VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

REKONSTRUKCE REPETITIVNÍCH ELEMENTŮ DNA

RECONSTRUCTION OF REPETITIVE ELEMENTS IN DNA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAN HYPSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JANKA PUTEROVÁ**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/21309/2017/xhypsk01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Hypský Jan, Bc.**
Obor: Bioinformatika a biocomputing
Téma: **Rekonstrukce repetitivních elementů DNA**
Reconstruction of Repetitive Elements in DNA
Kategorie: Bioinformatika

Pokyny:

1. Nastudujte základy molekulární biologie a dále se zaměřte na repetitivní elementy v DNA.
2. Nastudujte literaturu z dané oblasti a vytvořte přehled dostupných nástrojů a metod pro rekonstrukci repetic.
3. Navrhněte vhodný algoritmus nebo modifikujte existující algoritmus pro rekonstrukci repetitivních elementů.
4. Po konzultaci s vedoucí implementujte navržený algoritmus a otestujte na vhodně zvoleném vzorku dat.
5. Diskutujte dosažené výsledky a další možné pokračování projektu.

Literatura:

- Zytnicki M, Akhunov E, Quesneville H. 2014. Tedna: a transposable element de novo assembler. Bioinformatics. 1-3. doi: 10.1093/bioinformatics/btu365.
- Chu C, Nielsen R, Wu Y. 2016. REPdenovo: Inferring De Novo repeat motifs from short sequence reads. PLoS One. 11:1-17. doi: 10.1371/journal.pone.0150719.
- Wicker T et al. 2007. A unified classification system for eukaryotic transposable elements. Nat. Rev. Genet. 8:973-982. doi: 10.1038/nrg2165-c4.
- Novák P, Neumann P, Pech J, Steinhaisl J, Macas J. (2013). RepeatExplorer: a Galaxy-based web server for genome-wide characterization of eukaryotic repetitive elements from next-generation sequence reads. Bioinformatics. 29:792-3. doi: 10.1093/bioinformatics/btt054.
- Dále dle doporučení vedoucího.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Puterová Janka, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Eukaryotické genomy obsahují velké množství repetitivních struktur. Jejich detekce a sestavení patří dnes k hlavním výzvám bioinformatiky. Tato práce obsahuje hlavní klasifikaci repetitivní DNA a představuje implementaci nového *de novo* assembleru, zaměřeného na hledání a sestavení LTR retrotranspozonů a satelitní DNA. Assembler přijímá na svém vstupu krátké ready (single nebo pair-end), získané sekvenátory druhé generace (NGS). Tento assembler je založen na přístupu Overlap Layout Consensus.

Abstract

Eukaryotic genomes contain a large number of repetitive structures. Their detection and assembly today are the main challenges of bioinformatics. This work includes a classification of repetitive DNA and represents an implementation of a novel *de novo* assembler focusing on searching and constructing LTR retrotransposons and satellite DNA. Assembler accepts on his input short reads (single or pair-end), obtained from next-generation sequencing machines (NGS). This assembler is based on Overlap Layout Consensus approach.

Klíčová slova

repetitivní DNA, sestavení repetic, *de novo* assembler, LTR retrotranspozony, satelitní DNA, krátké ready, Overlap Layout Consensus

Keywords

repetitive DNA, repeats assembly, *de novo* assembler, LTR retrotransposons, satellite DNA, short reads, Overlap Layout Consensus

Citace

HYPSKÝ, Jan. *Rekonstrukce repetitivních elementů DNA*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Janka Puterová

Rekonstrukce repetitivních elementů DNA

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením paní Ing. Janky Puterové. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Hypský
21. května 2018

Poděkování

Za výborné vedení, odborné připomínky a celkovou podporu při tvorbě této diplomové práce bych rád poděkoval vedoucímu práce Ing. Jance Puterové. Poděkování patří i organizaci MetaCentrum za poskytnuté výpočetní zdroje a nástroje, které výrazně pomohly při tvorbě této práce.

Obsah

Kapitola 1

Úvod

Většina eukaryotických genomů je tvořena vysokým procentem repetitivních struktur DNA. Prvotní studie hodnotily repetitivní DNA jako odpadní, zbytečnou a bez funkce a proto nebyla repeticím věnována velká pozornost. Následné studie však prokázaly, že některé druhy repetitivních struktur mají v genomu důležitou roli (např. regulace rychlosti translačních procesů). Začala se proto projevovat potřeba tyto sekvence v genomu blíže zkoumat a zjistit jejich funkci a roli v genomu. Repetice lze rozdělit do mnoha rodin podle jejich struktury nebo způsobu šíření.

Aby bylo možné zjistit, které repetice jsou obsaženy v genomu, je nutné nejprve tento genomu sekvenovat. První generace sekvenačních technik (např. Sangerova metoda) produkovala velmi dlouhé fragmenty DNA dobré kvality. Celková cena sekvenování byla velmi vysoká a proto limitovala vznik nových projektů, zabývajících se studiem genomu. Tato situace se výrazně změnila s příchodem technik sekvenování druhé generace - NGS. Sekvenační techniky druhé generace produkují velké množství dat ve formě kratších fragmentů, než v případě technik první generace, nicméně cena sekvenování (i samotného nástroje) byla několika násobně nižší. Tyto skutečnosti zajistily širší dostupnost sekvenačních nástrojů vědeckých ústavům a tím i vznik nových projektů, zaměřených na studium genomů organismů, případně jeho částí - repetitivní DNA.

Sekvence DNA, získané při sekvenování (tzv. ready), nenesou žádnou informaci o pozici v genomu, ze které byly získány. Pro sestavení genomu je proto nutné zajistit správné uspořádání readů. Dnes se nejčastěji používají dva hlavní přístupy pro sestavení genomu: De Bruijn graf (DBG) a Overlap layout consensus (OLC). Obě metody přijímají na svých vstupech datové sety readů, produkované sekvenačními nástroji, a reprezentují je formou grafu. Hlavní odlišnosti u těchto sestavovacích metod spočívají v typu vytvořeného grafu a výpočetní náročnosti jednotlivých fází algoritmů, které také určují výhody a nevýhody obou přístupů. Metoda OLC se skládá ze tři fází. Nejprve jsou vstupní ready zarovnány vůči sobě a hledají se překryvy mezi nimi. Na základě těchto informací je vytvořen graf překryvů, ve kterém představují ready vrcholy a překryvy tvoří hrany. V další fázi algoritmu hledána cesta grafem přes všechny vrcholy grafu (tzv. hamiltonovská cesta). V posledním kroku je nalezená cesta vyhodnocena, čím je získána sekvence genomu. Druhý přístup DBG nejprve transformuje vstupní ready na menší úseky tzv. k-mery, ze kterých je pak vytvořen de Bruijn graf. Sekvence genomu je získána průchodem grafu přes všechny jeho hrany (tzv. Eulerova cesta).

Kapitola 2 popisuje základní rozdělení repetitivních elementů a bližší charakteristiku jednotlivých rodin. Další část práce se zaměřuje na současné techniky a dostupné sekvenovací nástroje (viz kap. 3). Důraz bude kladen na vlastnosti jednotlivých technologií -

především velikost, přesnost a množství nasekvenovaných dat. V kapitole 4 budou prezentovány hlavní přístupy pro sestavování genomu spolu s jejich vzájemným porovnáním. Na základě vyhodnocení dostupných přístupů pro sestavování repetic bude v kapitole 5 popsán návrh nového sestavovacího nástroje, založeného na přístupu Overlap Layout Consensus, který bude implementován v rámci této práce. Nástroj se bude skládat z několika samostatně spustitelných částí. K podrobnému popisu funkcí a implementací jednotlivých částí aplikace je vyhrazena kapitola 6. Následně budou přiblíženy způsoby testování assembleru a prezentovány dosažené výsledky. Závěru práce bude patřit celkovému zhodnocení aplikace spolu s přiblížením možných vylepšení programu do budoucna.

Kapitola 2

Repetitivní DNA

DNA eukaryot obsahuje vysoký počet nekódujících sekvencí. Podobně jako kódující DNA i nekódující může být unikátní a může se nacházet v genomu ve více identických nebo velmi podobných kopiích. Tyto sekvence DNA s vysokým množstvím kopií se nazývají repetitivní sekvence. Repetitivní DNA tvoří genové rodiny obvykle na základě příbuzných sekvencí a funkcí a lze je rozdělit na tandemové nebo rozptýlené [19, 25].

2.1 Tandemové repetice

Tandemové repetice jsou tvořeny identickými a nebo téměř identickými jednotkami. Tyto jednotky jsou v genomu umístěny za sebou. Na základě velikostí těchto jednotek, rozdělujeme tandemové repetice na satelitní, mikrosatelitní a minisatelitní DNA [25].

2.1.1 Satelitní DNA

Eukaryotické genomy jsou z části tvořeny vysoce repetitivní DNA označovanou jako satelitní DNA. Repetitivní jednotky tandemově se opakujících sekvencí (monomery) jsou uspořádány obvykle v poli stovek až tisíců kopií. V genomech proto zabírají i několik megabází. Například u rostlin tvoří satelitní DNA až 20% genomu, u hlodavců a hmyzu až 50%. Obvykle se délka těchto jednotek pohybuje od několika desítek až tisíců nukleotidů. Satelitní DNA se neliší od minisatelitní a mikrosatelitní DNA pouze velikostí monomerů, ale i ve způsobu jejich šíření a umístění v konkrétním genomu. Někdy dokonce bývá délka monomerů podobná délce mini- nebo mikrosatelitů. Satelitní DNA je vysoce rozmanitá, nicméně podobnost v rámci populace je vysoká (druhově specifická satelitní DNA) [13, 19].

2.1.2 Minisatelitní DNA

Tandemové repetice o velikosti větší než 9 nukleotidů se označují jako minisatelity. Minisatelitní DNA se nejčastěji nachází v subtelomerických oblastech chromozomů. Velikost minisatelitní DNA se mění podobně jako u klasických molekulárních mechanismů a jejich klasifikace nezáleží jen na velikosti repetitivní jednotky, ale také na možné funkci a distribuci v eukaryotických genomech [13].

2.1.3 Mikrosatelitní DNA

Mikrosatelitní DNA (mikrosatelity) se nachází ve všech doposud prostudovaných genomech organismů. Tento druh DNA je také znám jako jednoduché opakující se sekvence nebo

krátká tandemová opakování. Mikrosatelity jsou tvořeny tandemovými repeticemi o maximální délce devíti nukleotidů. Celkový počet repetitivních jednotek se pohybuje v rozmezí 5-40, kdy se mohou vyskytovat i série repetic. Tyto změny mají za následek různé délky alel. Mikrosatelitní DNA se nejčastěji nalézají v proteinově kódujících i nekódujících regionech a regulačních sekvencích ve formě nejčastěji trinukleotidu nebo dinukleotidu. Mikrosatelity jsou charakterické mutačním chovaním, které bývá mnohonásobně větší, než průměrná frekvence mutace v jiných částech genomu. Tyto mutace způsobují spíše změnu počtu repetitivních jednotek než bodovou mutaci. Z těchto důvodů je mikrosatelitní DNA velmi nestabilní v závislosti na počtu a délce repetitivních jednotek nebo čistoty repetic. Nejdůležitějším faktorem je ovšem počet jednotek opakovaní, kdy větší počet repetitivních jednotek způsobuje vyšší nestabilitu regionu. Podle těchto charakteristik mikrosatelitní DNA lze analyzovat a odhadovat úroveň genetické variability v populacích, genetickou rozmanitost a příbuzenské křížení [13, 19].

2.2 Transpozobilní elementy

Transpozobilní elemety (TE), nazývané též transpozony, jsou sekvence DNA, které mohou měnit svoji polohu v rámci stejného genomu. Téměř všechny živé organismy obsahují TE v hojném počtu. Transpozonové prvky se dělí do dvou hlavních tříd podle provedení transpozičního mechanismu. První třídou jsou tzv. retrotranspozony a v druhé třídě se nacházejí DNA transpozony. Obě třídy transpozonů obsahují jak autonomní, tak i neautonomní prvky. Rozdíl mezi autonomními a neautonomními elementy je ve způsobu transpozice. Autonomní transpozony obsahují tzv. otevřený čtecí rámec (ORF - z angl. Open Read Frame). ORF kóduje produkt nezbytný pro transpozici. Neautonomní transpozony ORF nemají nebo je poškozený a neschopný kódovat sekvence potřebné k transpozici. Obsahují však sekvence potřebné pro transpozici. Při přítomnosti aktivátoru transpozice (např. Ac u kukuřice) a sekvencí potřebných pro transpozici, dochází k transpozici neautonomních transpozonů [13, 19].

2.2.1 Retrotranspozony

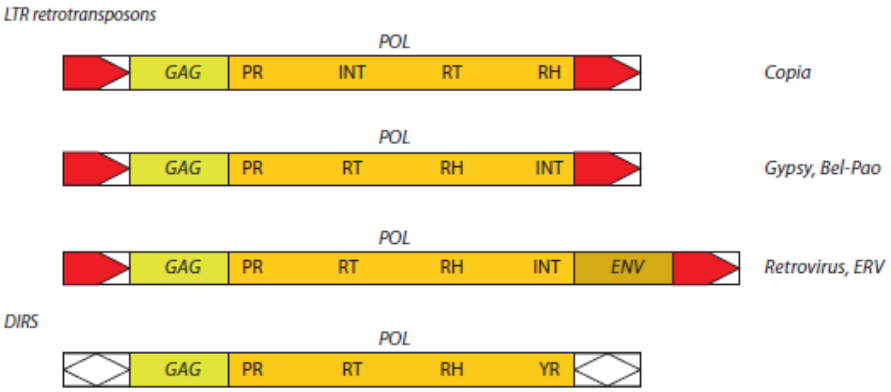
Retrotranspozony se vyznačují transponováním prostřednictvím RNA meziproduktů. RNA prvku je transkribována a následně se pomocí reverzní transkripce vytváří komplementární DNA, která se následně umístí na nové místo v genomu. Tento mechanismus transpozice se nazývá "copy & paste". Podle klasifikačního systému představeného Thomasem Wickerem et.al. [22] lze retrotranspozony rozdělit do pěti řádů: LTR retrotranspozony, DIRs, LINEs, PLE a SINEs [13].

LTR retrotranspozony

První řád zastupují LTR (z angl. Long Terminal Repeat) retrotranspozony. Jedná se o transpozony převládající především v genomech rostlin. Velikost LTR retrotranspozonů dosahuje velikosti několika set párů bází až 5 kbp (výjimečně 25 kbp). LTR retrotranspozony obsahují na obou koncích řetězce dlouhé terminální repetice. Tyto retrotranspozony můžeme dále rozdělit do tří hlavních linií: 1) Ty1/copia, 2) Bel-Pao a 3) Ty3/gypsy. Hlavním rozdílem těchto linií je enzymatické uspořádání domén (ilustrováno na obrázku 2.1). Tento typ retrotranspozonů je velmi podobný retrovirům s výjimkou nepřítomnosti env genu [13].

DIRs

DIRs (z angl. Dictyostelium intermediate repeat sequence) prvky tvoří druhý řád. Tyto prvky se vyvinuly z gypsy podobným LTR retrotranspozonů (obrázek 2.1). Oproti LTR retrotranspozonům obsahují tyrosin rekombinázu (YR) místo integrázy a repetitivní sekvence na koncích řetězce jsou invertované. Některé DIRs prvky mohou zahrnovat introny v rámci ORF [13, 19].



Obrázek 2.1: Retrotranspozony 1. a 2. řádu. LTR retrotranspozony mají na svých koncích dlouhé terminální repetice (červená barva) a obsahují geny GAG (zelená barva) a POL (oranžová barva). Gen POL obsahuje enzymatické složky PR - proteináza, INT - integráza, RT - reverzní transkriptáza a RH - RNáza H. Pořadí jednotlivých složek určuje skupinu LTR retrotranspozonů. Retroviry se liší od skupiny gypsy LTR retrotranspozonů obsahem ENV genu (světle hnědá barva). DIRs retroelementy neobsahují v genu POL integrázu. Ta je nahrazena tyrosin rekombinázou (YR). Také koncové repetice se liší. Ve srovnání s LTR retrotranspozonů jsou invertované [13].

LINEs

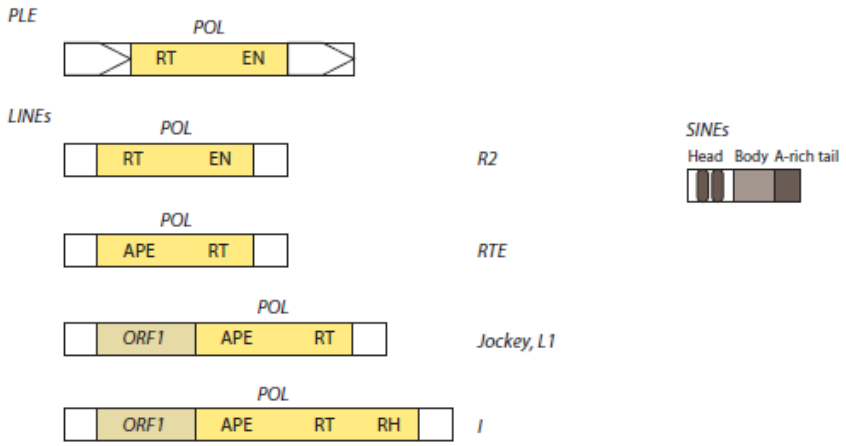
Retrotranspozony typu LINEs (z ang. Long Interspersed Nuclear Elements) jsou charakteristické absencí LTR elementů (obrázek 2.2). Na svém konci mohou obsahovat tandemovou repetici nebo poly(A) konec. Tato třída se může následně rozdělit na 5 rodin (R2, RTE, Jockey, L1 a I). Rozdíly mezi těmito rodinami jsou na obrázku 2.2. Velikost LINEs dosahují několika kilobází a v lidském genomu dosahují až 516 000 kopií (přibližně 17% celkového genomu) [13, 19].

PLE

Dalším typem retrotranspozonů jsou retrotranspozony PLE (Penelope-like). PLE jsou velmi podobné telomerázám. Ve většině případů Penelope-like prvky kódují otevřený čtecí rámec ORF (z angl. Open Read Frame) složený z domén reverzní transkriptázy [13].

SINEs

Posledním řádem jsou retrotranspozony SINEs (z angl. Short interspersed nuclear elements). Jedná se o neautonomních prvky, které jsou funkčně spojeny s LINEs. Prvky SINEs



Obrázek 2.2: Retrotranspozony LINEs a PLE. Rodiny R2 a RTE obsahují pouze jeden čtecí rámec (ORF). ORF u R2 obsahuje reverzní transkriptázu RT a následně endonuklázu EN. U RTE se ORF skládá z apurino-apyrimidové endonukleázy (APE) a RT. Zbylé rodiny LINEs retrotranspozonů obsahují dva ORF, označované jako ORF1 a ORF2. Funkce ORF1 je velmi podobná genu GAG. ORF2 obsahuje stejné enzymy jako RTE. U rodiny I je navíc v ORF2 obsažen enzym RNAnázy H (RH). Struktura retrotransponů SINEs (nalevo) se dělí na tři části: Hlava (Head) obsahuje Pol III gen promotoru, tělo (Body) obsahuje enzymy příslušné rodiny a konec (A-rich tail) je tvořen posloupností adeninu [13].

jsou vytvářeny pomocí retrotranspozice RNA polymerazy III. Tato RNA polymeráza je přepisována z reverzní transkriptázy kódované LINEs. SINEs nekódují proteiny a postrádají oblasti pro autonomní transkripci. Struktura SINEs se dělí do tří modulů - terminální hlava, vnitřní oblast a terminální konec. Hlava v sobě nese Pol III gen promotoru. Vnitřní oblast je velmi variabilní a specifické v rámci rodiny. Na konci SINEs se nachází region bohatý na adenin. Přítomnost těchto retrotransponů bývá často používaným kritériem pro hodnocení mezidruhových vztahů. Průměrná délka elementů SINEs bývá okolo 80-500 bp [13, 19].

2.2.2 DNA transpozony

DNA transpozony využívají pro své šíření konzervativní mechanismus "cut & paste". Při tomto procesu nevznikají RNA mezi produkty. Třída II je téměř výlučně zastoupena v eukaryotických organismech. DNA transpozony se dělí na dvě hlavní podtřídy: 1) TIRs a 2) Polintony a Helitrony [13, 19].

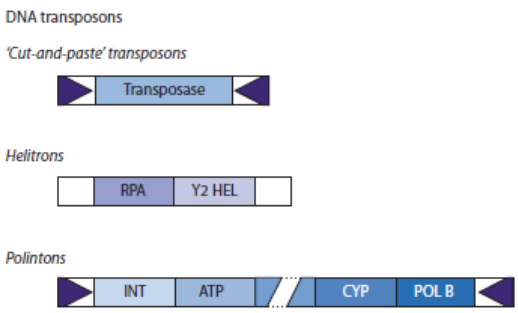
TIRs

Charakteristickým rysem této třídy je přítomnost koncových invertovaných opakování (TIR - z angl. Terminal inverted repeats) a obsahem pouze jediného ORF, kódujícího transponázu. Tato transponáza rozpoznává právě TIRs a začlení DNA transpozon na nové místo v genomu [13].

Polintony a Helitrony

Polintony a Helitrony se liší od TIRs způsobem šíření. Pro své šíření využívají replikační proces "copy & paste". Velmi dlouhé DNA transpozony o délce 15-20 kb se nazývají polintony (z angl. Polintons), známé též jako Mavericks. Polintony mají na obou koncích DNA řetězce TIRs o velikosti 100-1000 bp. Tyto transpozony kódují více než 10 proteinů (obrázek 2.3) [13].

Helitrony se nacházejí především v genomech rostlin. Na koncích řetězce mají helitrony vlásenkové struktury místo TIRs. Transpozice helitronů probíhá replikačním mechanismem otáčející se kružnice [1]. Helitrony kódují tyrosin rekombinázu s helikázovou doménu a replikační protein A (ilustrováno obrázkem 2.3) [13].



Obrázek 2.3: Stručný přehled struktur vybraných DNA transpozonů. Obecná struktura TIRs DNA traspozonů. Na koncích řetězce se nacházejí TIRs (tmavě modrá barva), které obklopují trasponázu. Rodina transpozonů Helitron. Helitrony nemají TIRs. Místo nich mají vlásenkové struktury (bílá barva). Mezi vlásenkovými strukturami se nachází replikační protein A (RPA) a tyrosin rekombináza s helikázovou doménou typu 2 (Y2 HEL). Poslední struktura repezentuje polintony. Politrony kódují více proteinů, proto jsou z hlediska struktury velmi variabilní. Mezi typické enzymy, které jsou v politronech obsaženy, patří integráza (INT), ATPáza (ATP), cysteinová proteáza (CYP) a B-typ DNA polymeráza (POL B). Na začátku a na konci polintonů se nacházejí TIRs [13].

[1] Princip mechanismu demonstrován zde: http://what-when-how.com/molecular-biology/rolling-circle-dna-replication-molecular-biology/

Kapitola 3

Technologie sekvenování DNA

První metodu sekvenování DNA řetězce představil v roce 1970 Frederick Sanger. Tato metoda dokázala dešifrovat kompletní geny, vyžadovala jen minimální manipulaci s toxickými látkami a radiozotopy. Z těchto důvodů se Sangerovo sekvenování stalo hlavním sekvenačním nástrojem 20. století. Sangerova technika sekvenování také, jako první sekvenační technika, dokázala sekvenovat lidský genom v rámci Human Genome Project. Nicméně nasekvenování lidského genomu ukázalo hlavní nevýhody této metody. Především se jednalo o rychlost, časovou náročnost a cenu sekvenování, které byly nevyhovující. Proto národní instituce pro zkoumání lidského genomu (NHGRI) zahájila program financování vývoje nových technik sekvenovaní. Důsledkem tohoto programu byl velký rozvoj sekvenačních technik nové generace - NGS (z ang. Next generation sequencing) [4].

3.1 Druhá generace sekvenování (NGS)

Oproti Sangerovu sekvenování, metody NGS sdílejí tři hlavní rysy. NGS nevyžadují bakteriální klonování fragmenů DNA, sekvenační reakce se provádí paralelně a nevyžadují elektroforézu (výstupem jsou přímo sekvence). NGS technologie produkují velké množství tzv. readů a to velmi vysokou rychlostí. Velikost readů je však velmi malá, což mělo za následek vznik zarovnávajících algoritmů (tyto algoritmy jsou popsány v kapitole 4). Mezi hlavní NGS technologie patří 454, SOLiD, Illumina nebo PGM (porovnání technologií je na obrázku 3.1 a 3.2) [4].

3.1.1 454 - Roche (Roche Applied Science)

V roce 2005 vznikla první technologie NGS založená na pyrosekvencování Roche 454. Princip pyrosekvencování spočívá ve sledování pyrofosforečnanu uvolněného během začleňování nukleotidu. Přesnost sekvenování se pohybuje kolem 99,9%. Nesporně významnou výhodou je rychlost sekvenace, která trvá několik hodin. Sekvenátor produkuje ready velikosti až 700 bp. Velkou nevýhodou této technologie vysoká cena činidel, potřebných pro sekvenování a relativně vysoká míra chyb u polybází delších než 6 bp [12].

3.1.2 SOLiD

SOLiD (Sequencing by Oligo Ligation Detection) využívá pro sekvenaci ligázu. Sekvenátor produkuje velké množství readů krátké délky, pohybující se v rozmezí od 35 - 85 bp. Právě

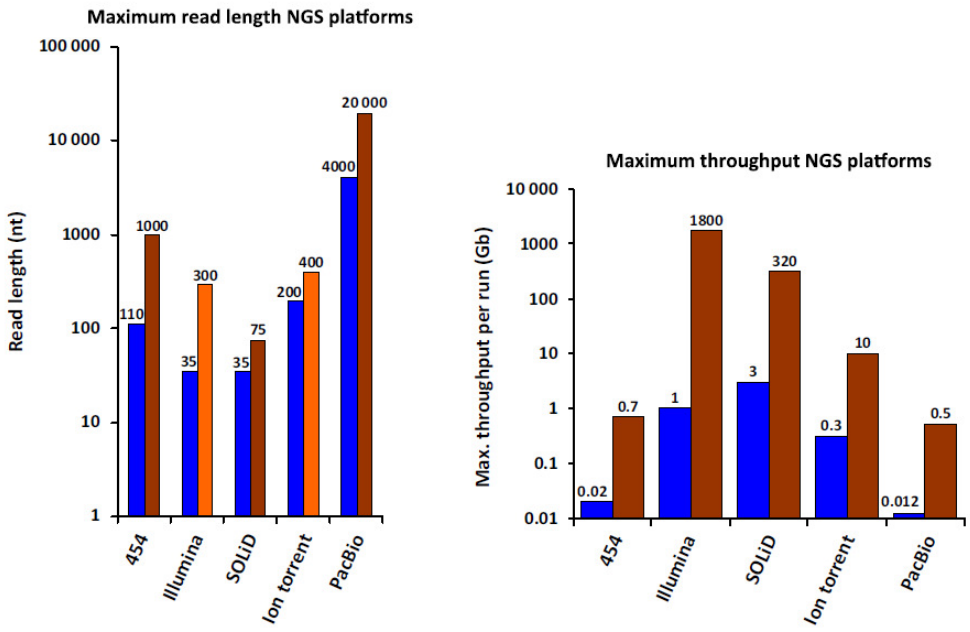
krátká délka readů je hlavní nevýhodou SOLiD. Naopak velkou výhodou je vysoká míra přesnosti, která dosahuje až 99,99% [12].

3.1.3 Illumina (Solexa)

V roce 2006 představila společnost Solexa Genome Analyzer (GA). Princip sekvenování je založen na sekvenování syntézou. Společnost Illumina v roce 2007 koupila Solexa a pokračovala ve vývoji GA. Velikost produkovaných readů dosahuje 75 - 150 bp s chybovostí menší než 2%. Hlavní předností této technologie je nízká cena sekvenování (0.07 $ za Mb) [12].

3.1.4 PGM (Ion Torrent)

Společnost Ion Torrent představila v roce 2010 novou techniku PGM (Personal Genome Machine). PGM využívá polovodičové technologie místo optické detekce nukleotidů. To má za následek snížení nákladů na výrobu a zmenšení velikosti sekvenačních přístrojů. Také rychlost sekvenace je mnohem vyšší [12].



Obrázek 3.1: Porovnání sekvenačních metod. Napravo je zobrazeno srovnání metod na základě délky readů. Je patrné, že PacBio generuje nejdelší ready. Obrázek nalevo reprezentuje maximální propustnost jednotlivých platforem. Zde dosahuje nejlepších výsledků Illumina [4].

3.2 Třetí generace sekvenování

Třetí generace sekvenovacích nástrojů přišla s novými požadavky a pohledy na sekvenování DNA a má dva charakteristické rysy. Zpracování informací probíhá u třetí generace v re-

Sequencer	454 GS FLX	HiSeq 2000	SOLiDv4	Sanger 3730xl
Sequencing mechanism	Pyrosequencing	Sequencing by synthesis	Ligation and two-base coding	Dideoxy chain termination
Read length	700 bp	50SE, 50PE, 101PE	50 + 35 bp or 50 + 50 bp	400~900 bp
Accuracy	99.9%*	98%, (100PE)	99.94% *raw data	99.999%
Reads	1 M	3 G	1200~1400 M	—
Output data/run	0.7 Gb	600 Gb	120 Gb	1.9~84 Kb
Time/run	24 Hours	3~10 Days	7 Days for SE 14 Days for PE	20 Mins~3 Hours
Advantage	Read length, fast	High throughput	Accuracy	High quality, long read length
Disadvantage	Error rate with polybase more than 6, high cost, low throughput	Short read assembly	Short read assembly	High cost low throughput

Obrázek 3.2: Podrobné porovnání výhod a nevýhod nástrojů, vytvořených na základě různých NGS technik. 1) 454 GS FLX - platforma 454 (Roche), 2) HiSeq 2000 - platforma Illumina, 3) SOLiDv4 - platforma SOLiD a 4) Sanger 3730xl - založeno na Sangerově sekvenování [12].

álném čase a oproti druhé generaci nevyžadují pro sekvenování PCR (z angl. Polymerase chain reaction) a sekvenování je proto rychlejší [12].

3.2.1 PacBio (Pacific Biosciences)

Jedním z hlavních nástrojů v této oblasti technologie je PacBio. PacBio generuje až 20 kbp dlouhé ready, proto se velmi hodí k de novo sestavování genomu. Princip metody spočívá v detekci fluorescence DNA, která je složena z fosfátově označených nukleotidů, syntetizovaných pomocí jedné DNA polymerázy. Detekce probíhá v reálném čase, což velmi zrychluje sekvenování [4].

3.2.2 MinION (Oxford Nanopore Technologies)

Společnost Oxford Nanopore Technoligies představila novou technologii sekvenování, která pracuje na principu identifikace jednotlivých molekul DNA procházejících přes nanopór. Sekvenátor MinION tuto technologii využívá a je primárně určen pro vykonávání velkého množství experimentů. Jedná se o miniaturní a přenositelné zařízení, které využívá pro komunikaci s počítačem rozhraní USB 3.0. MinION umožňuje produkovat ready dlouhé až několik stovek kilobází [20].

3.3 Shrnutí sekvenačních metod

V současnosti jsou nejvíce používané techniky pro sekvenaci DNA Illumina a SOLiD, především díky své vysoké propustnosti. Nicméně délka readů, které tyto techniky produkují, je velmi malá a nevhodná pro de novo sestavování genomů. Vlivem modernizace přístrojů však došlo k velkému zlepšení technologií NGS (především Illumina). Velikost readů se zvýšila, což umožnilo použít technologii Illumina pro de novo sestavení genomu. Také náklady na sekvenovaní výrazně poklesly. Společnost Illumina uvolnila před několika lety sadu sekvenačních nástrojů HiSeq X Ten. Tyto nástroje mohou generovat až 1,8 TB sekvencí s velmi nízkými náklady (cena se pohybuje pod 1000$ za genom). Nicméně cena zřízení HiSeq je

velmi vysoká a proto je využívána především velkými výzkumnými ústavy [4]. NGS technologie přinesly i nové typy výstupních dat. Jako velmi efektivní a užitečné se pro sestavování genomu ukázalo použití tzv. párových readů (též pair-end). Párové ready (angl. pared-end) se získávají sekvenování obou konců DNA fragmentu. Read jsou následně uloženy do knihovny readů a získavají příponu podle místa, z kterého byly sekvenovány (f,1 - pro počátek fragmentu DNA nebo r,2 - konec fragmentu DNA). Typicky se velikost těchto readů pohybuje kolem 200-300 bp.

Kapitola 4

Přístupy pro sestavení genomu

Sekvenační techniky nové generace zpřístupnily sekvenování širší veřejnosti. Rychlost a cena sekvenování již nebyla překážkou a proto mnoho výzkumných ústavů začalo vytvářet projekty, zaměřené na výzkum genomů. Aby bylo možné genom sestavit, bylo nutné vytvořit účinné sestavovače genomu. Postupně se tyto sestavovače (angl. assemblery) začaly specializovat, aby vyhovovaly novým potřebám výzkumu a vznikaly tak assemblery zaměřené na sestavovaní určitých typů genomů a metagenomů (např. genomy prokaryot/eukaryot) nebo sestavení podoblastí genomu (např. unikátní nebo repetitivní sekvence). Kromě specializace se od sebe sestavovače lišily i v případě techniky samotného sestavení. První assemblery zpracovávaly na vstupu dlouhé ready, získané pomocí Sangerovi metody a používaly přístup, kde byly nejdříve hledány vzájemné překryvy readů, pak byl na základě těchto překryvů vytvořen graf (v grafu představovaly ready uzly a překryvy hrany) a v posledním kroku byla výsledná sekvence genomu získaná průchodem grafu přes všechny uzly - hamiltonovská cesta. Tento velmi intuitivní přístup se nazývá Overlap Layout Consensus (OLC) a tvoří stěžejní část této práce. Metoda OLC není vždy zcela optimální řešení, protože hledání cesty přes všechny vrcholy je výpočetně náročné (jedná se o NP těžký problém). Proto se objevily i další přístupy, které začaly assemblery využívat. Především se jednalo o přístup založený na tvorbě de Bruijnova grafu (popsán níže).

4.1 De Bruijn graph (DBG)

V roce 1995 představil Ramana. M. Indury a Michael. S. Waterman algoritmus DBG. Z počátku byl tento algoritmus málo známý a rozšířený, především kvůli své neintuitivnosti a nízké použitelnosti. Zvrat však nastal s příchodem sekvenačních technologií, poskytujících krátké ready, jako Illumina nebo SOLiD. DBG algoritmus dosahoval na krátkých genomech velmi dobrých výsledků a proto se stal základem mnoha sestavovacích assemblerů jako například Velvet [23] nebo Tedna [24]. Princip činnosti algoritmu DBG je velmi podobný přístupu OLC a lze ho rozčlenit do několika fází: předzpracování dat, převod k-merů na graf (DBG), tvorba contigu, spojení contigů a odstranění mezer [11].

4.1.1 Předzpracování dat

Algoritmus DBG nevyužívá pro svou činnost přímo vstupní ready, ale provádí jejich transformaci na k-mery. K-mery jsou krátké úseky bází, extrahované z readů. Velikost k-merů bývá typicky v sestavovačích nastavitelná, maximálně však dosahuje několika set bází. Než je zahájena transformace readů, je potřeba provést jejich korekci. U vstupních readů je třeba

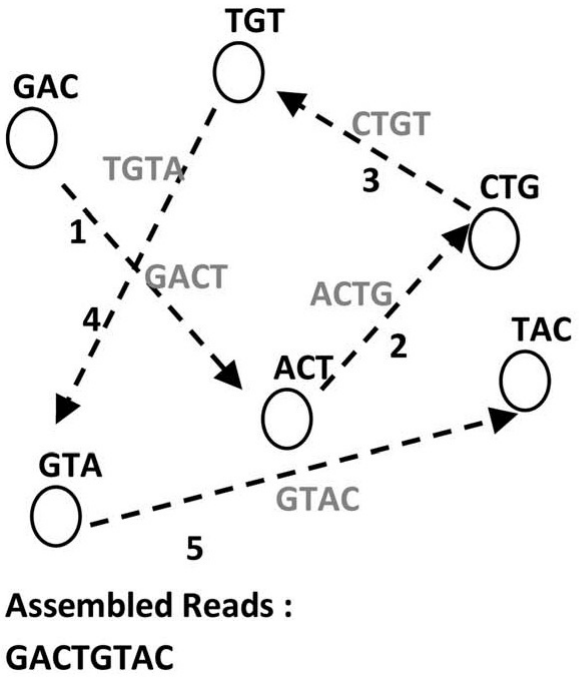
počítat s chybami (např. chybně sekvenováné báze). Tyto chyby se pak následně projevují při extrakci k-merů, kdy je získáváno velké množství chybných k-merů, které nejsou obsaženy v genomu. Proto je důležité vstupní ready filtrovat a případné chyby opravovat [11].

4.1.2 Sestavení DBG

Následně je třeba k-mery transformovat do formy de Bruijn grafu. Pro tvorbu tohoto grafu je nejprve nutné získat tzv. k-1 mery. Tyto k-1 mery jsou vytvářeny tak, že je původní k-mer zkrácen na svých koncích o jednu bázi. Vznikají tedy vždy dva k-1 mery, které jsou zaneseno do grafu jako vrcholy a k-mery pak tvoří v grafu hrany mezi vrcholy.

4.1.3 Tvorba contigu a jejich spojení

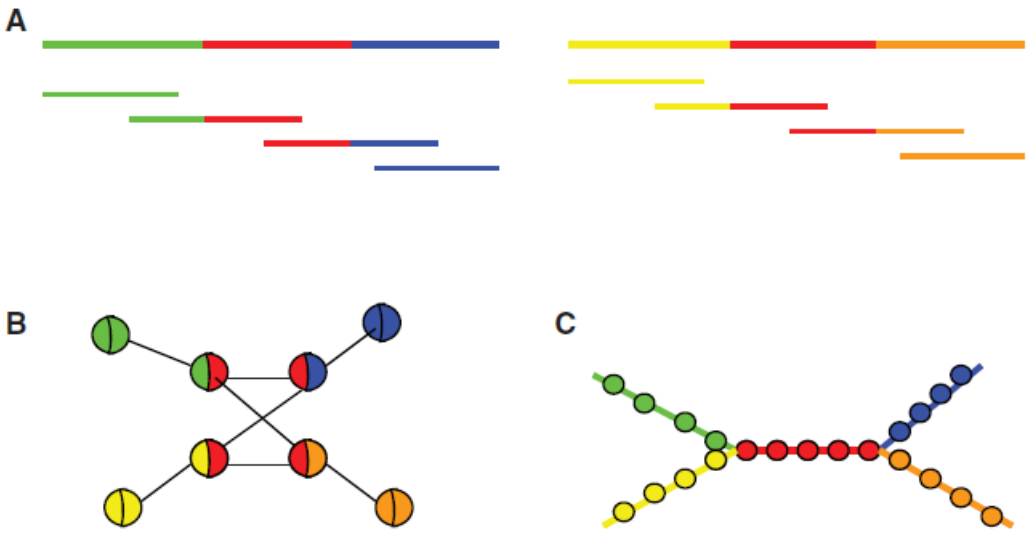
Dalším krokem algoritmu DBG je sdružení k-merů v grafu, které spolu sdílejí společný vrchol, do větších souvislých posloupností - tzv. contigů. Contigy se získávají hledáním cesty v grafu přes všechny hrany. V grafu je tedy hledána Eulerova cesta (obrázek 4.1). Časová složitost hledání řešení je lineární O(N). Rozlišujeme dva typy contigů podle toho, zda obsahují pouze unikátní nebo repetitivní sekvenci genomu. Na contig obsahující repetici se váže velké množství k-merů z contigů, obsahujících unikátních oblasti genomu [11].



Obrázek 4.1: Nalezení Eulerovy cesty v grafu. Cesta je získána průchodem grafu přes všechny hrany reprezentované k-mery. Číselný index u každé hrany určuje pořadí vyhodnocení hrany v rámci cesty [5].

Získané contigy je třeba seřadit a vzájemně spojit do souvislé sekvence, tak aby co nejvíce odpovídala sekvenci sekvenovaného genomu. Pro propojení contigů se využívají vlastnosti párových readů. Princip je velmi jednoduchý: je-li k-mer contingu v párové vazbě s k-merem jiného contigu, vznikne mezi těmito contigy vazba. Problémy při těchto vazbách nastávají především v oblasti repetic (obrázek 4.2). Ke contigu vstupujícího do repetice není

typicky jednoduché přiřadit odpovídající výstupní contig. Objevují se zde ale i další dva problémy, kdy není možné propojit contigy. První problém je možný vznik chybných cest v grafu (souvisí s fází předzpracování a řešení spočívá ve vyřazení chybných readů nebo k-merů z konstrukce grafů). Druhý problém jsou velmi krátké contigy. Typicky je řešení těchto problému složité a jedná se NP-problém [11].



Obrázek 4.2: Ukázka rozdílu při sestavování grafu překryvů metodami OLC a DBG. A) Nasekvenovaná data dvou oblastí genomu, které obsahují stejnou repetici (vyznačena červeně). B) Sestavení grafu překryvů pomocí metody OLC, kde jsou v případě repetic patrné zpětné vazby mezi začátkem a koncem repetice. C) Výsledné sestavení repetice metodou DBG [10].

4.1.4 Assembler pro sestavení repetic - Tedna

Nástroj Tedna je *de novo* sestavovač transpozonů, využívající DBG pro analýzu vysoce frekventovaných k-merů. Asembler nejprve vypočítává velikost pokrytí genomu jako maximum frekvenčního rozdělení k-meru. Za transpozony jsou považovány k-mery, jejichž výskyt je minimálně dvojnásobný než pokrytí genomu. Tedna následně rozdělí graf na jednotlivé komponenty a ponechává ty, které obsahují transpozony. Každá komponenta je samostatně vyhodnocena, kdy je hledána nejdelší cesta, která představuje sekvenci transpozonu. Asembler umí využít párové informace z readů pro projení jednotlivých komponent a získaní nejdelší možné sekvence repetice [24].

4.2 Graf řetězců (string graf)

Přístup založený na řetězcových grafech je kombinací OLC 4.3 a DBG, kdy hlavní myšlenkou je pracovat přímo se vstupními ready (stejně jako u OLC) bez převodu na k-mery a provádět rychlejší sestavení sekvence hledáním Eulerovy cesty v grafu (obdobně jako přístup DBG) [15]. V první fázi je prováděno klasické hledání překryvů readů „all-to-all". Jednotlivé překryvy pak vytvářejí intervaly pokrytého genomu. Pokud je read obsažen v in-

tervalu (je nalezeno překrytí, které celý read mapuje do již existujícího intervalu) je tento read odstraněn. V případě, že je nalezen překryv readu s intervalem, ale read není celou délkou obsažen v intervalu, je provedena konkatenace readu s intervalem. Tímto způsobem je odstraněno velké množství readů (typicky až 60%). Výsledné překryvy jsou zaneseny do orientovaného řetezcového grafu, kde jsou podobně jako přístupu OLC odstraněny všechny tranzitivní uzávěry. Vrcholy jsou tvořeny ready (nebo intervaly) a hrany reprezentují vazby mezi nimi. U hran také reprezentován počet těchto vazeb, kdy malé číslo typicky značí unikátní sekvence genomu a velké počet vazeb zase indukuje možné repetitivní úseky DNA. Následně je hledána Eulerova cesta grafem, čímž je získána výsledná sekvence genomu [15].

4.3 Metoda OLC

První OLC algoritmus byl publikován v roce 1980. Jedná se o velmi intuitivní způsob sestavování genomu, který lze použít, jak na velmi dlouhé ready poskytované Sangerovým sekvenováním a především na krátké ready, získané pomocí sekvenačních nástrojů druhé generace. Jak je již z názvu patrné, algoritmus provádí tři základní úkony: overlap (hledání překryvů), layout (sestavení grafu uspořádání) a consensus (vytvoření konsenzus sekvence). Přístup OLC byl od svého publikování několikrát upraven, kdy se jednalo především o použití nových algoritmů pro hledání překryvů, které značně usnadnily a zrychlily sestavení grafu překrytí [11].

4.3.1 Hledání překryvů (overlap)

Prvním krokem algoritmu OLC je hledání překryvů jednotlivých readů. Pro popis činnosti, výslednou kvalitu překryvů i celkovou dobu trvání fáze overlap je třeba definovat několik pojmů. Jedná se především o tyto pojmy (faktory) [11]:

- velikost genomu (G) - Velikost sestavovaného genomu nepřímo ovlivňuje především rychlost hledání překryvů. Jedná se o fixní parametr, kdy čím je větší délka sekvenovaného genomu, tím roste i počet readů, potřebným k popsání tohoto genomu. Vstupní datový soubor tedy obsahuje tedy větší množství řetězců bází (readů) získaným sekvenačním nástrojem, které je třeba zpracovat, což vede k vyšší spotřebě výpočetních zdrojů i času.
- délka readů (L) - Jedná se také o fixní parametr, který je určen zvoleným sekvenačním nástrojem.
- prahová hodnota (T) - Hodnota, sloužící k rozhodnutí o vytvoření překryvu dvou readů. Pokud je délka shody dvou readů větší než hodnota T, je vytvořeno propojení mezi těmito ready.

Princip hledání překryvu spočívá v zarovnání dvou readů a hledání jejich nejdelší společné části. Typicky je tento proces prováděn mezi všemi ready (tzv."all-to-all") a proto dosahuje časové složitosti O(N^2). Pro získání výsledného překryvu bylo v průběhu let použito několika technik. Obecně lze tyto techniky rozdělit do dvou základních skupin: hledající úplnou schodu (např. přístupy založené na sufixových stromech) a techniky, hledají nejlepší možnou schodu. První skupina je typicky využívána pro hledání překryvů na známých datech, které neobsahují šum a chyby. Pro svou činnost často používají pomocné datové struktury, které snižují celkovou časovou složitost. Reálná data však nikdy nejsou ideální. Ready obsahují určité procento chyb, které je závislé na použitém sekvenačním

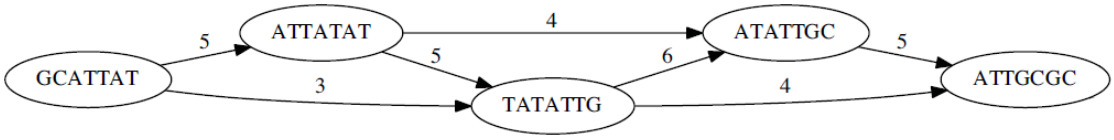
nástroji. Hledání úplné shody by bylo v tomto případě velmi neefektivní. V praxi se proto převážně používají techniky, založené na hledání nejdelšího překryvu na základě výpočtu a hledání nejlepšího skóre (tzv. dynamické programování). Dynamické programování se dokáže vypořádat s chybnými bázemi a umožňuje vložení mezer pro získání lepšího překryvu. Nejznámějším zástupcem této skupiny nástrojů je program BLAST, založený na dynamickém programování a v praxi široce používaný. BLAST bude použit v této práci pro hledání překryvů.

4.3.2 Sestavení grafu (layout)

Druhá fáze - layout tvoří stěžejní část algoritmu OLC, kdy použité metody, techniky a heuristiky výrazně ovlivňují rychlost a kvalitu výsledku. Principiálně se tento krok snaží provádět dvě činnosti: spojit vstupní ready do contigů a vytvořit z contigů supercontigy. Na výstupu se pak následně nachází seřazená posloupnost readů, které se již nacházejí na správných pozicích genomu (tzv. graf uspořádání) [2].

Vytvoření grafu překryvů

Po kroku overlap byly získány překryvy mezi jednotlivými ready. Aby bylo možné provést analýzu a sestavení genomu, je důležité transformovat vstupní data do vhodné datové struktury. Pro tento účel nejlépe vyhovuje forma grafu, kde ready představují vrcholy grafu a hrany grafu reprezentují překryvy (obrázek 4.3).



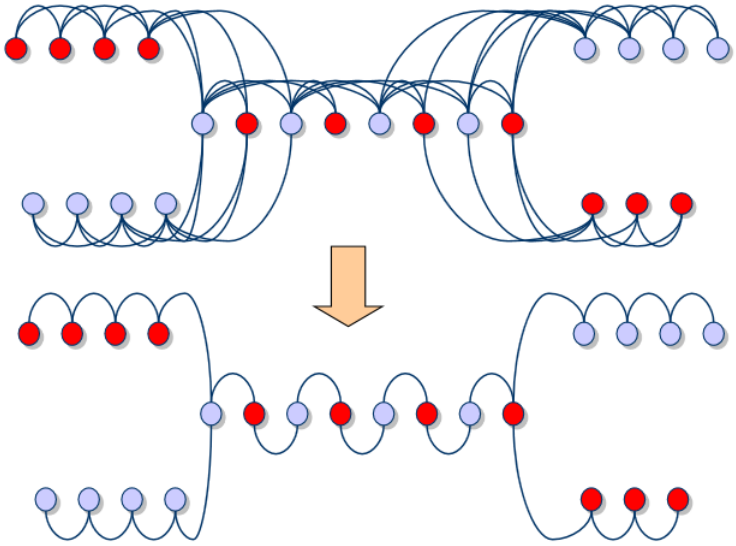
Obrázek 4.3: Příklad jednoduchého grafu překrytí. Vrcholy orientovaného grafu jsou tvořeny ready a hrany představují překryvy mezi nimi (ohodnocení hrany udává velikost překryvu) [6].

Spojení readů do contigů

Graf překryvů typicky obsahuje velké množství vrcholů a hran. Aby bylo možné spojit ready do contigů, je třeba tento graf upravit a zajistit tak co nejefektivnější tvorbu contigů. První úpravou grafu je odstranění tranzitivních uzávěrů mezi ready. Mějme ready r1, r2 a r3. Pokud existují hrany (překryvy) mezi ready (r1, r2), (r2, r3) a (r1, r3), pak je hrana (r1, r3) odstraněna. Výsledný graf se zjednoduší (obrázek 4.4). Následnou analýzou grafu získáme dva druhy contigů:

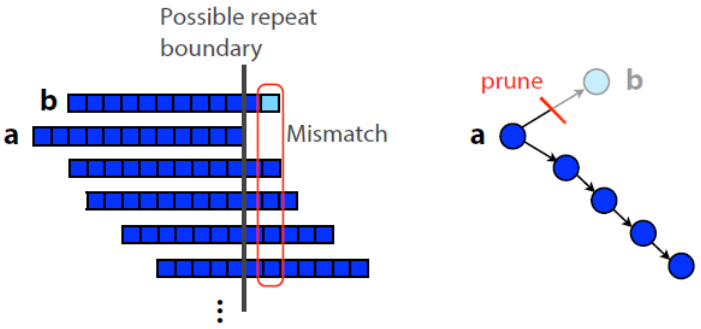
1. **Contigy unikátních sekvencí - unitigy** - Contigy obsahují pouze unikátní sekvenci genomu (např. gen), vazby na jiné contigy bývá výjimečné a jejich sestavení je obvykle triviální.

2. **Contigy repetitivních struktur** - Tyto contigy se vyznačují vysokým obsahem readů, které dosahují vysoké vzájemné hodnoty překryvu. Pro repetitivní contigy jsou dále charakteristické vazbami na velké množství unikátních contigů.



Obrázek 4.4: Graf překrytí před a po odstranění tranzitivních uzávěrů [14].

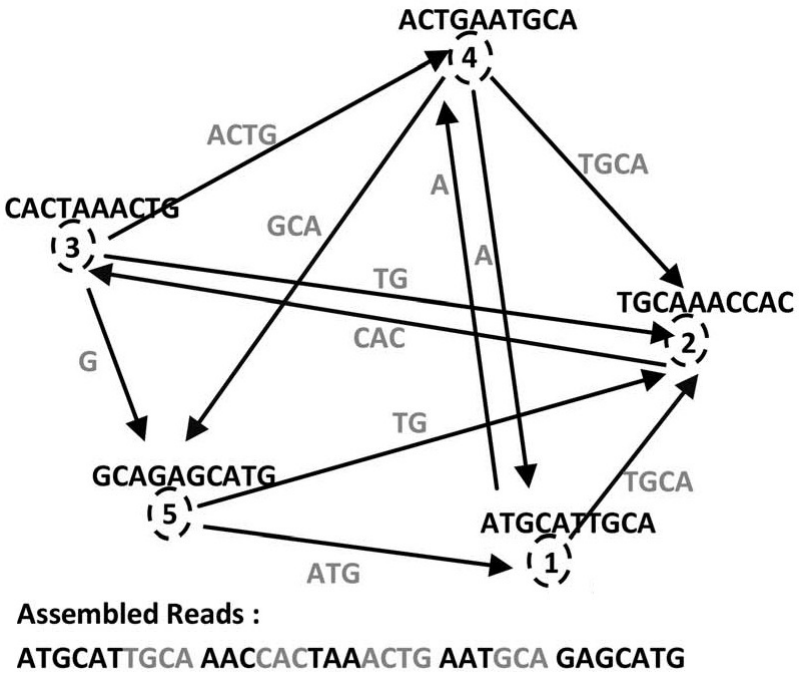
Při tvorbě unikátních úseků dochází i ke vzniku chybných cest v grafu (obrázek 4.5). Tyto cesty vznikají z důvodu chybných bází, získaných při sekvenování a při dalším vyhodnocení grafu by působily problémy, proto je nutné je odstranit. Nelze opomenout, že grafy mohou obsahovat cesty, které mají totožný výchozí a koncový vrchol a vytvářejí v grafu cykly (tzv. bubliny). Typicky se tyto cesty nacházejí v místech inzercí nebo delecí v repetici a mají vliv na délku a kvalitu výsledné cesty. Cykly jsou v grafu obecně nežádoucí, proto je snaha tyto cesty z grafu odstranit nebo přerušit.



Obrázek 4.5: Ukázka vzniku cesty v grafu z důvodu chybné báze a její odstranění [9].

Výsledná sestava je získána hledáním cesty, která prochází všemi vrcholy grafu a to právě jednou - tzv. hamiltonovská cesta (obrázek 4.6). Nalezení hamiltonovské cesty je z hlediska složitosti problematické, protože se jedná NP-úplný problém (polynominální časová složitost hledání). Tato skutečnost je často považována za největší slabinu tohoto přístupu.

Výsledkem první fáze kroku layout je graf contigů (obrázek 4.7). Na obrázku jsou patrné unikátní A, B, C contigy a contig obsahující repetici X. U unikátních contigů je z obrázku patrné, že kromě fragmentů obsahujících jedinečné sekvenci DNA obsahují také ready, které



Obrázek 4.6: Ukázka hledání hamiltonovské cesty v grafu překryvů. Cesta je získána průchodem přes všechny vrcholy (přestavující ready) grafu. Číselný index uvnitř každého vrcholu určuje pořadí vyhodnocení vrcholu v rámci cesty [5].

v sobě částečně obsahují repetice tzv. chimérické ready. Jelikož je účelem této práce hledat repetetivní elementy v DNA, bude nutné tyto chimérické ready vyhledávat a získat z nich informace o začátku a konci repetic.

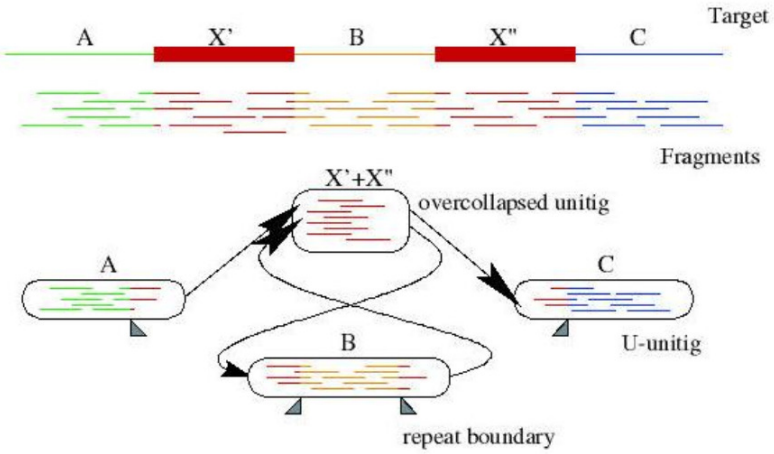
Tvorba supercontigů (scaffolding)

Posledním krokem fáze layout je seskupení contigů do větším celků - supercontigů. Tento krok se v assemblerech založených na přístupu OLC značně liší. U assemblerů nevyužívajících párové ready je sestavení supercontigů komplikované a je potřeba použít speciální heuristiky. U moderních assemblerů jsou už na vstupu typicky preferovaný pouze párové ready.

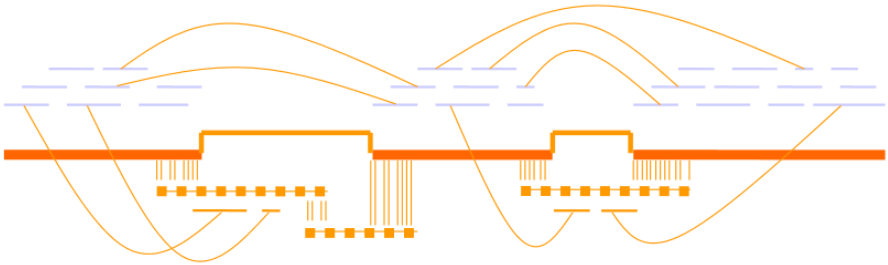
Párové informace readů určují vazby mezi contigy spolu s jejich orientací. Pokud dva unikátní contigy v sobě obsahují alespoň 2 párové vazby fragmentů jsou tyto contigy zařazeny do fronty pro vytvoření supercontigu. Z této fronty jsou přednostně zpracovávány contigy s vysokým počtem párových vazeb a menší mezerou mezi contigy. Vzniká graf supercontigů, který ještě není zcela úplný, protože obsahuje mezery mezi jednotlivými supercontigy. Mezery vznikají v místech repetic nebo v oblastech, které dosáhly nízkého pokrytí při sekvenování. Pro odstranění mezer se používáním párových vazeb získaných z contigů obsahujících repetice (demostrováno na obrázku 4.7) [17].

4.3.3 Consensus

Vstupní ready a následně vytvořené contigy mohou obsahovat chyby v podobě mezer nebo chybných nukleotidových bází, které zhoršují výslednou kvalitu výsledků. Je tedy nutné



Obrázek 4.7: Graf sestavení s jasně vyznačenými unikátními contigy A,B,C a contigem X, obsahujícím repetici [14].

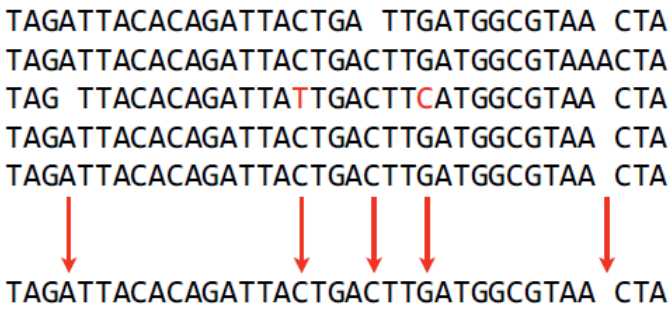


Obrázek 4.8: Zacelení mezer mezi supercontigy s pomocí párových informací a fragmentů contigů obsahujících repetice [14].

tyto báze vyhledat a opravit je. Algoritmus OLC využívá pro tento účel velmi jednoduchou metodu většinového hlasování. Princip většinového hlasování je zobrazen na obrázku 4.9. Všechny ready v rámci contigu jsou zarovnány pod sebe. Následně jsou porovnány báze na každé pozici. Do výsledné sekvence je zapsána nejčastěji se vyskytující báze (případně mezera) [9].

4.3.4 Assemblery založené na přístupu OLC

Na přístupu OLC bylo v průběhu let vytvořeno velké množství assemblerů, lišících se od sebe v použitých technikách hledání překryvů a zarovnání. Většina assemblerů se zaměřuje na sestavování celého genomu například Arachne [1], Celera Assembler [3] nebo CAP3 [8]. Typicky tyto assemblery nebyly navrženy pro hledání repetitivních struktur (jedním z důvodů byla prvotní teze, že repetitivní DNA nehraje žádnou roli v genomu - tzv. odpadní DNA). Až s novými poznatky o důležitosti repetitivní DNA se začaly objevovat první assemblery, konstruované přímo na hledání a analýzu repetic. Velmi známým a rozšířeným sestavovačem repetic se stal RepeatExplorer [16], který bude detailněji popsán v následující podkapitole.



Obrázek 4.9: Ukázka funkce většinového hlasování. Každý řádek představuje jednu zarovnanou sekvenci. Pro každý sloupec je vypočtena nejčastěji se vyskytující báze (případně mezera) [9].

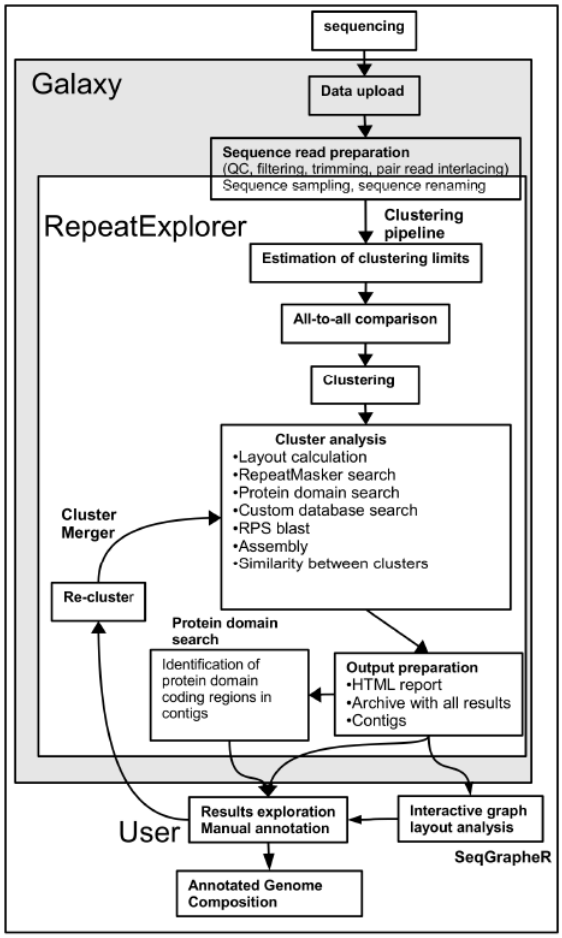
RepeatExplorer

RepeatExplorer je volně dostupná sada nástrojů umožňujících vyhledávání a následnou identifikaci repetitivních struktur v genomu. Assembler nevyžaduje pro svou činnost referenční databázi prvků a umožňuje proto zpracování dat tzv. *de novo*. RepeatExplorer byl implementován v prostředí Galaxy a je přístupný přes webové rozhraní, kde je však nutné počítat s limity serveru, jako například omezení paměti na 16 GB [16].

Celkové schéma jednotlivých komponent RepeatExploreru je zobrazeno na obrázku 4.10. Nejprve jsou na vstup nahrána data ve formě sekvenčních readů. Velikost těchto readů není omezena. Následně dochází k předzpracování vstupních dat sadou nástrojů platformy Galaxy, čímž se sníží jejich celková velikost. Dále je provedeno zarovnání readů s využitím clustering pipeline. Pro hledání překryvů je využita technika „all-to-all". Výsledkem je graf clusterů. Tento graf dále prochází pomocí sady nástrojů analýzou, kdy jsou identifikovaný a extrahovány jednotlivé sekvenční regiony nebo hledány proteinové domény. Získané výsledky mohou být následně uživateli prezentovány jak v textové tak i v grafické formě (pomocí SeqGrapher), kde může uživatel následně provádět manuální anotaci nalezených elementů [16].

4.4 Porovnání přístupů pro sestavení repetetivní DNA

Sestavení repetitivních úseků není v praxi jednoduché. Genomy obecně obsahují širokou škálu rodin repetic (popsaných v kapitole 2), které se nacházejí v mnoha různých variacích. Tyto problémy velmi ovlivňují použitelnost jednotlivých metod pro sestavení repetice. Aby bylo možné dosáhnout co možná nejlepších výsledků je nutné vybrat vhodnou techniku detekce a sestavení repetitivních sekvencí. Předchozí kapitoly představily celkem tři přístupy pro sestavení genomu, založené na tvorbě grafu překryvů (OLC), sestavení de Bruijn grafu (DBG) a vytváření grafu řetězců. Metoda řetězcových grafů provádí vysokou filtraci vstupních readů. To má za následek nižší opakovatelnost repetitivních úseků a jejich horší detekovatelnost v grafu. Tato technika tedy není pro hledání repetic příliš vhodná a nebude dále porovnávána. Celkový proces sestavení repetitivních elementů DNA můžeme rozdělit do několika fází, které současně poslouží jako hodnotící kritéria metod.



Obrázek 4.10: Schématické znázornění assembleru RepeatExplorer. [16].

4.4.1 Předzpracování dat

Prvním kritériem hodnocení metod je fáze předzpracovaní dat. Vstupní data obsahují typicky velké množství chyb, které mohou následně ovlivnit časovou a prostorovou složitost ostatních fází sestavení. V případě DBG je předzpracování dat nutné, protože každá chybná báze generuje v grafu novou cestu. To má za následek vznik rozsáhlého grafu s cestami, které ve skutečnosti v genomu neexistují a zvýší se tím časová i prostorová složitost algoritmu. Vstupní ready je tedy nutné opravit případně filtrovat, aby se výskyt případné chyby snížil na minimum. V praxi dostupné techniky filtrace a opravy jsou však složité a vyžadují nemalé výpočetní a paměťové zdroje.

Naproti tomu se metoda OLC dokáže velmi dobře vypořádat s chybnými bázemi ve vstupních datech. Fáze předzpracování dat tedy bývá typicky velmi jednoduchá nebo není v některých případech nutná a úplně chybí.

4.4.2 Sestavení grafu

Obě metody transformují vstupní data do formy grafu. Rozdíl mezi metodami je především v použitých datech pro vytvoření grafu. Metoda DBG musí nejprve provést transformaci readů na k-mery a následně vytvořit k-1 mery. To ovšem vyžaduje nemalé paměťové zdroje

a je navíc ztracena informace o spojitosti readu. Paměťová a časová náročnost spolu s počtem k-merů je závislá na délce readů a stanovené délce výsledných k-meru: čím je délka readu větší, tím je vyšší výpočetní složitost i počet k-merů. Jelikož k-mery tvoří vrcholy grafu, bývá vytvořený graf rozsáhlý a práce s ním by byla velmi složitá. Je tedy nutné na grafu provádět úpravy, které zjednoduší následné sestavení repetice.

Metoda OLC potřebuje před vytvořením grafu nalézt všechny překryvy mezi ready. V praxi se typicky se pro tento účel využívá technika dynamického programování, kdy je porovnání prováděno tzv. „all-to-all". Z hlediska potřebné paměti vyžaduje tato technika alokaci pole závislého na velikosti readů. Časová složitost je následně určena dobou nalezení nejlepšího překryvu a celkovém počtu readů. Počet vrcholů výsledného grafu překryvů je pak následně dán počtem vstupních readů a počet hran je určen počtem nalezených překryvů. Množství nalezených overlapů závisí především na nastavené hodnotě minimálního prahu překryvu a počtu opakování repetice v genomu. Typicky bývá repetice silně pokryta a počet nalezených hran je veliký. Následné hledání řešení (sekvence repetice) by bylo nadmíru obtížné. Po vytvoření grafu tedy běžně dochází k redukci hran, které jsou přebytečné (jedná se o tzv. tranzitivní hrany). Tato operace nebývá však v praxi zcela jednoduchá (např. problémy s cykly v grafu) a je třeba počítat s určitým výpočetním zatížením.

Repetice se v reálných genomech vyskytují v úplné délce jen zřídka. Velmi často dochází u repetic k vynechání určité části (delece) nebo naopak vložení unikátní sekvence do těla repetice (inzerce). Z toho vyplývá, že repetice stejného typu nemá vždy stejnou délku. Tyto vlastnosti se projevují i při tvorbě obou typů grafů. Takto poškozené repetice mohou vytvářet v grafu nové cesty, čímž se grafu zvýší jeho robustnost, a současně také vznikají chybné nebo slepé konce cest (tzv. dead-end), které mohou při následné sestavení repetice a hledání cesty grafem působit veliké potíže.

4.4.3 Sestavení repetice

Určení správné struktury repetice bývá v grafu velmi obtížné. Problematiku sestavení repetice lze rozdělit na dva hlavní problémy: nalezení začátku/konce repetice a nalezení optimální cesty, které nejlépe reprezentuje nalezený repetitivní element. Při hledání počátku repetice je třeba počítat s tzv. chimérickými ready. Chimérické ready obsahují část unikátní a repetitivní DNA. V grafu se s nimi lze typicky setkat nejen na začátku a konci grafu, ale také v místech inzerce unikátních sekvencí uvnitř repetice. V praxi jsou typicky tyto ready filtrovány. Výhodou filtrace je snazší nalezení sestavy repetice za cenu nižší výsledné kvality sestavy, kdy je nalezená repetice oříznuta na začátku a konci o několik desítek bází oproti skutečné délce repetice. Některé repetitivní rodiny (např. LTR retrotranspozony) obsahují na začátcích a koncích podobné sekvence. Pokud jsou tyto repetice v grafu nalezeny vytváří typicky kruhovou strukturu, kdy se začátky a konce repetice spojí. Podobné kruhové struktury jsou získány i v případě tandemového uspořádání repetice v genomu. Nalezení správného začátku repetice a tedy i místa, kde je možné graf přerušit bez poškození výsledné struktury repetice je složité, protože možná řešení tohoto problému bývají často efektivní pouze v případě určitého typu repetice.

Hledání výsledné cesty grafem tvoří hlavní rozdíl metod a v dnešní době i důvod, proč je dnes v praxi častěji preferována metoda DBG. Přístup OLC hledá hamiltonovskou cestu grafem s polynomiální časovou složitostí, zatímco DBG implementuje Eulerovu cestu s lineární časovou složitostí. Je tedy patrný nesrovnatelný rozdíl v časové složitosti obou metod, který se nejvíce projeví u rozsáhlejších grafů.

4.4.4 Vyhodnocení vlastností

Oba přístupy pro sestavení OLC a DBG mají své charakteristické vlastnosti, které ovlivňují jejich použitelnost pro sestavení repetic. Pokud jsou na vstupu data reprezentující nízké pokrytí genomu a lze očekávat případné chyby, je výhodnější volbou přístup OLC, který s chybou dokáže velmi dobře vypořádat. Tvorba grafu a jeho následné úpravy bývají u obou přístupů srovnatelně obtížné a není tedy v tomto případě možné rozhodnout, který přístup je v tomto ohledu výhodnější. Nicméně v hledání cesty grafem je mnohem výhodnější hledání cesty přes všechny hrany (Eulerova cesta) oproti cestě přes všechny vrcholy (hamiltonovská cesta). Hledání cesty je značná nevýhoda algoritmu OLC oproti metodě DBG. Na druhou stranu, OLC pracuje přímo s ready, nevyžaduje dodatečnou filtraci vstupních dat, neztrácíme informaci o spojitosti readů a je i výhodnější při používání párových readů. V současné době bývá v praxi preferováno spíše sestavování de Bruijn grafu před grafem překryvů. Důvodem je skutečnost, že většina nástrojů NGS poskytuje vcelku krátké ready, které ve výsledku produkují menší počet k-merů. Sestavovače založené na OLC nalézají spíše uplatnění v případech, kdy jsou na vstupu získávány delší sekvence nebo kdy kvalita těchto dat může být silně zatížena chybou.

Kapitola 5

Návrh řešení

V současné době existuje velké množství nástrojů pro sestavování repetic. Většina těchto nástrojů využívá přístup tvorby de Bruijn grafu (např. Tedna [24]). Rozdíly v nástrojích spočívají především v použití odlišných heuristik například pro detekci repetic, zjednodušení nebo vyhodnocení grafů. Výsledná kvalita sestavy je pak velmi silně závislá na genomu, z kterého jsou data získána. V praxi tedy nelze zcela jednoznačně rozhodnout, který z assemblerů je nejlepší, protože assembler je typicky optimalizován pro hledání specifických typů repetic (např. LTR-transpozony) a proto při sestavě ostatních rodin repetic nedosahuje zdaleka tak dobrých výsledků [21]. Při návrhu nového nástroje pro sestavení repetitivní DNA je proto nutné, položit si několik důležitých otázek:

- Jaká data lze na vstupu očekávat?
- Bude assembler zaměřen na hledání určitého typu repetic a jaký přístup bude nejvhodnější?
- Co má být výstupem nástroje?
- Jaký způsob testování bude použit?

5.1 Vstupní data

Vstup algoritmu tvoří datová sada nasekvenovaných readů uložená v souboru typu FASTA (případně FASTQ). Formát FASTA představuje v bioinformatice standard pro uložení a práci s biologickými daty. Každý záznam v souboru FASTA se skládá ze dvou částí, oddělených odřádkováním. První část, začínající znakem '>' (u FASTQ '@'), tvoří jedinečný popis, identifikaci a případně dodatečnou informaci o datech. Druhou část představují samotná data (typicky peptidové nebo nukleotidové sekvence). U formátu FASTQ je navíc přidána také informaci o kvalitě dat (skóre), oddělenou od sekvence symbolem '+'. Obsah souboru pak vypadá následovně:

FASTA:

```
>r1_1
CCCGGCTAAGGGAGGAGACCACCCCTCATATTGTCTTATGCCCAATTTCTGCCTCCAAAG
AAAGAARAAGTAAAAACTAAAAGGCAGAAATGAAATCCAC
>r1_2
GCCTCCAAAGAAAGAARAAGTAAAAACTAAAAGGCAGAAATGAAATCCACAGGCAGACAG
```

```
CCCGGCGCCACACCCTGGGCCTGGTAGTTAAAGATCGACC
```

FASTQ:

```
@ERR2014428.1 M02925:24:000000000-A9MAE:1:1101:10744:1000/1
NGCCTGTACTTGGAGACGGCGATTGAGCCGGTTTGGTACTTGGGCGAGCGCCTAGTTCGC
CAGTGCTTTACCGAGACTTTCGTCGTGCCGGTTGACCCGC
+
B8BCCGGGGGGGGGGGGGGGGGGGGGFGGGGGGGFGGGGGGGGGGGGGGGGGGGGGGG
GGFGGGGGGGGGGGGGGGGGGEGGGGGGGGGGGGGGGGG
```

I přesto, že FASTA tvoří standard pro uložení libovolných bioinformatických informací, v případě uložení readů (nebo párových readů) neexistuje jednotná standardizace popisu readů. Reálná data reprezentují typicky nízké pokrytí genomu a délka jednotlivých readů se může lišit (např. pokud data procházejí dodatečnými úpravami). Tato skutečnost může často komplikovat vývoj nástroje a je třeba s ní počítat.

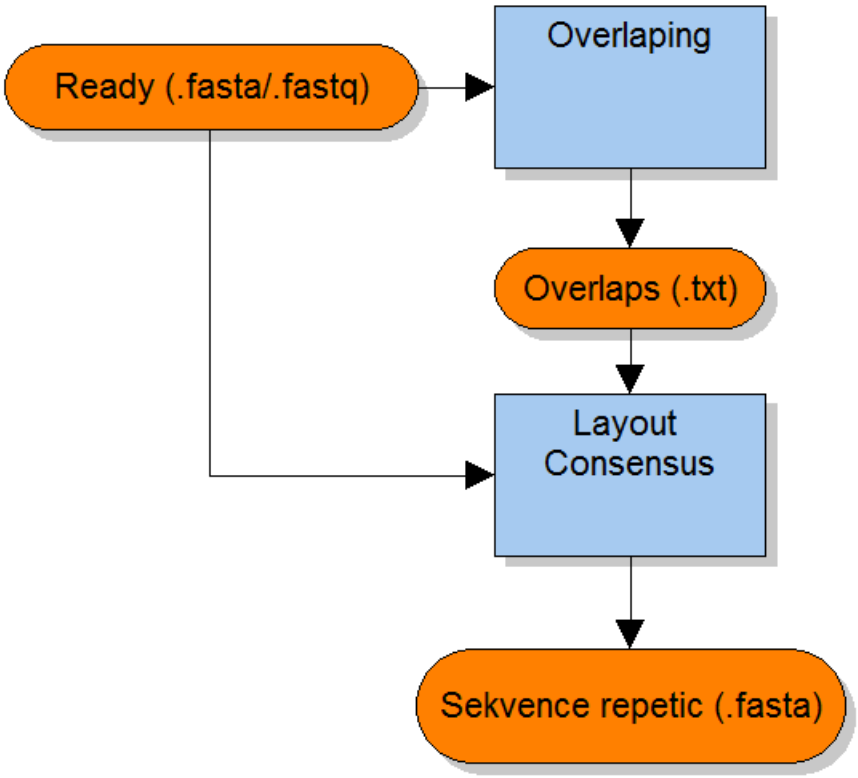
5.2 Zaměření assembleru

Po pečlivé úvaze byla pro implementaci zvolena metoda OLC. Důvodů k tomuto rozhodnutí je hned několik a vycházejí především z porovnání obou metod v kapitole 4.4. Hlavní předností metody OLC je především snazší fáze předzpracování, není nutná filtrace vstupních dat a metoda toleruje variabilitu repetic. Značnou nevýhodou přístupu OLC je hledání cesty přes všechny vrcholy grafu, kdy složitost hledání s robustností grafu roste. Tento problém bude nicméně redukován, protože repetice a nízké pokrytí genomu způsobí rozdělení grafu na jednotlivé části (tzv. komponenty grafu). Následně bude potřeba určit, zda daná komponenta grafu obsahuje/neobsahuje přímo repetitivní element. Hledání cesty a vyhodnocení sekvence bude provedeno pouze v případě detekce repetice. Tímto způsobem dojde k redukci nevýhody hledání cesty grafem a zlepší se i celková rychlost assembleru.

Repetice jsou velmi různorodou skupinou elementů v genomu. To má za následek vznik odlišných uskupení (tvarů) v grafu, které tyto repetice reprezentují. Útvary, obsahující repetice, obvykle dosahují vysoké koncentrace vrcholu grafu, které jsou vzájemně silně propojené (tzv. vysoká úroveň vrcholu - počet hran vedoucích z/do uzlu) oproti okolí vrcholům, obsahujícím unikátní sekvence. Odhalení repetice tedy nebývá v grafu příliš složité, díky vyšší propojenosti vrcholu v místě repetice. Způsob univerzální detekce začátku repetice je však komplikovaný a dostupná řešení (např. hledání vrcholu s nejvyšším stupněm) nemusí vždy fungovat správně. Některé metody analyzují podgraf a sestavují statistiky o jednotlivých vrcholech. Následně s využitím heuristických metod vyhodnocují pravděpodobných začátek/konec repetice, jiné metody pro změnu určují jako začátek vrcholy s nejvyšší úrovní vrcholu. Celkově však žádná technika nedokáže zaručit, že nalezený počátek repetice je skutečně počátkem libovolného repetitivního elementu v genomu a kvalitních výsledků dosahují na určitém typu repetic.

Vytvořit assembler pro hledání libovolných repetitivních prvků a jejich následné kvalitní sestavení je téměř nereálné. Lepší variantou je zaměřit assembler na hledání specifických typů repetic a dosahovat při sestavě takovýchto elementů kvalitních výsledků. Nástroj bude proto zaměřen na hledání elementů, které vytvářejí graf s kruhovou topologií. Typickými představiteli těchto repetitivních elementů jsou LTR retrotranspozony nebo satelitní DNA. LTR retrotranspozony obsahují na svém začátku a konci LTR (viz. kapitola 2). Pokud tyto terminální repetice obsahují silně konzervované úseky, které jsou na obou koncích podobné,

může docházet k vytvoření kruhovou strukturu repetice v grafu. Vrcholy, nesoucí nukleotidovou sekvenci LTR, budou navíc oproti ostatním částem více pokryté. K tomu faktu přispívá i skutečnost, že dlouhé terminální repetice se mohou v genomu nacházet i samostatně. Na úsek LTR bude proto, kromě vrcholu obsahujících repetitivní DNA, navázáno i velké množství vrcholů představujících chiméry. Počátek repetice bude tedy detekován ve vrcholu s nejvyšší koncentrací hran. Na tomto místě bude kružnice rozpojena a bude jednoznačně určen počátek a konec repetice. U repetic také nelze opomenout skutečnost, že repetitivní prvky se v genomu nacházejí v plné délce jen ojediněle a obsahují tzv. indely (inzerce a delece). Inzerce i delece vytvářejí v grafu nové větve. Tyto větve mají následně vliv na hledání a kvalitu výsledné cesty. Primárně bude assembler implementován pro použití na datovém centru **Metacentrum**[1], poskytujícím především robustní výpočetní, paměťové zdroje a širokou škálu modulů a nástrojů.



Obrázek 5.1: Blokové schéma struktury assembleru. Modré bloky představují jednotlivé samostatně spustitelné programy. Oranžové bloky reprezentují tok dat mezi jednotlivými částmi aplikace. Orientované hrany znázorňují tok dat mezi komponentami.

5.2.1 Koncepce assembleru

Koncipovat assembler, jako jeden samostatný program by bylo velmi nevýhodné. Pro testování a univerzálnější použití bude mnohem lepší rozdělení nástroje na několik samostatně spustitelných částí, kdy nebude nutné například hledání překryvů při každém spuštění pro-

[1] https://metavo.metacentrum.cz/

gramu, ale bude možné program spustit s již předem nalezenými překryvy. Assembler se bude tedy skládat ze dvou částí (obrázek 5.1):

- **Overlaping** - Nalezení, uložení a následná filtrace všech překryvů readů, nalezených nástrojem BLAST.
- **Layout & Consensus** - Aplikace ze získaných vstupních dat sestaví graf překryvů. Poté program nalezne a vyhodnotí repetitivní elementy v genomu a na výstupu je pak poskytnut soubor typu FASTA, obsahující sekvence sestavených repetic.

5.2.2 Výstupní data

Assembler produkuje na svém výstupu sekvence nalezených repetic. Kvalita a délka sestavené sekvence repetice bude silně ovlivněna nejen vstupními daty, ale zvoleným typem vyhodnocení (volba cesty grafem) repetice. Lze tedy očekávat možnou další práci s těmito informacemi, proto je nutné výstupní data poskytovat ve vhodném formátu. Obdobně, jako v případě vstupních dat, bude nejvhodnější variantou standardizovaný formát FASTA. Každý nalezený repetitivní prvek bude mít v souboru jedinečný identifikátor a úspěšně sestavenou sekvenci.

Kapitola 6

Implementace

Pro implementaci nového nástroje zvolit vhodný programovací jazyk, který bude nejvíce vyhovovat potřebám nástroje. Assemler by měl být snadno ovladatelný a použitelný. Byl zvolen jazyk Python, který poskytuje dobrou rychlost výpočtu, sílu skriptovacích jazyků (není třeba program překladat a použitelnost bude vyšší) a širokou škálu dostupných pomocných knihoven pro práci s bioinformatickými daty a formáty. Sestavovač bude rozdělen, jak již bylo uvedeno v kapitole 5, na dvě samostatně spustitelné aplikace. Jejich implementace a popis bude uveden v následujících podkapitolách.

6.1 Potřebné knihovny a nástroje

Assembler bude potřebovat pro svou činnost několik pomocných knihoven a nástrojů. Většina z těchto nástrojů je již přístupná v rámci Metacentra a není tedy nutná jejich dodatečná instalace. Jedná se především o stěžejní knihovny a nástroje:

- balíček **igraph**[1] - Jedná se o volně dostupnou balíček pro práci s grafy. Práce s knihovnami balíčku je díky množství implementovaných funkcí velmi snadná. Implicitně však *igraph* neumožňuje vykreslení vytvořených grafů.
- balíček **cairo**[2] - Balíček grafických funkcí *cairo* poskytuje nástroje pro práci s grafy. *Cairo* je přímo kompatabilní s *igraph*, je tedy možné snadné vykreslení grafu.
- balíček **biopython**[3] - Dnes již téměř standardní komponenta pro práci s genetickými daty. Biopython je kolekce funkcí a nástrojů pro rychlý a efektivní vývoj bioinformatických nástrojů.
- nástroj **MegaBlast**[4] - MegaBlast vyhledává překryvy mezi vstupními ready s využitím dynamického programování. Nástroj je volně dostupný ve formě přídavného modulu na serverech Metacentra.
- nástroj **FormatDB** - Program pro vytvoření databáze readů, kterou vyžaduje na svém vstupu MegaBlast. Aplikace je volně dostupná ve formě přídavného modulu na serverech Metacentra.

[1] http://igraph.org/python/
[2] https://cairographics.org/pycairo/
[3] http://biopython.org/
[4] https://wiki.metacentrum.cz/wiki/TGICL

6.2 Nalezení překryvů

V současné době existuje široké spektrum programů, které problematiku hledání overlapů již efektivně řeší. Není tedy nutné implementovat nový nástroj. Metacentrum primárně poskytuje modul MegaBlast, dosahují dobrých výsledků, a proto byl tento program zvolen pro řešení této fáze assembleru. MegaBlast je vysoce multifunkční a na svém vstupu přijímá množství dodatečných parametrů. Implicitně je provedeno hledání překryvu mezi vstupním souborem readů (označeny Q -query) s přiloženou databází readů, vytvořenou modulem `formatdb` (označeny S). Modul produkuje na svém výstupu textový soubor, ve kterém jsou nalezené překryvy uloženy na samostatném řádku v tabulkové formě (tzn. jednotlivé informace oddělují mezery). Formát uložení překryvu je následující: název Q, délka Q, počátek překryvu na Q, konec překryvu na Q, název S (s předponou lcl|), délka S, počátek překryvu na S, konec překryvu na S, identita (shoda) v místě překryvu, délka překryvu, nutnost reverze. Pro názornou ilustraci je příklad možného výstupu uveden níže.

```
r8_1 100 1 76 lcl|r17_1 100 25 100 100 148 74 +
```

Počet a kvalita nalezených překryvů primárně závisí především na následujících parametrech, které je možné nastavit:

- **identity** - hodnota identity určuje minimální procentuální shodu bází, pro vytvoření překryvu,
- **min_overlap** - minimální délka překryvu dvojice readů,
- **num_mismatch** - maximální počet mezer, které je možné použít pro nalezení nejlepšího překryvu.

MGBlast nerozlišuje od sebe porovnávané ready. Jsou proto například vyhledány i překryvy readu na sebe sama, které jsou při tvorbě grafu překryvů nežádoucí. Výstupní soubor překryvů implicitně vyžaduje filtraci. K tomu účelu byl implementován skript *filter_mgblast.py*. Program vyžaduje celkem tři parametry: vstupní soubor překryvů, jméno výstupního souboru a minimální odsazení readů od sebe.

6.3 Sestavení repetic

Implementace programu pro sestavení repetitivní elementů se nachází v souboru *olc.py* a tvoří jádro celé této práce. Nejdůležitější funkce jsou popsány v rámci této kapitoly. Program začíná spuštěním hlavní funkce `main()`. V `main()` jsou postupně spouštěny ostatní funkce a probíhá zde i kontrola vstupních parametrů. Celkem aplikace očekává pětici parametrů:

- **-r <read file>** - Vstupní soubor nasekvenovaných readů. Přijímány jsou formáty FASTA a FASTQ.
- **-l <overlaps file>** - Soubor obsahující nalezené překryvy mezi ready.
- **-t práh filtrace vrcholu <int>** - Nastavení prahové hodnoty pro filtraci neužitečných vrcholu v grafu (např. vrcholy bez překryvu nebo vrcholy, obsahující unikátní sekvence genomu). Tento parametr má uplatnění pouze u silně pokrytých genomů. Na datech, představující nízké pokrytí, je doporučeno nastavit hodnotu na 0.

- **-p typ cesty grafem** - Volba mezi hledáním nejdelší (LP) a nejkratší (SP) cesty grafem.
- **-o <output file>** - Jméno výstupního souboru, obsahujícího sekvence nalezených repetic.

6.3.1 Tvorba grafu

Pokud vstupní parametry vyhovují, je zavolána funkce `createOverlapGraph()` pro vytvoření grafu překryvů. Funkce nejprve vytvoří strukturu orientovaného grafu, vytvořením instance třídy `igraph.Graph`. Následně jsou načteny všechny ready v příslušném vstupním souboru a je provedena jejich transformace do podoby vrcholů grafu. Každý vrchol obsahuje informace o jedinečném ID readu, nukleotidové sekvenci a typu (v případě párových readů). Poté je načten soubor s nalezenými překryvy. Kromě důležitých informací o překryvu (např. délka překryvu, počátek/konec překryvu) je pro hrany vypočítána váha, protože ready v reálných datových sadách nemusí mít vždy shodnou délku. Tato skutečnost může komplikovat úpravy grafu, protože kratší ready produkují typicky i kratší délku překryvu, což může vést k vyčlenění vrcholu z výsledné cesty grafem. V programu byla proto váha hrany vypočítána z délky překryvu vzhledem k délce obou readů (funkce `adaptiveOverlap(x)`). Graf následně prochází sadou úprav s cílem odstranění zbytečných vrcholů a hran. Tyto operace však mají negativní vliv v následující fázi algoritmu, kdy je hledán začátek repetice 6.3.2. Před úpravami grafu jsou proto uloženy aktuální hodnoty stupně vrcholu (proměnné IN a OUT), které slouží k určení počátku repetice. Následně jsou z grafu odstraněny všechny vrcholy nevyhovující prahové hodnotě, určené parametrem programu.

6.3.2 Hledání repetic

Vytvořený graf překryvů je rozdělen na jednotlivé shluky. Shlukování je provedeno standardní knihovní funkce `clusters()`. Metoda vyhledá v grafu všechny souvislé struktury a ponechá pouze ty, které jsou kruhové struktury. Nalezené shluky reprezentují potenciální repetitivní prvky v genomu. Pro každou takto nalezenou strukturu v grafu je volána funkce `createLayout()`. Funkce `createLayout()` nejprve vyhledá vrchol s nejvyšším stupněm a označí tento vrchol jako počátek repetice. V tomto místě dochází k rozpojení grafu a označení předpokládaného konce elementu (vrchol, který má nejlepší překryv s počátečním vrcholem). Z takto upraveného grafu je již teoreticky možné hledat hamiltonovskou cestu, nicméně velké množství tranzitivní hran by vedlo k vysoké časové složitosti hledání. Program proto provádí nejprve odstranění tranzitivity v grafu. Většina kvalitních a efektivních algoritmů pro řešení problému tranzitivity vyžaduje směrový acyklický graf (tzv. DAC). Případné cykly typicky nevedou k řešení nebo v horším případě je vráceno nekvalitní řešení. Pro odstranění cyklů v grafu je v aplikaci použita standardní funkce `feed_arc_set()` (FAC), pracující na principu vyhledání tzv. spanning tree v grafu. Metoda vyhledá nejmenší počet hran, které je nutné odstranit pro acyklicitu grafu. Při hledání řešení bere funkce v úvahu váhu hran a přednostně jsou odstraněny hůře hodnocené hrany. Na výsledný graf je následně aplikováno odstranění tranzitivity voláním funkce `hsuTransiReductionAlgo()`. Princip odstranění tranzitivy je založen na sestavení matice cest (angl. path matrix), které je následně při rozhodování zda je hrana tranzitivní. V této fázi programu je již graf velmi jednoduchý a vhodný pro hledání cesty grafem. Jelikož je již znám předpokládaný začátek a konec repetic je pro hledání cesty použita standardní knihovní funkce `get_shortes_path()`, která vyhledá nejkratší cestu grafem mezi detekovaný počátkem a koncem repetice. Důvod, proč

byla použita nejkratší cesta, je především v možnosti zanesení případných inzercí do výsledné repetice. Tyto inzerce by mohly působit problém i při použití standardní implementace algoritmu hledání hamiltonovské cesty, jelikož by algoritmus nebyl schopen vyhledat cestu přes všechny vrcholy grafu.

6.3.3 Vyhodnocení nalezené repetice

Výsledné cesta je reprezentována jako seznam vrcholů. Poslední krokem algoritmu je nalezení consensus sekvence k nalezené cestě - funkce `makeConsensus()`. Krok vyhodnocení repetice je řešen pomocí cyklu přes všechny vrcholy cesty, kdy se v každém kroku provádí zarovnání a vyhodnocení překryvu dvou po sobě následujících vrcholů. Následně je získaná sekvence o seknuta na svém konci o část tvořící překryv s počátečním vrcholem která je redundantní (vyplývá z kruhové struktury původního grafu) a uložena do zadaného souboru typu FASTA (všechny výstupní soubory repetic se primárně nachází v adresáři output).

Příklad spuštění

- *./python olc.py -r test1-reads.fasta -l test1-overlaps.txt -t 0 -o output.fasta*

6.4 Spuštení na serveru Metacentrum

Aplikace byla primárně navrhnuta pro použití na serverech Metacentra, protože vyžaduje velké množství pomocných modulů a knihoven, které jsou zde již dostupné. Aby bylo možná snadná manipulace a spuštění podprogramů aplikace, vznikly k tomuto účelu pomocné skripty: *run_assembler.sh* - pro spuštění hlavní aplikace pro hledání repetic, *make_genom.sh* - vytvoření umělého genomu a *make_reads.sh* - vytvoření setu párových readů. Potřebné parametry a příklady spuštění jsou dostupné v hlavičkách souborů a nebudou zde proto uvedeny.

Kapitola 7

Testování

V této kapitole bude popsáno testování nástroje na umělých/generovaných a reálných datech. Pro potřeby testování byl proto implementován generátor umělého genomu a generátor readů. Podrobnější popis obou programů je v podkapitolách 7.1 a 7.2. Nástroj bude testován na několika sadách readů, získaných z uměle vytvořených genomů. Celkem budou použity čtyři testovací sady, které prověří činnost a chování nástroje za odlišných vstupních podmínek (např. procentuální zastoupení repetic v genomu, pokrytí genomu, zašumění dat, atd.). První dva datové sety mají posloužit pouze k demonstraci a otestování funkčnosti aplikace. Třetí test ukazuje chování assembleru, kdy jsou repetice silně zatíženy delecemi. Současně bude tento test simulovat skutečné datové sety, získané ze sekvenátorů. Poslední testování proběhlo přímo na reálných datech.

7.1 Generátor umělého genomu

V rámci aplikace byl implementován v jazyce Python 2.7 jednoduchý generátor unikátního genomu s názvem *genomGenerator.py*. Generátor přijímá na svém vstup soubor FASTA, obsahující repetice, které mají být v genomu obsaženy. Následně je volána funkce pro vytvoření požadovaných variant repetitivních elementů `makeRepeatsVariants()`. Parametry repetic jsou plně nastavitelné - počet opakování, inzerce, delece nebo mutace. Poté aplikace vygeneruje náhodnou sekvenci DNA (`makeUniSeq()`) a vytvořené repetice vloží na náhodné pozice v genomu. Celková velikost výsledného genomu je předávána parametrem. Vytvořený genom je uložen do souboru FASTA funkcí `writeGenomToFasta()`. Pokud nelze požadovaný genom vytvořit (např. byla zadána příliš malá velikost genomu nebo chybné vstupní parametry), program nevygeneruje genom a vypíše pouze standardní chybové hlášení.

Parametry generátoru

Generátor vyžaduje celkem 9 povinných parametrů:

- **-r <input file>** - Název vstupního souboru typu FASTA s uloženými repetitivními elementy.
- **-g genom_size <float>** - Velikost výsledného genomu v MB.
- **-n num_rep <int>** - Počet opakování repetice v genomu.
- **-i num_inserts <int>** - Celkový počet inzercí v rámci jednoho typu repetice. Délka inzerce je náhodná minimálně však délky 50 bp.

- **-d num_deletes <int>** - Počet delecí v repetici.
- **-v num_variants <int>** - Počet mutovaných repetic v genomu.
- **-t tandem_layout <int>** - Pravděpodobnost tandemového uspořádání jednotlivých repetic v genomu v %.
- **-o <output file>** - Název výstupního souboru typu FASTA s uloženým genomem.

Příklad spuštění

- *./python genomGenerator.py -r repeats.fasta -g 1 -n 20 -i 8 -d 5 -v 4 -t 60 -o output.fasta*

7.2 Simulace párových readů

Nad uměle vytvořeným genomem je pro potřeby testování nutné vytvořit datový set single nebo pair-end readů, simulujících ready nástrojem Illumina. Z tohoto důvodu byl naimplementován jednoduchý generátor readů *readGenerator.py*, který primárně umožňuje generovat ready libovolné délky s možností simulace chyby sekvenačního nástroje.

Parametry generátoru readů

- **-g <input file>** - Název vstupního souboru typu FASTA s uloženým genomem (např. test1-genom.fasta).
- **-n read_number <int>** - Počet nasimulovaných readů.
- **-l read_len <int>** - Zvolená délka readů.
- **-f frag_len <int>** - Délka fragmentu, ze kterého jsou párové ready získány.
- **-e sek_error** - Simulace chyby sekvenátoru vyjádřená v %.
- **-o <output file>** - Název výstupního souboru typu FASTA s uloženými ready.
- **-s (-p)** - Volba mezi single a pair-end ready.

Příklad spuštění

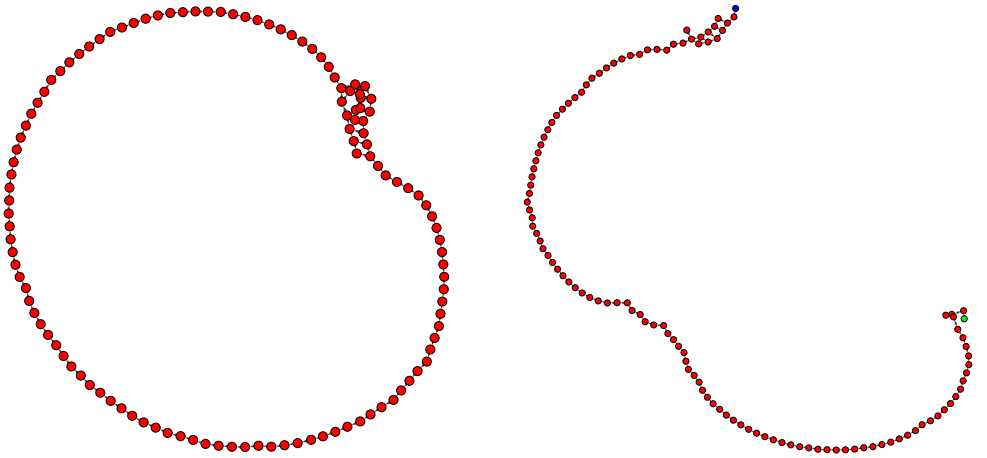
- *./python readGenerator.py test1-genom.fasta 120 100 test1-reads.fasta -s*

7.3 Testování na umělých datech

Testovací sada 1

Pro první testování byl vybrán uměle vytvořený genom, který obsahuje pouze jeden LTR retrotranspozon. Z databáze dosud objevených a anotovaných repetitivním elementů (Giri RepBase[1]) byla pro tento účel vybrána repetice *MAGGY*. Jedná se o krátký LTR retrotranspozon délky 5638 bází, který byl objeven v genomu myši obecné. Účelem prvního testování bylo především ověřit funkčnost aplikace a schopnost detekce repetice. Umělý genom a následná simulace readů byly vytvořeny s následujícími parametry:

[1] http://www.girinst.org/repbase/update/browse.php



(a) Nalezení repetice v genomu. Repetice je na obrázku zobrazena červenou barvou.

(b) Ukázka rozseknutí LTR retrotranspozonu. K rozseknutí repetice dochází v oblasti LTR. Zelenou barvou je reprezentován detekovaný počátek repetice a modrou její konec.

Obrázek 7.1: Demonstrace činnosti algoritmu při fázi zpracování grafu.

- **délka genomu**: 15000 bp
- **počet opakování repetice**: 2
- **zastoupení repetice v genomu**: 81%
- **tandemové uspořádání repetic** : 0%
- **délka readů**: 100
- **počet readů**: 120
- **šum** : 0%
- **pokrytí genomu**: 0.75x

V grafu byla podle očekávání nalezena pouze jedna souvislá struktura, která byla vyhodnocena jako repetitivní. Ostatní vrcholy byly vyhodnoceny jako unikátní části genomu a došlo k jejich odstranění. Na obrázku 7.1 je vykreslena struktura nalezeného LTR retrotranspozonu. Na první pohled je zde patrné, že repetice utvořila kruhovou strukturu. Současně dobře viditelný úsek s vysokým pokrytím (na obrázku vpravo nahoře), který představuje dlouhou terminální repetici na obou koncích LTR retrotranspozonu. Následně se algoritmus pokusil rozseknout kruhovou strukturu v místě vrcholu s nejvyšší hodnotou stupně a také odstranit všechny hrany působící cyklicitu grafu. Rozseknutí kružnice proběhlo v místě obsahujícím LTR, což byl očekávaný výsledek vzhledem k LTR retrotranspozonu.

Graf je nyní acyklický a program může provést redukci tranzitivních hran a vyhledání cesty grafem. Assembler úspěšně vyhledal cestu v grafu, která dosahovala délky 5651 bází (tabulka 7.1). Původní délka repetitivního elementu byla pouze 5638 bází. Důvodem prodloužení sekvence bylo začlenění několika unikátních bází (13 bp), které se nacházely

	MAGGY
Skutečná délka repetice (bp)	5638
Nalezená délka reptice (bp)	5651

Tabulka 7.1: Výsledky vyhodnocení sekvence repetice MAGGY. LTR retrotranspozon byl sestaven téměř v původní délce.

v posledním vrcholu cesty (read, tvořící tento vrchol byl chimérický). Jelikož byla délka unikátních části velmi malá oproti repetitivní části a došlo k začlenění tohoto vrcholu do celkové struktury repetice.

Testovací sada 2

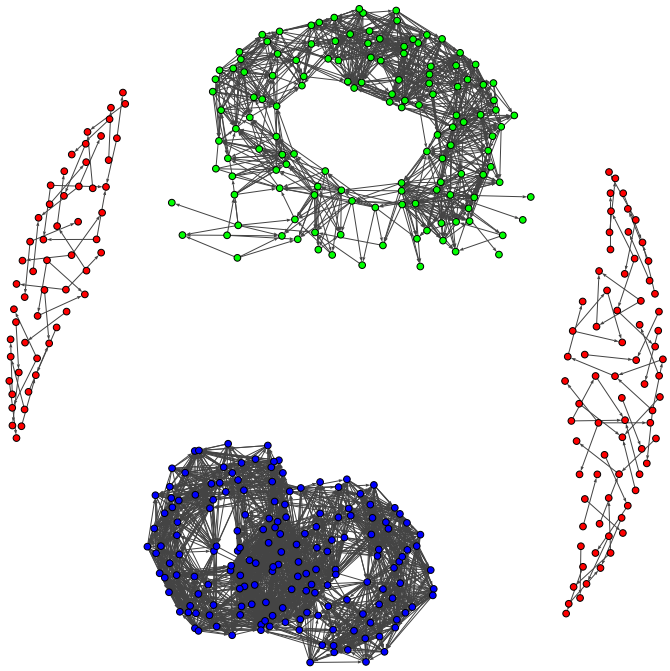
Z předchozího testování je patrné, že assembler dokázal nalézt a vyhodnotit zadaný repetitivní element. V případě jediné repetice tedy použité algoritmy fungují korektně. Cílem druhé sady testování proto bude ověření, že aplikace dokáže nalézt také satelitní elementy, které se budou nacházet v genomu a díky svému tandemovému uspořádání by měly vytvářet kruhovou strukturu. Z databáze RepBase byly pro účely testování vybrány dva satelity: *GSAT_MM* délky 471 bp, nalezený v genomu myši obecné (*Mus musculus*) a kratší *BMSAT1* z genomu plejtváka obrovského (*Balaenoptera musculus*) o celkové délce 422 bp. Dále bude sledováno chování algoritmu pokud budou repetice zatíženy inzercemi, delecemi a mutacemi. Výsledná datová sada byla proto vytvořena s následujícími parametry:

- **délka genomu**: 30000 bp
- **počet opakování repetice**: 20
- **zastoupení repetice v genomu**: 68,5%
- **variace repetice**: 4 inzerce, 7 delece, 2 mutace
- **tandemové uspořádání repetic** : 65%
- **délka readů**: 100
- **počet readů**: 500
- **šum**: 0%
- **pokrytí genomu**: 1.66x

Pro spuštění aplikace byl vytvořen graf překryvů, který obsahoval celkem 2 souvislé komponenty. Assembler je tedy schopen nalézt a rozlišit od sebe dva repetitivní elementy, které se nalézají v genomu (obrázek 7.2). U nalezených repetic je na první pohled patrné, že obě tvoří kruhovou strukturu. Satelit *GSAT_MM* byl sestaven v délce 585 bp. V tomto případě jednoznačně došlo k začlenění inzerce do celkové délky repetitivního prvku, protože skutečná délka satelitu je pouze 471 bp (tabulka 7.2). Sestavená délka satelitu *BMSAT1* byla o 5 bází kratší než je skutečná délka této repetice, což indikuje začlenění delece do výsledné sekvence repetice. Je tedy zřejmé, že indely mají v assembleru vliv na výslednou délku sestavené repetice.

	GSAT_MM	BMSAT1
Skutečná délka repetice (bp)	471	422
Nalezená délka reptice (bp)	585	417

Tabulka 7.2: Výsledky druhého testování. Ani jeden satelit nebyl sestaven satelit v plné délce. Projevil se zde vliv inzercí a delecí na výslednou délku cesty.



Obrázek 7.2: Graf nalezených repetic v genomu. Detekovány byly oba satelity GSAT_MM (modrá barva) a BMSAT1 (zelená barva).

Testovací sada 3

Předchozí testovací sady byly získány z relativně malých umělých genomu a pokrytí genomu bylo vysoké. V praxi ovšem assemblery pracují s rozsáhlými datovými sety (řádově statisíce readů), kdy pokrytí genomů nebývá příliš vysoké (5-30%). Třetí test proto vytvoří mnohem větší umělý genom a bude zde simulováno nízké pokrytí genomu. Umělý genom bude obsahovat celkem 3 LTR retrotranspozony a jeden satelitním element. Kromě již představeného LTR retrotranspozonu *MAGGY* a satelitu GSAT_MM byly do genomu přidány elementy:

- ALISEI - LTR retrotranspozon délky 3924 bp, nalezený u smrku ztepilého (*Picea abies*).
- HOBS - LTR retrotranspozon délky 6639 bp, obsažený v genomu houby sněť kukuřičná (*Ustilago maydis*).

Vytvořené repetice budou obsahovat velké množství variací, které lépe odpovídá skutečnému výskytu repetic v genomu. Navíc bude při sekvenování simulována chyba sekvenátoru, nastavená na 5%. Při testu bude tedy možné sledovat chování assembleru za specifických

	Délka repetice (bp)	Skutečná délka (bp)	Typ
REPET_ELEMENT_0	3013	3924	ALISEI
REPET_ELEMENT_1	405	422	GSAT_MM
REPET_ELEMENT_2	2751	6639	HOBS
REPET_ELEMENT_3	2777	5638	MAGGY

Tabulka 7.3: Shrnutí výsledků třetího testování. Nalezeny byly čtyři repetitivní elementy v genomu. Délka všech sestavených elementů se od skutečné délky vždy lišila. Hlavní vliv na délku repetic měla vysoká hodnota delecí v repeticích.

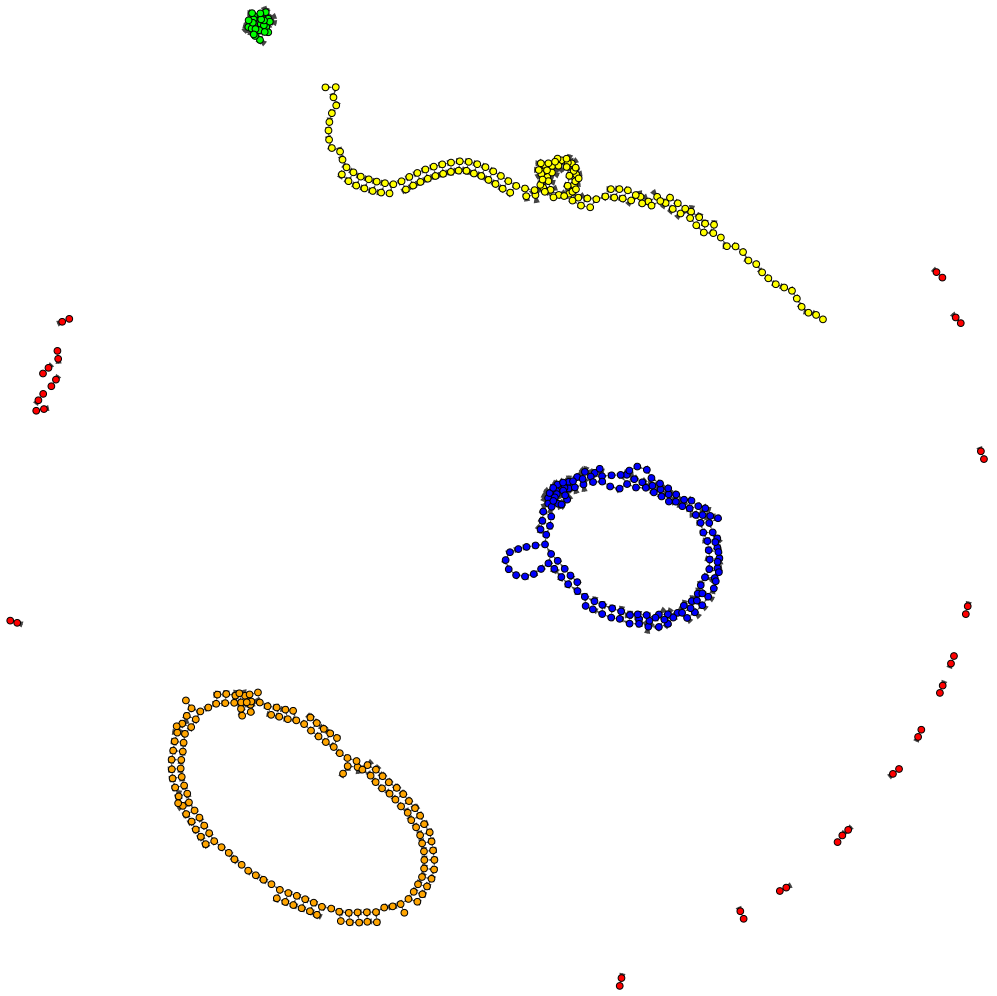
podmínek, které odpovídají reálným datovým sadám. Datová sada byla vygenerována s následujícími parametry:

- **délka genomu**: 1 Mb
- **počet opakování repetice**: 40
- **zastoupení repetice v genomu**: 58,4%
- **variace repetice**: 7 inzerce, 19 delece, 8 mutace
- **tandemové uspořádání repetic** : 25 %
- **délka readů**: 150
- **počet readů**: 1000
- **šum** : 5%
- **pokrytí genomu**: 0.15x

Chování assembleru na zadané datové sadě a dosažené výsledky jsou velmi zajímavé (tabulka 7.3). Program nalezl v genomu všechny vložené repetice. Ovšem ani jedna z nalezených repetic nebyla sestavena v původní délce. Tento výsledek testu bylo však možné očekávat vzhledem k volbě vstupních parametrů. Vysoká hodnota delecí a mutací v repeticích měla přímí vliv na možnost sestavení repetice v plné délce. Nízké pokrytí genomu a menší zastoupení repetitivní složky v genomu oproti předchozím testům mělo výrazný vliv na nalezení a sestavení repetic. Dlouhé LTR retrotranspozony nedosáhly sestavení celé své sekvence a nalezená sekvence byla vždy výrazně menší. Nejvíce byla zkrácena délka repetice HOBS (na obrázku 7.3 reprezentována žlutou barvou). Kromě již zmíněných faktorů měla na zkrácení této repetice vliv neuzavřenost struktury repetice, Nedošlo k úplnému pokrytí vnitřní oblasti repetice (vzniklo rozpojení struktury), které se projevilo při hledání cesty grafem - cesta byla sestavena pouze od počátku až po rozpojení. Na základě těchto výsledků, lze usoudit, že assembler nedosahuje příliš dobrých výsledků, pokud jsou repetice silně zatíženy indely, nebo pokud je hledaná repetice rozpojena.

7.4 Testování na reálných datech

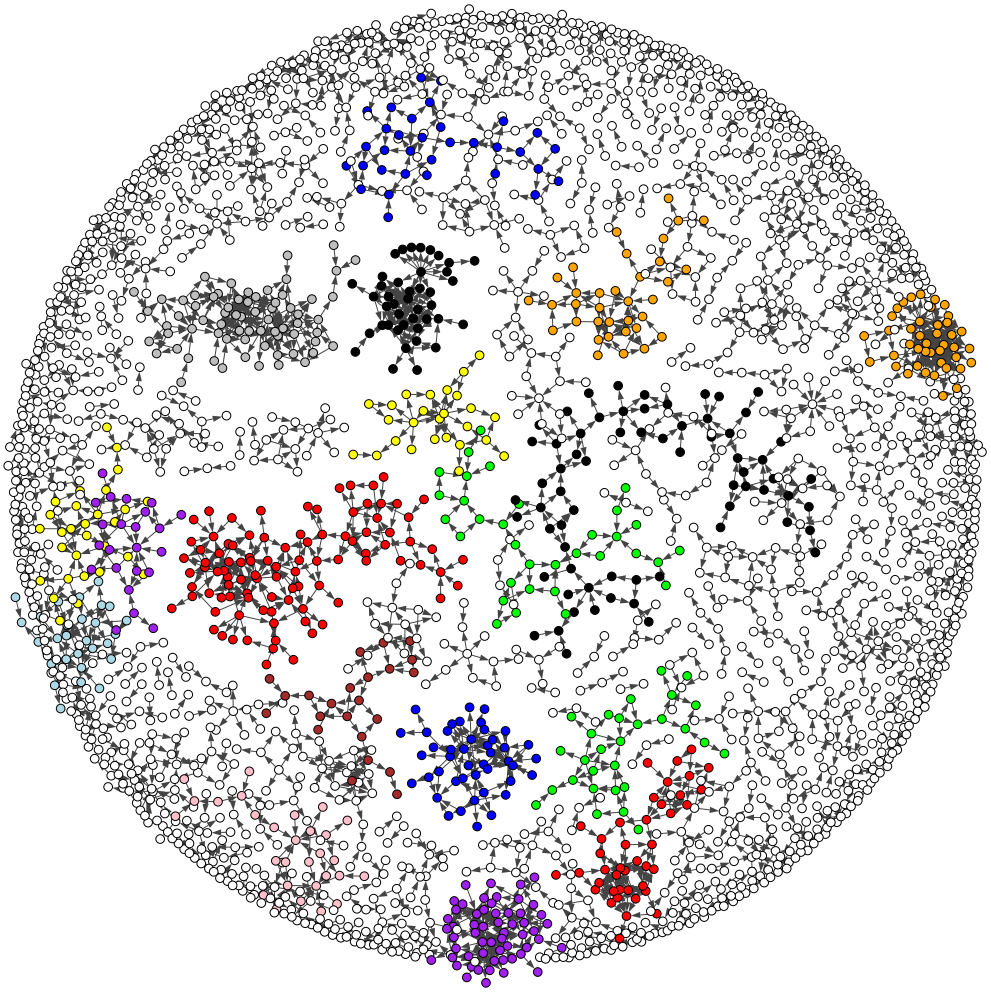
Předešlé testy byly prováděny na simulovaných datech, která mají svá omezení a nemohou zcela pokrýt libovolné variace repetic v genomu. Pro poslední testovaní bude proto použit



Obrázek 7.3: Graf překryvů pro testovací sadu číslo 3. Aplikace detekovala všechny struktury repetic obsažené v genomu: satelit GSAT_MM (zelená), ALISEI (modrá), HOBS (žlutá) a MAGGY (oranžová).

vzorek reálných dat, který prokazatelně obsahuje dostatečné množství LTR retrotranspozonů a bude možné vyzkoušet jejich detekci a sestavení. Byl zvolen datový set přiložený k článku [18], zabývajícím se studiem chromozomu Y rostliny silenky širokolisté (*Silene latifolia*). Bližší analýza obsahu genomu této rostliny zjistila přítomnost vysokého procenta LTR retrotranspozonů (přibližně 70% celkového genomu). Genom byl sekvenován nástrojem Illumina Nextera a bylo získáno celkem 3,114,716 readů a pokrytí genomu dosahovalo zhruba 66%. Testovat celý datový set by bylo komplikované i za předpokladu použití výpočetních zdrojů Metacentra (některé použité nástroje mají limitní omezení pro vstupní data). Ze získaných dat byl proto vybrán mnohem menší vzorek obsahující 10,000 readů (5,000 párových readů), na kterém byl implementovaný nástroj testován a výsledky jsou zobrazeny na obrázku 7.4. Assembler v přiložených datech detekoval a sestavil celkem 11 repetitivních elementů navzdory faktu, že byl vybrán velmi malý vzorek readů, který nedosahoval pokrytí ani 1% z celkového počtu readů. To lze tedy považovat za dobrý výsledek.

Posouzení kvality sestavení a identifikace repetic je však složitá, protože neexistuje přesný seznam repetitivních elementů, které se ve vybraném vzorku dat nacházejí.



Obrázek 7.4: Ukázka grafu překryvů při testování na reálných datech. V grafu jsou patrné struktury představující repetitivní elementy (odlišeny barevně) od zbylých nerepetitivních částí grafu (bílá barva). Z detekovaných struktur bylo sestaveno celkem 11 repetic.

Kapitola 8

Závěr

Tato diplomová práce byla zaměřena na implementaci nového nástroje pro detekci repetitivních elementů v genomu. Pro tvorbu práce bylo potřeba nastudovat a vytvořit stručný přehled současných metod klasifikace a sestavení repetitivních DNA OLC a DBG, spolu s dostupnými assemblery, které se touto problematikou zabývají. Oba přístupy byly důkladně porovnány s ohledem na hledání repetic v genomu. Na základě dostupných informací o vlastnostech jednotlivých metod sestavování, byl navržen nový nástroj, založený na přístupu OLC, pro vyhledávání satelitní DNA a LTR retrotranspozonů, které vytvářejí kruhovou strukturu v grafu překryvů. Aby bylo možné otestovat funkčnost assembleru, bylo nutné vytvořit programy pro generaci umělého genomu, obsahujícího nastavitelný počet a možné variace repetitivních prvků, a simulaci readů. Následně byly vytvořeny čtyři testovací sady, které měly ověřit funkčnost a vlastnosti implementovaného assembleru. Poslední testovací sada představovala reálná data, získaná při sekvenování genomu rostliny *Silene latifolia*.

Z výsledků testování je patrné, že assembler dokázal dobře vyhledat a sestavit repetitivní element vytvářející kruhovou strukturu. Délka a kvalita nalezené sekvencí se při jednotlivých testováních lišila. Hlavní vliv na dosažené výsledky měly především použité techniky pro detekci repetic v grafu a následné úpravy grafu překryvů (např. odstranění cyklů v grafu). Metoda `clusters()` pro hledání shluků v grafu se projevila účinná u silněji pokrytých genomů. Při nízkém pokrytí genomu, kdy nebylo zajištěno pokrytí celé délky repetice, metoda sice dokázala některé repetice rozpoznat v grafu a nicméně následné sestavení těchto elementů neposkytovalo vždy dobré výsledky. Další potencionální problém byl odhalen v případě odstraňování cyklů v grafu. Technika FAC odstranila z grafu nejmenší počet hran působící v grafu cyklicitu, kdy jako rozhodovací kritérium byla brána pouze váha hrany. Ve většině případů došlo k správnému odstranění hrany, představující unikátní sekvenci genomu, nicméně v některých případech, kdy repetice v sobě obsahovala dobře pokrytou dlouhou inzerci, vybrala metoda chybnou hranu, což mělo za následek úplné začlenění inzerce do sekvence repetice.

S ohledem na zjištěné nevýhody assembleru by další rozšíření algoritmů mělo být zaměřeno na jejich odstranění. Jednou z možností vylepšení aplikace je použití lepší techniky detekce repetic v grafu. Se zajímavým řešením detekce repetic přišel nástroj RepLong [7] (článek dostupný od 1.4.2018), který analyzuje vztahy příslušnosti vrcholů a na jejichž základě vytváří komunity, vzájemně si blízkých vrcholů.

Literatura

[1] Batzoglou, S.: ARACHNE. *Genome Research*, ročník 12, č. 1: s. 177–189, ISSN 10889051, doi:10.1101/gr.208902.
URL http://www.genome.org/cgi/doi/10.1101/gr.208902

[2] Burge, C.; Gifford, D.; Fraenkel, E.: Foundations of Computational Systems Biology. Massachusetts Institute of Technology: MIT OpenCourseWare, 2014.
URL
https://ocw.mit.edu/courses/biology/7-91j-foundations-of-computational-and-systems-biology-spring-2014/lecture-slides/MIT7_91JS14_Lecture6.pdf

[3] Denisov, G.; Walenz, B.; Halpern, A. L.; aj.: Consensus generation and variant detection by Celera Assembler. *Bioinformatics*, ročník 24, č. 8, 2008-04-11: s. 1035–1040, ISSN 1367-4803, doi:10.1093/bioinformatics/btn074.
URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btn074

[4] van Dijk, E. L.; Auger, H.; Jaszczyszyn, Y.; aj.: Ten years of next-generation sequencing technology. doi:10.1016/j.tig.2014.07.001.
URL http://linkinghub.elsevier.com/retrieve/pii/S0168952514001127

[5] El-Metwally, S.; Hamza, T.; Zakaria, M.; aj.: Next-Generation Sequence Assembly. doi:10.1371/journal.pcbi.1003345.
URL http://dx.plos.org/10.1371/journal.pcbi.1003345

[6] Gifford, D. K.: Genome Assembly. MIT Computer Science and Artificial Intelligence Lab, 2014.
URL
https://ocw.mit.edu/courses/biology/7-91j-foundations-of-computational-and-systems-biology-spring-2014/lecture-slides/MIT7_91JS14_Lecture6.pdf

[7] Guo, R.; Li, Y.-R.; He, S.; aj.: RepLong. *Bioinformatics*, ročník 34, č. 7, 2018-04-01: s. 1099–1107, ISSN 1367-4803, doi:10.1093/bioinformatics/btx717.
URL https://academic.oup.com/bioinformatics/article/34/7/1099/4596943

[8] Huang, X.: CAP3. *Genome Research*, ročník vol. 9, č. issue 9, 1999: s. 868–877, ISSN 10889051, doi:10.1101/gr.9.9.868.
URL http://www.genome.org/cgi/doi/10.1101/gr.9.9.868

[9] Langmead, B.: Overlap Layout Consensus assembly. John Hopkins Whiting School of Engineering Baltimore, 2012.
URL
http://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_olc.pdf

[10] Langmead, B.: De Bruijn Graph assembly. John Hopkins Whiting School of Engineering Baltimore, 2015.
URL
https://www.cs.jhu.edu/~langmea/resources/lecture_notes/assembly_dbg.pdf

[11] Li, Z.; Chen, Y.; Mu, D.; aj.: Comparison of the two major classes of assembly algorithms. *Briefings in Functional Genomics*, ročník vol. 11, č. issue 1, 2012-02-16: s. 25–37, ISSN 20412649, doi:10.1093/bfgp/elr035.
URL https://academic.oup.com/bfg/article-lookup/doi/10.1093/bfgp/elr035

[12] Liu, L.; Li, Y.; Li, S.; aj.: Comparison of Next-Generation Sequencing Systems. *Journal of Biomedicine and Biotechnology*, ročník vol. 2012, 2012: s. 1–11, ISSN 11107243, doi:10.1155/2012/251364.
URL http://www.hindawi.com/journals/bmri/2012/251364/

[13] López-Flores, I.; Garrido-Ramos, M.: The Repetitive DNA Content of Eukaryotic Genomes: s. 1–28. doi:10.1159/000337118.
URL http://www.karger.com/doi/10.1159/000337118

[14] Martínek, T.: Sestavování DNA. Vysoké učení technické Brno, Fakulta informačních technologií, 2017.
URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FBIF-IT%2Flectures%2Fsestavovani_dna.pdf&cid=10787

[15] Myers, E. W.; Hamza, T.; Zakaria, M.; aj.: The fragment assembly string graph. doi:10.1093/bioinformatics/bti1114.
URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bti1114

[16] Novak, P.; Neumann, P.; Pech, J.; aj.: RepeatExplorer. *Bioinformatics*, ročník vol. 29, č. issue 6, 2013-03-14: s. 792–793, ISSN 13674803, doi:10.1093/bioinformatics/btt054.
URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btt054

[17] Pop, M.; Salzberg, S.; Shumway, M.: Genome sequence assembly. *Computer*, ročník 35, č. 7, 2002: s. 47–54, ISSN 0018-9162, doi:10.1109/MC.2002.1016901.
URL http://ieeexplore.ieee.org/document/1016901/

[18] Puterova, J.; Kubat, Z.; Kejnovsky, E.; aj.: The slowdown of Y chromosome expansion in dioecious Silene latifolia due to DNA loss and male-specific silencing of retrotransposons. *BMC Genomics*, ročník 19, č. 1, 2018: s. –, ISSN 1471-2164, doi:10.1186/s12864-018-4547-7.
URL
https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-4547-7

[19] Shcherban, A. B.: Repetitive DNA sequences in plant genomes. *Russian Journal of Genetics: Applied Research*, ročník vol. 5, č. issue 3, 2015: s. 159–167, ISSN 20790597, doi:10.1134/S2079059715030168.
URL http://link.springer.com/10.1134/S2079059715030168

[20] Technologies, O. N.: MinION - Portable, real-time biological analyses. Oxford: Oxford Nanopore Technologies, 2016.
URL https://nanoporetech.com/products/minion

[21] Vollmers, J.; Wiegand, S.; Kaster, A.-K.; aj.: Comparing and Evaluating Metagenome Assembly Tools from a Microbiologist's Perspective - Not Only Size Matters! *PLOS ONE*, ročník 12, č. 1, 2017-1-18: s. e0169662–, ISSN 1932-6203, doi:10.1371/journal.pone.0169662.
URL http://dx.plos.org/10.1371/journal.pone.0169662

[22] Wicker, T.; Sabot, F.; Hua-Van, A.; aj.: A unified classification system for eukaryotic transposable elements. *Nature Reviews Genetics*, ročník 8, č. 12, 2007: s. 973–982.
URL http://www.nature.com/nrg/journal/v8/n12/abs/nrg2165.html#top

[23] Zerbino, D. R.; Birney, E.: Velvet. *Genome Research*, ročník 18, č. 5, 2008-02-21: s. 821–829, ISSN 1088-9051, doi:10.1101/gr.074492.107.
URL http://www.genome.org/cgi/doi/10.1101/gr.074492.107

[24] Zytnicki, M.; Akhunov, E.; Quesneville, H.: Tedna. *Bioinformatics*, ročník vol. 30, č. issue 18, 2014-09-04: s. 2656–2658, ISSN 13674803, doi:10.1093/bioinformatics/btu365.
URL https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btu365

[25] Šeda, O.; Liška, F.; Šedová, L.: Genetické haraburdí - repetitivní DNA. 2006.
URL http://biol.lf1.cuni.cz/ucebnice/repetitivni_dna.htm

Příloha A

Obsah CD

- `aplikace` – Adresář s implementací assembleru.
  - `data` – Vstupní data assembleru.
  - `output` – Výstup aplikace.
  - `src` – Zdrojové soubory assembleru.
  - *make_genom.sh* – Skript pro vytvoření genomu na MetaCentru.
  - *make_reads.sh* – Skript pro vytvoření readů na MetaCentru.
  - *make_overlaps.sh* – Skript pro vytvoření překryvů na MetaCentru.
  - *run_assembler.sh* – Skript pro spuštění assembleru na MetaCentru.
- `dokumentace` – Adresář se soubory dokumentace.
  - `doc` – Zdrojové soubory dokumentace.
  - `xhypsk01.pdf` – Dokumentace práce ve formátu pdf.
- `README.txt` – Soubor s popisem použití aplikace.