ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra počítačů

Nástroj pro objevování sekvenčních motivů v RNA-Seq datech

Tool for sequence motif discovery in RNA-Seq data

Diplomová práce

Studijní program: Otevřená informatika

Specializace: Bioinformatika

Vedoucí práce: RNDr. Martin Pospíšek, Ph.D.

Petr Schimperk

Praha, Květen 2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Schimperk** Jméno: **Petr** Osobní číslo: **487010**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Bioinformatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Nástroj pro objevování sekvenčních motivů v RNA-Seq datech

Název diplomové práce anglicky:

Tool for sequence motif discovery in RNA-Seq data

Pokyny pro vypracování:

Cíle a kontrolní body:
• Seznámit se s problematikou vyhledávání sekvenčních motivů RNA ve výpočetní biologii.
• Analyzovat nástroj vyvinutý Janem Holčákem v rámci předchozí diplomové práce nazvané: „Hledání sekvenčních motivů v mRNA selektovaných vazbou na translační iniciační faktory z rodiny eIF4E."
• Navrhnout úpravy a rozšíření pracovního toku připraveného v odkazované diplomové práci. Půjde zejména o následující rozšíření:
a) Bude umožněn obecnější vstup. Dosavadní řešení bylo navrženo pro analýzu dat z jednoho konkrétního experimentu. Nově bude možné pracovat na vstupu jednak s uživatelem definovaným seznamem genů, jednak přímo se sekvenačními daty z RNA-Seq.
b) Bude navržen algoritmus pro porovnání, sdružení a analýzu RNA motivů získaných z existujících detektorů motivů.
c) Bude navržen algoritmus pro hledání sekvenčních motivů v datech z RNA-Seq. Algoritmus zohlední sekvenční odlišnosti mezi referenčními sekvencemi genomů, respektive transkriptomů uvedenými ve veřejných databázích a sekvencemi RNA v aktuálním vzorku.
d) Algoritmus navržený v bodu c) bude upraven tak, aby kromě kvalitativních rozdílů v sekvencích analyzovaných mRNA zohlednil též rozdílnou četnost různých mRNA ve zkoumaném vzorku.
• Rozšířený nástroj ověřit vyhledáním motivů v RNA-Seq datech dodaných vedoucím práce. Klíčová je efektivita a přehlednost reportované množiny motivů.

Seznam doporučené literatury:

The MEME Suit - https://meme-suite.org/meme/index.html
The HOMER - http://homer.ucsd.edu/homer/motif/rnaMotifs.html
Avinash Achar, Pål Sætrom, RNA motif discovery: a computational overview, Biol Direct 2015 Oct 9:10:61. doi: 10.1186/s13062-015-0090-5.
Další odborná literatura je součástí přehledu, který připraví student v rámci práce

Jméno a pracoviště vedoucí(ho) diplomové práce:

RNDr. Martin Pospíšek, Ph.D. katedra počítačů FEL

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

Datum zadání diplomové práce: **12.02.2024** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **21.09.2025**

RNDr. Martin Pospíšek, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

Datum převzetí zadání	Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24.5.2024 Petr Schimperk .....

Poděkování

Rád bych poděkoval svému vedoucímu RNDr. Martinu Pospíškovi, Ph.D. za cenné rady, připomínky a odborné vedení práce. Dále bych rád poděkoval své rodině za pomoc a podporu.

Tato práce vznikla za podpory projektu Národní institut virologie a bakteriologie (Program EXCELES, ID: LX22NPO5103), financovaného Evropskou unií – Next Generation EU a projektu OP JAK CZ.02.01.01/00/22_008/0004575 RNA pro terapii, spolufinancováno Evropskou unií.

Abstrakt

Hledání a analýza sekvenčních motivů jsou důležitým krokem při zkoumání funkcí genů a regulace genové exprese. Pro řešení této úlohy v současnosti existuje více než 150 heuristických i neheuristických algoritmů využívajících různorodé programovací přístupy. Výběr z neustále zvětšující se množiny programů je časově náročný, stejně jako jejich následná instalace a korektní spuštění. Tato diplomová práce se zabývá tvorbou aplikace, která má za cíl urychlit výběr a zautomatizovat instalaci oněch nástrojů a také poskytnout základní kvalitativní analýzu nalezených motivů. Pro běh aplikace je využita Java a pro většinu použitých nástrojů je využita virtualizace pomocí kontejnerů, což dohromady umožňuje širokou podporu napříč běžnými operačními systémy. Samotné jádro aplikace je naprogramováno jako jednoduchý daty řízený graf úloh (*data-driven task graph*), do kterého uživatel přidává kroky reprezentující určitou funkci.

Klíčová slova

RNA, sekvenční motiv, hledání nových motivů, virtualizace, aplikace řízená daty.

Abstract

Search and analysis of sequence motifs are important steps in studies of gene functions and regulation of gene expression. Currently, to address this task, there are more than 150 both heuristic and non-heuristic algorithms using a variety of programming approaches. Selecting from this ever-expanding set of applications is time-consuming, as is subsequent installation and proper execution. This thesis focuses on developing an application to quicken selection and automate the installation of such tools, and to provide basic qualitative analysis of found motifs. The application runs on Java and for most of the tools used container virtualization is employed, combined allows for broad support across common operating systems. The core of application is programmed using a simple data-driven task graph, to which the user adds steps representing a certain function.

Key words

RNA, sequence motif, de-novo motif discovery, virtualization, data-driven application.

Obsah

Seznam zkratek **10**
1 Úvod **11**
2 Vyhledávání sekvenčních motivů **12**
2.1 Genetická informace **12**
2.2 Motivy **13**
2.2.1 Reprezentace motivů 13
2.3 De novo hledání motivů **14**
2.3.1 Pravděpodobnostní algoritmy 15
2.3.2 Local search algoritmy 16
2.3.3 Algoritmy využívající strojové učení 16
2.3.4 Neheuristické algoritmy 17
2.4 Porovnávání motivů **18**
2.5 Hledání motivů v datech ze sekvenování **19**
2.6 Formáty sekvencí a motivů **20**
3 Analýza výchozí aplikace **22**
3.1 Analýza vybraných částí dle adresářů kódu **22**
3.1.1 „bioplachta_containers“ 22
3.1.2 „generate_fasta“ 23
3.1.3 „find_motifs“ 24
3.1.4 „lead2gold“ 25
3.1.5 Ostatní části aplikace 26
3.2 Vyhodnocení analýzy **26**
4 Popis vytvořené aplikace **27**
4.1 Základní popis aplikace **27**
4.1.1 Konfigurační formát a definice experimentu 28
4.1.2 Argumenty aplikace 28
4.1.3 Grafické rozhraní aplikace 29
4.1.4 Rozhraní příkazové řádky 30
4.1.5 První spuštění aplikace 30
4.2 Funkcionality aplikace a jejich implementace **30**
4.2.1 Definiční a vstupní kroky 31
4.2.2 Výstupní kroky 32

4.2.3 Krok na stažení sekvencí pomocí Biomart API ... 32
4.2.4 Krok pro de novo hledání motivů ... 33
4.2.5 Krok pro hledání motivů v sekvenci ... 34
4.2.6 Kroky na porovnání motivů ... 35
4.2.7 Implementace mechanismu kroků ... 35
4.2.8 Rozhraní virtualizace a zásuvné moduly ... 37
4.2.9 Ostatní implementační detaily ... 37
5 Diskuse ... 39
5.1 Porovnání s podobnými aplikacemi ... 39
5.1.1 Tmod ... 39
5.1.2 Galaxy ... 40
5.1.3 MEME-suite, DynaMIT a GimmeMotifs ... 40
5.2 Možnosti rozšíření funkcionalit ... 41
6 Testování ... 43
6.1 Testování založené na ověřitelných datech ... 43
Závěr ... 46
Bibliografie ... 47
Seznam příloh ... 50
Příloha C – ukázky GUI aplikace ... 51

Seznam zkratek

API	programovací rozhraní aplikace	application programming interface
CLI	rozhraní příkazové řádky	command line interface
DNA	deoxyribonukleová kyselina	deoxyribonucleic acid
EM	EM algoritmus	expectation-maximization algorithm
EOF	konec souboru	end of file
GUI	grafické uživatelské rozhraní	graphical user interface
HMM	skrytý Markovův model	hidden Markov model
JRE	prostředí pro běh jazyka Java	Java runtime environment
JSON		JavaScript Object Notation
mRNA	mediátorová RNA	messenger RNA
NGS	sekvenování nové generace	next generation sequencing
PFM	poziční matice frekvencí	position frequency matrix
PMS	vyhledávání vložených motivů	planted motif search
PPM	poziční matice pravděpodobností	position probability matrix
PWM	poziční matice vah	position weight matrix
RNA	ribonukleová kyselina	ribonucleic acid
TFBS	vazebné místo transkripčního faktoru	transcription factor binding site
tRNA	transferová RNA	transfer RNA
UTR	nepřekládaná oblast	untranslated region
WSL	podpora běhu Linux na Windows	Windows subsystem for Linux
XML	rozšiřitelný značkovací jazyk	extensible markup language

1 Úvod

Obecnou definicí motivu v datových vědách je kratší model nebo vzor dat, který se vyskytuje na více místech ve vstupních datech. Může jít například o hledání vzoru v časových řadách, nebo hledání společného podgrafu napříč několika grafy. Hledání různých motivů je také důležitým krokem při výzkumu v několika oblastech molekulární biologie, jmenovitě např. při analýze vazebný míst pro transkripční faktory (TFBS) nebo při modelování sekundárních struktur RNA a proteinů. Ačkoliv lze motivy hledat pomocí biologických experimentů, je výhodnější využít kombinace predikce a následné validace pomocí vhodně navrženého experimentu. Kvůli tomu a také kvůli narůstajícímu množství sekvencí ke zpracování se hledání motivů stalo jedním z větších podoborů bioinformatiky.

Tato práce se zabývá problematikou hledání nových sekvenčních motivů, pro kterou již byla vytvořena řada algoritmů postavená na různých programovacích přístupech. Hlavním problémem těchto algoritmů jsou rozdíly v distribucích programů a uživatelských rozhraních, kdy poměrná část nabízí přívětivější online webovou stránku, ale zbytek používá různé formáty jak pro vstup a argumenty, tak i pro výstup. Při použití vícero programů, což je vzhledem k jejich rozdílným přístupům doporučováno, se přidá i problém porovnání výsledků, protože většina algoritmů používá rozdílné metriky kvality. Kromě porovnání je možné nalezené motivy sloučit a tím zjistit, zdali neexistuje nějaký společný obecnější motiv.

Výstupem diplomové práce bude aplikace, jejímž účelem je spojit už existující nástroje k vytvoření ucelené funkcionality – hledání nových motivů v RNA sekvencích pomocí více programů najednou. Jako základ pro tuto část budou analyzovány a použity již existují nástroje vytvořené Janem Holčákem. Důraz bude kladen také na přívětivost uživatelského rozhraní, ale při zachování možností podrobné konfigurace a snadné rozšiřitelnosti. Posledním cílem je možnost využití informace ze sekvenačních dat – počtu sekvenačních čtení zasahujících do oblasti jednoho konkrétního nálezu motivu.

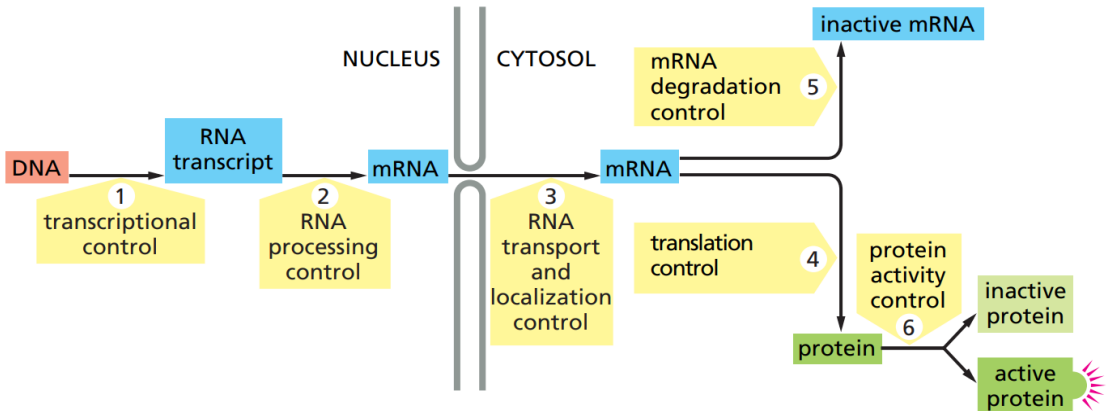
2 Vyhledávání sekvenčních motivů

2.1 Genetická informace

Všechny buněčné organismy uchovávají svojí genetickou informaci v DNA, kterou přepisují do RNA. Obě tyto nukleové kyseliny jsou polymerní řetězce složené z nukleotidů, každý nukleotid je složen z pětiuhlíkatého cukru ribózy, fosfátu a aromatické báze obsahující dusík. Základními bázemi jsou: adenin, guanin, cytosin a buď u DNA thymin nebo u RNA uracil. Těmto bázím jsou přiřazena kódová písmena podle prvních písmen jejich názvů, tj. A = adenin, dále G, C, T, U. Páteřní řetězec polymeru tvoří střídající se cukr a fosfát, na cukru je pak navázána báze. Tento řetězec má dva rozdílné konce, které se označují 5′ a 3′ podle čísel příslušných uhlíků koncové ribózy, které jsou v rámci páteřního řetězce svázány fosfodiesterovou vazbou.

DNA má strukturu dvoušroubovice, kde se spolu párují dva komplementární řetězce nukleotidů pomocí slabých nevazebných interakcí – vodíkových můstků. Tím vznikají páry bází (angl. *base pair*), jejich počet pak udává délku kyseliny. Komplementární bází k adeninu je thymin a k cytosinu guanin.

Úsek DNA, který nese konkrétní dědičnou genetickou informaci s nějakým významem, se nazývá gen. Části genů, které kódují proteiny nebo funkční RNA, jsou přepisovány do RNA. RNA má obvykle jedno řetězec, může ale tvořit sekundární sktruktury, ve kterých se krátké úseky řetězce párují s jinými úseky stejného řetězce podle komplementarity bází, případně i s jinými RNA. Mezi typy RNA patří například mediátorová mRNA nebo transferová tRNA, jiné typy mohou sloužit jako katalyzátory biochemických reakcí nebo mít jiné regulační funkce. mRNA, neboli také kódující RNA, je následně přeložena do proteinu v procesu zvaném translace, tento tok genetické informace se dohromady nazývá centrálním dogmatem molekulární biologie. [1, 10]



Obrázek 1 – centrální dogma molek. biologie včetně kontrolních mechanismů [10]

2.2 Motivy

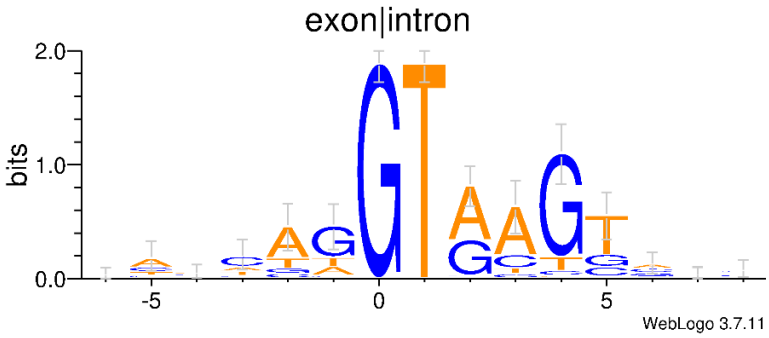
Motivy v RNA, DNA nebo proteinech se dělí na dva typy: sekvenční a strukturní. Sekvenční motiv je jednotky až nižší desítky monomerů dlouhý úsek řetězce s nějakým konkrétním významem. Motivem může být např. vazebné místo transkripčního faktoru (TFBS). Většina sekvenčních motivů jsou taková místa, která jsou detekována nějakou jinou makromolekulou, mají tedy biologický význam a jsou evolučně více konzervována než okolní sekvence. Strukturní motivy mohou být delší než sekvenční motivy a reprezentují určité vzory v sekundární struktuře RNA nebo proteinů – např. *helix-turn-helix* u proteinů vázajících se do velkého žlábku DNA. [2, 3, 10]

V bioinformatice lze využít i tzv. *network* motiv – podgraf společný pro více grafů (*networks*) nebo v rámci jednoho grafu. Uplatňuje se např. při hledání souvislostí v metabolických a regulačních drahách nebo protein-protein interakcích, které se dají vyjádřit jako orientovaný graf. [12]

2.2.1 Reprezentace motivů

Motiv lze reprezentovat buď textově, číselně nebo vizuálně. Nejjednodušeji lze motiv zapsat pomocí tzv. konsenzu – řetězce znaků, kde pro každou pozici je znak reprezentující jeden nebo více nukleotidů (např. AATGN, kde N[1] reprezentuje libovolný nukleotid). Konsenzus lze také zapsat pomocí gramatiky (ekvivalent k přechozímu příkladu AATG[ATGC]). Číselnou reprezentací jsou poziční 2D matice (profily motivů), ke pro každou pozici a pro každý možný symbol je uvedená hodnota:

- PFM – frekvence – absolutní počet výskytu symbolu na pozici
- PPM – pravděpodobnosti – lze ji vytvořit z PFM, suma sloupce je rovna 1
- P(S)WM/PSSM – váhy/skóre – lze spočítat z PPM např. pomocí *logit* funkce



Obrázek 2 – formát WebLogo ukazující motiv přechodu mezi exonem a intronem [5]

[1] Doporučené značení nukleotidů – *Table 1. Summary of single-letter code recommendations* v https://www.ncbi.nlm.nih.gov/pmc/articles/PMC341218/pdf/nar00303-0014.pdf

Běžně používaným vizuálním formátem je WebLogo (obrázek 2). WebLogo je grafem, kde na vodorovné ose jsou jednotlivé pozice a na svislé ose míra konzervace, pro každý symbol vyjádřené jako rozdíl maximální entropie na konkrétní pozici a entropie symbolu na stejné pozici. Pro každou pozici jsou vyobrazena písmena s nenulovou entropií, která jsou vertikálně škálována invertovanou mírou entropie (tj. čím menší entropie, tím vyšší písmeno, pozice s malou maximální entropií jsou vyšší než pozice s velkou maximální entropií). U DNA a RNA se škála entropie pohybuje od 0 do 2, protože pro vyjádření čtyř symbolů stačí 2 bity. [4, 5]

2.3 De novo hledání motivů

Planted motif search (PMS/LDMS) neboli de novo hledání motivů je NP-úplná bioinformatická úloha, jejímž vstupem jsou sekvence nukleotidů nebo aminokyselin a výstupem jsou motivy. Z matematického hlediska je úloha definována následovně: vstupem je *t* sekvencí, každá o délce *n*, výstupem je množina všech motivů, každý o délce *l*, pro které platí: pro každý výstupní motiv *x* se v každé vstupní sekvenci nachází podřetězec o délce *l*, který má Hammingovou vzdáleností vůči *x* maximálně *d*. V praxi však přesné hodnoty *l* ani *d* dopředu neznáme. Úloha může mít různé varianty: s omezením délky motivu *l* pomocí intervalu, omezením počtu výskytu v každé ze vstupních sekvencí (OOPS[1] – jeden výskyt na sekvenci, ZOOPS – jeden nebo nula výskytů, ZOMOPS – neomezeně výskytů), předem určený nebo omezený typ motivu aj.

Úloha je problematická zejména kvůli sumární délce vstupních sekvencí, motiv se může nalézat kdekoliv, a nejednoznačnosti hledaných motivů – na každé pozici se mohou vyskytovat až 4 nukleotidy nebo až 23 proteinogenních aminokyselin. Kvůli náročnosti úlohy se primárně využívají heuristické (aproximační) algoritmy, ale existují i algoritmy prohledávající celý stavový prostor. Heuristické algoritmy mají z definice tu nevýhodu, že nemusí naleznout globální optimum, ale oproti exaktním algoritmům mohou být i řádově rychlejší. [4, 6, 14]

[1] *(Zero or) One (or More) Occurence(s) Per Sequence*

Náročnější variantou je hledání nových motivů s mezerami (*gapped motifs*), ve kterých na libovolných místech mohou být mezery o libovolných délkách, v sumě však nesmí přesáhnout předem definovanou konstantu. Mezerou se rozumí, že na dané pozice může být libovolný symbol z abecedy. Omezenější definicí je pouze jedna mezera o omezené délce, kterou lze řešit pomocí hledání bez mezer a následným sloučením motivů, jejichž výskyt je bližší než definovaná konstanta. [19]

Algoritmů řešících tuto úloha existuje velké množství. Některé z nich byly vytvořeny pro nějaký specifický experiment a prokázaly pro daný experiment svou specifickou účinnost. Většinou se ale jedná o obecné algoritmy testované komparativně proti jiným přístupům, kdy se ukazuje, že pro určitá data jsou některé přístupy lepší než jiné a obráceně. V následujících podkapitolách jsou popsány některé algoritmické přístupy, kterých využívají i algoritmy se zaměřením na hledání strukturních motivů.

2.3.1 Pravděpodobnostní algoritmy

Implementace algoritmů v této skupině hojně využívají EM algoritmu nebo jsou založeny na stejném principu jako EM. Jeho nevýhodou je, že nachází pouze lokální optimum dle prvotních hodnot, je tedy nutné algoritmus vícekrát spustit s jinými hodnotami. Principu podobném EM využívají programy založené na algoritmu Gibbsova vzorkování (např. BioProspector nebo AlignACE) – tento algoritmus je oproti EM nedeterministický, protože používá *Markov chain Monte Carlo* metodu při výběru dalšího kroku. Jeho nevýhodou je ale výpočetní náročnost, která je větší než u EM. Dalšími vylepšeními EM metody jsou MEME, STREME[1] a MEMERIS algoritmy, které EM využívají jako základ pro pozdější slučování a eliminaci nevhodných kandidátů na motivy, interně reprezentovaných pomocí PWM. K tomu využívají skóre přepočtené na p-hodnotu, která udává pravděpodobnost, že nahodilá sekvence o stejné délce jako motiv získá lepší skóre. Dle [4] je nejlepším vylepšením EM metody algoritmus MCEMDA[2], který kandidáty motivů postupně vylepšuje pomocí Monte Carlo přístupu a EM algoritmu, čímž řeší problém lokálního optima.

[1] *Multiple Em for Motif Elicitation* a *Sensitive, Thorough, Rapid, Enriched Motif Elicitation* - https://meme-suite.org/meme/
[2] *Monte Carlo EM Motif Discovery Algorithm* - https://doi.org/10.1109/tcbb.2008.103

Přístup sloučení vícero algoritmů využívá také ssHMM algoritmus, který interně místo PWM využívá skrytý Markovův model (HMM) spolu s Gibbsovým vzorkováním. HMM pro reprezentaci profilu motivu spolu s kovariančním modelem využívá COVE algoritmus. CMfinder algoritmus pak spojuje dohromady přístupy MEME a COVE algoritmů, ale je převážně zaměřen na hledání strukturních motivů. [4, 6, 13, 20]

2.3.2 Local search algoritmy

Skupina algoritmů, jejichž principem je optimalizace prvotního řešení, tj. po nalezení libovolného řešení se snaží najít lepší „sousední“ řešení. Z definice pak vyplývá, že všechny tyto algoritmy nemusejí najít globální optimum. První podskupinou jsou algoritmy využívající obecné techniky pro local search algoritmy. Využívají buď záměny podčásti motivu za jinou (*pattern-based*) nebo hledají pomocí větvení v prostoru profilů motivů (*profile-based branching*). Algoritmus TEIRESIAS nejdříve hledá kratší části motivů, které mají parametricky omezený počet nejednoznačných symbolů a vyskytují se alespoň v parametricky definovaném počtu sekvencí, tyto části pak rozšiřuje na celé motivy.

Druhou podskupinou jsou algoritmy založené na grafech (kde hrany reprezentují sousednost řešení např. definovanou pomocí Hammingovy vzdálenosti). Algoritmus WINNOWER převádí úlohu hledání motivu na hledání kliky v neohodnoceném grafu, na vylepšením je pak algoritmus SP-STAR, který eliminuje hrany s nízkým ohodnocením a za pomoci dalších heuristik dosahuje lepší doby běhu. MotifCut algoritmus řeší úlohu převodem na hledání podgrafu s maximální hustotou. Obecnou nevýhodou grafových algoritmů může být vyšší paměťová náročnost. [4, 6, 13, 20]

2.3.3 Algoritmy využívající strojové učení

Oproti *local search* algoritmům algoritmy strojového učení dokážou rychleji najít globální optimum. Této skupině algoritmů také nahrává dnešní překotný vývoj softwaru i hardwaru v oblasti hlubokého strojového učení, který může dále snížit časovou náročnost, případně zlepšit kvalitu při zachování stejné časové náročnosti.

Do první podskupiny patří algoritmy využívající evolučních algoritmů, které využívají principů genetických mechanismů křížení, mutace a selekce a jsou proto méně náchylné na počáteční nastavení hodnot. Algoritmus FMGA začíná na řetězcích kratší délky, ze kterých vybere kandidáty pro další generaci, *x* nejlepších a část pomocí váženého náhodného výběru, a ty pak rozvine v novou generaci. Vylepšením FMGA přístupu jsou např. algoritmy GAMOT nebo GAME, který dosahuje v některých typech reálných sekvencí lepších výsledků[1] než MEME a další algoritmy založené na EM. GEMFA algoritmus navíc využívá právě EM, kdy použitím genetických mechanismů řeší problém lokálního minima EM algoritmu. Odlišnější jsou algoritmy MOGAMOD a MHABBO, které kombinují evoluční algoritmus s *multi-objective* optimalizací, která slouží k upravení parametrů konkrétních genetických mechanismů.

Druhou podskupinou jsou algoritmy založené na rekurentních neuronových sítích (RNN) – příkladem algoritmus KEGRU využívající word2vec pro reprezentaci k-merů nebo algoritmus DESSO, které predikuje TFBS a cis-regulační motivy s využitím detekce tvaru DNA. HLARPBP algoritmus využívá *long short-term memory* (LSTM) spolu s *attention* mechanismem, což je přístup běžně používaný pro zpracování jazyka (*natural language processing*, NLP). [4, 6, 13, 20]

2.3.4 Neheuristické algoritmy

Existují i exaktní algoritmy, které prohledávají celý možný stavový prostor. Jejich nevýhodou je často exponenciální doba běhu, ale pro řešení nějakým způsobem omezené úlohy se objevují algoritmy s polynomiální dobou běhu. Obecně jsou tedy použitelné spíše na malé soubory vstupních dat a na hledání velmi krátkých motivů, nebo lze použít speciálně upravené algoritmy[2] pro běh na výpočetních clusterech.

Tuto skupinu lze rozdělit na dvě, případně třetí, která využívá principů prvních dvou skupin dohromady. Prvním principem je iterace přes vzory motivů (*pattern-driven*) – l-mery z množiny $|\Sigma|^l$, kde Σ je abeceda symbolů. Druhým principem je rozdělení celého vstupu na vzorky (*sample-driven*) – množinu až $t \cdot (n - l + 1)$ l-merů. Tyto l-mery jsou z definice úlohy již hotovými motivy, ale lze l nahradit menším číslem k a tyto k-mery využít jako základ pro hledání pomocí skenování jejich okolí. [4, 6, 13, 20]

[1] Tabulka 5 – porovnání algoritmu GAME proti EM, https://doi.org/10.1093/bioinformatics/btl147
[2] Např. https://doi.org/10.1093/bioinformatics/btv466

Algoritmů patřících do této skupiny je mnoho, část z nich volí hledání hrubou silou, Algoritmy lze dále dělit podle datové struktury využité pro hledání: hledací strom nebo trie, hashovací datová struktura, graf nebo vzácněji i *integer linear programming* (ILP) nebo *branch and bound* metoda. Jmenovitě největší část z této skupina obsahuje v názvu PMS a nějaký přívlastek nebo číslovku (např. recentnější qPMS9 nebo v rámci této skupiny často porovnávaný PMSprune), z jiných např. SMILE, PAMPA, Weeder nebo VOTING, který využívá mechanismu hlasování, kdy každá sekvence obsahující konkrétní motiv mu dá právě jeden hlas. [4, 6, 13, 20]

2.4 Porovnávání motivů

Vzhledem k velkému množství převážně heuristických nástrojů na hledání motivů se doporučuje jich využít několik, tak aby se maximalizovala pravděpodobnost nalezení všech hledaných motivů. To vede na otázku, jak motivy mezi sebou porovnávat. Porovnávat se motivy dají dle biologických vlastností nebo z matematického hlediska. Nejobecnějším zápisem motivu je matice pravděpodobností, na kterou lze jiné matice jako PFM nebo PWM snadno převést. Oproti jiným hodnotám má pravděpodobnost výhodu fixního rozsahu od 0 do 1, lze tedy spočítat „vzdálenost“ matice obdobně jako vzdálenost vektorů. Tento přístup není ale úplný, neboť nevyužívá znalosti o problematice motivů. Kromě přímého porovnání matic existují také algoritmy založené na statistických metodách, např. Pearsonův χ^2 test, Pearsonův korelační koeficient nebo Fisherův exaktní test, které porovnávají, zdali jsou si dva sloupce (tj. jednu konkrétní pozice prvního motivu proti nějaké pozici druhého motivu) matice podobné. [9, 22, 23]

Nástroj MOTIFSIM umožnuje porovnávat libovolný počet vstupních seznamů motivů v různých formátech. Nejprve všechny motivy převede na PPM a také k nim vytvoří komplementární verze. Poté pro všechny možné dvojice motivů spočítá skóre podobnosti a tím je seřadí od nejpodobnějších po ty, které jsou na prahu, který si uživatel zvolil před spuštěním nástroje. Výstupem jsou pak seznamy dvojic podobných motivů: pro každý vstupní seznam zvlášť a jeden pro globální porovnání napříč seznamy. Nástroj také nabízí vytvoření obrázku se stromem[1], který podobně jako fylogenetický strom ukazuje skupiny podobných motivů. [9]

[1] Ukázka od autorů nástroje MOTIFSIM: http://motifsim.org/download/results/v2-2/R_06-18-18_1442_Global_Matching_Motif_Tree.png

Druhým využitím porovnání motivů podle podobnosti je hledání v existujících databázích motivů (např. JASPAR nebo TRANSFAC). Tuto funkcionalitu nabízí algoritmus Tomtom, který kromě zpracování dat z databází implementuje vlastní metodu pro porovnání motivů. Ta však v jádru používá funkci pro porovnání dvou sloupců a k ní sedm implementací, z nichž čtyři jsou zmíněné v prvním odstavci této kapitoly. Na výstupu algoritmus vrací takové motivy, jejichž podobnost je lepší než nastavený práh významnosti pomocí E- nebo q-hodnoty. [22]

2.5 Hledání motivů v datech ze sekvenování

Motivy lze hledat i v sekvencích, které pocházejí přímo z NGS, např. RNA-seq nebo ChIP-seq. Většina moderních sekvenátorů generuje FASTQ soubory, ze kterých je nutné nejprve vytvořit původní sekvenci pomocí de novo zarovnání nebo zarovnáním proti referenčnímu genomu do BAM souboru. Ze zarovnané sekvence si uživatel dle svého uvážení vybere část, ve které chce hledat, např. oblasti zkoumaných genů či jejich části. Kvalita přípravy sekvencí je klíčovým faktorem pro nalezení potenciálně biologicky signifikantních motivů, jakékoliv sekvence navíc, až už zdvojené nebo pro daný experiment zbytečné, zanáší do hledání šum.

Po přípravě sekvencí následuje samotné de-novo hledání motivů. Kromě primární vstupní sekvence, ve které chceme motivy hledat, jsou běžné ještě dva argumenty: sekvence pozadí a omezení délky hledaného motivu. Omezení délky slouží hlavně ke zkrácení doby běhu algoritmů, protože jinak by mohly být hledány i motivy řádově ve stovkách nukleotidů dlouhé, což většinou není žádoucí. Sekvence pozadí slouží ke kontrole FP (*false positive*) motivů, příkladem ve vstupní sekvenci může existovat podřetězec s relativně málo výskyty a být tedy potenciální kandidátem na motiv, ale ve skutečnosti se může jednat o běžný podřetězec s velkým výskytem mimo vstupní sekvenci. Tento přístup je v literatuře referován jako diskriminační hledání motivů.

Poslední částí je vyhodnocení nalezených motivů, které se může skládat z více kroků: porovnání motivů mezi sebou případně proti existujícím databázím (kapitola 2.4), sloučení motivů v nové motivy, nové, ale globální, ohodnocení na vstupních sekvencích (protože různé programy používají různé metriky kvality), aj. Poté už je jen na uživateli, co s nalezenými motivy udělá. [6, 11, 24]

2.6 Formáty sekvencí a motivů

FASTA – člověkem čitelný formát pro zápis sekvence spolu s jejím obecným popisem. Jedná se o seznam záznamů, který je složen z řádku popisu začínajícím symbolem ‚>' a z jedno nebo více řádků sekvence. Popis může obsahovat libovolná data, není proto možné se snažit o nějakou interpretaci, je na uživateli, aby zajistil, že popis obsahuje data, která daný program očekává.

FASTQ – soubor typicky obsahující data ze sekvenace, tj. sekvence jednotlivých readů a data o jejich kvalitě. Formátově se jedná o rozšíření FASTA, kdy pro každou sekvenci existuje ještě jedna sekvence ASCII[1] symbolů od ‚!' do ‚~' vyjadřující kvalitu od nejmenší po největší – kvalita zde znamená pravděpodobnost, že se dané pozici opravdu nachází daný nukleotid. Obdobně jako u FASTA i zde popis může obsahovat libovolná data, ale při běžném využití obsahuje formát využívaný sekvenátorem použitým při sekvenaci, data lze tedy o něco lépe interpretovat. [11]

SAM/BAM/BAI – *Sequence Alignment/Map* textový formát, v binární verzi BAM s indexovacím BAI souborem. Formát slouží k uložení informací o zarovnání čtení (ze sekvenace) vůči referenční sekvenci (typicky genomu) a rozlišuje dva typy čtení: lineární – může obsahovat inzerce, delece nebo být částečně nenamapované, ale je pouze na jednom vlákně, chimérické – původní čtení nelze reprezentovat jako jedno lineární, např. došlo k mapování jedné části na ‚+' vlákno a druhé na ‚–' vlákno, ale lze je po rozdělení na části zapsat jako vícero lineárních čtení. Každé zarovnané čtení obsahuje alespoň 11 povinných údajů, vybrané údaje: samotné čtení a jeho kvalita, pozice a kvalita zarovnání na referenční sekvenci, řetězec „CIGAR" obsahující pro každý nukleotid informaci o porovnání proti referenční sekvenci (shoda, inzerce, delece, …). [15]

MEME/TRANSFAC – textové ASCII formáty, které slouží pro zápis motivů v maticové podobě. V nejjednodušší podobě obsahují název motivu (unikátní v rámci jednoho souboru) a matici pravděpodobností v případě MEME formátu nebo matici frekvencí v případě TRANSFAC formátu. Dále mohou obsahovat různá metadata specifikující vlastnosti motivu, např. verzi nebo taxon organismu. [8]

[1] ASCII a UTF-8 jsou běžně používaná kódování textových souborů

GFF3 – textový UTF-8[1] formát pro popisování úseků sekvencí, primárně genomů, a jejich vlastností nebo příznaků. Formátově se jedná o 9tisloupcové CSV s tabulátorem jako oddělovačem. Prvních 8 sloupců jsou fixní typy jako pozice a typ vlastnosti nebo vlákno, poslední sloupce je řetězec reprezentující mapování `klíč=hodnota;klíč=…`, který slouží pro libovolná data, ale některé klíče jsou zadefinované: `Dbxref` je reference do nějaké databáze, `parent` ukazuje na nadřazenou vlastnost (např. konkrétní transkript ukazuje na gen, ze kterého pochází). [17]

BED – má obdobný formát jako GFF3, ale slouží primárně pro označování a podbarvení konkrétních úseků sekvencí. Na rozdíl od GFF3 poskytuje pouze jedno obecné pole pro popis a souřadnice jsou indexovány od nuly, GFF3 indexuje od 1. [18]

3 Analýza výchozí aplikace

Tato diplomová práce vychází z aplikace Jana Holčáka [7], kterou bylo nutné předem zanalyzovat a vybrat ty části, které půjdou použít pro tvorbu nové aplikace. Analýza byla provedena v rámci projektu předcházejícímu této práci a byla zaměřena na funkčnost jednotlivých částí, které jsem podrobil testování[1], a také na snadnou integraci do dalších programů pomocí kontejnerové virtualizace. V následujících kapitolách je popsán stav, který vznikl stažením přílohy diplomové práce [7] (složka `project_source_code`) a drobnými změnami při testování a evaluaci [příloha A – složka `projekt`, soubor `thesis.diff`, případně skript `download_unpack_patch_thesis.sh` pro stažení a aplikování změn]. Součástí projektu bylo také vypracování návodu na použití této aplikace, ze kterého jsou zde zakomponovány pouze některé části.

3.1 Analýza vybraných částí dle adresářů kódu

3.1.1 „bioplachta_containers“

Každý podadresář obsahuje soubory potřebné k vytvoření kontejneru ve virtualizační aplikaci kompatibilní s rozhraním Docker. Program nabízí kompatibilitu s platformou Docker a Singularity, přidání další platformy je relativně jednoduché – stačí aby byla schopna zpracovat „Dockerfile“ definici kontejneru a uměla při spuštění kontejneru přijmout mapování adresářů (angl. *volume mapping*). Pro tento účel byly autorem vytvořeny `build.sh` skripty a hlavní skript `build_all.sh`, které dohromady slouží k vytvoření všech obrazů kontejnerů pro Docker virtualizaci. Bohužel pro Singularity je nutné nejdřív buď vytvořit Docker obrazy a ty převést na Singularity obrazy nebo zkonvertovat soubory Dockerfile do formátu, který Singularity umí zpracovat. Autor zvolil první možnost, kdy hotové Docker obrazy nahrál na veřejně přístupný portál[2], díky čemuž zjednodušil instalaci pro další uživatele.

[1] Testováno na Windows 11 s využitím WSL 2 (Debian 11.8) a Singularity 3.11.5.
[2] Profil autora s jeho nahranými Docker obrazy https://hub.docker.com/u/plachta11b

Možnost využití obecně závisí na následujícím: pro část programů, které autor ve své aplikaci použil, Docker obrazy poskytují přímo tvůrci konkrétních programů, zde lze předpokládat zajištění vývoje do budoucna. Z Docker obrazů vytvořených autorem práce je část programů pravděpodobně již bez jakéhokoliv aktivního vývoje. Ale u těch, které ještě aktivní vývoj mají, by bylo ideální do budoucna zajistit alespoň základní formu manuálních aktualizací. V době psaní této práce budou využity všechny Docker obrazy, tak, jak je autor vytvořil, kvůli návaznosti na další části jeho práce, případné aktualizace a vylepšení mohou být předmětem dalšího vývoje.

Pokud by chtěl uživatel přidat další program na hledání motivů, který nemá vlastní Docker obraz, pak může v tomto adresáři manuálně vytvořit soubory pro sestavení nového obrazu. Samotné přidání obrazu do aplikace je středně náročným úkolem – vyžaduje úpravu na několika místech a vytvoření skriptů pro předání vstupu a zpracování výstupu programu.

3.1.2 „generate_fasta“

Skripty napsané v programovacím jazyce Bash v tomto adresáři slouží k převodům formátů nukleotidových sekvencí, zarovnání čtení ze sekvenace proti referenčnímu genomu, manipulacím se sekvencemi (vystřihování/slučování, tvorba konsenzus sekvencí, …), stahování referenčních genomů a jejich anotací či tvorbě nových randomizovaných sekvencí pro testování.

V adresáři `pipelines` se nacházejí skripty začínající `generate_fasta_`, které připraví FASTA soubory pro hledání motivů. První se sufixem `bedtools` slouží k tvorbě FASTA souborů přímo ze sekvenace a při běžném použití vyžaduje ještě několik jiných skriptů k předpřípravě dat. Druhý končící `biomart` stáhne pomocí Biomart API 5‘ a 3‘ UTR oblasti (lze stáhnout i *gene_exon, cdna* a *coding* oblasti) z předdefinovaných genů, poté je mírně upraví pro hledání motivů. Poslední způsob generace je pomocí autorem vytvořeného programu PPRSG, který vygeneruje sekvence na základě vstupního modelu a parametrů a poté do nich vloží uměle vytvořené motivy. Takto vygenerované sekvence jsou určené pro kvalitativní a kvantitativní testování programů hledajících motivy, jelikož je předem známý výsledek.

Některé skripty obsahují konkrétní cesty ke vstupním souborům a vyžadují nastavení dvou enviromentálních proměnných: `DATASET_eIF4E` – fixní adresářová struktura s daty z jedné konkrétní sekvenace[1], `MF_PROJECT_DATA_DIR` – adresář pro ukládání výstupů, stažených souborů apod. se subjektivně zbytečně složitou vnitřní strukturou (viz ukázky cest k souborům v následující kapitole). Z těchto tří důvodu by bylo velmi náročné použít tyto skripty v další aplikaci – převedení prvních dvou na argumenty skriptů vyžaduje rozsáhlejší změny v kódu skriptů.

3.1.3 „find_motifs“

Tento adresář je svojí vnitřní strukturou podobný adresáři `generate_fasta`, ze kterého jako vstup pro hledání motivů přebírá FASTA soubory. Druhou funkcí skriptů je převádění a sloučení výstupů programů hledajících motivy.

V adresáři `pipelines` se nacházejí skripty `run_pipeline.sh` a `run_pipeline_biomart.sh`, které spustí hledání motivů v FASTA souborech ze sekvenace a z Biomart API. Oba skripty využívají seznam zvolených programů ve skriptu `pipelines/scripts/search.sh`, kde lze za/odkomentováním řádku zvolit, zdali bude program součástí hledání nebo ne. U některých programů jsou uvedeny poznámky o kvalitě výsledků či době běhu. Po spuštění skriptu jsou postupně prohledány všechny zvolené datasety, seznam zvolených datasetů lze upravit v souboru `generate_fasta/pipelines/prefix_filter` v adresáři výstupních souborů.

Výstup hledání je uložen do složky `find_motifs` v adresáři výstupů – obsahuje výsledky hledání motivů i jejich souhrny z následné konverze. Struktura složky je:
`[název datasetu]/result_motifs/[název programu na hledání motivů]/[název konkrétního hledání]/[je-li program framework s více algoritmy pak cesta obsahuje „tag_[název algoritmu]“]`
, kde název konkrétního hledání je ve formátu:
`out_[minimální délka motivu]to[maximální délka motivu]_[„withbg“ nebo „nobg“ podle toho, zdali byla zadána sekvence pozadí (tj. proběhlo diskriminační hledání)]_[název vstupního FASTA souboru, ve kterém se hledaly motivy]`.

[1] Tato data byla poskytnuta vedoucím práce, měl jsem je k dispozici při testování a měl je k dispozici i autor analyzované aplikace.

Ve složce jednoho hledání lze nalézt kompletní výstup programu (včetně logů) a výstup konverze pomocí autorova programu lead2gold – tyto soubory začínají na `result_converted` a jedná se normální textové soubory. Konverze probíhá do třech formátů: seznam konsenzů motivů, MEME a TRANSFAC soubory s maticemi výskytů nukleotidů v motivech.

Celý tento mechanismus interně používá podčást autorem nazvanou Paraffin. Jedná se o skripty přímo ovládající jednotlivé kontejnery s programy na hledání motivů nebo jejich konverzi, které mají společný vstupní skript `containter.sh`, kterým se buď spustí hledání nebo konverze pomocí autorova programu lead2gold. Tento skript slučuje názvy parametrů jednotlivých programů a poskytuje kvalitní rozhraní pro jejich spouštění, proto jsem si tento skript jako jediný vybral k vytvoření rozhraní mezi autorovou a mojí aplikací.

Poslední významnou částí je podadresář `similarity/cluster_similar` se skripty pro porovnání nalezených motivů v rámci jednoho datasetu. Skript `similarity/compare/compare.sh` pak slouží k porovnání dvou předchozích porovnání. Za zmínku stojí ještě skript `logo.sh` pro zobrazení motivů ve formátu WebLogo. Tyto skripty jsou ale nestabilní nebo nefunkční, nebyly proto použity při vývoji nové aplikace.

3.1.4 „lead2gold“

lead2gold je autorem vytvořený program pro konverzi výstupů jednotlivých programů na hledání motivů do formátu MEME (pravděpodobnostní matice výskytu nukleotidu) a TRANSFAC (matic počtu výskytů nukleotidů), tak aby další zpracování výstupů bylo snazší a výstupy šlo porovnat mezi sebou.

3.1.5 Ostatní části aplikace

Ve zdrojovém kódu programu se místy vyskytují soubory pro měření výkonu a doby běhu jednotlivých částí, obecně tyto soubory obsahují v názvu `perf` nebo `performance`. Předposledním adresářem s kódem je `motif_description`, kde se nacházejí skripty spouštějící program Tomtom pro porovnání jednoho motivu proti databázi motivů[1]. Posledním adresářem je `mp_prepare`, který společně s několika skripty v adresáři `generate_fasta` slouží k transformacím dat z jedné konkrétní sekvenace a svojí syntaxí kódu i výstupem se významně odlišují od zbytku aplikace, proto tento adresář nebyl dále analyzován.

3.2 Vyhodnocení analýzy

Cílem analýzy bylo zjistit jaké funkce poskytuje aplikace Jana Holčáka, zdali tyto funkce dělají to, co mají, a také jaká je jejich využitelnost pro další vývoj.

Aplikace má funkční a kvalitně provedou část pro manipulaci s Docker obrazy, tato část bude využita v nové aplikaci. Část pro manipulaci se sekvencemi, ať už lokálně vytvořenými nebo staženými z internetu, byla otestována a je z větší části funkční, ale i tak pro další použití je nevhodná kvůli fixním cestám k souborům. Tato část bude v nové aplikaci naimplementována od základu znova, ale tak, aby byly přibližně zachovány nabízené funkce. Nástroje pro následné zpracování nalezených motivů jsou pouze naznačeny a nepodařilo se mi je spustit, alespoň je lze využít jako základ pro další rešerši.

Obecnou nevýhodou je uzavřenost pouze na RNA sekvence. V DNA sekvencích lze hledat také, ale programy, které umožnují přepínat mezi DNA a RNA vstupem, jsou nastaveny na RNA, a proto se motivy s komplementárním výskytem nemusí objevit ve výstupu hledání. Tento nedostatek z časových důvodů bude zachován.

[1] Tato část programu bohužel nebyla otestována kvůli následujícímu souboru http://alternate.meme-suite.org/meme-software/Databases/motifs/motif_databases.12.19.tgz, který mi nešlo stáhnout.

4 Popis vytvořené aplikace

Cílem je vytvořit aplikaci, která naváže na předchozí aplikaci (kapitola 3) a přidá nové funkce. Předchozí aplikace byla napsána v jazyce Bash s využitím virtualizace, což umožnuje relativně snadné spuštění na většině unixových operačních systémů. Problém může nastat při spuštění na operačním systému Windows, který v základní instalaci nemá interpret jazyka Bash a také může nastat problém při převodu cest k souborům (unixové systémy používají jako oddělovače znaky ‚/' a ‚;', kdežto Windows používají ‚\' a ‚:'). Tento problém lze obejít pomocí WSL a spustit celou aplikaci v něm. Rozhraním je pouze příkazová řádka.

Nově vytvořená aplikace má název *Programmable Motif Discovery Tool* je napsána v Javě 21, tento jazyk jsem zvolil proto, abych předešel případným problémům při běhu na různých operačních systémech a také proto, že v základní distribuci JRE je dostupná GUI knihovna, což mi umožnilo vytvořit jednoduché grafické rozhraní pro konfiguraci programu. Stejně jako předchozí aplikace podporuje nově vytvořená aplikace virtualizaci pomocí Docker a Singularity.

4.1 Základní popis aplikace

Vzhledem k různorodosti biologických experimentů při hledání motivů a kvůli existenci dvou uživatelských rozhraní, GUI a CLI, je základním prvkem prostředkem v aplikaci konfigurace „experimentu", který se skládá z jednotlivých výpočetních kroků, které si zvolí dle svých potřeb. Aplikace proto nemá konkrétní fixní vstup ani výstup, oboje záleží na vstupních a výstupních krocích. I přesto lze primární vstup definovat jako nějakou formu sekvencí a primární výstup jako nějaký formát nalezených motivů.

Tento přístup může subjektivně být o něco složitější na manipulaci než předávaní argumentů v příkazové řádce. Ale pro lze argumentovat, že pro běžného uživatele je pohodlnější konfiguraci vytvořit pomocí GUI než číst dokumentaci k jednotlivým argumentům. Konfiguraci je možné uložit a následně spustit nebo načíst na jiném stroji, což lze přirovnat k manipulaci se seznamem příkazů pro příkazovou řádku uloženém ve formě skriptu. To také umožňuje předem vytvořenou složitější konfiguraci snadno spustit na hardwarově výkonnějším stroji (např. výpočetním clusteru), kde je k dispozici pouze CLI.

4.1.1 Konfigurační formát a definice experimentu

Konfiguraci experimentu lze uložit jako soubor ve formátu JSON, který jsem zvolil kvůli možnosti snadné editaci libovolným textovým editorem a kvůli široké podpoře napříč jinými programovacími a skriptovacími jazyky. Lze tedy např. měnit cesty ke vstupním souborům při sekvenčním spouštění aplikace nad různými sadami dat.

Jeden experiment (rsp. jeho konfigurace) se skládá z libovolného počtu kroků, kdy každý krok může mít libovolný počet argumentů, které slouží jako nastavení daného kroku, a vstupní a výstupní proměnné, které slouží ke vzájemnému spojování kroků, spojené kroky pak vytváří graf úloh (*task graph*). Argumenty mají konkrétní datový typ a většinou jasně definovaný rozsah hodnot. Proměnné jsou mapovány na soubory na disku uživatele, jejich datový typ je na venek uživateli skryt – interně je reprezentován jako slabý datový typ pomocí předpokládané přípony souboru.

4.1.2 Argumenty aplikace

Aplikace má 3 argumenty:

- `-c` nebo `--headless` – pokud tento přepínač není zadán, spustí se GUI verze aplikace, při zadání tohoto přepínače se spustí verze s CLI a je vyžadován přepínač `-e`
- `-e` nebo `--experiment` následovaný cestou ke konfiguraci experimentu – aplikace načte tuto cestu automaticky, v GUI verzi se otevře editační menu, v CLI verzi se experiment rovnou spustí. Data vygenerovaná při běhu experimentu jsou uložena v adresáři, který se vytvoří ve stejné složce jako soubor experimentu a jmenuje se `[název experimentu bez přípony souboru]_data`. Tento adresář je kdykoliv možné smazat, důležitá je pouze konfigurace
- `-r` nebo `--runtime-dir` následovaný cestou k adresáři pro ukládání globálních dat aplikace. Není-li argument zadán, nebo je prázdný, pak je cesta shodná s cestou, ze které byla aplikace spuštěna. Struktura adresáře: konfigurační soubor aplikace `global_config.json`, výchozí adresář pro ukládání experimentů a jimi vytvořených dat `experiments`, adresáře pro uložení dočasných souborů aplikace `singularity_repo`, `docker_repo` a `paraffin_framework`. Dalšími podsložkami jsou ekvivalenty adresářů ve zdrojovém kódu pro statická data: `language` – pro jazykové mutace, `templates` – pro předlohy konfigurací experimentů, a `genetic/genome` – pro definice referenčních genomů.

4.1.3 Grafické rozhraní aplikace

GUI je implementováno pomocí knihovny Swing. Skládá ze tří menu: základní menu, editační menu jednoho konkrétního experimentu a menu pro zobrazení aktuálně spuštěného experimentu. Za zmínku stojí ještě menu nastavení (kapitola 4.1.5).

Základní menu se otevře při spuštění aplikace a nabízí seznam experimentů uložených ve výchozí složce, seznam předloh a tlačítka pro vytvoření/otevření experimentu a nastavení. Při kliknutí na předlohu se vytvoří nový experiment, předlohy ve možné vytvářet editací a zkopírováním konfigurace experimentu do složky `templates` v globálním adresáři aplikace. (obrázek 3 v příloze C)

V editačním menu lze přidávat, přesouvat a odebírat jednotlivé kroky, u každého kroku je možné nastavit argumenty, vepsat názvy výstupních proměnných a zvolit/vepsat názvy vstupních proměnných – aby zvolení proměnné fungovalo, je třeba provést kontrolu konfigurace, během které se zjistí, jaké proměnné z předchozích kroků lze použít. V horní liště lze pak zkontrolovat konfiguraci, spustit, uložit, znovu načíst z disku nebo smazat experiment včetně vygenerovaných dat (v současné verzi aplikace jsou tlačítka s negativním nebo destruktivním významem bez potvrzovacího dialogu). Experiment se také uloží při zavření aplikace, je ale lepší ho před každým spuštěním/zavřením manuálně uložit, aby se zabránilo ztrátě konfigurace. Při kontrole konfigurace se u každého vstupu může objevit oznámení, problém nebo chyba upozorňující uživatele na další informace ohledně zadaných hodnot. Téměř všechny prvky tohoto menu mají popisky (*tooltips*), které detailněji vysvětlují, co daný krok dělá nebo argument ovlivňuje. (obrázek 4 v příloze C)

V menu zobrazujícím postup spuštěného experimentu je u každého kroku lišta zobrazující procentuální postup a jeho výstup. Výstupem kroku může být textový status nebo speciální typ výstupu, např. tlačítko otevírající soubor či složku nebo obrázek WebLoga. Dole se nachází tlačítko na zastavení experimentu, nejedná se vždy o okamžité zastavení experimentu, někdy je třeba počkat na dokončení části kroku, obzvlášť pokud ona část běží ve virtualizaci. Zavření aplikace při běhu experimentu se v současné verzi aplikace nedoporučuje, důvodem je v současné verzi nedostatečná ochrana na ukončení virtualizačních procesů. (obrázek 5 v příloze C)

4.1.4 Rozhraní příkazové řádky

Aplikace je možné spustit i v režimu bez grafického rozhraní. Tento režim rovnou spustí experiment, pokud je jeho konfigurace v pořádku, jinak skončí chybovou hláškou. Běh experimentu lze zastavit posláním EOF (*end of file*) symbolu[1]. Stejně jako při GUI režimu se v současné verzi aplikace nedoporučuje okamžité ukončení běhu (pomocí Ctrl+C). Textový výstup CLI není určen pro parsování a je primárně informací pro uživatele, speciální výstupy kroků jsou většinou převeditelné na text, výjimkou jsou např. obrazové výstupy. Skončí-li běh chybou, pak je nastaven příslušný nenulový *exit code*[2].

4.1.5 První spuštění aplikace

Při prvním spuštění, nezávisle na zvoleném rozhraní, aplikace uživatele vyzve, aby si vybral, jakou virtualizační aplikaci chce použít – zobrazí se nabídka s volbou aplikace a detekovanou verzí. V režimu s GUI je navíc možné zvolit jazyk aplikace, zapnout zobrazení experimentálních funkcí nebo např. změnit počet jader pro paralelní výpočty. Jazyk a jiná nastavení CLI lze změnit v globální konfiguraci aplikace. (obrázek 7 v příloze C)

4.2 Funkcionality aplikace a jejich implementace

Následující kapitoly popisují jednotlivé kroky a části aplikace a jejich implementace. Orientační seznam *packages*[3] a jejich obsahu:

- `cz/schimpet/pmdt` – obsahuje obecné třídy aplikace a `Main` třídy a všechny následují *packages*
- `cli` – třídy implementující CLI
- `container` – rozhraní pro ovládání virtualizačních aplikací
- `experiment` – mechanismus kroků a konfigurace experimentu
- `gui` – třídy využívající knihovnu Swing pro implementaci GUI
- `plugin` – rozhraní zásuvných modulů
- `step` – implementace jednotlivých kroků, jejich datových typů a všechny k tomu potřebné pomocné třídy, veškeré funkce specifické pro hledání motivů jsou zde
- `util` – podpůrné třídy aplikace

[1] Klávesová zkratka Ctrl+Z pro Windows, Ctrl+D pro většinu unixových operačních systémů
[2] Tabulka s kódy je v příloze A v souboru `src/main/java/cz/schimpet/pmdt/cli/exit_code_map.txt`
[3] *package* je ekvivalent adresáře v programovacím jazyce Java

4.2.1 Definiční a vstupní kroky

První skupinou kroků jsou takové, které slouží pouze k definici nějaké proměnné. Jejich implementace obsahuje, až na FASTA soubor, pouze validaci vstupu. V současné verzi aplikace lze zadefinovat cesty k souboru (formáty: FASTA, BAM, MEME motivy, JSON motivy – interní formát aplikace), seznam genů nebo transkriptů ve formátu Ensembl – `ENSyyyGx` nebo `ENSyyyTx`, kde `x` je 11místné číslo genu nebo transkriptu a `yyy` je třípísmenná zkratka druhu organismu, kterou lze v případě člověka vynechat.

Speciální funkci má krok pro zavedení FASTA proměnné, který generuje interní metadata ke vstupní sekvenci. Metadata poskytují obecné informace o sekvenci bez nutnosti číst nebo modifikovat původní FASTA soubor. Při generování metadat se interpretují názvy FASTA sekvencí, které mohou obsahovat libovolné informace, současná verzi aplikace umí detekovat název chromosomu a souřadnice v rámci chromosomu – rozpoznáván je pouze UCSC[1] formát `chr[název chromosomu]:[start]-[konec]` – start a konec tvoří uzavřený interval indexovaný od 1, lze také použít formát `[start]+[délka]`, pokud je sekvence složena z více nesouvislých úseků, lze opakovat buď celý formát, nebo jen interval přes středník nebo čárku. Ostatními metadaty jsou: volitelně název a verze referenčního genomu, délka a počet jednotlivých symbolů v sekvenci.

Do této skupiny lze zařadit i krok na filtrování FASTA souboru podle názvu sekvence. Tento krok má jako jediný argument textové pole, které je interpretováno následovně: každý neprázdný řádek, který není komentář, může začínat `r:` nebo `c:` nebo ničím. Prefix `r:` slouží pro zadání regulární výrazu, řádek bez prefixu nebo s prefixem `c:` je interpretován jako hledání pomocí exaktního podřetězce.

[1] Formát pojmenovaný podle *University of California, Santa Cruz*, jejíž *Genomics Institute* provozuje webovou stránku *Genome Browser*, která tento formát používá

4.2.2 Výstupní kroky

Primární výstupem aplikace jsou motivy, aplikace nabízí následující formáty motivů: konsenzus motivu, WebLogo obrázek a MEME soubor se všemi motivy (obsahující jejich PPM). První dva formáty lze buď uložit do souboru nebo zobrazit přímo v aplikaci, MEME soubor lze pouze uložit. U každého motivu, který vznikl jako výstup nějakého kroku, je v obecném popisku uvedeno, ze kterého programu nebo kroku pochází ve formátu `prefix:sufix`, kde prefix je název aplikace a sufix je název algoritmu, ze kterého motiv pochází. Možné prefixy jsou „paraffin“ (pro motivy z hledání pomocí programů frameworku Paraffin, sufixy jsou názvy programů pro hledání motivů) nebo „pmdt“ (pro motivy vytvořené v této aplikaci, sufix je zatím jediný možný „motifsim“ pro sloučené motivy v krocích z kapitoly 2.4).

Pro formát WebLogo jsem vytvořil virtualizační kontejner, který je složen z minimálního kernelu Alpine Linux ve verzi 3.20 + Python3 interpretu + WebLogo3 knihovny[1]. Při tvorbě jsem narazil na problém, kdy byla vyžadována dnes již relativně stará verze „9.50-r1“ knihovny ghostscript. V návaznosti na to jsem musel přidat *package* repositář ve verzi 3.11, abych mohl knihovnu stáhnout. (obrázek 8 v příloze C)

Sekundárním výstupem jsou soubory, které vyprodukují programy použité v rámci některých kroků, např. u programů na hledání nebo porovnání motivů to mohou být HTML soubory vhodné pro zobrazení uživatelem. Některé tyto soubory jsou prezentovány i jako výsledky kroku, ve všech ostatní případech je lze dohledat v adresáři s daty experimentu, který se nachází ve stejné složce jako soubor experimentu.

4.2.3 Krok na stažení sekvencí pomocí Biomart API

Tento krok umožňuje stáhnout oblasti genů nebo transkriptů bez potřeby manuálně vyplňovat uživatelské webové rozhraní, ale na rozdíl od webové rozhraní poskytuje velmi malou část ze všech možností. Při stahování je nutné vyčistit neplatné sekvence, deduplikovat názvy sekvencí (některé programy striktně vyžadují unikátní názvy sekvencí) a vygenerovat metadata (viz druhý odstavec kap. 4.2.1). Uživatel má možnost také volitelně deduplikovat sekvence transkriptů, někdy jsou pod různými ID transkriptů na nukleotid stejné sekvence.

[1] https://weblogo.threeplusone.com/ a https://github.com/WebLogo/weblogo

Během implementace bylo pro mě velkým problémem se vyznat v několikero API, které Ensembl potažmo Biomart nabízí. Nakonec jsem se rozhodl, že raději, i přes obtíže s XML konfigurací, využiji stejnou API, kterou využívají uživatelé při osobní návštěvě na stránce https://www.ensembl.org/biomart/martview/.

4.2.4 Krok pro de novo hledání motivů

Vstupem tohoto kroku jsou jeden nebo dva FASTA soubory (sekvence, ve které se budou hledat motivy, a sekvence kontrolního pozadí), množina programů pro hledání motivů a interval délky hledaných motivů, který primárně slouží pro omezení délky běhu programů. Výstupem je množina nalezených motivů. Pokud nejsou nalezeny žádné motivy, pak experiment skončí chybovou hláškou.

Množina programů je rozdělena na dvě podmnožiny, kdy druhá obsahuje nezcela funkční nebo nezprovozněné programy. Tato podmnožina je zobrazena pouze pokud jsou zapnuty experimentální funkce v nastavení aplikace.

V tomto kroku je využita část aplikace analyzovaná v kapitole 3 – framework Paraffin napsaný v jazyce Bash. Tento framework nabízí mnoho programů na de novo hledání motivů a slučuje jejich rozhraní za jeden vstupní skript `container.sh`. Mezi argumenty tohoto skriptu patří vstupní sekvence, výstupní složka, vybraný program na hledání motivů a režim – buď hledání nebo konverze výsledků. Kód skriptu tvoří překlad vstupních argumentů, následovaný vybráním a voláním skriptu obsluhující argumenty zvoleného programu a zakončený voláním kontejneru s programem.

Integrace tohoto skriptu v mojí aplikaci je složena z následujících částí:

- nejprve je nutné stáhnout a extrahovat framework z přílohy autorovy diplomové práce, extrahuje se pouze potřebná část, na kterou se zároveň aplikují mnou vytvořené drobné úpravy (viz třída `ParaffinFrameworkDownloader`, úpravy: opravy skrytých chyb, přemostění výstupu skriptů tak, abych mohl ovládat spuštění kontejneru ve své aplikaci). Upravená část je uložena do složky `paraffin_framework` v globálním adresáři aplikace.
- následuje instalace všech kontejnerů ze souboru `paraffin_framework/find_motifs/repository.config` pomocí virtualizačního rozhraní, které uživatel zvolil v nastavení aplikace

- samotné hledání motivů – každý program pro hledání motivů má v kódu aplikace zvlášť definovanou svoji konfiguraci, ve které je upraveno, jak se konkrétní program chová na výstupu nebo jaké odlišné argumenty navíc potřebuje. Tato konfigurace je spolu s argumenty kroku při spuštění transformována na argumenty skriptu `container.sh`, který je poté spuštěn v kontejneru s interpretem jazyka Bash a který vrátí název kontejneru, příkaz programu s argumenty, mapování složek kontejneru (*volumes*) aj. Po spuštění programu na hledání motivů následuje druhé spuštění skriptu `container.sh` v režimu `convert`, který slouží ke zjištění argumentů pro program lead2gold, který je následně spuštěn. Výstupem tohoto programu jsou MEME soubory motivů s názvem `result_converted.pwm`, bohužel tyto soubory nemají fixní cestu a ani počet, proto je nutné tyto soubory hledat pomocí regulárního výrazu.

Spouštění programů je řízeno počtem dostupných vláken procesoru, tento počet lze upravit v nastavení aplikace. Tento přístup je poněkud problematický, některé programy jsou jednovláknové a jiné umějí samy využít všechna vlákna procesoru, může tedy dojít k přehlcení procesoru a tím zbytečnému prodloužení doby běhu. Řešení tohoto problému jsem z časové náročnosti odložil na později, neboť bych musel každý program analyzovat v lepším případě čtením manuálu, v horším případě čtením kódu nebo analýzou systémových volání (pokud program není open-source).

4.2.5 Krok pro hledání motivů v sekvenci

Tento krok je implementován pomocí programu FIMO z MEME-suite frameworku, který prohledá pro všechny pozice ve vstupní sekvenci a pro každý nález motivu spočítá *log-likelihood ratio* skóre, ze kterého spočítá p-hodnotu – pravděpodobnost, že nahodilá sekvence o stejné délce jako motiv bude mít lepší skóre, než je skóre motivu, a také q-hodnotu – minimální *false discovery rate* (FDR), při které lze nález motivu považovat za statisticky signifikantní. Maximální prahovou p-hodnotu na výstupu je možné nastavit jako argument. Pokud není zaškrtnut „Jednoduchý režim“, pak program zobrazí pro každý signifikantní nález příslušný úsek sekvence a mj. přidá HTML výstup.

Výstupem kroku je seznam motivů s p-hodnotou menší než nastavený práh, v GUI je možné procházet všechny nalezené pozice seřazené dle p-hodnoty. Při čtení výstupu programu jsou zahozeny takové nálezy, které v rámci jedné sekvence spadají do dvou úseků (modelový příklad: při stažení 5‘ UTR oblastí transkriptů je možné, že stažená sekvence vznikla sloučením všech UTR oblastí daného transkriptu, pokud program touto informací disponuje, pak je nutné nálezy motivů spadající do dvou úseků považovat za neplatné). [8]

Krok také nabízí možnost spočítat speciální „rc-skóre“. Pro tuto funkci je nutné vyplnit vstupní proměnnou SAM/BAM, která poskytuje počet čtení pro konkrétní úsek dat ze sekvenace. Tento počet může být rozdílný, buď se jedná o všechna čtení zasahující do jednoho nálezu motivu, nebo pouze o ta čtení, která překrývají celou oblast nálezu. Rc-skóre jsem definoval jako skóre motivu krát počet čtení, což by přibližně mělo reprezentovat, že méně kvalitní motiv s vyšším počtem čtení „zvítězí“ nad kvalitním motivem s minimálním počtem čtení. Motivy, ke kterým nebylo nalezeno žádné čtení, mají nutně nulové rc-skóre. Stejně jako u p-hodnoty je nutné brát v potaz to, že rc-skóre je závislé na délce motivu, a to jak při výpočtu samotného normálního skóre motivu, tak i při výpočtu počtu zahrnutých readů. (obrázek 5 v příloze C)

4.2.6 Kroky na porovnání motivů

Aplikace nabízí dva způsoby porovnání motivů: hledání podobných motivů v jedné množině nebo hledání podobných motivů ze dvou množin. Pro oba kroky je použit program MOTIFSIM (viz kapitola 2.4). První krok slouží pro sloučení nalezených motivů, kdy při použití vícero programů na hledání motivů je možné, že se některé motivy budou opakovat nebo si budou velmi podobné. Druhý krok umožnuje najít podobné motivy mezi dvěma množinami, analogicky se dá ten krok přirovnat k průniku dvou množin. Důležitým argumentem je procento podobnosti dvou motivů, kterým se nastaví práh, kdy mají být dva motivy označené za podobné. [9] (obrázek 6 v příloze C)

4.2.7 Implementace mechanismu kroků

Jeden krok se skládá z definice kroku (kód v aplikaci) a hodnot konfigurace (které zadal uživatel). Definice je složena ze seznamu argumentů a proměnných, funkce pro jejich validaci a funkce provádějící krok samotný. Provedení experimentu se skládá z následujících částí:

1) uživatel definuje seznam kroků včetně jejich argumentů a názvů proměnných
2) konfigurace experimentu je zkontrolována a je vytvořen graf provazující jednotlivé proměnné. Obsahuje-li konfigurace chyby, pak experiment nelze spustit, pokud pouze varování, má uživatel na výběr, jestli experiment spustí nebo upraví
3) následuje spuštění experimentu, kdy se kroky popořadě provedou. Krok může během svého provedení zobrazovat uživateli procentuální postup a textový a/nebo speciální výstup. Pro zrychlení při opakovaném spuštění je implementována mezipaměť konfigurace, kdy se současná konfigurace porovná s konfigurací předchozí spuštění. Provedení kroku je přeskočeno, pokud jsou stejné argumenty + vstupní proměnné pocházejí také z přeskočených kroků + existují soubory výstupních proměnných + krok sám o sobě umožnil využití mezipaměti + uživatel nezaškrtnul přeskočení mezipaměti před spuštěním experimentu. Mezi spuštěním dvou kroků může proběhnout transformace proměnných, pokud je to nutné (viz následující odstavec).

Jelikož mezi soubory různých přípon mohou existovat konverze, tak bylo naimplementováno hledání nejkratší cesty mezi dvěma typy proměnných. Tato cesta je reprezentuje transformační funkci mezi výstupní proměnnou jednoho kroku a vstupní proměnnou druhého kroku. Náročnější uživatelé mohou tuto konverzi obejít manuálně definovanými typy proměnných přímo v konfiguračním souboru experimentu. V současné verzi aplikace jsou implementovány převody mezi různými formáty motivů a také naivní extrakce FASTA sekvence z BAM/SAM souboru (uživatel je ale varován, poněvadž je lepší, když si tuto extrakci provede sám podle svých potřeb). (obrázek 4 v příloze C)

Pro zajištění uživatelské přívětivosti bylo implementováno několik vrstev tzv. *crash pads*, jejichž funkcí je se za všech okolností pokusit o zachování běhu aplikace, není-li to možné tak alespoň o zachování uživatelských dat. Jednotlivé vrstvy:

0) nejnižší vrstvu slouží pro případy, které se už opravdu zachránit nedají, v takových případech se uloží všechna uživatelská data a uživateli vyskočí dialog s chybovou hláškou, která je primárně určena pro vývojáře
1) kód obsluhující spuštění kroků běží na samostatném vlákně, takže pokud nastane obdoba 0) ale při běhu experimentu, tak samotná aplikace zůstane nedotčena
2) každý krok je izolován od ostatních, pokud tedy nastane chyba při vykonávání funkce kroku, pak se ostatní data o běhu experimentu uloží

3) vlákna kroků, které využívající paralelizaci, se chovají obdobně jako 2), tj. například při hledání motivů, pokud skončí chybou jeden algoritmus, tak ostatní doběhnou a uživatel je pouze informován o tom, kolik algoritmů nedoběhlo a proč

4.2.8 Rozhraní virtualizace a zásuvné moduly

Vzhledem k definici kroků a jejich argumentů je téměř celý kód aplikace je psaný *data-driven* přístupem s využitím registrů typů. To umožňuje snadné rozšíření aplikace pomocí zásuvný modulů (pluginů), které jsou implementovány pomocí Java SPI, jenž je reprezentováno třídou `Plugin`. Každý modul může registrovat svoje kroky a jiné typy nebo rozhraní virtualizace, případně přidávat překlady. Všechny kroky implementované v současné verzi aplikace jsou registrovány ze statického `RootPlugin`.

Rozhraní virtualizace umožňuje podporovat různé virtualizační hypervizory, v současné verzi aplikace je převzatá podpora z frameworku Paraffin – Docker a Singularity. Implementace dalšího hypervizoru vyžaduje následující základní funkce: detekci verze hypervizoru, příkaz pro instalaci nebo sestavení nového obrazu (z Dockerfile nebo ekvivalentního souboru) a spuštění příkazu pro konkrétní obraz v kontejneru pro jednorázové využití.

4.2.9 Ostatní implementační detaily

Jako jedna z problematických části implementace se ukázala možnost překladu rozhraní. U CLI je vyžadováno, aby CLI bylo nastaveno v kódování UTF-8, což bohužel stále ještě není dnes standardem[1]. U GUI je problematické využití knihovny Swing, která nemá podporu změny jazyka u aktuálně zobrazených elementů UI, uživatel musí buď změnit menu nebo restartovat celou aplikaci. Proto musely být vytvořeny třídy, které obalují překládaný text a odkládají provedení samotného překladu až na dobu, kdy je nutné překlad zobrazit. Pro validaci souborů s překladem jsem přidal do *build* skriptu funkci na kontrolu celého kódu, která hledá klíče překládaných textů a kontroluje, zdali překlady existují pro všechny v současnosti přidané jazyky.

[1] Např. Windows vyžaduje příkaz `chcp 65001`, u OS založených na Linuxu bývá UTF-8 standardem

Kvůli nemalému počtu datových struktur zapisovaných na disk a kvůli možnému vývoji do budoucna jsem využil knihovnu *DataFixerUpper*[1] pro abstraktní definice datových struktur, kdy je každá sktruktura definována kodekem. Kodek obsluhuje jak zápis, tak i čtení dohromady, čímž je eliminováno nebezpečí chyby při rozdílu mezi kódem pro čtení a pro zápis. Tato knihovna nabízí také verzování struktur, které nepotřebuje udržovat kód ke čtení a převodu dat ze souborů starších verzí – stačí pouze abstraktní definice změn mezi jednotlivými verzemi a převod už provede sama knihovna.

[1] https://github.com/Mojang/DataFixerUpper

5 Diskuse

5.1 Porovnání s podobnými aplikacemi

5.1.1 Tmod

Tmod je program s GUI pro operační systém Windows, vytvořený v roce 2009 a napsaný v jazyce C++, který slučuje v té době 12 používaných programů na de-novo hledání motivů. Pro běh programů využívá prostředí Cygwin, které emuluje prostředí Linuxových operačních systémů. Pro použití programů v tomto prostředí je nutné je překompilovat, což se u některých programů neobešlo bez nutných změn. Jinak kromě problémů spojených s prostředím Cygwin slibují autoři snadné přidání nových algoritmů na hledání motivů.

Z uživatelského hlediska je obrazovka rozdělena na tři části: výběr programu nebo nástroje, jeho nastavení a prostor pro obecný textový výstup. Programy lze spouštět buď zvlášť nebo v dávkovém režimu, který je vhodný pro postupné zpracování vícero souborů v jednom spuštění. Ke každému programu je třeba napsat konverzi výstupních souborů na motivy reprezentované konsenzuální sekvencí, tyto motivy jsou pak ohodnoceny pomocí BioOptimizer nástroje. [26]

Porovnání – nejdřív musím objektivně vytknout reprezentaci motivů pomocí konsenzů, která může být v některých případech dostatečná, ale při jejím použití nutně dochází ke ztrátě části informace. Z dalšího objektivního hlediska program poskytuje výrazně méně funkcí než mnou vyvinutá aplikace, ale na druhou stranu nabízí režim dávkového spuštění, který lze v současné verzi mé aplikace vytvořit pouze pomocí skriptování a využití CLI (tj. s využitím prostředků třetí strany). Subjektivně mi přijde, že uživatelského prostředí je o něco lepší, protože při konfiguraci lépe využívá prostor uživatelského rozhraní, na druhou stranu výstup je pouze textový, moje aplikace nabízí i WebLoga.

5.1.2 Galaxy[1]

Galaxy Project je velký framework vyvíjený od roku 2005 a poskytují mnoho funkcí napříč různými obory převážně molekulární biologie. V současnosti jsou přidávány funkce i pro jiné výzkumné účely z oblasti biologie nebo lékařství. V základu je dostupný jako veřejná webová stránka[1], ale je možné využít i lokálně pomocí CLI (relativně složitější než u mé aplikace) nebo si spustit lokální webovou instanci.

Porovnání – nástroj staví na podobných principech a nabízí podobné funkcionality: možnost nakonfigurovat si přesně to, co potřebuji, designu „*write once run anywhere*" – spuštění kdykoliv a kdekoliv (včt. výpočetních clusterů a cloudových datacenter), možnost sdílení a reprodukovatelnosti postupů a výstupů, a možnost přidání vlastních pluginů/nástrojů. Subjektivně bych tedy řekl, že vzhledem k modernímu webovému designu a možnosti online sdílení výsledků je nástroj z hlediska uživatelské přívětivosti znatelně lepší, ale objektivně velkou nevýhodou je dostupnost pouze pár vybraných nástrojů z MEME-suite frameworku. Proto bych považoval tento framework pro analýzu motivů za základní minimum, i přesto mi společně se znalostí Gradle[2] posloužil jako dobrá inspirace.

5.1.3 MEME-suite, DynaMIT a GimmeMotifs

MEME-suite[3] je rozsáhlejší skupina nástrojů pro komplexní analýzu sekvenčních motivů, která vznikla rozšířením o další funkcionality z původního programu MEME vydaném v roce 1995. Pro hledání motivů nabízí několik nástrojů v závislosti na typu hledaných motivů a vstupních sekvencí. Framework je v omezené míře dostupný jako veřejná webová stránka, ale možné si stáhnout zdrojové kódy, případně jednoduše využít Docker obraz. Z pohledu rozšiřitelnosti je možná pouze přímá editace zdrojového kódu, není dostupné žádné konkrétní API.

[1] https://galaxyproject.org/ a https://usegalaxy.org/

[2] Gradle je *build* systém pro kompilaci a vývoj aplikací psaných nejen v Javě, jedná se o komplexní systém založený na propracovaném grafu úloh, který lze pomocí pluginů rozšířit a využít takřka pro cokoliv

[3] https://meme-suite.org/meme/index.html

DynaMIT je sada nástrojů napsaná v jazyce Python s důrazem na podporu běhu napříč operačními systémy vyvíjená mezi lety 2015 až 2018. Instalovat lze jako *pip package* nebo pomocí autory předpřipraveného virtualizačního obrazu pro VirtualBox. Aplikace nabízí 16 algoritmů pro de-novo hledání motivů, 7 algoritmů pro sloučení podobných motivů do clusterů a 7 výstupních a zobrazovacích metod. Aplikace umožňuje přidání nových funkcionalit pomocí implementace konkrétních tříd. Spuštěním vyžaduje konfigurační soubor, který lze vytvořit s pomocí jednoduchého GUI programu, bohužel zde chybí popisy argumentů. [27]

GimmeMotifs je framework zaměřený na analýzu motivů transkripčních faktorů. Je také napsaný v jazyce Python a je vyvíjený od roku 2011. Nainstalovat lze pouze jako Python *package*. Framework nabízí řadu nástrojů: pro de-novo hledání motivů nabízí 17 algoritmů včetně možnosti diskriminačního hledání, hledání nalezených motivů v sekvencích či databázích nebo např. různá porovnání množin motivů. Nástroje je možné spustit pouze pomocí CLI, výstup je většinou pouze textový, ale je možné vygenerovat WebLoga a klíčové nástroje včetně de-novo hledání poskytují HTML soubor. [28]

Porovnání – všechny tři výše zmíněné nástroje mají subjektivně nedostatečné nebo žádné GUI, což významně snižuje uživatelskou přívětivost. Objektivně ale nabízejí obdobné funkce jako mnou vyvinutá aplikace, MEME-suite jich dokonce nabízí víc. Možnosti rozšíření jsou víceméně omezené na skriptování, které uživatel musí stejně použít, pokud si chce uchovat postup, jak k výsledkům došel. Ale vzhledem k existenci projektů BioConda pro jazyk Python a Bioconductor pro jazyk R a jim podobným, to nelze brát vysloveně za objektivní nedostatek.

5.2 Možnosti rozšíření funkcionalit

Během vývoje jsem narazil na několik problémů, které bych rád vyřešil, ale z časových důvodů jsem se místy omezil pouze na poznámky v aplikaci, které popisující daný problém uživateli, případně na *to-do* komentáře v kódu.

Prvním problémem, který se objevil při testování, je fakt, že běžný uživatel operačního systému Windows nemá práva na vytváření tzv. souborových *soft links*, které může na operačních systémech založených na Linuxu vytvářet kdokoliv. Aby tyto tato práva získal, musí administrátor provést nejednoduchou editaci hluboko v nastavení systému. Řešením tohoto problému je úplně přestat *soft links* používat a namísto toho implementovat obdobnou funkcionalitu.

Mnoho drobných problémů přináší nedostatečný popis Biomart API, kde je k dispozici hodně filtrů a atributů, ale ne všechny jsou pro mě, jako člověka s nepřímou znalostí molekulární biologie, pochopitelné. Během vývoje jsem např. narazil na problém, že neumím správně nastavit filtry tak, abych stáhnul *flank* oblasti transkriptů, aniž by to skončilo chybovou hlášku (které odkazovaly na chybu serveru). Také je velmi obtížné zjistit, který atribut souřadnic patří ke kterému typu sekvence, respektive nenašel jsem obecný atribut, který by vrátil souřadnice přesně k té sekvenci, kterou chci stáhnout. Obecně by se tedy dalo rozšířit stahování z Biomart API tak, aby uživatel mohl přímo zadat dotaz ve formátu XML, který lze vygenerovat na stránkách, čímž by měl uživatel plnou kontrolu nad staženými daty. Kromě této funkce bych rád přidal stahování podle verzí sekvencí, nyní je možné zadat jenom ID a tím stáhnout nejnovější verzi transkriptu – respektive při zadání ID bez verze by se při prvním stažení tato ID rozšířila o nejnovější verzi tak, aby se zachovala replikovatelnost konfigurace.

Potřebným rozšířením je zakomponování modernějších algoritmů pro hledání nových motivů, obzvlášť z oblasti hlubokého strojového učení, neboť nástroj Paraffin se omezuje spíše na starší, ale v literatuře často zmiňované, algoritmy. K tomuto rozšíření lze přidat možnost přepínání mezi DNA a RNA hledáním, případně i hledání v proteinových sekvencích, které ale vyžaduje zavedení reprezentace proteinogenních aminokyselin, které není tak explicitně zakonzervované napříč organismy jako nukleotidová abeceda (která je navíc reprezentovatelná včetně všech možných kombinací symbolů v rámci běžné abecedy ‚A' až ‚Z').

Další funkcionalitou, kterou bych rád přidal a měl ji také naznačenou autor analyzované práce, je prohledání existujících databází vůči právě nalezeným motivům. K tomu lze použít např. dříve zmíněný program Tomtom z MEME suite, nebo přímo soubory z databází.

Kromě rozšíření spjatých s motivy by bylo vhodné, pro přenositelnost a stabilitu výsledků, do souboru konfigurace experimentu zakomponovat verze použitých kontejnerů, v případě nesouladu s právě instalovanými verzemi by byl uživatel vyzván buď k instalaci jiných verzí, nebo k odsouhlasení natrvalo a přepsání na lokálně instalovanou verzi. Kromě přenositelnosti výsledků je možné také rozšiřovat způsoby prezentace výsledků, např. v současné verzi se neukládají extrahované výsledky z kroků na podobnost motivů, nebo přidat možnosti pro použití pro molekulární biologii běžných programů jako IGV apod. Za zmínku také stojí ne zcela vyřešené problémy z vývoje jako plánování využití CPU při hledání motivů.

6 Testování

Jednotlivé funkce byly zevrubně testovány během vývoje jak nad umělými daty menších rozměrů, tak i nad soubory z reálných experimentů, což by mělo zajistit dostatečnou stabilitu aktuální verze aplikace. Na konci vývoje proběhlo testování uživatelské přívětivosti a celkové funkcionality pomocí ověřitelných dat.

Testování uživatelské přívětivosti bylo provedeno formou prezentace vedoucímu práce, který si měl možnost program vyzkoušet. Výstupem byly připomínky k současnému stavu a nápady na rozšíření funkcionality. Osobně bych si vytknul přidávání a přesouvání kroků, kde, pokud bych měl více času, bych přidal *drag-and-drop* mechanismus, případně společně s propojováním kroků přes proměnné bych raději použil grafovou strukturu (uzel = krok, hrana = proměnná).

6.1 Testování založené na ověřitelných datech

První testovací dataset byl vytvořen podle zveřejněného článku o TOP (*terminal oligopyrimidine*) motivech v 5‘ UTR oblastech [21]. Tyto motivy začínají jedním C, které je následováno cca 4 až 15 pyrimidinovými nukleotidy, po kterých následuje oblast bohatá na G. V příloze A zmíněného článku je uvedeno 92 genů a jejich 5‘ UTR sekvence, ve kterých jsou žlutě vyznačeny TOP motivy. K těmto genům se mi podařilo v databázi Ensembl najít 80 kanonických transkriptů a nad těmi spustit hledání motivů podle předlohy „Search in transcripts from Biomart“ – finální konfigurace experimentu je v příloze B. Celkem bylo nalezeno 7 motivů a další 3 byly vytvořeny jejich sloučením.

Motiv	Zdroj	# def	# ≤ 15	#	Min. p-val	Min. q-val
CTCTTTC	DynaMIT - GLAM2	66	63	283	1,35e-04	0,0326
CTCTTTCAC	DynaMIT - GLAM2	65	67	251	6,48e-06	0,0272
CTCTTTCCGTYCCCSGG	sloučení	53	54	241	2,32e-05	0,0363
CTCTTTCCSYYCCCSGG	DynaMIT - GLAM2	52	54	242	2,21e-05	0,0427
CTGCTTTC	DynaMIT - GLAM2	45	60	266	2,53e-04	0,3750
TCTTTCCGT	DynaMIT - MDscan	43	69	289	9,01e-06	0,0321
CCTCTTTCMCYYCCCSGG	sloučení	40	45	209	1,45e-04	0,1720
CTGCTGCCRCC	BaMM	19	38	254	6,55e-06	0,0254
GCTGCYGCCRYC	sloučení	9	35	270	6,54e-06	0,0214
GCBGCCGCCRTC	DynaMIT - MDscan	7	32	239	1,66e-06	0,0046

Tabulka 1 – nalezené motivy a jejich vlastnosti, řazeno podle 3. sloupce, (WebLoga na obrázku 8 v příloze C)

V tabulce 1 jsou uvedeny všechny výstupní motivy a k nim:

- zdroj motivu – motiv buď byl nalezen konkrétním programem nebo byl vytvořen sloučením podle podobnosti
- # def – počet transkriptů, ve kterých je alespoň jeden nález motivu reprezentovaný regulárním výrazem `^C[CT]{4+}`, který odpovídá začátku definice, tj. nezabýval jsem se analýzou oblastí bohatých na G
- # ≤ 15 – počet transkriptů, ve kterých je alespoň jeden nález motivu takový, že první pozice nálezu je menší nebo rovna 15. pozici transkriptu. V článku jsou motivy vždy na začátku sekvence bez informace o tom, zdali se jedná o začátek 5‘ UTR oblasti nebo ne. U 19 transkriptů stažených pomocí Biomart API tomu tak nebylo, proto jsem započítal i nálezy dostatečně blízko počátku transkriptu
- # – celkový počet nálezů ve vstupních transkriptech
- minimální p- a q-hodnota – napříč všemi nálezy ve vstupních transkriptech

Motivy jsem seřadil podle nálezů, které z části odpovídají definici. K tomu jsem přidal aproximaci, která je založena na odhadu ze článku, že se hledané motivy vyskytují na začátku 5‘ UTR oblastí, a také lépe reprezentuje motivy, které nemají počáteční C.

První motiv (CTCTTTC) jako jediný zcela odpovídá definici 5‘ TOP motivu. Motivy na 2. až 4. místě obsahují šum v podobě oblastí za 1. motivem, který je jejich společným podřetězcem. 6. motiv také obsahuje šum, ale hlavně mu chybí počáteční C, přesto má největší počet nálezů „# ≤ 15“, což lze vysvětlit pomocí jiného článku [25], kde autoři zmiňují i existenci 5‘ TOP motivů, jejichž prvním nukleotidem je U. Zajímavý je 5. motiv, který obsahuje G na 3. pozici, a proto neodpovídá definici 5‘ TOP motivu, jeho nálezy ve vstupních transkriptech však definici motivu již splňují.

Kvůli krátkosti nalezených motivů jsem byl nucen při jejich evaluaci zvolit prahovou p-hodnotu 0,05 (která je považována za horní hranici pro statistickou signifikanci), jejímž důsledkem je vysoký počet všech nálezů. Bohužel ani ta nestačilo na to, aby 1. motiv měl více „# ≤ 15“ nálezů než 2., ačkoliv by mít měl, jelikož je jeho exatním podřetězcem a měl by proto mít stejný nebo větší počet nálezů do 15. pozice transkriptu. To lze vysvětlit závislostí p-hodnoty na délce motivu (p-hodnota má tendenci být menší s rostoucí délkou motivu). Zajímavostí je relativně vysoká minimální q-hodnota 5. a 7. motivu (který vznikl sloučením ze mj. 5. motivu), kterou bych si vysvětlil tím, že se velmi podobají ostatním motivům, takže jejich nálezy budou obdobné, ale zároveň část nálezů bude obzvlášť u 5. motivu znatelně odlišná.

Nejlepším algoritmem se ukázal GLAM2, který je v základu dostupný v MEME suite, ale zde je integrován pomocí frameworku DynaMIT. Program AlignACE byl při hledání vypnut, protože obecně produkuje velké množství motivů s vysokou variabilitou nukleotidů. Je překvapením, že z 18 algoritmů, které doběhly, pouze 3 našly nějaké motivy (GLAM2, MDscan a BaMM). Během vývoje aplikace se počet programů, které na obdobně velkých testovacích sadách našly nějaké motivy, pohyboval okolo 10.

Závěr

Cílem práce bylo zanalyzovat, upravit a rozšířit již existují aplikaci na hledání motivů o nové funkce. Mnou vytvořená aplikace přidává GUI, ve kterém si uživatel vybere kroky reprezentující různé funkce, tyto kroky spojí a tím vytvoří ucelenou funkcionalitu. Implementované funkce jsou: stažení sekvencí pomocí Biomart API nebo vložení cest k vlastním sekvencím, de-novo hledání motivů, které využívá výše zmíněnou existují aplikaci, porovnání motivů podle podobnosti s možností jejich sloučení, ohodnocení motivů podle jejich nálezů v zadané sekvenci s možností využití četnosti čtení ze sekvenačních dat v místě jednotlivých nálezů, uložení motivů v několika formátech a další drobné funkce.

Aplikace je napsaná v jazyce Java ve verzi 21 a využívá kontejnerové virtualizace pro izolaci běhu programů třetí strany. Aplikace byla vyvíjena s ohledem na replikovatelnost a přenositelnosti výsledků, vytvořený sled kroků lze uložit jako konfiguraci do souboru a spustit kdykoliv a kdekoliv. Dalším aspektem vyvinuté aplikace je její široké API umožňující ostatním programátorům přidávat pluginy s různými funkcemi. Zmínit lze ještě CLI, které vyžaduje předem vytvořenou konfiguraci, a obstojné zabezpečení proti pádu aplikace a ztrátě dat.

Během vývoje byla aplikace průběžně testována, na konci vývoje došlo k otestování uživatelské přívětivosti a k otestování na ověřitelných datech, kdy bylo experimentálně zjištěno, že aplikace je schopna najít očekávané motivy. Funkce, která využívá četnost čtení ze sekvenačních dat při ohodnocení motivů, byla otestována pouze na stabilitu, poněvadž se jednalo o data, u kterých nejsou ověřené výstupy.

Bibliografie

1 JELÍNEK, Jan a Vladimír ZICHÁČEK. Biologie pro gymnázia. 7. roz. vyd. Olomouc, 2004, s. 298-326. ISBN 80-7182-177-2.

2 *Wikipedie: Otevřená encyklopedie: Sequence motif* [online]. ©2024 [cit. 2024-04-07]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Sequence_motif&oldid=1189961680.

3 *Wikipedie: Otevřená encyklopedie: Structural motif* [online]. ©2024 [cit. 2024-04-07]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Structural_motif&oldid=1140249516

4 MOHANTY, Satarupa a kol. A Review on Planted (l, d) Motif Discovery Algorithms for Medical Diagnose. Sensors. 2022, 22(3). Dostupné z: https://doi.org/10.3390/s22031204

5 SCHNEIDER, T. D. a R. M. STEPHENS. Sequence logos: a new way to display consensus sequences. Nucleic Acids Research. 1990, 18(20). Dostupné z: https://doi.org/10.1093/nar/18.20.6097

6 HASHIM A. Fatma, Mai S. MABROUK a Walid AL-ATABANY. Review of Different Sequence Motif Finding Algorithms. Avicenna J Med Biotechnol. 2019, 11(2), s. 130–148. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6490410/

7 HOLČÁK Jan: Hledání sekvenčních motivů v mRNA selektovaných vazbou na translační iniciační faktory z rodiny eIF4E [online]. [cit. 2024-01-15]. Dostupné z: https://dspace.cvut.cz/handle/10467/90251

8 Charles E. Grant, Timothy L. Bailey, William Stafford Noble. FIMO: scanning for occurrences of a given motif. *Bioinformatics*. 2011. Dostupné z: https://doi.org/10.1093/bioinformatics/btr064

9 Ngoc Tam L. Tran, Chun-Hsi Huang. MOTIFSIM: A Web Tool for Detecting Similarity in Multiple DNA Motif Datasets. *BioTechniques*. 2015. Dostupné z: https://doi.org/10.2144/000114308

10 ALBERTS, Bruce a kol. Molecular Biology of the Cell. 5. vyd., 2005. ISBN 978-0-8153-4105-5

11 J. A. Carriço, M. Rossi, J. Moran-Gilad, G. Van Domselaar, M. Ramirez. A primer on microbial bioinformatics for nonbioinformaticians. *Clin Microbiol Infect*. 2018. Dostupné z: https://doi.org/10.1016/j.cmi.2017.12.015

12 PATRA, Sabyasachi a Anjali MOHAPATRA. Review of tools and algorithms for network motif discovery in biological networks. 2020. Dostupné z: https://doi.org/10.1049/iet-syb.2020.0004

13 Avinash Achar, Pål Sætrom. RNA motif discovery: a computational overview. *Biology Direct*. 2015, 10(61). Dostupné z: https://doi.org/10.1186/s13062-015-0090-5

14 EVANS, Patricia A., Andrew D. SMITH a H.Todd WAREHAM. On the complexity of finding common approximate substrings. *Theoretical Computer Science*. 2003, 1(3), s. 407-430. Dostupné z: https://doi.org/10.1016/S0304-3975(03)00320-7

15 Sequence Alignment/Map Format Specification. 2023-11-16. Dostupné z: https://samtools.github.io/hts-specs/SAMv1.pdf

16 Ensembl. GFF/GTF File Format - Definition and supported options. [cit. 2024-05-16]. Dostupné z: https://www.ensembl.org/info/website/upload/gff.html

17 Lincoln Stein. Generic Feature Format Version 3. 2020-08-18. Dostupné z: https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md

18 Ensembl. BED File Format - Definition and supported options. [cit. 2024-05-16]. Dostupné z: https://www.ensembl.org/info/website/upload/bed.html

19 Morris Michael 1, François Nicolas, Esko Ukkonen. On the complexity of finding gapped motifs. *Journal of Discrete Algorithms*. 2010, 8(2), s. 131-142. Dostupné z: https://doi.org/10.1016/j.jda.2009.12.001

20 Geir Kjetil Sandve 1, Osman Abul, Vegard Walseng, Finn Drabløs. Improved benchmarks for computational motif discovery. *Biology Direct*. 2015. Dostupné z: https://doi.org/10.1186/s13062-015-0090-5

21 Eric Cockman, Paul Anderson a Pavel Ivanov. TOP mRNPs: Molecular Mechanisms and Principles of Regulation. *Biomolecules*. 2020. Dostupné z: https://doi.org/10.3390/biom10070969

22 Shobhit Gupta, John A Stamatoyannopoulos, Timothy L Bailey & William Stafford Noble. Quantifying similarity between motifs. *Genome Biology*. 2007. Dostupné z: https://doi.org/10.1186/gb-2007-8-2-r24

23 Dustin E. Schones, Pavel Sumazin, Michael Q. Zhang. Similarity of position frequency matrices for transcription factor binding sites. *Bioinformatics*. 2004, 21(3), s. 307-313. Dostupné z: https://doi.org/10.1093/bioinformatics/bth480

24 Emma Redhead, Timothy L. Bailey. Discriminative motif discovery in DNA and protein sequences using the DEME algorithm. *Bioinformatics*. 2007. Dostupné z: https://doi.org/10.1186/1471-2105-8-385

25 Lucas Philippe, Antonia M. G. van den Elzen, Maegan J. Watson a Carson C. Thoreen. Global analysis of LARP1 translation targets reveals tunable and dynamic features of 5′ TOP motifs. 2020. Dostupné z: https://doi.org/10.1073/pnas.1912864117

26 Hanchang Sun, Yuan Yuan, Yibo Wu, Hui Liu, Jun S. Liu. Tmod: toolbox of motif discovery. *Bioinformatics*. 2010. Dostupné z: https://doi.org/10.1093/bioinformatics/btp681

27 Erik Dassi, Alessandro Quattrone. DynaMIT: the dynamic motif integration toolkit. *Nucleic Acids Res*. 2016. Dostupné z: https://doi.org/10.1093/nar/gkw119

28 van Heeringen S. J. a Veenstra G. J. C. GimmeMotifs: a de novo motif prediction pipeline for ChIP-sequencing experiments. *Bioinformatics*. 2011. Dostupné z https://doi.org/10.1093/bioinformatics/btq636

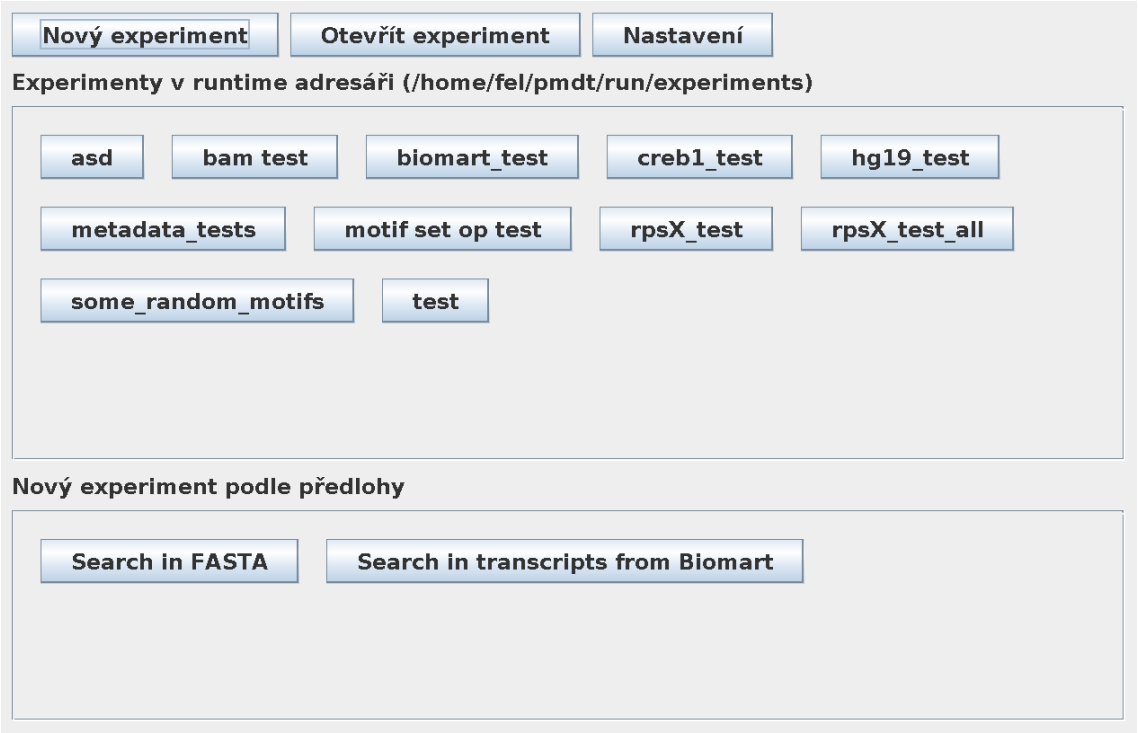
Seznam příloh

A Zdrojový kód aplikace (`A_source_code.zip`) – obsahuje `README.md` s manuálem ke kompilaci a seznamem nutných závislostí, kompletní zdrojový kód aplikace a kompilační konfiguraci pro Gradle

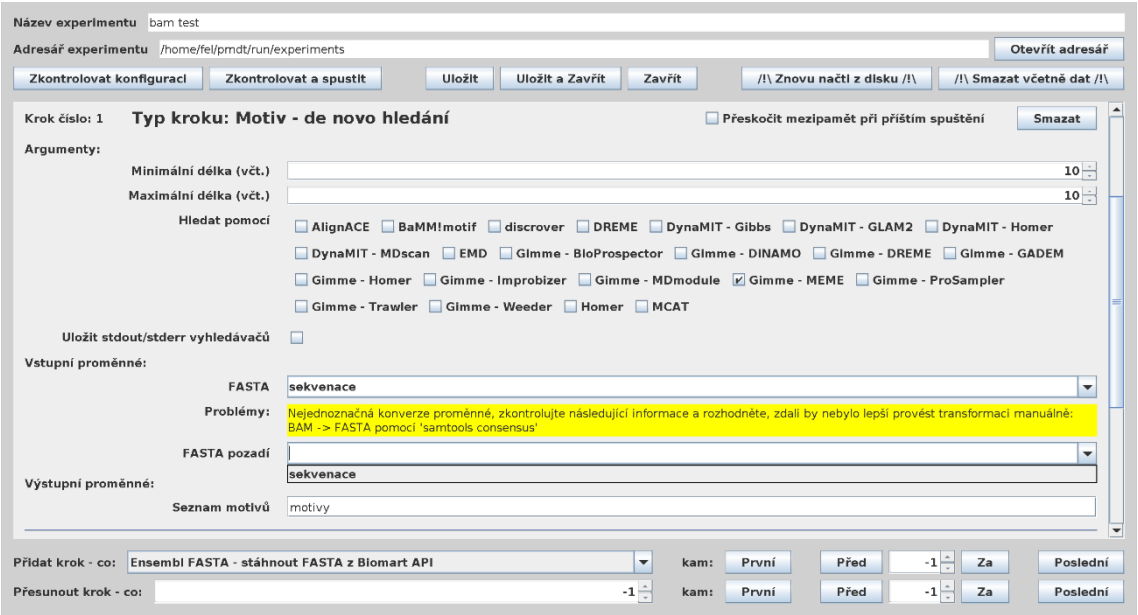
B Soubory použité nebo vzniklé při testování v kapitole 6.1 (`B_dataset.zip`) – obsahuje `README.txt` popisující obsah všech ostatních souborů a složek

C Obrázky – ukázky GUI aplikace

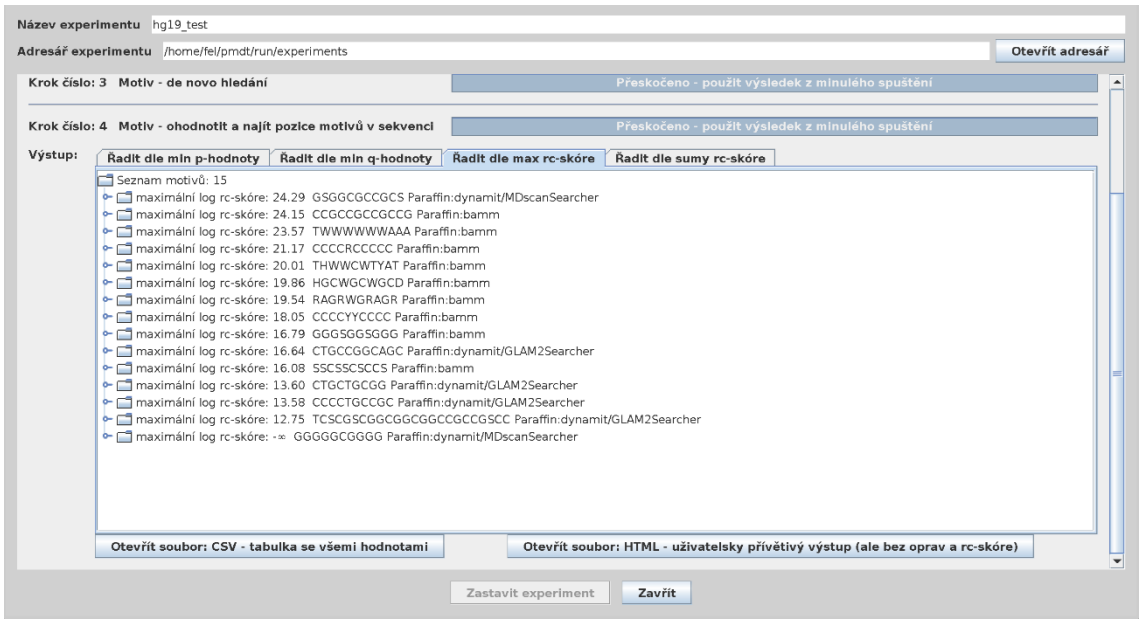
Příloha C – ukázky GUI aplikace



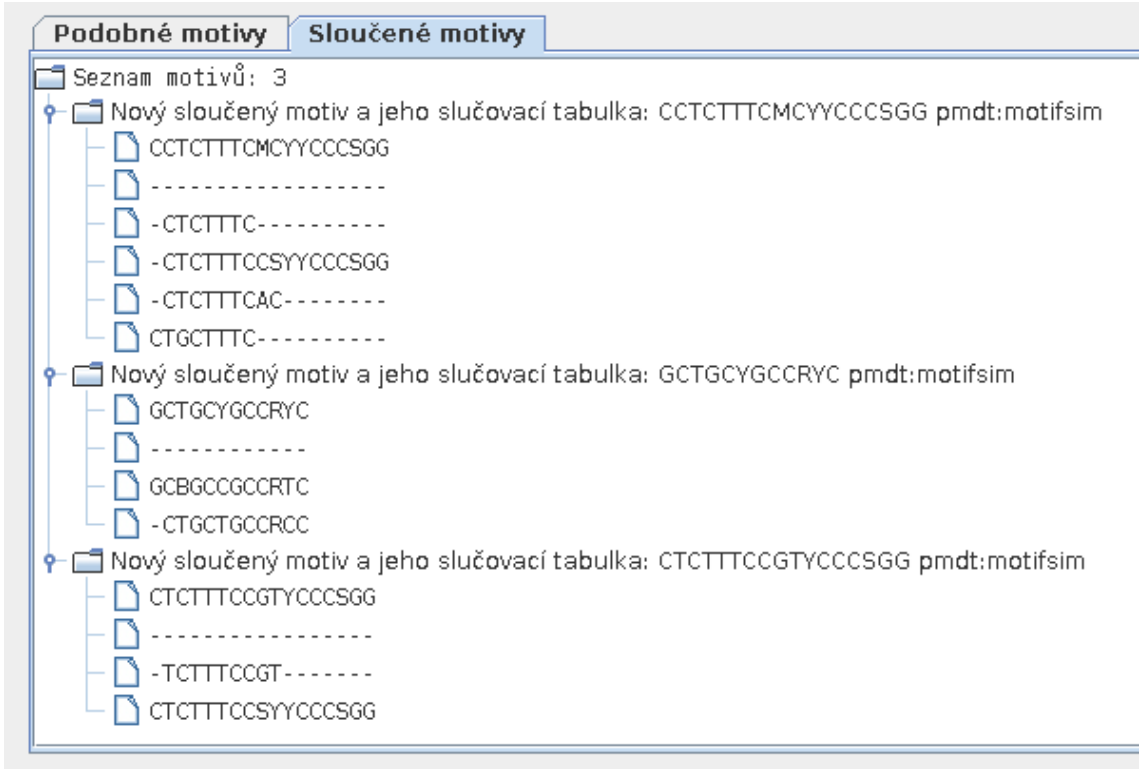
Obrázek 3 – základní menu aplikace



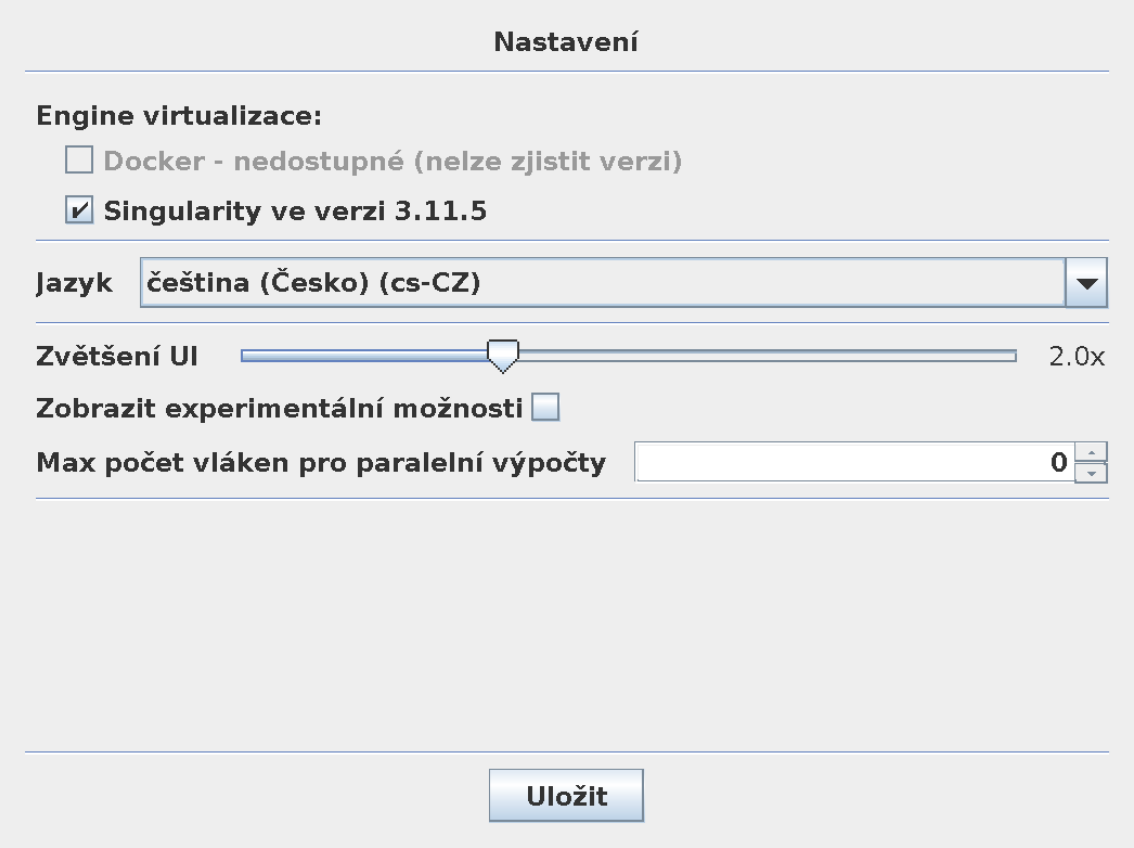
Obrázek 4 – konfigurace kroku pro de-novo hledání motivů s ukázkou transformační funkce proměnné, která vyžaduje pozornost uživatele



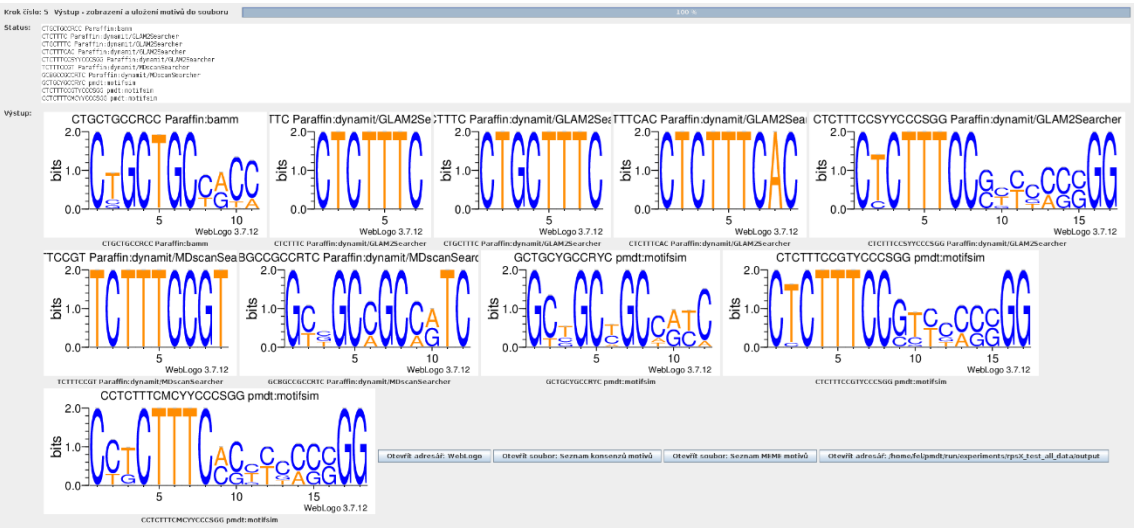
Obrázek 5 – výstup kroku pro ohodnocení nálezů motivů společně s ukázkou menu s dokončeným experimentem



Obrázek 6 – výstup kroku pro slučování motivů dle podobnosti



Obrázek 7 – nastavení aplikace



Obrázek 8 – ukázka generování výstupních souborů motivů, jsou zobrazeny motivy, které byly nalezeny při testování v kapitole 6.1