaných úsekov. Bolo by možné si potenciálne template sekvencie predpripraviť mimo algoritmu predikcie, potom by ale bolo treba takto pripravené dáta prepočítavať pri zmene v databázi. Vzhľadom na to, že celý tento proces trvá pár sekúnd rozhodli sme sa robiť to vždy odznova počas predikcie.

V ďalších krokoch iterujeme cez všetky vybrané nekonzervované úseky sekvencie a pre každý aplikujeme rovnaké kroky. Najprv si nekonzervovvaný úsek rozšírime. To znamená, že k nemu pridáme z oboch strán určitý počet nukleotidov (my používame parameter 5). Keďže pracujeme s nekonzervovaným úsekom musí existovať na každej strane minimálne 1 nukleotid, ktorý tento úsek ohraničuje (s výnimkou začiatku a konca sekvencie). Za pomoci týchto rozširujúcich nukleotidov sme schopní superpoziciovať (správne umiestniť) v nasledujúcich krokoch vybraný fragment sekundárnej template štruktúry do primárnej template štruktúry.

V kroku 3b postupne zarovnávame rošírený nekonzervovaný úsek s vyhovujúcimi predpripravenými štruktúrami z druhého kroku. V prípade, že zarovnanie splňuje určené podmienky, vyberieme príslušnu štruktúru za sekundárny template pre spracovávaný nekonzervovaný úsek a v prehľadávaní ďalej nepokračujeme. Podmienky pre výber používame nasledujúce: podobnosť zarovnania minimálne

70% a maximálne 95%. Maximálnu podobnosť volíme preto, aby sme ako template nepoužili štruktúru skoro úplne zhodnú s target štruktúrou. To by síce bolo žiaduce pri naozajstnej predikcii, kedy chcem dostať čo najlepší výsledok, ale nežiadúce pre testovanie funkčnosti predikcie, kedy by nám to zvyšok predikcie výrazne zjednodušilo. Pri predikcií neznámej štruktúry tiež typicky nepoznáme žiadnu veľmi podobnú štruktúru neznámej štruktúre, inak by sme ju použili ako primárny template.

V ďalšom kroku musíme namapovať nukleotidy zo sekundárnej template štruktúry do nekonzervovaného úseku primárnej template štruktúry (ktorá je už namapovaná na target sekvenciu) podľa zarovnania. To znamená pre každý použitý nukleoti zistiť, na aký nukleotid bol v zarovnaní namapovaný a zmeniť jeho index v pdb súbore. Okrem toho musíme zmeniť aj chain ID v prípade, že sa nezhodujú chain ids primárnej a sekundárnej template štruktúry.

V poslednom kroku sme skúšali dva rôzne prístupy.

1. Superpoziciovanie sekundárnych template štruktúr do dlhých nekonzervovaných úsekov primárnej template štruktúry.

2. Uložiť fragmenty vypĺňajúce dlhé nekonzervované úseky do separátnych súborov.

Prvý spôsob je superpoziciovanie nukleotidov, ktorými sme rozšírili z oboch strán nekonzervovaný úsek a našli preň vhodnú sekundárnu template štruktúru.Tieto dodatočne pridané nukleotidy superpoziciujeme na odpovedajúce nukleotidy v primárnej template štruktúre. Inak povedané to znamená vhodným spôsobom rotovať a posúvať sekundárnu template štruktúru ako celok tak, aby sme minimalizovali RMSD nukleotidov, ktoré sa nachádzajú v oboch štruktúrach a majú rovnaké indexy. Na toto sme použili *Superimposer* implementovaný v knižnici *BioPython* Bio. Po vhodnej rotácií a translácií terciárnej template štruktúry z nej odstránime pridané rozširujúce úseky (pridané v metóde *ExtendGap*) a nukleotidy čiastočne pokrývajúce nekonzervovaný úsek skopírujeme do primárnej template štruktúry. Takýmto spôsobom v ideálnom prípade vyplníme jeden dlhý nekonzervovaný úsek v primárnej template štruktúre, fragmentom štruktúry zo sekundárnej target štruktúry. Vo výsledku by sme pri použití tohoto prístupu mali template štruktúru, ktorá má dodatočne doplnené dlhé nekonzervované úseky vhodnými fragmentmi ostatných štruktúr.

Druhý prístup je jednoduchší a pozostáva len z uloženia správne namapovaných fragmentov sekundárnej template štruktúry do separátneho pdb súboru. Vo výsledku by sme teda mali pdb súbor s primárnou template štruktúrou a pre každý dlhý nekonzervovaný úsek v tejto štruktúre, by sme mali pdb súbor s fragmentom so sekundárnej template štruktúry vypĺňajúcej tento nekonzervovaný úsek. Fragmenty sukundárnych template štruktúr obsahujú informáciu iba o relatívnej polohe nukleotidov v jednom fragmente a poloha štruktúr v rôznych nekonzervovaných úsekov je nezávislá.

Posledná zmena, platná pri oboch dvoch prístupoch je, že vzhľadom nato, že sme našli sekundárne template štruktúry pre nekonzervované úseky nebudeme používať metódu *ProcessLongUnconservedParts* z pôvodného algoritmu, ktorá vyčleňovala predikciu dlhých nekonzervovaných úsekov do samostatnej predikcie algoritmom FARFAR.

Pre lepšie vysvetlenie uvádzame pseudokód algoritmu používajúceho viacero template štruktúr spolu s integráciou do pôvodného algoritmu. Pseudokód vychádza z pseudokódu algoritmu Trooper so sekundárnou štruktúrou predstaveného v piatej kapitole Sekcia 5.2.

```
Main(fastaTarget, fastaTemplate
        , secStrTemplate, pdbTemplate)
{
    CheckTemplateMapping
        (fastaTemplate,pdbTemplate)
    alignment := Align
        (fastaTarget, fastaTemplate)
    alignment1 := UseSlidingWindow
        (alignment)
    alignment2 := ProcessGaps
        (alignment1)
    conservedParts := CopyConservedParts
        (alignment2, pdbTemplate)
    mappedConservedParts := MapConservedParts
        (conservedParts, alignment2)
    conservedPartsSecStr := CopyConservedPartsSecStr
        (alignment2, secStrTemplate)
    mappedConservedPartsSecStr := MapConservedPartsSecStr
        (conservedPartsSecStr, alignment2)
    predictedSecStr := PredictUnconservedSecStr
        (mappedConservedPartsSecStr)
    mappedConservedParts := MultipleTemplatesModule
        (mappedConservedParts)
    longParts[] := ProcessLongUnconservedParts
        (mappedConservedParts, secStr)
    shortParts[] := ProcessShortUnconservedParts
        (mappedConservedParts, secStr)
    predictedParts[] := PredictUnconservedParts
        (longParts[], shortParts[])
    finalModel := ConnectPredictedParts
        (predictedParts)
}

MultipleTemplatesModule(mappedConservedParts)
{
    gaps[] := IdentifyLongUnconservedRegions
        (mappedConservedParts)
    potentialTemplateStructures[] = FindPotentialTemplates
        (database)
    foreach gap in gaps
    {
        extendedGap = ExtendGap(5 nk from each side)
        secTemplate := FindTemplateForGap
            (extendedGap, potentialTemplateStructures)
```

```
45          preparedSecTemplate := MapTemplateToGap
46              (secTemplate)
47          mappedConservedParts := IntegrateNewTemplateStr
48              (prepared_sec_template)
49      }
50      return mappedConservedParts
51  }
52
53      FindTemplateForGap(gap, potentialTemplateStructures)
54      {
55          alns = []
56          foreach sequence in potentialTemplateStructures:
57          {
58              alns = EmbossAlign(sequence, gap)
59          }
60          bestAln = alns.SelectBest()
61          return bestAln
62      }
63
64      MapTemplateToGap(secTemplate)
65      {
66          from secTemplate select only relevant nukleotides
67          renumber secTemplate to match unconserved region
68          rename chain to match the primary template
69      }
70
71      IntegrateNewTemplateStr(preparedSecTemplate)
72      {
73          superimpose prepared template to gap
74          OR
75          save secondary template to new file
76      }
```

6.3 Integrácia do existujúceho algoritmu

Integácia do algoritmu si vyžiadala niekoľko úprav v pôvodnom algoritme a implementácií.

Kostra algoritmu, používajúceho viacero template štruktúr:

1. Vybrať primárnu target štruktúru.
2. Zarovnať target a template sekvencie pomocou algoritmu globálneho zarovnania.
3. Identifikovať dlhé nekonzervované úseky v zarovnaní.
4. Pre každý nekonzervovaný úsek nájsť vhodnú sekundárnu target štruktúru pomocou semiglobálneho zarovnania.

5. Integrovať príslušné fragmenty zo sekundárnych template štruktúr do konzervovaných úsekoch z template štruktúry (pomocou superpozície), alebo fragmenty uložiť do separátnych súborov.

6. V prípade vyplnenia nekonzervovaného úseku sekundárnou template štruktúrou nemusíme vyčleňovať extra predikcie pre dlhé nekonzervované úseky.

7. Upravíme vstupy pre FARFAR

8. Pokračujeme predikciou algoritmom FARFAR.

Prvé tri kroky sa v ničom nelíšia od pôvodného algoritmu. Najprv musíme získať hlavnú template štruktúru, pričom nezáleží na tom, či ju už dostaneme na vstupe, alebo ju budeme algoritmicky hľadať nejakú vhodnú v našej stiahnutej databáze štrutúr. Následne zarovnáme target a template sekvencie, aplikujeme algoritmus posuvného okienka, ošetríme medzery v zarovnaní a dostaneme tak nekonzervované úseky, ktoré musíme predikovať. Z nich vyberieme tie dlhé a pokúsime sa pre ne nájsť sekundárne template štruktúry. Ktoré použijeme podľa popisu algoritmu uvedeného v predchádzajúcej sekcií.

V nasledujúcich krokoch nepoužijeme metódu *ProcessLongUnconservedParts*, ktorá vyberá a separátne pripravovuje predikciu dlhých nekonzervovaných úsekov, pretože tieto by mali byť doplnené zo sekundárnych template štruktúr a preto táto metóda stráca význam. Okrem toho musíme upraviť vstupné súbory a volania algoritmu FARFAR o novo pridané nukleotidy a súbory. Následne v prípade, že používame druhú variantu algoritmu s viacerými template súbormi, musíme upraviť automatizačné shell scripty, aby tieto súbory nakopírovali na správne miesto spolu s ostatnými vstupnými súborm algortimu FARFAR.

6.4 Časová zložitosť

V tejto sekcií popíšeme, ako sa líši časová zložitosť od pôvodného algoritmu Trooper. Jediná zásadná funkcionalita, ktorá pribudla, je implementovaná v metóde *MultipleTemplatesModule*. Z pohľadu asymptotickej zložitosti je zaujímavá hlavne metóda *FindTemplateForGap* slúžiaca na vyhľadanie sekundárnych template štruktúr. Metóda zarovnáva nekonzervovaný úsek na každú sekvenciu v databáze. Z toho vychádza časová zložitosť $O(ngl^2)$, pričom n je počet štruktúr v databáze, g je počet nekonzervovaných úsekov (ich počet však bude funkčne závislý od dĺžky target štruktúry - čím dlhšia, tým viac nekonzervovaných úsekov) a l je dĺžka štruktúry v databázi. Vo výsledku sa nám teda asymptotická časová zložitosť zhoršila z $O(l^2)$ na $O(ngl^2)$.

Reálna dĺžka behu algoritmu výrazne narástla. Merali sme ju na počítači s procesorom Intel Core i5-6200Li 2,8GHz a jeden beh metódy *FindTemplateForGap* trval pri nekonzervovanom úseku dĺžky 7-25 nukleotidov približne 30 sekúnd, pričom sme ale gap zarovnávali na celú databázu štruktúr a nie len tie dĺžky 50-500 nukleotidov. Celková dĺžka behu je potom závislá na počete dlhých nekonzervovaných úsekov, ktoré predikovaná štruktúra obsahuje (pre štruktúry dĺžky 50-500 nukleotidov a podobnosti s primárnym template-om 60-90% sú to najčastejšie 0 až 3 úseky).

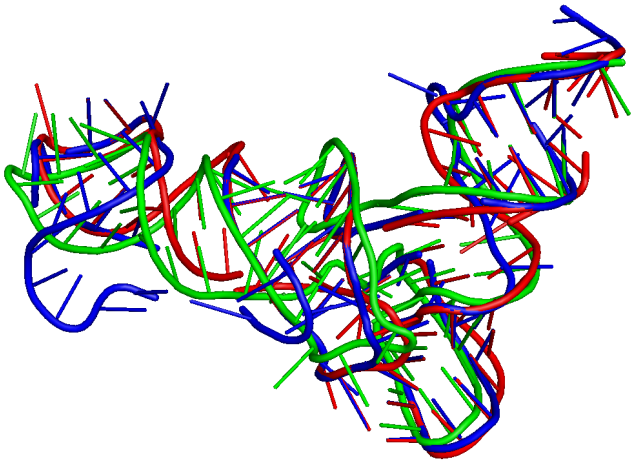
Zvýšená časová náročnosť by sa možno dala riešiť použitím algoritmov FASTA alebo BLAST, ktoré používajú heuristické metódy na vyhľadanie najpodobnejších sekvencií. Vzhľadom na to, že predikcia predikcia algoritmom FARFAR trvá oveľa dlhšie, rozhodli sme sa tento problém neriešiť.

6.5 Experiment a výsledky

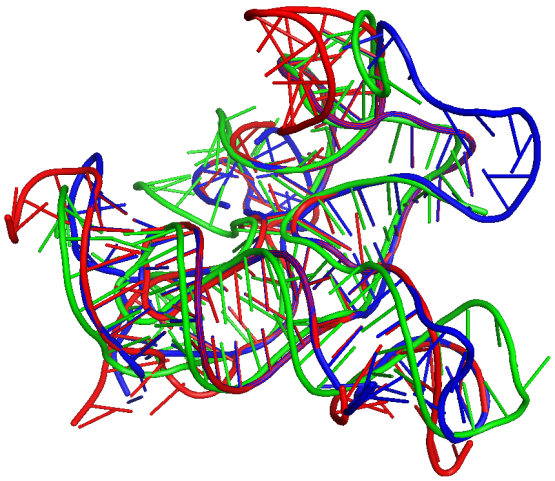
Pre experiment sme volili rovnaké podmienky a vstupné parametre, ako pre predchádzajúce dva experimenty. Predikovali sme tiež s použitím sekundárnej štruktúry, pretože predikcia so sekundárnou štruktúrou, ako aj predikcia s viacerými template štruktúrami, sú navzájom kompatibilné. Experimnety sme vykonávali na oboch variantách metódy *IntegrateNewTemplateStr*.

Prvý spôsob predikcie vypĺňaním nekonzervovaných úsekov v primárnej template štruktúre sa nám nepodarilo dokončiť, pretože sme naše zásahy do štruktúry nedokázali zladiť s algoritmom FARFAR. Vzhľadom na to, že pre nás funguje ako čierna skrinka, a neexistuje podrobná dokumentácia, ktorá by vysvetľovala príčiny chýb, ktoré môžu nastať sme s týmto prístupom narazili na problém, kedy FARFAR hneď na začiaku spadne na chybu *"FoldTree in pose does not have the right number of jumps to match chunk_res"*. Podarilo sa nám len zistiť, že chyba nastáva pri prevode dodaného template pdb súboru do internej reprezentácie FARFAR-u, čo môže byť spôsobené pravdepodobne len superpoziciovaním sekundárnej template štruktúry.

Preto sme implementovali druhý, jednoduchší spôsob, v ktorom vynechávamé superpoziciovanie sekundárnej template štruktúry a dodávame ju v separátnych pdb súboroch. Tento prístup je horší v tom, že superpoziciovanie dodaných sekundárnych template štruktúr musí vykonať FARFAR bez informácie o okolitých nukleotidov sekundárnej template štruktúry, a preto nie je prehľadávaný priestor zmenšený tak, ako by tomu bolo v prvej variante algoritmu. Pri tomto prístupe sa nám už podarilo časť štruktúr napredikovať, ale algoritmus stále trpel veľkou chybovosťou oproti verziám bez použitia viacerých template štruktúr. Napredikovali sme spolu *219 štruktúr* (*43%* z počtu napredikovaných štruktúr pôvodnoým algoritmom) a z toho v 39 z nich sa predikovalo pomocou viacerých template štruktúr (pre ostatné štruktúry sa buď sekundárna template štruktúra nepodarila nájsť, alebo sa v predikovanej štruktúre nenachádzali dostatočne dlhé nekonzervované úseky). V týchto *39* štruktúrah sme dosiahli priemernú RMSD na úrovni *10,72Å*, čo predstavuje zhoršenie oproti pôvodnému algoritmu, ktorý na rovnakých štruktúrach dosiahol priemerné RMSD hodnoty *6,16Å*. Napriek tomu, že celkový priemer predikcií sa zhoršil Obrázok 6.2, podarilo sa nám z 39 štruktúr napredikovať *8* s nižšou RMSD Obrázok 6.3. Pri bližšom preskúmaní výsledných štruktúr sme zistili, že výsledný priemer RMSD najviac zhoršila predikcia dvoch štruktúr a to *1FIR_A* a *3 L0U_A*, ktoré boli pomocou rôznych template štruktúr napredikované spolu *25 krát*. Na obrázku Obrázok 6.2 je viditeľný nespojitý úsek, na ktorý bola použitá sekundárna template štruktúra a predikcia ktorého sa kvôli tomu výrazne zhoršila. Predpokladáme teda, že keby sme dokázali určiť vhodnú sekundárnu template štruktúru, mohli by sme získať lepšie výsledky.



Obrázek 6.2: Príklad zhoršenia, v predikcií štruktúry 3L0U_A pri použití template štruktúry 1FCW_A spôsobeného použitím viacerých template štruktúr. Zelenou farbou je znázornená experimentálne získana štruktúra, modrou predikcia používajúca viacero template štruktúr (RMSD = 10,764Å) a červenou štruktúra napredikovaná algoritmom z kapitoly 5 (RMSD = 4,58Å).



Obrázek 6.3: Príklad zlepšenia, v predikcií štruktúry 4QLN_A pri použití template štruktúry 4QK9_A vďaka použitiu viacerých template štruktúr. Zelenou farbou je znázornená experimentálne získana štruktúra, modrou predikcia používajúca viacero template štruktúr (RMSD = 6,081Å) a červenou štruktúra napredikovaná algoritmom z kapitoly 5 (RMSD = 8,157Å).

Závěr

Primárnym cieľom tejto práce bolo rozšíriť pôvodný algoritmus predikcie terciárnej štruktúry za pomoci vzorovej štruktúry o dva moduly, ktoré by zlepšili problematickú predikciu štruktúr obsahujúcich dlhé nekonzervované úseky. Okrem toho sme chceli predikciu zautomatizovať tak, aby nevyžadovala manuálne zásahy, a následne naše výsledky porovnať s algoritmom ModeRNA.

Prvý modul, ktorý vkladá do predikcie terciárnej štruktúry medzikrok predikcie sekundárnej štruktúry, sme úspešne navrhli, implementovali a otestovali. Z testovania vyplynulo, že časť predikovaných štruktúr sa zlepšila a časť naopak zhoršila, za čo, ako sme analýzou ukázali, môžu hlavne nepresne napredikované sekundárne štruktúry target molekuly. Návrh druhého modulu, pomocou ktorého vieme pri predikcii jednej target štruktúry použiť viacero template štruktúr, sa ukázal byť zložitejší. Prvé dva návrhy algoritmu sa pri testovaní ukázali ako nepoužiteľné, ale tretiu verziu algoritmu sa nám podarila naimplementovať a odladiť do takej miery, že sme boli schopní získať výsledky indikujúce, že pri nájdení správnej sekundárnej template štruktúry sme schopní výsledok predikcie zlepšiť, ale naopak pri použití nevhodnej štruktúry sa môžu výsledky výrazne zhoršiť. Implementáciu algoritmu sa nám podarilo úspešne automatizovať do takej miery, že na hromadnú predikciu štruktúr a vyhodnotenie výsledkov stačí spustiť štyri shell skripty. Vďaka tomu sme dokázali výsledky algoritmu Trooper porovnať s výsledkami algoritmu ModeRNA, pričom sa ukázalo, že náš algoritmus bol vo verzii so sekundárnou štruktúrou v celkovom priemere RMSD o niečo presnejší.

Výsledky našej práce ukázali, že obe metódy majú potenciál, aby pri použití správnej sekundárnej štruktúry, alebo ďalšej vhodnej template štruktúy, dokázali výsledky predikcie zlepšiť. Okrem toho sme ukázali, že naše aktuálne výsledky sú porovnateľné s konkurenčným algoritmom ModeRNA. V budúcnosti by sme sa práve chceli zamerať na zlepšenie predikcie sekunárnych štruktúr a ďalšie vyladenie hľadania vhodných template štruktúr. Takisto by sme chceli zbaviť nášu implementáciu závislosti na algoritme FARFAR a predikciu nekonzervovaných úsekov zabezpečovať vlastným riešením.

Seznam použité literatury

Biopython. https://biopython.org/. Accessed: 2019-07-10.

Alberts B, Johnson A, L. J. a. k. (2002). *Molecular Biology of the Cell*. 4th edition. Garland Science, New York. ISBN 0-8153-3218-1.

Berman, H. M., W. J. F. Z. G. G. B. N. W. H. a. S. I. N. (2000). The protein data bank. *Nucleic Acids Research*, (28), 235–242.

Biesiada, M., Purzycka, K. J., Szachniuk, M., Blazewicz, J. a Adamiak, R. W. (2016). *Automated RNA 3D Structure Prediction with RNAComposer*, pages 199–215. Springer New York, New York, NY. ISBN 978-1-4939-6433-8. doi: 10.1007/978-1-4939-6433-8_13. URL https://doi.org/10.1007/978-1-4939-6433-8_13.

Boniecki, M. J., Lach, G., Dawson, W. K., Tomala, K., Lukasz, P., Soltysinski, T., Rother, K. M. a Bujnicki, J. M. (2015). SimRNA: a coarse-grained method for RNA folding simulations and 3D structure prediction. *Nucleic Acids Research*, **44**(7), e63–e63. ISSN 0305-1048. doi: 10.1093/nar/gkv1479. URL https://doi.org/10.1093/nar/gkv1479.

Brudno, M; Malde, S. P. A. (2003). Glocal alignment: finding rearrangements during alignment. *Bioinformatics*, (19), 54–62.

Crick, F. (1970). Central dogma of molecular biology. *Nature*, **227**(2), 561–563.

Das R., Karanicolas J., B. D. (2010). Atomic accuracy in predicting and designing noncanonical rna structure. *Nature methods*, (7), 291–294.

Eddy, S. (2004). Genome regulation by long noncoding rnas. *Nature Biotechnology*, (22).

Felden, B. (2007). Current opinion in microbiology. *Current opinion in microbiology*, (10), 286–291.

Flores, S., Sherman, M., M Bruns, C., Eastman, P. a Altman, R. (2011). Fast flexible modeling of rna structure using internal coordinates. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*, **8**, 1247–57. doi: 10.1109/TCBB.2010.104.

Frellsen, J., Moltke, I., Thiim, M., Mardia, K. V., Ferkinghoff-Borg, J. a Hamelryck, T. (2009). A probabilistic model of rna conformational space. *PLOS Computational Biology*, **5**(6), 1–11. doi: 10.1371/journal.pcbi.1000406. URL https://doi.org/10.1371/journal.pcbi.1000406.

Galvanek, R. a Hoksza, D. (2017). Template-based prediction of rna tertiary structure using its predicted secondary structure. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2238–2240. doi: 10.1109/BIBM.2017.8218009.

Galvanek, R., Hoksza, D. a Pánek, J. (2016). Template-based prediction of rna tertiary structure. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1897–1900. doi: 10.1109/BIBM.2016.7822808.

Galvánek, R. (2016). Predikce terciární struktury rna na základě předlohy.

Gibbs, Adrian J.; McIntyre, G. A. (1970). The diagram, a method for comparing sequences. its use with amino acid and nucleotide sequences. *Biochem*, (16), 1–11.

Gruber AR, Lorenz R, B. S. N. R. H. I. (2008). The vienna rna websuite. *Nucleic Acids Research*, (36).

Jenny Gu, P. E. B. (2009). *Structural Bioinformaticsl*. 2th edition. Wiley-Blackwell, New Jerseyl. ISBN 978-0-470-18105-8.

Krupovic, M., Blomberg, J., Coffin, J. M., Dasgupta, I., Fan, H., Geering, A. D., Gifford, R., Harrach, B., Hull, R., Johnson, W., Kreuze, J. F., Lindemann, D., Llorens, C., Lockhart, B., Mayer, J., Muller, E., Olszewski, N. E., Pappu, H. R., Pooggin, M. M., Richert-Pöggeler, K. R., Sabanadzovic, S., Sanfaçon, H., Schoelz, J. E., Seal, S., Stavolone, L., Stoye, J. P., Teycheney, P.-Y., Tristem, M., Koonin, E. V. a Kuhn, J. H. (2018). Ortervirales: New virus order unifying five families of reverse-transcribing viruses. *Journal of Virology*, **92**(12). ISSN 0022-538X. doi: 10.1128/JVI.00515-18. URL `https://jvi.asm.org/content/92/12/e00515-18`.

Lu, X. a Olson, W. K. (2003). 3DNA: a software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures. *Nucleic Acids Research*, **31**(17), 5108–5121. ISSN 0305-1048. doi: 10.1093/nar/gkg680. URL `https://doi.org/10.1093/nar/gkg680`.

Moore, P. B. (1999). *The RNA World*. 2th edition. Cold Spring Harbor Laboratory, New Haven. ISBN ISBN 0-87969-561-7.

Needleman S. B., W. C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, (48), 443–453.

Noller, H. (1984). Structure of ribosomal rna. *Annual Review of Biochemistry*, (53), 119–162.

Qian, B., Raman, S., Das, R., Bradley, P., McCoy, A. J., Read, R. J. a Baker, D. (2007). High-resolution structure prediction and the crystallographic phase problem. *Nature*, **450**(7167), 259. doi: 10.1038/nature06249. URL `https://app.dimensions.ai/details/publication/pub.1001896118andhttp://europepmc.org/articles/pmc2504711?pdf=render`. Exported from https://app.dimensions.ai on 2019/05/26.

Ramlan, E. a Zauner, K.-P. (2008). An extended dot-bracket-notation for functional nucleic acids.

Rice P., L. I. a A., B. (2000). Emboss: The european molecular biology open software suite. trends in genetics. *Trends in Genetics.*, (16), 276–277.

Rinn, J. a Chang, H. (2012). How do rna folding algorithms work? *Annual Review of Biochemistry*, (81), 145–166.

Rother, M., Rother, K., Puton, T. a Bujnicki, J. M. (2011). RNA tertiary structure prediction with ModeRNA. *Briefings in Bioinformatics*, **12** (6), 601–613. ISSN 1477-4054. doi: 10.1093/bib/bbr050. URL `https://doi.org/10.1093/bib/bbr050`.

Rother K, Rother M, B. M. P. T. B. J. (2011). Rna and protein 3d structure modeling: similarities and differences. *Journal of Molecular Modeling*, (10), 2325–2336.

Ryu, W.-S. (2017). Chapter 2 - virus structure. In Ryu, W.-S., editor, *Molecular Virology of Human Pathogenic Viruses*, pages 21 – 29. Academic Press, Boston. ISBN 978-0-12-800838-6. doi: https://doi.org/10.1016/B978-0-12-800838-6.00002-3. URL `http://www.sciencedirect.com/science/article/pii/B9780128008386000023`.

Schrödinger, LLC (2015). The PyMOL molecular graphics system, version 1.8.

Schudoma C., May P., N. V. W. D. (2010). Sequence-structure relationships in rna loops: establishing the basis for loop homology modeling. *Nucleic Acids Research*, (38), 970–980.

Sharma, S., Ding, F. a Dokholyan, N. V. (2008). iFoldRNA: three-dimensional RNA structure prediction and folding. *Bioinformatics*, **24**(17), 1951–1952. ISSN 1367-4803. doi: 10.1093/bioinformatics/btn328. URL `https://doi.org/10.1093/bioinformatics/btn328`.

Smith T. F., W. M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, (147), 195–197.

Stephen F. Altschul, Mark S. Boguski, W. G. J. C. W. (1994). Issues in searching molecular sequence databases. *Nature Geneticsvolume*, (6), 119–129.

Yunjie Zhao, Yangyu Huang, Z. G. Y. W. J. M. . Y. X. (2012). Automated and fast building of three-dimensional rna structures. *Scientific Reports*.

Seznam obrázků

Seznam tabulek

A. Přílohy

Príloha obsahuje skripty a dáta, ktoré boli použité pri písaní tejto práce. Skkripty a dáta sú tiež dostupné na `https://github.com/galvaner/Trooper`.