VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

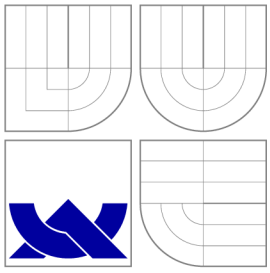
# OPTIMALIZACE ZAROVNÁNÍ DAT Z NEXT-GENERATION SEKVENOVÁNÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

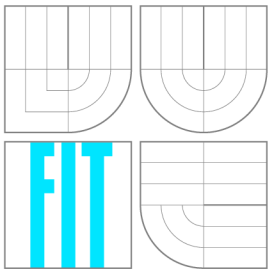
AUTOR PRÁCE
AUTHOR

Bc. VOJTĚCH ŠALANDA

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# OPTIMALIZACE ZAROVNÁNÍ DAT Z NEXT-GENERATION SEKVENOVÁNÍ

OPTIMIZATION OF THE NEXT-GENERATION SEQUENCING DATA ALIGNMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VOJTĚCH ŠALANDA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. IVAN VOGEL

BRNO 2014

# Abstrakt

Tato práce přináší možnost optimalizace nástrojů pro zarovnání sekvenčních čtení, která jsou produktem sekvenačních technik 2. generace. Výsledné zarovnání existujících nástroje lze ovlivnit různým nastavením parametrů. Za tímto účelem byl vytvořen optimalizační framework, který pomocí algoritmu diferenciální evoluce nalezne optimální nastavení zvolených parametrů. Cílem optimalizace je maximalizace přesnosti zarovnání. Funkčnost frameworku byla ověřena na datových sadách jak reálných, tak generovaných sekvenčních čtení. Díky přesnějšímu zarovnání lze získat přesnější podobu referenční sekvence pro navazující analýzy genetických vlastností.



# Abstract

This thesis presents the possibility of shorts DNA reads alignment tools optimization. These short DNA reads are products of Next-Generation Sequencing technologies. The results produced by existing alignment tools can be influenced by setting different values of parameters. For this purpose, it was developed an optimization framework to find the optimal values of selected parameters. This framework is based on Differencial Evolution algorithm and its main goal is to maximize the alignment accuracy. The functionality of the framework was tested on both real and generated data sets of short DNA reads. As a result of more accurate alignment, it can be obtained more exact reference sequence which is suitable for the subsequent analysis of genetic characteristics.



# Klíčová slova

Sekvenování nové generace (NGS), zarovnávací nástroje, sekvenční čtení, optimalizace zarovnání.

# Keywords

Next-Generation Sequencing, alignment tools, short read, alignment optimization.


# Citace

Vojtěch Šalanda: Optimalizace zarovnání dat z next-generation sekvenování, diplomová práce, Brno, FIT VUT v Brně, 2014

# Optimalizace zarovnání dat z next-generation sekvenování

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Ivana Vogela a uvedl jsem všechny literární prameny, ze kterých jsem čerpal.

.....
Vojtěch Šalanda
20. května 2014

## Poděkování

Vřele děkuji svému vedoucímu panu Ing. Ivanu Vogelovi za vedení při přípravě a vypracování této práce. Jeho důsledný přístup, připomínky a rady mi pomohly v řešení mnohých problémů.

# Obsah

# Kapitola 1

# Úvod

Součástí molekulárně-genetických metod analýzy biologického materiálu je sekvenování DNA. Existuje mnoho sekvenačních metod, jejichž společným vstupem je biologická struktura ve formě vlákna DNA/RNA s neznámou primární strukturou, což je posloupnost nukleotidů. Sekvenací a následným zarovnáním lze tuto sekvenci zrekonstruovat a zjistit posloupnost nukleotidů. Následnou analýzou lze získat cenné informace o genech, jejich funkcích nebo geneticky podmíněných chorobách. Uplatnění nachází tyto metody zejména v medicíně při studiu dědičných a nádorových onemocnění nebo v genetice při sestavování genomů organizmů, studiu vzájemné příbuznosti nebo identifikaci genů a kvantitativní analýze transkriptomů.

Next-generation sekvenování je skupina metod vyznačujících se masivním paralelismem a vysokou propustností, díky čemuž jsou tyto metody označovány jako HTS (*high-throughput sequencing*). Zavedení těchto technologií přineslo nové možnosti ve zkoumání nukleotidových sekvencí uvnitř buňky. Výrazně poklesly náklady, rozšířilo se pole působnosti a nové přístroje umožnily zkoumat DNA/RNA sekvence v prokaryotických i eukaryotických buňkách. Problematika HTS technologií bude popsána ve 2. kapitole.

Výsledkem činnosti sekvenačních přístrojů je množina krátkých sekvenčních čtení (*reads*), jejichž vlastnosti a různé typy budou popsány ve 3. kapitole.

Díky produkci obrovského množství dat a existenci různých sekvenačních technologií je třeba vyvíjet bioinformatické nástroje pro zpracování těchto dat produkovaných většinou komerčně využívanými přístroji. Základním bioinformatickým problémem je správné mapování těchto přečtených sekvencí na referenční sekvenci nebo sestavení původní sekvence *de novo*. V současné době je možno si vybrat z 60 různých nástrojů, přičemž mnoho z nich není starších než 7 let a jejich vývoj probíhal současně s vývojem sekvenačních technik. Ve 4. kapitole budou popsány algoritmy a datové struktury, které jsou používány při mapování, možnosti optimalizace a speciální zarovnání.

Cílem zarovnávacích nástrojů je především maximalizace přesnosti ať už v úloze anotace genomových sekvencí nebo v případě studia výskytů mutací a polymorfismu. Výslednou přesnost těchto nástrojů lze ovlivnit nastavením různých parametrů. Hlavním úkolem této práce je návrh a implementace optimalizačního frameworku, který za pomocí evolučních optimalizačních technik nalezne optimální hodnoty zvolených parametrů s cílem maximalizovat přesnost zarovnání. Základním požadavkem je možnost aplikace na libovolný zarovnávací nástroj. V 5. kapitole bude popsán proces výběru zarovnávacích nástrojů pro testování frameworku.

6. kapitola je věnována implementaci jednotlivých částí frameworku, realizaci diferenciální evoluce jako nástroje optimalizace a popisu použitých formátů souboru pro uložení

sekvence, sekvenčních čtení a zarovnání.

V 7. kapitole je pak popsán proces vytvoření datových sad jak generovaných, tak reálných sekvenčních čtení. Výsledky experimentů provedených nad těmito datovými sadami jsou zobrazeny v podobě tabulek a grafů. Součástí této kapitoly je i popis možných navazujících postupů a experimentů, které je možné díky snadné rozšiřitelnosti optimalizačního frameworku realizovat.

# Kapitola 2

# Next-generation sekvenování

Existující sekvenační metody lze dělit podle společných znaků do několika skupin. První skupinou jsou metody 1. generace, jejichž zástupcem je pomalá Sangerova metoda. Dříve byla tato metoda dominantní na trhu téměř 20 let a významně se podílela na sestavení kompletního lidského genomu v roce 2000 [6]. V průběhu této doby byla metoda vylepšována, avšak stále vykazovala jistá omezení, která zapříčinila vývoj sofistikovanějších metod a vývoj Sangerovy metody byl zastaven po určitém nízkém počtu vylepšení. Sangerova metoda je založena na sekvenování určitého úseku, který vždy končí stejným typem nukleotidu (A/C/G/T).

Další skupinou jsou metody 2. generace. Tyto nové metody uskutečňují strategie založené na kombinaci přípravy templátu, sekvenování a následného zobrazení, genomového zarovnání k referenční známé sekvenci a metod sestavení genomu z nasekvenovaných fragmentů. Hlavním cílem využití těchto metod je zrychlení sekvenačního procesu, možnost paralelní sekvenace a snížení nákladů, které byly v případě Sangerovy metody velmi vysoké. Metody NGS (*next-generation sequencing*) produkují obrovské množství sekvenčních čtení, což jsou krátké nasekvenované úseky DNA. U některých přístrojů je uvedena dokonce miliarda čtení v jednom běhu. Tyto NGS metody nachází uplatnění zejména v opakovaném znovusestavení lidského genomu, což vede k detailnějšímu a kvalitnějšímu studiu genetických změn a jejich vlivu na nemoci a zdraví člověka. Schopnost nasekvenovat celý genom nejen u člověka, ale i příbuzných organismů přispěla k rozsáhlým srovnávacím evolučním studiím, které byly doposud nerealizovatelné. Na trhu existují sekvenační přístroje, které se specializují na konkrétní problémy. Každá metoda a přístroj vykazuje různé vlastnosti a při aplikaci na konkrétní problém i různou přesnost, což vede ke srovnávání metod, hodnocení a stanovení vhodnosti jejich použití.

Dále existuje ještě třetí skupina metod 3. generace, jejímž představitelem je metoda Nanopore [3]. Vývoj v této oblasti dále pokračuje a cílem je možnost nasekvenování lidského genomu za 100 dolarů a s tím související vývoj personální genomiky [28]. Při vyšetření pacienta by byl k dispozici jeho genom, který by napomáhal určení diagnózy. Navzdory velkému zájmu o dosavadní pokroky a cíle této techniky není Nanopore významně komerčně využíván, a proto jsou metody NGS stále dominantními na trhu sekvenačních technologií.

V této kapitole budou popsány NGS techniky a přístroje Illumina/Solexa, Roche/454 a SOLiD/LifeTechnologies. Názvy jednotlivých přístrojů nesou názvy firem, které je vyvinuly. Důležitými aspekty v oblasti zarovnání, které je věnována tato práce, jsou délka a kvalita nasekvenovaných sekvenčních čtení. Zároveň bude zmíněn pokrok těchto metod, jejich limity a předpokládaný budoucí vývoj.

## 2.1 Příprava templátu

Prvním společným krokem všech NGS metod je příprava templátového vzorku. Templát je označení vlákna, ke kterému se bude během sekvenačního procesu vytvářet komplementární vlákno. Pro jeho přípravu existují dva používané přístupy:

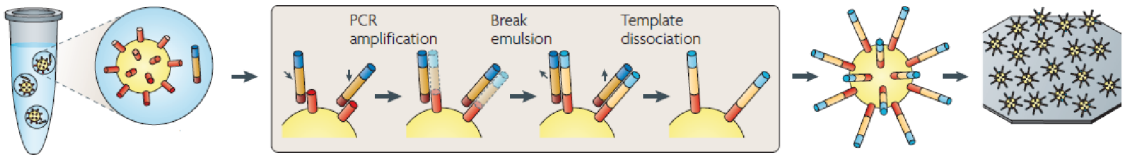
- zesílení templátu pomocí techniky PCR,
- použití jedné molekuly jako templátu.

Současné NGS metody štěpí genomovou DNA na náhodných místech na krátké fragmenty. Fragmenty lze použít pro přípravu jak jednoduchých templátů, tak mate-pair templátů, u kterých je třeba více genetického materiálu. Templát je připevněn a fixován na pevném povrchu, což umožňuje miliardám jinak sekvenčních reakcí probíhat paralelně a tím mnohonásobně zrychlit celý sekvenační proces.

### 2.1.1 Zesílení templátu

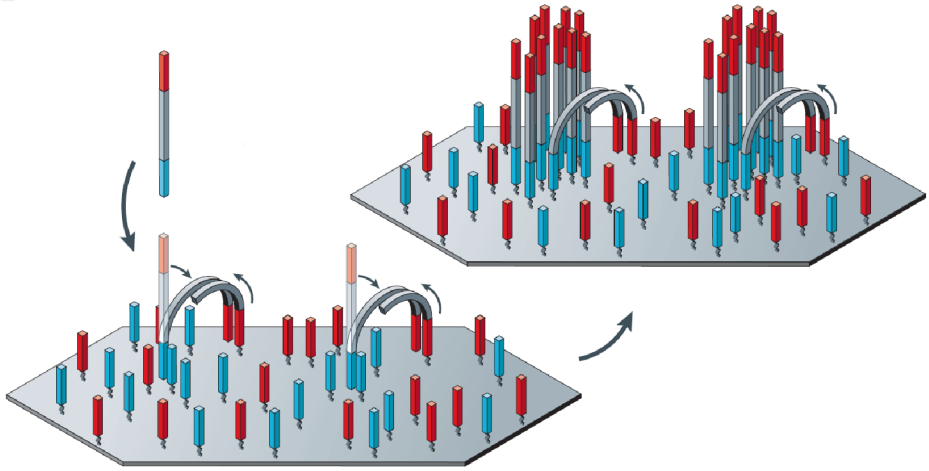
Většina systémů pro zobrazování sekvencí není navržena za účelem rozpoznání konkrétního nukleotidu na základě jediného signálu. Proto je nutné zesílit templát, což zesílí i výsledný signál a usnadní jeho rozpoznání.

První ze dvou standardních metod je technika emulzní PCR (emPCR) [7]. Její výhodou je použití v nebuněčném prostředí, čímž se lze vyhnout ztrátám některých genomových sekvencí. Tyto ztráty jsou nejčastějším problémem při klonování uvnitř bakteriální buňky. Nejprve se vytvoří knihovna jednovláknových fragmentů. Dále jsou jednotlivé fragmenty na obou koncích obohaceny o adaptory sestávající z univerzálních vazebných míst pro primery. Dvoušroubovice je v další fázi rozpletena do dvou samostatných vláken při denaturaci. Každé jednotlivé vlákno DNA je zachyceno na chromatinové kuličce, která má na svém povrchu primery. Dále následuje amplifikace tak, že k zachycenému vláknu se vytvoří vlákno komplementární od každého primeru na povrchu kuličky. Na sklíčku jsou v poslední fázi pomocí chemické reakce uspořádány jednotlivé kuličky do pravidelné mřížky a připraveny k dalšímu použití v rámci NGS. V případě sekvenace párových sekvenčních čtení je templátem úsek dvouvláknové DNA (viz kapitola 3). Postup zesílení templátu je znázorněn na obrázku 2.1.




Obrázek 2.1: Technika PCR v krocích [39].

Druhou metodou je zesílení *solid-phase* [11], které využívá přístroj Illumina. Na pevném povrchu sklíčka jsou ve vysoké hustotě rozmístěny přímé a reverzní primery. K primeru je dále navázán templát, který má být zesílen. V místě navázání templátu dojde k jevu zvanému *bridge amplification*. Existence dvou druhů primeru pro navázání opačných konců templátu umožňuje vznik mostu podle obrázku 2.2. K navázanému templátu se vytvoří komplementární vlákno. Dvě vlákna jsou oddělena a proces se opakuje. Po jeho skončení vzniká 100-200 milionů shluků v místech navázání původního templátu.



Obrázek 2.2: Zesílení templátu metodou *solid-phase* [39].

### 2.1.2 Jednomolekulový templát

Metody použití jednomolekulového templátu využívají jednodušší a přímočařejší postup. Množství potřebné genomové DNA je menší než $1\,\mu g$, což je výhodou oproti daleko náročnějším metodám zesílení templátu. Další výhodou je absence techniky PCR, díky které mohou vzniknout nežádoucí nukleotidové mutace. Zesílením tak může vzniknout zkreslený výsledek, což působí problémy při zarovnání či znovusestavení původní sekvence nebo při RNA-seq [53], kdy dochází ke snížení reprezentativnosti mRNA molekul.

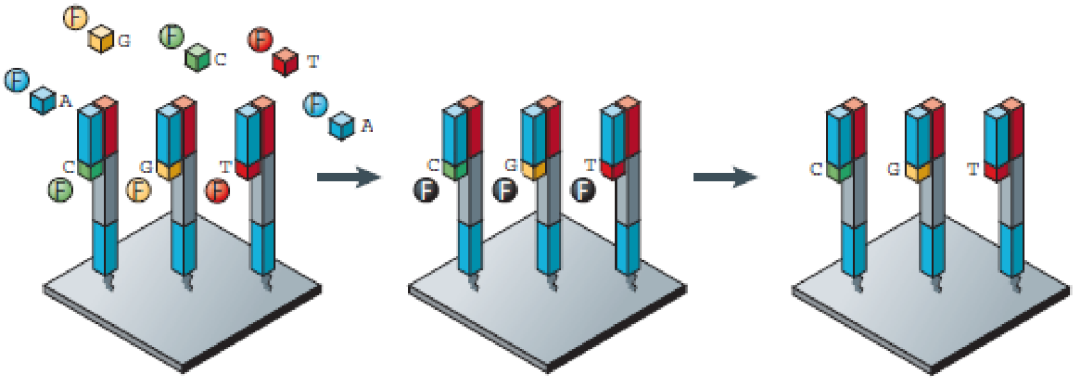
V první fázi dojde k fixaci jednotlivých primerů na pevný podklad. Fragmenty o velikosti 200-250 bází jsou obohaceny adaptory a hybridizací připojeny k primerům. Po navázání DNA polymerázy dochází k vytvoření komplementárního vlákna DNA k templátu. Druhou variantou je fixace jednotlivých templátů k pevnému povrchu a navázání volných primerů. V obou případech je možné na destičce rozmístit několik miliard těchto dvojic vlákno-primer. Třetí způsob spočívá ve fixaci DNA polymerázy k pevnému povrchu [8]. Na fixovanou molekulu jsou navázány dvojice vlákno-primer. Tímto přístupem lze sekvenovat templáty o délce až 10 000 bází, čehož lze využít při real-time sekvenování.

## 2.2 Sekvenování a zobrazení

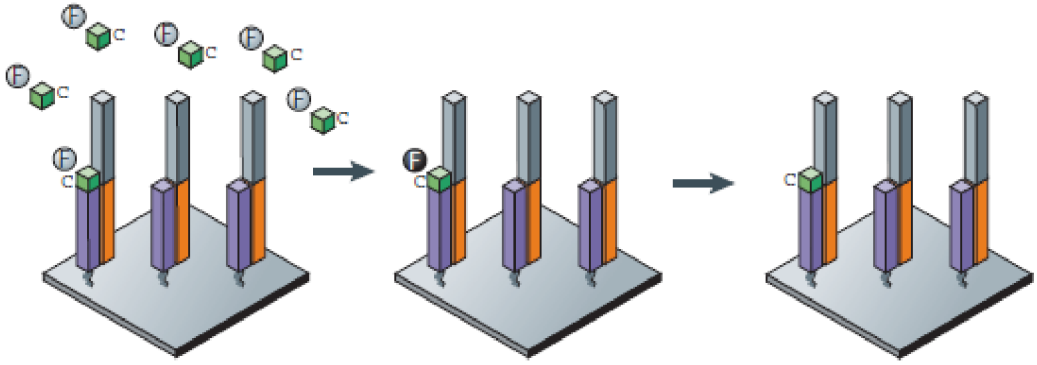
Po vytvoření a případném zesílení templátu je dalším krokem sekvenace. Při zesílení templátového vzoru vzniká populace identických templátů, z nichž každý prošel sekvenačním procesem. Při zobrazení je pozorován zesílený signál, který vzniká součtem signálů ze shodných nukleotidů. Vznik jednoho signálu z více dílčích klade důraz na účinnost procesu, eliminace mutací a dokončení celého procesu. V případě, že proces sekvenace skončí předčasně, nedojde k vytvoření celého komplementárního vlákna a výsledné sekvenční čtení má nízkou kvalitu. Všechny zmíněné poruchy zvyšují fluorescenční šum nebo zkrácení čtení [9]. U druhého typu k těmto chybám nedochází, ale templáty jsou v každém cyklu náchylné na adice nukleotidů či sond. Tzv. shášející efekt (*dephasing*), který vzniká mezi barvami sousedních molekul nebo při začlenění určité tmavé sondy, zase způsobuje delece ve výsledném čtení.

### 2.2.1 Sekvenace terminací (CRT)

Metoda CRT spočívá v použití fluorescenčně modifikovaných nukleotidů, které fungují jako terminátory sekvenace [38]. DNA polymeráza je navázána na dvojici templát-primer. V každém kroku dojde k navázání právě jednoho fluorescenčně modifikovaného nukleotidu, který je komplementární k prvnímu volnému templátovému nukleotidu ve směru sekvenace. Dojde k terminaci díky chemické modifikaci navázaného nukleotidu. V dalším kroku dojde k odstranění chemické fluorescenční značky, která identifikuje daný nukleotid. Pro zachování správného a kontrolovaného chodu jsou jednotlivé fáze zakončeny fází odplavení nepotřebného materiálu. Celý cyklus může být buďto jednobarevný podle obrázku 2.4, kdy je v roztoku právě jeden typ nukleotidu, nebo čtyřbarevný podle obrázku 2.3, kdy jsou v roztoku všechny 4 typy nukleotidů.




Obrázek 2.3: Sekvenace přístroje Illumina/Solexa se čtyřbarevným cyklem [39].




Obrázek 2.4: Sekvenace přístroje Helicos/BioScience s jednobarevným cyklem [39].

Čtyřbarevný cyklus je použit v přístroji Illumina/Solexa a jednobarevný používá Helicos/BioScience. Použité reverzibilní terminátory, což jsou zmíněné modifikované nukleotidy, se používají ve dvou variantách s 3' blokovaným a 3' neblokovaným koncem.

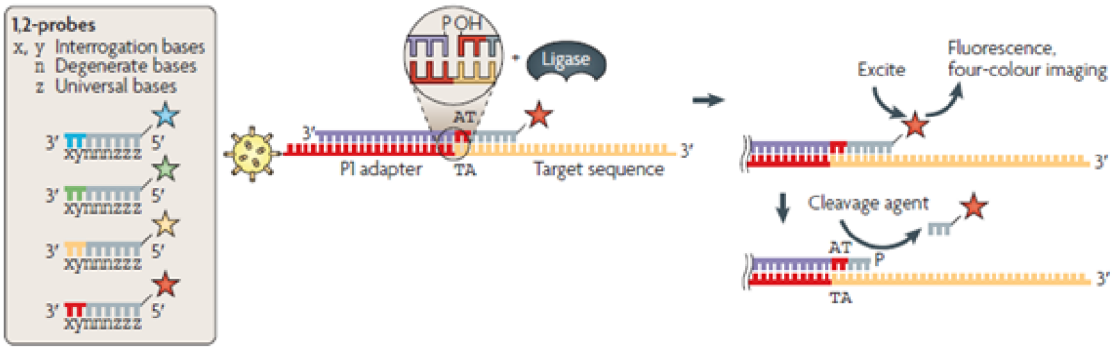
V případě 3' blokovaného konce je třeba štěpení dvou chemických vazeb. Tím se odštěpí fluorescenční barvivo a dojde k obnově původní -OH skupiny, čímž se uvolní vlákno pro další cyklus. Čtyři barvy jsou detekovány úplným vnitřním fluorescenčním odrazem (TIRF) za

pomocí dvou laserů. Nejčastější chybou je záměna guaninu za jiný typ nukleotidu, další problémy může způsobit zesílení AT-bohatého či CG-bohatého templátu, což vede ke zkreslení výsledného zobrazení. Při detekci SNVs (*single nucleotide variants*) dochází k 99 % shodě s existujícími SNVs, falešná shoda u nově objevených SNVs dosahuje minimálně 2,5 %. Výsledky pochází ze zarovnání pomocí bioinformatických nástrojů MAQ [33] a ELAND [13].

Nevýhodou je složitý proces modifikace 3' konců, který vyžaduje nutnost mutovat i DNA polymerázy. Naproti tomu 3' neblokovaný konec spočívá v připojení malé chemické sloučeniny. Vzniklý terminátor je rozpoznán všemi DNA polymerázami [54] a používá se ve dvou typech podle druhu modifikace. V obou případech dochází k terminaci syntézy a obě strategie jsou účinnější než varianta s 3' blokovaným koncem.

### 2.2.2 Sekvenace ligázou (SBL)

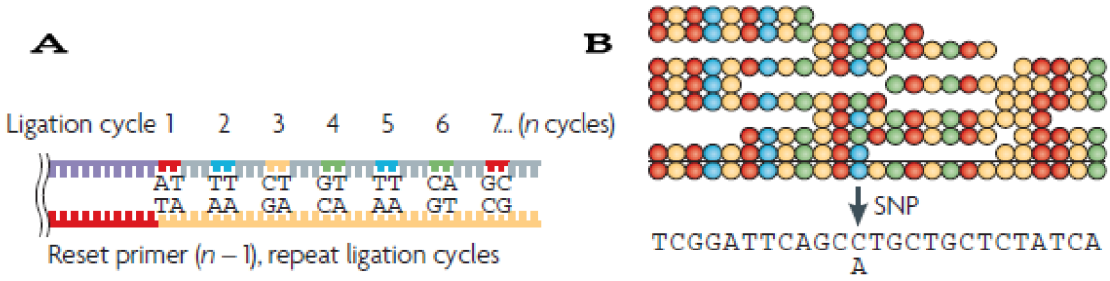
Použití DNA ligázy je alternativní způsob k nejčastěji používané metodě CRT, která používá DNA polymerázu a reverzibilní terminátory. Enzym DNA ligáza slouží ke spojování krátkých DNA sond do delších celků při replikaci buňky. V případě SBL jsou fluorescenčně označeny jednonukleotidové nebo dvounukleotidové sondy. Tyto sondy jsou navázány ke komplementárnímu vláknu na volnou pozici za primerem. Zbylé sondy jsou odplaveny. Dále se podle fluorescenčního značení identifikují připojené sondy [26]. Proces pokračuje odstraněním fluorescenční skupiny nebo připojením nového primeru. Sekvenace je znázorněna na obrázku 2.5.




Obrázek 2.5: Sekvenace pomocí DNA ligázy přístroje SOLiD/Life [39].

Po skončení jednoho kola je nutné posunout místo navázání primeru, aby se nasekvenovala i místa, která v předchozím kole tvořila primer. Počet kol je stejný jako délka primeru, jak ukazuje obrázek 2.6.

Využití DNA ligázy při sekvenování je jednou z vlastností přístroje SOLiD/Life [52]. V tomto případě je použito dvounukleotidových sond, jejichž výhodou je schopnost opravovat chyby v jednotlivých kolech a možnost detekce SNVs. Templát je zesílen technikou emPCR. V rámci jednoho kola se provede deset připojení sondy. Ve druhém kole dojde k připojení primeru délky o jedna kratší a pokračuje se stejným způsobem. Při následném barevném dekódování lze detekovat dvě pozice současně a díky více kolům je možné opravit vzniklé chyby. Barevné výsledky jsou poté zlinearizovány a výsledek je zarovnán k referenční sekvenci podle obrázku. K přečtení kompletní sekvence je nutné provést nejméně 5 kol, což je ovlivněno délkou použitých sond.

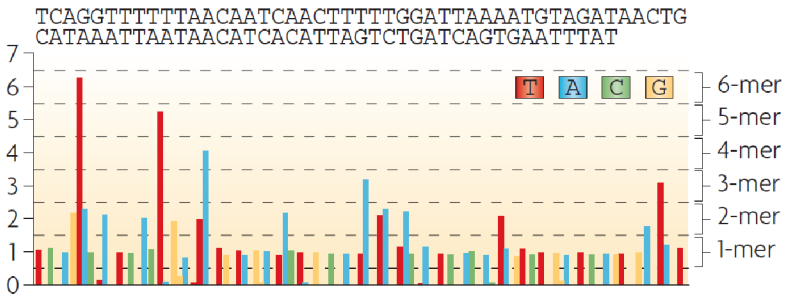



Obrázek 2.6: Naznačení cyklů přístroje SOLiD (A), rekonstrukce DNA sekvence, označeno místo polymorfismu (B) [39].

Výhodou přístroje SOLiD je vysoká kvalita výsledných sekvenčních čtení. Jejich délka je však pouze 50 bází, což závisí na počtu cyklů v rámci jednoho kola. Krátká čtení mohou působit problémy při následném zarovnání zejména v regionech s častými repeticemi. Problém AT-bohatých a CG-bohatých sekvencí je stejný jako v případě CRT.

### 2.2.3 Sekvenace adicí

Sekvenování pomocí adice je označováno jako pyrosekvenování (*pyrosequencing*). Hlavním rozdílem oproti původním přístupům je použití nemodifikovaných nukleotidů. Modifikace se týká DNA polymerázy, která je obohacena o přesné množství nukleotidů dNTP (*deoxyribonucleotide triphosphate*). DNA polymeráza připojí za primer komplementární dNTP a její činnost je ukončena. Cyklus je obnoven při dodání další dNTP. Pořadí a intenzita světelných efektů jsou zaznamenány do tzv. flowgramu, ze kterého lze určit primární strukturu podle obrázku 2.7.




Obrázek 2.7: Sekvenace adicí přítroje Roche/454 [39].

Příkladem využití metody pyrosekvenování je přístroj Roche/454. Příprava templátu probíhá pomocí techniky emPCR, následně je 1-2 miliony kuliček rozmístěno v PTP (*PicoTiterPlate*) jamkách, mezi kterými je snížena interakce, což zvyšuje kvalitu výsledku. V ideálním případě je v každé jamce pouze jedna kulička. Jednotlivé dNTP jsou distribuovány do PTP jamek v předem určeném pořadí. Pomocí kamery jsou snímány světelné signály, které jsou produktem enzymatické reakce. Počet dNTP je přímo úměrný síle výsledného signálu pro až 6 homopolymerních repetic (6 po sobě jdoucích stejných nukleotidů).

Výhodou této platformy je vhodnost použití při mate-pair sekvenování, protože nepotřebuje dvojnásobnou dobu na rozdíl od jiných metod. Nejčastější chybou během pyro-

sekvenování je inserce či delece. Délka výsledných čtení se pohybuje průměrně kolem 330 bází.

## 2.3 Zarovnání a sestavení

Po dokončení sekvenačního procesu jsou dvě možnosti dalšího pokračování. V případě, že není známa referenční sekvence, dochází k sestavení *de novo*, což znamená, že výsledkem bude původní genom, který vznikne sestavením jednotlivých čtení za sebou, přičemž se využije vzájemného překrytí pro stanovení pořadí. Druhá možnost vyžaduje znalost referenční sekvence, ke které budou nasekvenovaná čtení zarovnána. Konkrétní strategie je určena na základě biologické aplikace, nákladů, úsilí a časové úvahy. Zarovnání k referenční sekvenci se provádí za účelem objevení nových a potvrzení již objevených SNVs. U nově objevených SNVs je nutná následná validace. Důležitým cílem je detekce genových variací za účelem zjištění příbuznosti různých organismů.

Zarovnání sekvenčních čtení k referenční sekvenci má své limity, které jsou způsobeny výskytem častých repetic. V případě pokusu zarovnat čtení k cizí referenční sekvenci je vysoká pravděpodobnost, že toto zarovnání selže [13]. K řešení problému repetic lze použít čtení typu mate-pair nebo pair-end (viz kapitola 3). Kromě informace o primární sekvenci dvou úseků obsahují ještě dodatečnou informaci o jejich vzdálenosti v referenční sekvenci, což vede k přesnějšímu zarovnání.

# Kapitola 3

# Sekvenční čtení

Výsledkem činnosti všech sekvenačních přístrojů je knihovna nasekvenovaných sekvenčních čtení. Jak vyplývá z předchozí kapitoly, jedná se o fragmenty různých délek od 30 bází do více než 1000 bází, u kterých je známá primární struktura DNA/RNA. Součástí vstupu je i kvalita jednotlivých čtení, která je definována pro každý nasekvenovaný nukleotid. Délka sekvenčních čtení a jejich kvalita v různých místech je specifická pro jednotlivé NGS platformy. Existují tři základní kategorie používaných sekvenčních čtení podle typu sekvenace:

- single-end čtení,
- mate-pair čtení,
- pair-end čtení.

Jednotlivé kategorie se liší výsledkem a množstvím informace o čtení, popřípadě o dvojici čtení v případě posledních dvou kategoriích. Některé platformy NGS se specializují pouze na jednu kategorii a jejich cílem je množina co nejdelších sekvenčních čtení vysoké kvality.

Nezávisle na výběru typu čtení je možné sekvenovat různá biologická data, která lze rozdělit do tří základních kategorií:

- DNA,
- RNA,
- další sekvence (proteiny).

V této práci je kladen důraz zejména na první kategorii, která zahrnuje sekvenaci genomové DNA v buňkách zkoumaného organismu. Do druhé kategorie spadá sekvenace transkriptomu, což je soubor všech genových RNA produktů. Příkladem je metoda RNAseq [43]. Do třetí skupiny patří další možné sekvence, jako jsou proteiny, které jsou analyzovány na základě interakce s DNA (metoda ChIPseq).

## 3.1 Single-end čtení

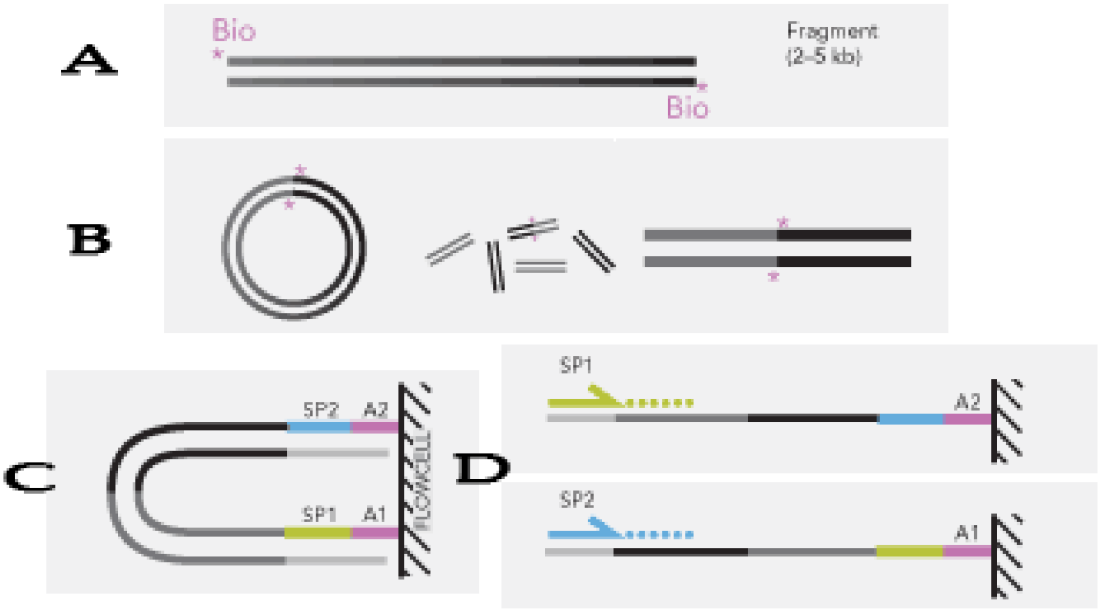
První kategorii tvoří nejjednodušší typ sekvenčních čtení. Jedná se o fragmenty standardní, výše zmíněné délky, která je ovlivněna použitou platformou a s ní související technologií. Každé čtení nese informaci o primární struktuře jednovláknové DNA/RNA a kvalitě nasnímaného signálu pro každý nukleotid. Jeho výhodou je jednoduchost vytvoření, protože

používá standardní metody sekvenace (viz kapitola 2). Tato čtení lze použít jak při zarovnání, které je v tomto případě jednodušším procesem než v případě párových čtení, tak při sestavení *de novo*. Výhodou je tedy jednoduchost při dalším zpracování.

Nevýhodou je možná nejednoznačnost při zarovnání i sestavení *de novo*. V případě příliš krátkých čtení může dojít k vícenásobnému zarovnání. Tento aspekt se zejména při sestavení *de novo* může projevit neúplnou rekonstrukcí hledaného genomu. K těmto problémům dochází zejména v případě, že nasekvenovaný genom obsahuje větší množství repetic.

## 3.2 Mate-pair čtení

Knihovna *mate-pair sequencing* je vytvořená ze sekvenčních čtení, které obsahují inserty o velikosti 2-5 kb. Jedná se o typ párových čtení, které kromě nasekvenované oblasti obsahují dodatečnou informaci o velikosti insertu. Tato čtení a jejich knihovny jsou využitelné v mnoha biologických aplikacích, jako je sestavení *de novo*, detekce strukturální variability při genomovém zarovnání nebo dodatečná kontrola při zarovnání k obecně referenční sekvenci. Lze kombinovat data z knihovny mate-pair s druhým typem párových čtení (pair-end), čímž vzniká silná kombinace dlouhých čtení, které nesou dostatečné informace pro přesné zarovnání nasekvenovaných čtení k referenčnímu genomu nebo jeho sestavení *de novo*.




Obrázek 3.1: Průběh sekvenování mate-pair čtení: fragmentace (A), cirkularizace a vytvoření fragmentů v okolí spoje (B), vytvoření shluku (C), sekvenování čtení (D) [19].

Proces vytváření mate-pair čtení začíná fragmentací vlákna DNA podle obrázku 3.1. Vzniknou tak fragmenty o velikosti 2-5 kb, což je v porovnání se single-end čteními více než desetinásobek délky. Takto vzniklé fragmenty jsou potom na jednom konci označeny biotinem upravenými dNTP (nukleotidy). Pomocí těchto dNTP a chemické reakce mezi konci dojde k cirkularizaci fragmentů, podle obrázku. Fragmenty, které nevytvoří uzavřené smyčky, jsou odstraněny. Následně jsou v okolí spojů z těchto kruhových čtení odštěpeny

fragmenty délky 400-600 bází a je odstraněna biotinová značka uprostřed nich, jak je znázorněno na obrázku 3.1 (B). Tyto kratší sekvence jsou dále obohaceny adaptory na obou koncích a probíhá proces zesílení pomocí techniky PCR. Průběh sekvenace je znázorněn na obrázku 3.1 (D). Po vzniku shluku je nasekvenováno první vlákno, poté je vytvořen nový shluk a je nasekvenováno i druhé vlákno [19].

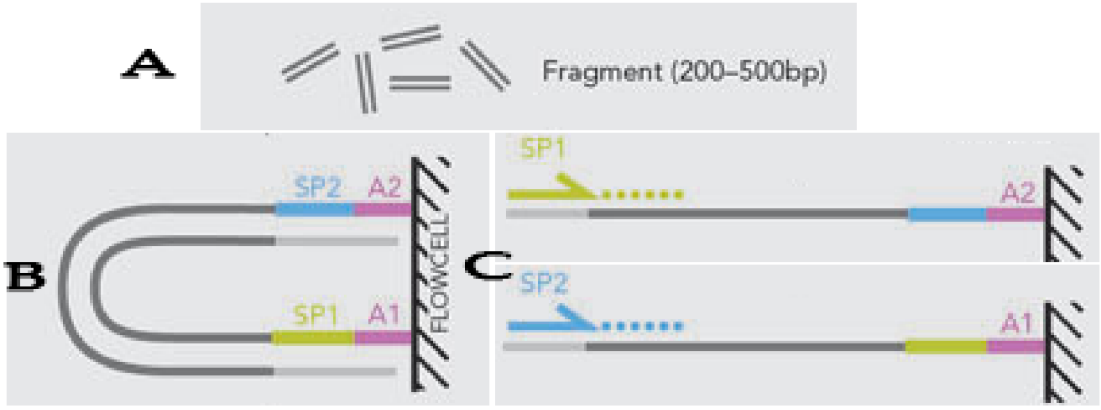
Každé čtení je tedy definován jako dvě části sekvence, které byly nasekvenovány po vzniku shluku technikou PCR, a délkou fragmentu, která se pohybuje od 2 kb do 5 kb. Při následném zarovnání lze těchto informací využít ke zvýšení citlivosti a přesnosti výsledku [19].

Výhodou je efektivní protokol, který není závislý na konkrétním genomu ani použité next-generation platformě. Další výhodou jsou nízké požadavky na množství materiálu, které jsou okolo 10 $\mu g$ pro jeden běh přístroje.

## 3.3 Pair-end čtení

Obdobou mate-pair knihoven je knihovna párových čtení druhého typu s názvem pair-end. Princip spočívá ve vytvoření fragmentů o velikost 200-500 bází, což je v porovnání s fragmenty mate-pair asi desetkrát méně. Princip zarovnání je stejný jako v případě mate-pair. Při tomto procesu se kromě mapování nasekvenované části využívá informace o velikosti insertu. V případě pair-end čtení se jedná o insert asi desetkrát menší než v případě mate-pair [20].

Vytvořené fragmenty jsou obohaceny adaptory. Pomocí techniky PCR dojde k zesílení a vzniku shluků podle obrázku 3.2 (B). V rámci shluků probíhá sekvenační proces nejprve z prvního konce. Po obnovení shluku se nasekvenuje sekvence z druhého konce, jak je znázorněno na obrázku 3.2 (C). Délka nasekvenovaných částí je z každé strany typicky 75 bází, tedy jedno párové čtení sestává z 2 krát 75 bází a dodatečné informace o velikosti insertu 200-500 bází. Při jednom běhu sekvenačního přístroje vznikne až 200 milionů sekvenčních čtení.




Obrázek 3.2: Průběh sekvenování pair-end čtení: fragmentace (A), vytvoření shluku (B), sekvenování čtení (C) [20].

Kromě zmíněné výhody dodatečné informace vykazuje párové sekvenování další výhody. Pokud se jedná o množství DNA ve zkoumaném vzorku, potřebuje párové sekvenování stejné

množství jako single-end sekvenování. Nevyžaduje methylaci DNA a lze jej použít při více biologických aplikacích. Díky použití pair-end čtení lze dosáhnout vyšší kvalitu zarovnání.

## 3.4 Shrnutí

Výsledek sekvenace je knihovna sekvenčních čtení, což jsou nasekvenované fragmenty určité délky, která koresponduje s použitou NGS technologií. Kromě jednoduchých single-end čtení lze vytvořit také párová čtení, která kromě informace o primární struktuře úseku DNA obsahují ještě informaci o velikosti insertu mezi dvěma čteními v páru. Velikost insertu pair-end čtení je asi desetkrát menší než v případě čtení mate-pair.

# Kapitola 4

# Zarovnání sekvenčních čtení k referenční sekvenci

Navazujícím krokem je zpracování vstupních dat a jejich zarovnání či sestavení *de novo* pomocí bioinformatický nástrojů (mapperů). Zarovnání sekvencí je obecně jedním z nejdůležitějších bioinformatických problémů. Hnací silou vývoje maperů je obrovské množství dat, které produkují HTS technologie firem Illumina, Roche, SOLiD nebo Helicos a které je řádově v Gbp na jeden přístroj za den. V důsledku těchto podmínek bylo rychle zjištěno, že nestačí dosavadní nejlepší zarovnávací nástroje na tak velké množství dat, což tedy vedlo k velkému nárůstu počtu mapperů na trhu. Tyto nástroje byly navrženy tak, aby se dokázaly vyrovnat s vlastnostmi a omezeními, kterými lze charakterizovat jednotlivé HTS technologie, jako jsou krátká sekvenční čtení (Illumina, SOLiD, Helicos), nehomogenní kvalita čtení na celé jeho délce (Illumina), speciální kódování sekvenčních čtení (SOLiD), nízká pravděpodobnost inserce (Illumina) nebo nízká pravděpodobnost substituce (Helicos). Nástroje pro zarovnání krátkých sekvenčních čtení jsou obecně rychlejší a přesnější.

V následující kapitole budou popsány jednotlivé algoritmy a mapovací techniky spolu se zástupci z řad mapperů, jejichž principy jsou postaveny na těchto algoritmech. Algoritmy lze srovnávat podle jejich časové a paměťové složitosti v porovnání s citlivostí při zarovnání jednotlivých sekvenčních čtení. V poslední části budou popsány přístupy v zarovnání různých typů čtení.

## 4.1 Mapovací techniky

Problém mapování HTS dat lze definovat takto:

| Název proměnné | Popis proměnné |
|---|---|
| $Q$ | množina sekvenčních čtení |
| $S$ | množina referenčních sekvencí |
| $k$ | prahová hodnota pro porovnání |
| $d$ | funkce vzdálenosti, $d : Q \times S \rightarrow R$ (reálná čísla) |

Problém spočívá v nalezení $\forall q \in Q$ všech podřetězců $m \in S$, které splňují všechna omezení a zároveň platí následující vztah:

$$d\left(q, m\right) \leq k \tag{4.1}$$

Výsledkem je množina $M_q = \{m \mid d(q, m) \leq k\}$, která se nazývá *matches*. Omezující podmínky úzce souvisí s použitou HTS technologií a použitým datovým typem. Datový typ je dán typem sekvenace (viz kapitola 3). Hlavním cílem je nalézt správnou pozici $\forall q \in Q$, kde $Q$ obsahuje velké množství prvků. V rámci tohoto nalezení je třeba brát v úvahu chyby během HTS procesu vzniku množiny $Q$ a strukturální variace. Kvůli těmto požadavkům je třeba zvolit správnou distanční funkci, jejímž výsledkem bude aproximační přiřazení. Taková funkce většinou počítá shody (*matches*) a neshody (*mismatches*) v odpovídajících si nukleotidových bázích po namapování, vložení nukleotidu nebo začlenění mezery (*gap*) s pravděpodobností spojenou s výskytem v sekvenčních čtení.

## 4.2 Rozdělení algoritmů

Pro optimální řešení problému mapování jsou navrženy rychlé algoritmy, které jsou nejčastěji založeny na vytváření pomocný datových struktur. Tyto struktury se nazývají indexy a tvoří se jak pro referenční sekvenci, tak pro množinu sekvenčních čtení. Někdy se tvoří pro obojí současně. Na základě vlastností indexů lze rozdělit algoritmy do dvou skupin:

- algoritmy s hashovací tabulkou,
- algoritmy se sufixovými stromy.

## 4.3 Algoritmy s hashovací tabulkou

Základem těchto algoritmů je použití hashovací tabulky a hashovací funkce. Hashovací tabulka neboli tabulka s rozptýlenými položkami je datová struktura, ve které se využívá hashovací funkce pro přístup k uloženým položkám. Hashovací funkce transformuje klíč na index do primárního (rozptylového, hashovacího) pole. V tomto poli se následně vyhledává sekvenčním způsobem. Základními požadavky na hashovací funkci jsou rychlost a vytváření co nejmenšího počtu kolizí [17].

Příkladem nástrojů, jejichž princip je založený na hashovací funkci, je nástroj BLAST [2]. Algoritmus nástroje BLAST a jeho variant je založen na principu zarovnání menší části sekvenčního čtení, které se nazývá semínko (*seed*). Následně dochází k rozšíření zarovnání od místa namapovaného semínka. BLAST uchovává v hashovací tabulce pozici každého $k$-meru, což je subsekvence o délce $k$. Konstanta $k$ má standardně hodnotu 7-11 pro zarovnání DNA sekvence a hodnotu 2-3 pro sekvence proteinů. Každý $k$-mer je klíčem pro vyhledávání v hashovací tabulce, kde lze nalézt všechny možné výskyty $k$-meru v sekvenci. Výsledné zarovnání je hodnoceno tzv. skórovací maticí, která přiřazuje ohodnocení jednotlivým kombinacím v páru. Příkladem skórovací matice je BLOSUM62 [40], kterou používá program BLAST. Tato skórovací matice vznikla na základě empirických údajů získaných při zarovnání datasetu sekvencí, které vykazovaly 62 % podobnost. Nejlépe hodnocené dvojice odpovídajících si nukleotidů se označují zkratkou HSP (*high-scoring pairs*).

Po nalezení pozice semínka následuje jeho rozšíření oběma směry bez akceptace mezer a spojování dílčích rozšířených úseků do větších celků podle principu *seed-and-extend*. Toto rozšiřování je ukončeno v případě, že celkový počet HSP klesne pod stanovenou hodnotu prahu. Tímto je ukončena první fáze, která nebere v úvahu výskyt mezer. Ve druhé fázi dojde ke zpřesnění celkového skóre pomocí algoritmu Smith-Waterman [49]. Výsledkem je zpřesněné, statisticky významné lokální zarovnání.

Základní BLAST algoritmus byl vylepšen a přizpůsoben různým typům zarovnání. Následující techniky popisují mapování datasetu krátkých sekvencí na velkou genomovou sekvenci stejného biologického druhu.

### 4.3.1 Vylepšení seedingu

Pojmem *seeding* se označuje proces lokalizace semínka na referenční sekvenci. Původní algoritmus BLAST vyžadoval, aby bylo nalezeno 11 po sobě následujících shod pro DNA sekvenci. V případě výskytu 10 % mutace, kterou lze simulovat jako chybně nasekvenovaný nukleotid na každé desáté pozici, však není možné zarovnat semínko o délce 11 a zarovnání bude neúspěšné. Tento problém řeší daleko citlivější přístup prostorově rozšířeného semínka *spaced seed*. V tomto případě je místo požadavku na přesnou shodu 11 po sobě jdoucích nukleotidů použita binární šablona shody. Například šablona 111010010100110111 vyžaduje také 11 shod nukleotidů, které však nemusí nutně následovat po sobě, ale jsou vyžadovány v místě, které v šabloně odpovídá „1“. V místě, kde je v šabloně „0“, není shoda vyžadována. V porovnání s $k$-merem se šablonou 11111111111 je zarovnání dvou sekvencí se 70 % podobností zvýšena citlivost o 55 % [37]. Tento vylepšený *spaced seed* je specifikován váhou, která udává počet shod, což je v tomto případě 11. Nevýhodou tohoto přístupu je větší a složitější hashovací tabulka a jí odpovídající hashovací funkce.

První program, který využil pro zarovnání krátké sekvence prostorově rozšířeného semínka, byl Eland. Konkrétně pracoval se šesti šablonami prostorového semínka, které pokrývaly celé krátké sekvenční čtení. Na základě použití těchto šablon bylo garantováno, že budou identifikovány dvě neshody v zarovnání vždy alespoň jednou ze šablon. Zároveň platí, že nezáleží na místě výskytu těchto neshod v zarovnání. Mapper SOAP [34] převzal tuto strategii s tím rozdílem, že indexuje referenční genom místo sekvenčního čtení. Další zlepšení přinesly nástroje SeqMap [21] a MAQ [33], které rozšiřují tuto metodu povolením $k$ neshod v zarovnání. V souvislosti s tímto nárůstem ale potřebují $\binom{2k}{k}$ šablon, což vede k exponenciálnímu nárůstu počtu šablon v závislosti na proměnné $k$. MAQ garantuje nalezení 2 neshod zarovnání v prvních 28 bázích každého čtení, což je nejkvalitnější úsek sekvenčního čtení, které produkuje přístroj Illumina. Tato vlastnost zlepšuje částečnou shodu při nalezení potenciálního mapovacího místa.

RMAP [47], který je založen na Baeza-Yates-Perleberg algoritmu, používá odlišnou sadu binárních šablon. Snižuje potřebný počet šablon pro detekci $k$ neshod zarovnání na $k + 1$. Nárůst šablon je v tomto případě lineární a detekce neshod je stále dostatečně efektivní. Nicméně v důsledku snížení počtu šablon je snížena jejich váha. Tato strategie není vhodná v kombinaci s hashovací tabulkou, protože existuje mnoho kandidátů míst, kam lze namapovat odpovídající sekvenční čtení.

Zlepšení přišlo v podobě nalezení optimální cesty pro hledání minimálního počtu prostorových semínek na základě známé délky sekvenčních čtení a požadavku na citlivost a využití paměti. Příkladem je program ZOOM [36], který za pomoci 5 binárních šablon s váhou 14 identifikoval všechny možné dvojice neshod zarovnání na sekvenčních čteních délky 32 bází. Pro srovnání, RMAP používá 3 šablony s váhou 10, již zmíněný Eland používá 6 šablon s váhou 16, ale kvůli redukci nároků na paměťový prostor používá pro indexaci v hashovací tabulce pouze 12,5 bází. Vzhledem k tomu, že časová složitost algoritmu je (aproximačně) přímo úměrná váze $q$, počtu šablon $m$, počtu sekvenčních čtení $n$ a délce genomu $L$, ZOOM má lepší teoretickou složitost díky omezení paměti.

Je velmi náročné držet v paměti RAM celou hashovací tabulku pro $q > 15$, kde $q$ je váha šablony. Homerova verze [16] hashovací tabulky měla dvouúrovňové indexační schéma

pro jakékoliv delší $q$. Tabulka byla zkonstruována pro jakoukoliv délku šablony $j$ takovou, že $j < q$, kde $q$ je typicky 14. Pro nalezení klíče délky $q$ a více bylo nutné vyhledat podle klíče délky $j$ a dále byla pomocí binárního vyhledávání nalezena cílová pozice v seznamu pozic pro konkrétní klíč délky $j$. Výsledná časová složitost byla logaritmická, což je oproti běžné konstatní složitosti jen o něco málo horší. Navíc byla velikost tabulky nezávislá na velikosti $q$. Podobný přístup byl zvolen pro Eland a MAQ algoritmus, ale v tomto případě bylo použito indexování sekvenčních čtení místo genomu.

Další algoritmy této skupiny většinou využívají prostorově rozšířené semínko s odlišnými šablonami, které jsou specifické pro referenční genom a citlivost mapování. Všechny tyto aplikace jsou příčinou velké oblíbenosti těchto semínek při mapování krátkých sekvenčních čtení.

### 4.3.2 Q-gram filtr a vícenásobné zarovnání

Potenciální problém s běžnými i prostorově rozšířenými semínky je problém mezer, protože mezery není možno uvnitř semínek nijak reprezentovat. Mezery jsou často nalezeny později pomocí techniky dynamického programování, což je rozšiřující krok základního algoritmu. Jinou alternativou detekce mezer je vytvoření možných mezer na každé pozici semínka [34]. Q-gram filtr [5], který je implementován například v nástroji SHRiMP [46], poskytuje jedno z možných řešení pro vytváření indexu, který umožňuje mezery. Tento filtr je založen na pozorování, že při výskytu zarovnávané sekvence (*query*) délky $w$ s nanejvýš $k$ rozdíly (mezera či neshoda v zarovnání) existuje $(w + 1) - (k + 1) * q$ podřetězců délky $q$, které sdílí tato sekvence s ostatními podřetězci délky $w$ v databázi [22]. Q-gram filtr je stejně jako technika rozšířených semínek silně závislý na rychlosti vyhledávání v hashovací tabulce. Tyto metody se liší v tom, že v případě použití semínek je namapováno pouze jeden semínko, které je následně rozšiřováno, zatímco v případě q-gram filtru je namapováno většinou více kratších semínek. Možnost vícenásobného namapování semínek na sekvenci je častou vlastností tzv. *capillary* mapperů, jejichž zástupcem je nástroj BLAT. U těchto mapperů se jedná se o hlavní techniku urychlení zarovnání delších sekvenčních čtení.

### 4.3.3 Rozšiřování semínek

V souvislosti s použitím dostatečně dlouhých prostorově rozšířených semínek v porovnání s délkou zarovnávaného čtení je možné problematiku rozšiřování semínek zanedbat. Není také třeba provádět vícenásobné zarovnání bez mezer. Provádí se pouze první krok zarovnání podle šablony semínka, který je ve srovnání s dynamickým programováním mnohem rychlejší a jednodušší. Nicméně i tak bylo v této oblasti dosaženo několik optimalizačních pokroků oproti původní verzi BLAST. Největší přínosem v oblasti rychlosti zarovnání bylo rozšíření standardního algoritmu Smith-Waterman pomocí vektorizace. Základní myšlenkou bylo paralelní zarovnání pomocí vektorového počítače, což je architektura SIMD (*Single Instruction Multiple Data*). V jednom cyklu CPU je pak možno zpracovat více částí zarovnávané sekvence současně. Příkladem instrukcí jsou SSE nebo MMX instrukce, přičemž pomocí SSE2 instrukcí, které jsou součástí x86 CPU, došlo ke zrychlení algoritmu Smith-Waterman až na desetinásobek původní rychlosti [10]. Této výhody bylo využito v nástrojích Novoalign nebo SHRiMP.

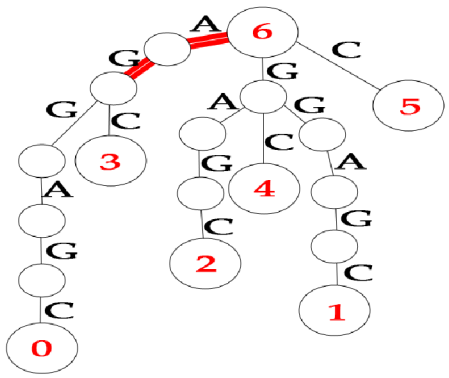
Dalšího zlepšení bylo dosaženo pomocí dynamického programování v okolí semínek. Vylepšení se stalo součástí prvního kroku mapování semínka. Výsledkem byla redukce celého prostoru, protože již nebylo nutné zbytečně prohledávat daleko vzdálené oblasti od místa namapovaného semínka. Navíc bylo zjištěno [42], že krátká sekvence může být zarovnána

v celé své délce k cílové sekvenci délky $L$ s maximálně $k$ neshodami nebo mezerami s časovou složitostí $O(kL)$. To znamená, že nezáleží na délce zarovnávané sekvence. Tento objev pomohl zrychlit zarovnání, které bylo omezováno pomalým dynamickým programováním.

## 4.4 Algoritmy se sufixovými stromy

Algoritmy založené na sufixových nebo prefixových stromech (*suffix and prefix tries*) zpřesňují problém spárování zarovnávané sekvence s konkrétním místem na referenční sekvenci. Standardně se jedná o proces sestávající z 2 kroků. Prvním krokem je identifikace přesného spárování. Ve druhém kroku se pomocí přesného spárování vytvoří nepřesné (obsahující chyby) zarovnání. K nalezení a identifikaci přesného spárování se používá přesné reprezentace sufixových nebo prefixových stromů. Příkladem takové reprezentace je sufixový strom, rozšířené sufixové pole [1] nebo FM-index [12]. Výhoda využití stromů spočívá v zarovnání k více identickým místům referenční sekvence současně, protože tyto identické podřetězce lze ve stromu nalézt při procházení od kořene k listu. To je velká výhoda oproti hashovací tabulce, protože pro zarovnání sekvenčních čtení s identickými podřetězci určité délky se ve stromě provádí pouze jedno prohledávání. Metody následného zpřesnění zarovnání jsou nezávislé na výběru datové struktury v předchozím kroku. Lze tedy přesné zarovnání provést pomocí struktury FM-index a následné zpřesnění pomocí sufixového stromu.

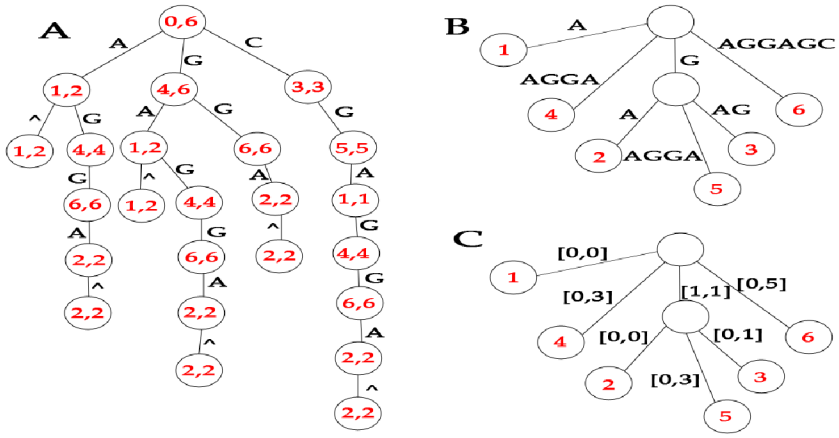
Sufixový strom je datová struktura, která uchovává všechny sufixy daného řetězce, čímž poskytuje rychlé spárování řetězců. Konstrukce sufixového stromu pro řetězec AGGAGC je naznačena na následujícím obrázku. Pro každý sufix zkoumaného řetězce se vytvoří cesta od kořene k listu, kde každý uzel odpovídá indexu v řetězci a na každé hraně je jeden z nukleotidů A/C/G/T. Červeně je naznačena cesta pro vyhledání pozice podřetězců AG. Z obrázku je zřejmé, že se jedná pozice 0 a 3 v referenční sekvenci. Při konstrukci se postupuje od nejkratších sufixů k nejdelším.




Obrázek 4.1: Sufixový strom s naznačeným vyhledáním řetězce AG.

Časová složitost vyhledání ve stromu je v případě existence přesné shody zarovnávané sekvence (čtení) a podřetězce v referenční sekvenci $O(n)$, kde $n$ označuje délku sekvenčního čtení. Z toho vyplývá, že časová složitost nezávisí na délce referenční sekvence, což umožňuje použití i v případě zarovnání k rozsáhlým genomům. Prostorová složitost sufixového stromu je $O(L^2)$, což vykazuje kvadratickou závislost na délce referenční sekvence a paměť je v tomto případě velmi neefektivně využita. Do paměti RAM by se v takovém případě nedal vytvořit ani index bakteriálního genomu, který má řádově tisíce bází.

Přílišnou paměťovou složitost redukuje komprimovaná varianta prefixového stromu, který se od sufixového stromu liší konstrukcí pro sufixy reverzního řetězce. Pro oba typy stromů lze však aplikovat stejné algoritmy pro vyhledání shody, proto jsou ve svém použití rovnocenné. Na obrázku 4.2 (A) je znázorněn prefixový strom v nekomprimované variantě pro řetězecAGGAGC. Číslo v uzlech udává indexy do pole sufixů. Tyto indexy odpovídají takovým sufixům, jejichž prefixem je prefix původního řetězce. Na obrázcích 4.2 (B) a 4.2 (C) jsou znázorněny komprimované varianty prefixového stromu pro řetězec AGGAGC. V případě obrázku 4.2 (B) jsou na hranách podřetězce a na obrázku 4.2 (C) jsou pouze intervaly podřetězců referenční sekvence.




Obrázek 4.2: Prefixové stromy: prefixový strom nekomprimovaný (A), komprimované varianty (B, C) [32].

Teoretická paměťová složitost komprimovaného sufixového stromu je $L * \log_2 L + O(L)$ v případě využití operace *rank-selection* [41], avšak i nejlepší bioinformatické nástroje potřebují pro reprezentaci jednoho nukleotidu 12-17 bytů paměti, čili stále je reprezentace lidského genomu v paměti nemožná. Zástupcem nástrojů používajících sufixový strom je MUMmer [25].

Tento problém byl nakonec vyřešen konstrukcí již zmíněného sufixového pole a rozšířeného sufixového pole. Konstrukce sufixového pole spočívá v seřazení všech sufixů referenčního řetězce lexikograficky do pole, přičemž prvním řetězce v poli je vždy prázdný řetězec označený „$“.

```
6 $AGGAGC
5 C$AGGA G
4 GC$AGG A      SA = (6,3,0,5,2,4,1)
3 AGC$AG G  BWT = (CG$GGAA)
2 GAGC$A G
1 GGAGC$ A
0 AGGAGC $
```

Obrázek 4.3: Sufixové pole a Burrows-Wheelerova transformace [32].

Rozšířené sufixové pole sestává ze standardního sufixového pole a několika pomocných polí. Požadavky na paměť jsou 6,25 bytů na 1 nukleotid. Je zachována lineární časová složitost vyhledávání přesné shody a současně je časová složitost nižší než v případě standardních sufixových polí. Na principu rozšířených sufixových polí jsou postaveny nástroje

Vmatch [1] a Segemehl [14]. S cílem ještě více snížit paměťovou náročnost a zefektivnit celé vyhledávání byla zavedena struktura FM-index, která je založena Burrows-Wheelerovu transformaci (BWT) [4]. Na obrázku 4.3 je naznačena transformace pro referenční sekvenci AGGAGC. Při lexikografickém seřazení sufixů, které jsou cyklicky rozšířeny na celou sekvenci, se do pole BWT zapíše vždy poslední znak. Velkým přínosem je skutečnost, že lze nalézt v prefixovém stromě uzel potomka z uzlu rodiče v konstantním čase za pomoci zpětného prohledávání v této struktuře. Časová složitost vyhledávání přesné shody je stejná jako v případě stromů. V oblasti paměťové složitosti je výhodou návrh FM-indexu jako komprimované datové struktury. Velikost v paměti může být menší než referenční sekvence v případě, že obsahuje repetice. Ve skutečnosti FM-index není komprimovaný kvůli dosažení vyššího výkonu během zarovnání. Abeceda řetězce DNA je velmi malá, proto není třeba komprimace. V praxi při použití FM-indexu je třeba 0,5-2 byty na každý nukleotid, což závisí na konkrétní implementaci a na vstupních parametrech. Díky těmto nízkým paměťovým požadavkům lze reprezentovat index celého lidského genomu na 2-8 GB. Mezi mappery, které používají FM-index, patří zejména Bowtie a novější verze Bowtie2 [27], BWA [29] a SOAP2 [35]. Zejména nástroje Bowtie patří díky nízké náročnosti a vysoké rychlosti k nejpoužívanějším.

Hledáním vhodné datové struktury pro reprezentaci DNA zarovnání je problém, který je obecně řešen i mimo oblast bioinformatiky. V literatuře jsou popsány rozsáhlé teorie [44] zarovnání řetězců, speciálně zaměřené na krátké řetězce. Nicméně tyto tradiční zarovnávací algoritmy hledají zejména přesné shody, což je v případě zarovnání DNA sekvencí odsunuto na druhé místo za rychlost, která je preferována především.

### 4.4.1 Zpřesnění zarovnání u sufixových/prefixových stromů

V navazujícím zpřesnění zarovnání není důležité, který z vyjmenovaných přístupů je použit pro zarovnání s přesnou shodou. Stejně jako v prvním kroku je i zde nejrozšířenější strukturou FM-index. Nástroje MUMmer a Vmatch nejprve zarovnají kratší úseky na základě maximálního počtu unikátních shod (MUMs), shod a repetic. Pomocí zarovnání s mezerami (*gapped*) jsou spojovány krátké úseky s přesnou shodou (*exact match*). Podobný přístup využívá nástroj Segemehl, kde je zarovnání iniciováno nejdelší prefixovou shodou u každého sufixu a dále je zarovnání dokončeno pomocí zarovnání s mezerami a je redukován počet nesprávných zarovnání.

Mezi další přístupy patří vzorkování referenční sekvence pomocí průchodu stromem od kořene k listu (*top-down*). Následné zarovnání je realizováno metodou dynamického programování nebo pomocí zavedení heuristik. Příkladem heuristického zarovnání jsou mappery Bowtie a BWA. Hlavním rozdílem oproti dynamickému programování je možnost omezit počet neshod v zarovnání a mezer a také vyžadují zarovnání celého sekvenčního čtení, proto průchod stromem může být omezený na prohledávání jen do určité úrovně. Alternativním postupem je prohledání všech možných kombinací neshod a mezer, což vede ke zcela přesnému zarovnání.

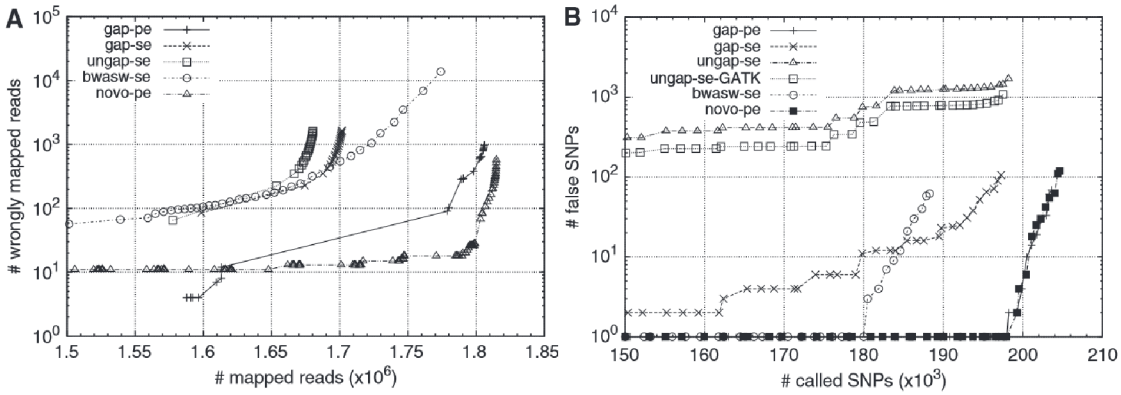
## 4.5 Zarovnání s využitím vlastností sekvenčních čtení

Zmíněné přístupy jsou obecně navržené postupy při zarovnání krátké sekvence k referenční sekvenci. Vzhledem k pokročilým sekvenačním metodám HTS a novým typům sekvenčních čtení vykazují i zarovnávací nástroje specifické vlastnosti pro dostatečně citlivé zarovnání těchto čtení.

### 4.5.1 Zarovnání s mezerami

Původní krátká sekvenční čtení produkovaná přístroji Illumina a SOLiD byla velmi výhodná při zarovnání bez mezer. Zarovnání s mezerami (*gapped*) je však neúměrně pomalejší a velmi neefektivní v případě nástrojů založených na strategii *seed-and-extend*. V průběhu vývoje však docházelo k prodlužování sekvenčních čtení a zarovnání s mezerami našlo své uplatnění. Stále je však výpočetně velmi náročné a ne ve všech případech je jeho použití nutné.

Podle grafu A na obrázku 4.4 je zřejmé, že při použití zarovnání s mezerami je citlivost o několik procent vyšší než při zarovnání bez mezer. Ovšem přesnost ve smyslu redukce počtu chyb v zarovnání zůstala stejná. Porovnávají se křivky gap-se a ungap-se, které odpovídají zarovnání single-end čtení.




Obrázek 4.4: Srovnání citlivosti zarovnání s mezerami a bez mezer [32].

Daleko větší význam má zarovnání s mezerami v případě studia SNPs [45]. Zarovnání bez mezer je velmi náchylné k falešné lokalizaci SNPs způsobené nižší citlivostí, jak vyplývá z obrázku 4.4 (B). Falešné SNPs v případě zarovnání s mezerami vzniká v blízkosti neodhalené mezery. Efektivní algoritmy, jako jsou BWA a Novoalign, používají zarovnání s mezerami. Jejich použití je vhodné při studiu genomových variant a k potlačení nesprávného zarovnání sekvenčních čtení.

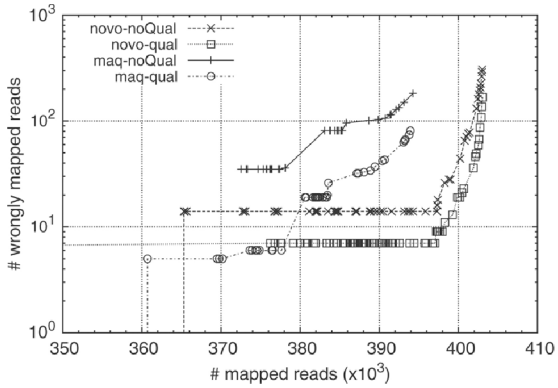
### 4.5.2 Využití párových sekvenčních čtení

Párová sekvenční čtení obsahují dodatečnou informaci o velikosti insertu, kterou lze při zarovnání využít. Velikost insertu je vzdálenost dvou čtení tvořících pár. Díky této vlastnosti se lze vyhnout chybnému zarovnání sekvenčních čtení k referenční sekvenci. V porovnání se zarovnáním single-end čtení lze dosáhnout vyšší citlivosti a specifičnosti, jak je znázorněno na obrázku 4.4 (A). Křivka označená *novo-pe* odpovídá zarovnání nástrojem Novoalign a křivka *(un)gap-pe* odpovídá zarovnání párových čtení nástrojem BWA. Analogicky křivka *bwasw-se* odpovídá nástroji BWA-SW [30] a *ungap-se-GATK* je zarovnání s použitím GATK pro snížení počtu falešných SNPs. Z grafu A vyplývá, že křivka *novo-pe* překonává křivku *gap-pe*. Přesnost určení SNPs je v tomto případě stejná.

Při zarovnání párových čtení se standardně postupuje obdobným způsobem jako u nepárových sekvenčních čtení. V první fázi se nezávisle zarovnají obě části bez ohledu na jejich reálnou vzdálenost na vlákně DNA. Ve druhé fázi se provede kontrola vzdálenosti s určitou odchylkou, zda je možné takové mapování akceptovat. Velikost odchylky je možné zadat explicitně, jak je tomu například u mapperu Bowtie.

### 4.5.3 Využití Quality Score při zarovnání

Při hodnocení zarovnání sekvenčních čtení se využívá skórovací funkce, která se aplikuje na každý nukleotidový pár v zarovnání. Výsledkem je skóre, které udává míru správnosti zkoumaného zarovnání. Vhodným nastavením prahové hodnoty, jak je tomu například u programu LAST [24], lze regulovat přesnost zarovnání, tedy zarovnání s nižším skórem jsou automaticky odfiltrovány. Podle studie v článku [48] bylo zjištěno, že použití Quality Score při zarovnání zlepšuje výslednou přesnost zarovnání. Tato metoda je založena na znalosti pravděpodobnosti výskytu chybné báze v sekvenčním čtení určité délky. Neshoda v zarovnání je ohodnocena nižší penalizační hodnotou v případě vyšší pravděpodobnosti výskytu chyby v sekvenčním čtení. Z grafu na obrázku 4.5 je zřejmý rozdíl v počtu nesprávně namapovaných sekvenčních čteních při aplikaci pravděpodobnostního modelu Quality Score a při jeho absenci.




Obrázek 4.5: Srovnání počtu nesprávně zarovnaných sekvenčních čtení při použití Quality Score [32].

Při tomto zarovnání bylo použito párových sekvenčních čtení délky 51 bází nasimulovaných pomocí nástroje MAQ z lidského genomu. Pravděpodobnost výskytu substituce byla 0,085 % a pravděpodobnost výskytu inzerce 0,015 %. Tyto statistiky se však mění v závislosti na použité množině sekvenčních čtení, proto je nutné provést analýzu této množiny před samotným zarovnáním.

### 4.5.4 Zarovnání dlouhých sekvenčních čtení

U dlouhých sekvenčních čtení je větší pravděpodobnost, že budou obsahovat vícenásobné inzerce a více strukturálních variací, v důsledku čehož může docházet k následnému nesprávnému zarovnání k referenční sekvenci. Základní vlastností nástrojů, které pracují s těmito delšími čteními je schopnost akceptovat možné mezery v zarovnání. Není tedy možné zarovnávat celé sekvenční čtení, ale pouze jeho část. Všechny současné nástroje, které zarovnávají dlouhá sekvenční čtení k referenčnímu genomu, využívají hashovací tabulku nebo FM-index pro zarovnání semínka. Následné zarovnání celé sekvence je realizováno pomocí algoritmu Smith-Waterman. Tento algoritmus se vyznačuje velmi citlivou detekcí inzercí a umožňuje zarovnání s částečnou shodou.

### 4.5.5 Speciální zarovnání

Zarovnávací metody lze optimalizovat přímo pro konkrétní sekvenační platformu, která byla použita při tvorbě sekvenčních čtení. Jednotlivé platformy dávají vytvořeným čtením určitá specifika. Všechny doposud popsané metody a přístupy se týkají zejména sekvenčních čtení vytvořených platformou Illumina, která je na trhu sekvenačních přístrojů dominantní [39]. V případě dalších platforem jsou při zarovnání použita jistá specifika, která odpovídají použitým sekvenačním technikám (viz kapitola 2).

Zarovnání sekvenčních čtení z přístroje SOLiD je příkladem těchto specifických přístupů. Při sekvenování metodou SOLiD se zkoumají vždy dvě sousední báze současně. Každý dinukleotid, kterých může být jednou z 24 možných kombinací, je kódován jednou ze 4 možných barev, čímž vzniká čtyřbarevné spektrum kódů podle obrázku obrázek 4.6 (A). Po dekódování vzniká řetězec nukleotidů. Problémem je vysoká citlivost na jednobodovou chybu, která má za následek rozšíření chyby ve směru dekódování podle obrázku 4.6 (D). Možným řešením je barevné zakódování a reverzace referenční sekvence podle obrázku 4.6 (B, C).


Obrázek 4.6: Zarovnání sekvenčních čtení přístroje SOLiD [32].

Nevýhodou tohoto přístupu je dvojí chyba v barevném zakódování v případě jednobodové mutace podle obrázku 4.6 (E). Dvě po sobě jdoucí barevné změny jsou proto upřednostněny před změnami, které nejsou bezprostředně sousední. Dokonalejší variantou je upravený algoritmus Smith-Waterman se znalostí barvy, který používá například nástroj SHRiMP [46]. Hlavní výhodou tohoto přístupu je detekce inzerce během zarovnání bez nutnosti dodatečné analýzy.

Speciální přístup vyžadují i sekvenční čtení vzniklá například při sekvenaci transkriptomu. Hlavním důvodem je alternativní sestřih, ke kterému dochází během posttrankripčních úprav. Během sestřihu dojde k vyštěpení intronů z primárního transkriptu. Sekvenční čtení pořízená na rozhranní dvou exonů pak nelze standardním přístupem zarovnat. Jedním z řešení je použít při zarovnání množinu známých či predikovaných hranic sestřihu. Tuto množinu lze získat pomocí prvotního standardního zarovnání pouze těch čtení, která jsou zcela obsažena uvnitř exonu. Při okrajích shluků namapovaných čtení jsou poté predikovány hranice sestřihu. Tento princip využívá například nástroj TopHat [51].

# Kapitola 5

# Výběr programů k optimalizaci

Současně existuje více než 60 různých mapovacích nástrojů založených na různých technikách zarovnání a algoritmech vytváření indexu (viz kapitola 4). Díky tomu je velmi obtížné zvolit vhodný nástroj pro řešení daného problému. Záleží na použité sekvenační technologii, kvalitě a délce sekvenčních čtení nebo zda se jedná o nasekvenovaný genom či transkriptom. Některé nástroje jsou přímo navrženy pro konkrétní sekvenační platformu. Pro účely této práce byla zvolena platforma Illumina, která je dominantním přístrojem na trhu sekvenačních technologií a všechny mapovací nástroje jsou schopny zarovnávat sekvenční čtení produkované tímto přístrojem [39].

Důležitým aspektem výběru nástroje je zajímavá vlastnost, u které lze na základě provedených experimentů předpokládat vysoký potenciál při optimalizaci. Mezi takové vlastnosti patří povolení neshody při počátečním zarovnání a možnost zjemnění indexu pro detailnější vyhledání počáteční shody, jak je tomu například u programu BLAT, který je prvním vybraným nástrojem. Další zajímavá vlastnost je možnost použití prostorově rozšířeného semínka (viz kapitola 4) a jeho proměnlivé délky. Díky tomuto semínku lze také dosáhnout vyšší citlivosti a přesnosti zarovnání, proto druhým vybraným nástrojem je program LAST, u kterého lze kombinovat prostorově rozšířené semínko se semínkem s proměnlivou délkou.

Dalším důležitým krokem po výběru konkrétního nástroje je optimální nastavení parametrů zarovnání, aby bylo dosaženo požadované přesnosti. Cílem této práce je vytvoření nástroje pro nalezení takových parametrů, aby programy dosáhly co největší přesnosti zarovnání. Je zřejmé, že vedlejším důsledkem zvýšení přesnosti bude pokles rychlosti a zvýšení paměťové náročnosti v případě vytvoření detailnějšího indexu. Rychlost bude sekundárním kritériem při optimalizaci parametrů. V případě, že dvě konfigurace zvolených parametrů budou vykazovat stejnou přesnost, bude vybrána rychlejší varianta.

## 5.1 BLAT

Jednou z hlavních výhod programu BLAT je především jeho rychlost v porovnání s nástrojem BLAST, který je základním nástrojem pro zarovnání dvou sekvencí. V případě velmi podobných sekvencí je rychlost až desetkrát vyšší [23]. Index pro vyhledávání v referenční sekvenci se vždy tvoří pouze jednou pro zarovnání všech sekvenčních čtení, což také přispívá k celkové vyšší rychlosti. Program BLAT indexuje referenční sekvenci, čímž se liší od nástroje BLAST, který indexuje zarovnávané sekvenční čtení. Po vytvoření indexu se provádí zarovnání, které lze rozdělit do dvou částí:

- fáze prvotního vyhledání (*search stage*),

- fáze zpřesňujícího zarovnání (*alignment stage*).

V první fázi se vyhledají oblasti, které jsou homologní s částí zkoumané sekvence. Cílem této fáze je zredukovat prohledávací prostor potenciálních zarovnání. Ve druhé fázi pak dojde ke zpřesnění prvotního zarovnání.

Využívá se principu zarovnání semínka a jeho následného rozšíření [23]. Zarovnání semínka je jednoduchá a efektivní metoda nalezení shody délky $k$, což je délka semínka. Ke každému semínku délky $k$ sekvenčního čtení se vyhledá množina potenciálních a standardně nepřekrývajících se zarovnání. Zarovnání jednotlivých nukleotidů v rámci semínka je vzájemně nezávislé, tedy pravděpodobnost nalezení homologní oblasti k semínku lze vyjádřit vzorcem 5.1.

$$P = M^k \tag{5.1}$$

Proměnná $M$ odpovídá pravděpodobnosti shody mezi homologními oblastmi přibližně 0,98 % [23].

Ve druhé fázi dochází ke zpřesnění zarovnání. V tomto procesu existují určitá omezení, což poskytuje možnost dodatečné optimalizace. V případě, že výsledek prvotního zarovnání je téměř přesný, lze tento přístup použít i v jeho základní formě. Algoritmus spočívá v generování seznamu možných částečných zarovnání sekvenčního čtení a homologní oblastí, jejíž velikost je typicky 50-200 nukleotidů. Semínko délky $k$ se iterativně prodlužuje, dokud nedojde k nalezení pouze jedné shody nebo není dosažena maximální délka. Dále se zarovnání rozšíří na maximum při zachování přesné shody. Takto nalezené oblasti se spojují do celkového zarovnání a prázdná místa se doplňují s cílem minimalizovat počet mezer. Dále se minimalizuje počet neshod a počet inzercí a delecí.

V porovnání s vysokou rychlostí a efektivitou v případě zarovnání téměř identických sekvencí je rychlost i přesnost mnohem nižší v případě, kdy je podobnost sekvencí nižší než 90 %. Klíčem k optimální rychlosti a zejména maximální přesnosti je správné sestavení indexu pro prvotní vyhledání homologních oblastí. Tato práce se věnuje zejména optimalizaci těch parametrů, které ovlivňují vlastnosti vytvořeného indexu. I malá změna velikosti semínka má výrazný vliv na rychlost i přesnost.

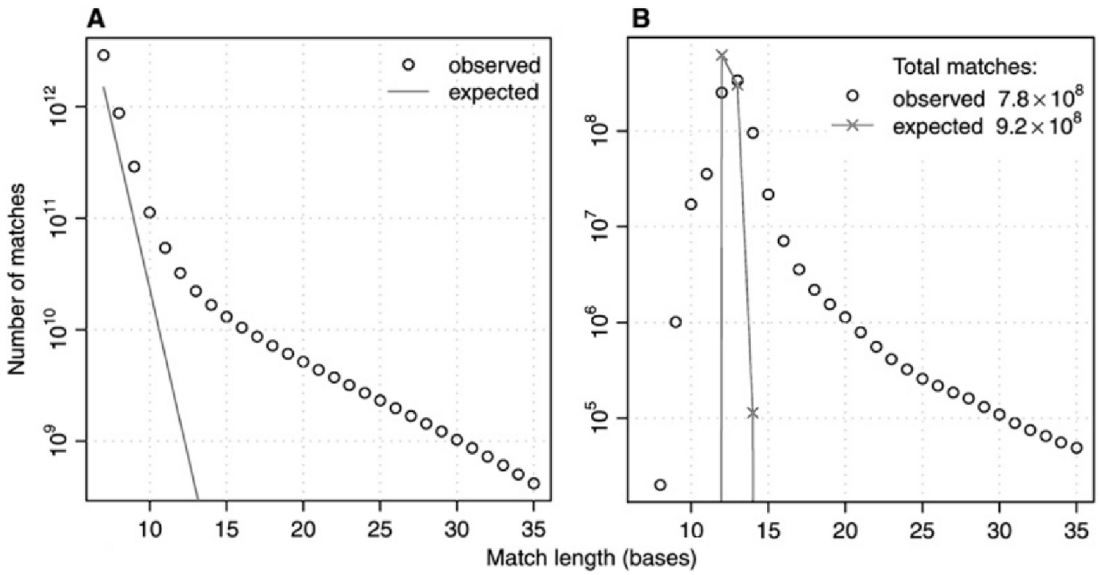
## 5.2 LAST

Klasická metoda *seed-and-extend*, která je použita v programu BLAT, je v případě programu LAST nahrazena prostorově rozšířeným semínkem s proměnlivou délkou. Výhodou této metody je garance lineárního nárůstu času i počtu shod namísto kvadratického, což se projeví zejména v případě použití referenčních sekvencí délek řádově v gigabázích. Hlavním úkolem však stále zůstává hledání podobností mezi dvěma obrovskými datovými sadami. Stejně jako nástroj BLAT vychází LAST ze standardního zarovnávacího nástroje BLAST.

V případě strategie *seed-and-extend* se jedná o velmi rychlé nalezení primární shody. Při použití kratšího semínka dochází k sice přesnějšímu, ale časově daleko náročnějšímu zarovnání, které je současně paměťově náročné, protože velikost indexu roste se zkracováním semínka. Naopak prodlužování semínka vede ke snížené přesnosti a citlivosti zarovnání [24]. Principem programu LAST je iterativní prodlužování semínka, dokud frekvence výskytu semínka v referenční sekvenci neklesne pod určitou mez. Tuto adaptivní délku lze pomocí

parametrů nastavit na fixní, čímž se ovšem ztrácí jedna z klíčových vlastností programu LAST.

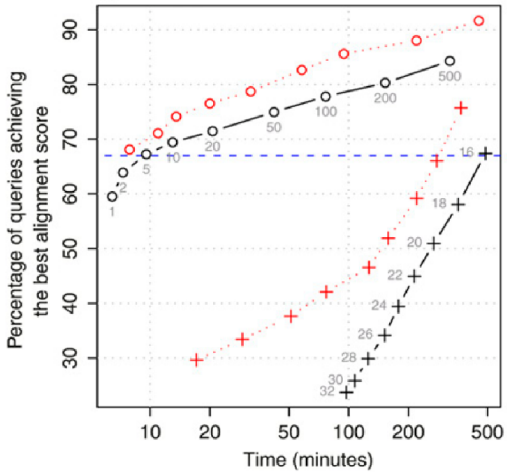
Vhodnost použití adaptivní délky semínka vyplývá z povahy biologických sekvencí. V případě sekvence s rovnoměrným rozložením bází je vhodnější použít fixní délku semínka, ale vlastnosti biologických sekvencí nevykazují rovnoměrné rozložení bází. Tyto sekvence obsahují repetice, což jsou opakované výskyty určitých vzorů, a mohou být AT bohaté, což odpovídá více než 80 % zastoupení bází adeninu a tyminu. Na grafech na obrázku 5.1 je znázorněno, jak délka semínka ovlivňuje počet zarovnání. V tomto případě jsou použita adaptivní semínka s maximálně desetinásobným výskytem v referenční sekvenci. Při použití adaptivního semínka bylo nalezeno 777 milionů zarovnání při nejčastější délce 12-13 bází, jak je naznačeno na obrázku 5.1 (B). Tento výsledek lze porovnat se zarovnáním pomocí semínka s fixní délkou v rozmezí 7-35 bází na obrázku 5.1 (A). Křivky *expected* odpovídají v obou případech zarovnání dat s rovnoměrným rozložením bází a křivky *observed* odpovídají zarovnání lidského chromozomu X k myšímu chromozomu X. Pro snížení počtu zarovnání k cílovým 777 milionům při použití fixního semínka je třeba použít délku 32 bází, což však vede ke snížení citlivosti. Z experimentu vyplývá, že použitím adaptivního semínka lze dosáhnout přesnosti fixního semínka délky 32 bází a citlivosti fixního semínka délky 13 bází, proto je adaptivní semínko vhodnější než fixní [24].




Obrázek 5.1: Srovnání výsledků zarovnání s použitím adaptivního semínka a semínka s fixní délkou [24].

Druhou klíčovou vlastností programu LAST je použití prostorově rozšířeného semínka, což je dominantou moderních zarovnávacích metod. Princip rozšířeného semínka spočívá v binární šabloně, která reprezentuje místa, na kterých je při zarovnání vyžadována shoda (viz kapitola 4). U programu LAST je tento princip kombinován se semínkem s adaptivní délkou tím způsobem, že se šablona, která reprezentuje prostorově rozšířené semínko, cyklicky opakuje, dokud nevyplní celé adaptivní semínko. Výhoda použití prostorově rozšířeného semínka je znázorněna na obrázku 5.2. Použití prostorově rozšířeného semínka (červená křivka) je z hlediska procentuální úspěšnosti zarovnaných sekvenčních čtení výhodnější než použití standardního semínka. Výsledek platí jak pro semínko s adaptivní délkou (kolečka),

tak pro semínko s pevnou délkou (křížky). V tomto experimentu byl použit genom myši jako referenční sekvence a optimální rozšířené semínko se šablonou 111010010100110.




Obrázek 5.2: Srovnání použití standardního a prostorově rozšířeného semínka [24].

Kombinace obou vlastností programu LAST nabízí velký potenciál při hledání optimálního nastavení parametrů. Zejména prostorově rozšířené semínko bylo předmětem mnoha dosavadních optimalizací [24]. Jednou z nich je zavedení *subset seeds*, u kterých je využito rozdělení dusíkatých bází do dvou typů. Puriny (A, G) a pyrimidiny (C, T) lze v tomto případě považovat za třídy ekvivalence, což dává těmto semínkům možnost záměny A za G nebo C za T. Větší uplatnění však tato metoda nachází při zarovnání proteinů, které lze rozdělit do většího počtu tříd podle podobných vlastností.

## 5.3 Výběr parametrů k optimalizaci

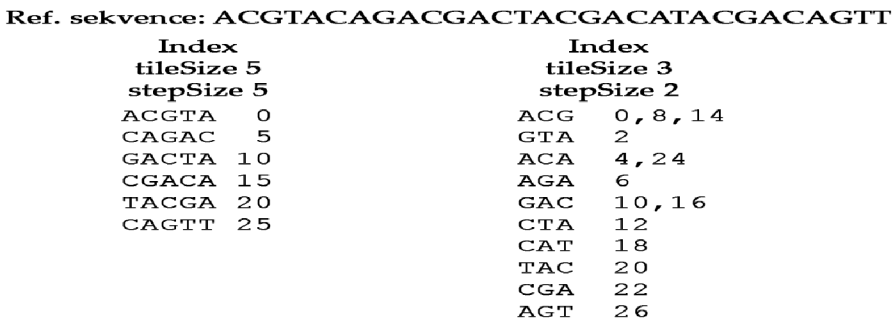
Přesnost a rychlost zarovnání každého nástroje je výrazně ovlivněna nastavením parametrů. Lze zjemnit jak vytváření indexu, tak následné zarovnání. V rámci zarovnání lze povolit či zakázat mezery, použít speciální skórovací matice nebo filtrovat potenciálně nesprávná zarovnání. Popis činnosti jednotlivých parametrů lze najít na manuálových stránkách každého nástroje včetně výchozích hodnot těchto parametrů. Některé parametry mohou nabývat hodnot pouze z omezeného intervalu z důvodu zachování efektivity nástroje a zamezení použití takových hodnot, které nepovedou k odpovídajícím výsledkům.

Výběr parametrů souvisí s konkrétním cílem, kterým je maximální přesnost zarovnání. Vybrané parametry musí tedy ovlivňovat zejména vytvoření indexu pro vyhledání počáteční shody. Dalším klíčovým nastavením je povolení neshod v počátečním zarovnání semínka. V obou případech se očekává možnost zlepšení citlivosti a přesnosti zarovnání. Naopak parametry ovlivňující penalizace mezer, jejich maximální délky a hodnoty není v této práci nutné brát v potaz, protože k zarovnání budou použita sekvenční čtení z nasekvenovaného genomu či jeho části. Penalizace a omezení mezer nachází uplatnění spíše v zarovnání čtení z nasekvenovaného transkriptomu, kde se projevuje alternativní sestřih. Vybrané parametry pro program BLAT jsou zobrazeny v tabulce 5.1.

| Název parametru | Popis funkce | Výchozí hodnota |
|---|---|---|
| tileSize | Velikost semínka | 11 (pro DNA) |
| stepSize | Délka indexačního kroku | tileSize |
| oneOff | Počet neshod v počátečním zarovnání | 0 |

Tabulka 5.1: Parametry k optimalizaci programu BLAT.

Na obrázku 5.3 je znázorněn rozdíl vytvořeného indexu v závislosti na parametrech velikosti semínka (tileSize) a indexačního kroku (stepSize). V prvním případě je délka indexačního kroku stejná jako délka semínka, díky čemuž se semínka nepřekrývají. Stejně je tomu i při výchozím nastavení parametrů. Druhý případ znázorňuje značné zjemnění celého indexu. Jsou použita kratší semínka a indexační krok je kratší než semínko, což vede k hustšímu pokrytí, ale také k nárůstu vyhledávací tabulky.




Obrázek 5.3: Vytvoření indexu pro prvotní zarovnání programem BLAT.

Nevýhodou programu BLAT je chybějící podpora pro prostorově rozšířená semínka, která však využívá druhý vybraný program LAST. Tento program se skládá ze dvou samostatných programů, které řeší dvě oddělené části celého zarovnání. Prvním programem je LASTDB, který vytváří index pro vyhledání počáteční shody. Následuje činnost druhého programu LASTAL, jehož výsledkem je finální zarovnání. Parametry k optimalizaci programu LASTDB jsou zobrazney v tabulce 5.2.

| Název parametru | Popis funkce | Výchozí hodnota |
|---|---|---|
| m | Prostorově rozšířené semínko (šablona) | 1 |
| w | Délka indexačního kroku | 1 |

Tabulka 5.2: Parametry k optimalizaci programu LASTDB.

Výchozí hodnota prostorově rozšířeného semínka je řetězec „1“. Při zarovnání semínka se hledá pouze přesná shoda. Indexační krok má délku 1, což je nejmenší možná hodnota. V případě programu LASTAL jsou parametry pro optimalizaci znázorněny v tabulce 5.3.

| Název parametru | Popis funkce | Výchozí hodnota |
|---|---|---|
| e | Minimální skóre zarovnání | 40 |
| l | Maximální délka adaptivního semínka | $\infty$ |
| m | Mezní frekvence výskytu při rozšiřování semínka | 10 |

Tabulka 5.3: Parametry k optimalizaci programu LASTAL.

Nastavením prvního parametru lze z výsledku odstranit taková zarovnání, která nejsou z důvodu nízkého skóre zřejmě nesprávná. V této práci je ovšem prioritní maximalizace počtu správných zarovnání, proto tento parametr není stěžejní. Druhým parametrem lze nastavit limit pro prodlužování délky semínka a třetí parametr omezuje prodlužování délky semínka maximálním počtem výskytů v referenční sekvenci. Hledání optimálních hodnot těchto parametrů je odděleno od optimalizace vytvoření indexu. Hlavním důvodem je zmíněná existence dvou oddělených programů, z nichž každý řeší jinou část problému zarovnání.

# Kapitola 6

# Návrh a implementace frameworku

Cílem práce je vytvoření frameworku pro hledání optimálních vstupních parametrů zarovnávacích nástrojů. Základním předpokladem je možnost použití na jakýkoliv zarovnávací nástroj, u kterého lze nastavit vstupní parametry, a tím ovlivnit výslednou přesnost, rychlost, nebo jiné měřitelné veličiny.

Množství parametrů a rozsah jejich možných hodnot je důvodem proč hledat jejich optimální nastavení. Výchozí hodnoty parametrů vychází většinou ze standardních hodnot používaných programem BLAST (viz kapitola 4). Příkladem je délka semínka u programu BLAT. Semínko má délku 11 bází při zarovnání sekvenčních čtení k DNA sekvenci a v případě proteinů má délku 5 bází. Velikost indexačního kroku je stejně velká jako velikost semínka, což vede k nepřekrývajícímu se mapování, které ovšem nemusí být dostatečně přesné. Výhodou použití výchozích hodnot je vysoká rychlost zarovnání. Nevýhodou je nižší přesnost zarovnání, která snižuje následně přesnost navazujících analýz. Z toho důvodu je cílem optimalizace maximalizace přesnosti. Druhým příkladem je prostorově rozšířené semínko u programu LAST, u kterého nejsou výchozí hodnoty známy. Pro sekvenční čtení delší než 50 bází je hledání optimálních šablon pro semínka velmi náročným problémem, proto je vhodné použít za účelem optimalizace evolučních technik.

Z obou případů vyplývá nutnost optimalizovat zarovnávací nástroje za účelem maximalizace přesnosti zarovnání. Složitý problém nalezení optimálního nastavení souvisí s velikostí stavového prostoru s více dimenzemi, který je utvářen rozsahem hodnot zkoumaných parametrů. Z toho důvodu je vodné při řešení problému optimalizace použít evoluční techniky.

## 6.1 Diferenciální evoluce

Evoluční techniky jsou založeny na myšlence optimalizace hodnot nezávislých proměnných pomocí evolučních algoritmů, které jsou inspirovány přírodními procesy. Mezi tyto techniky patří například genetický algoritmus, který je současně nejstarší používanou technikou v této skupině, evoluční strategie, genetické programování, simulované žíhání či diferenciální evoluce, která je použita v této práci. Tyto techniky se vzájemně liší v určitých částech algoritmu, ale obecný přístup k řešení problému je pro všechny stejný. Spočívá ve vhodném zakódování zadaného problému do chromozomu, který se označuje jako kandidátní řešení. Použití modelu chromozomu má přímou návaznost na průběh evolučních procesů v přírodě. Jednotlivé proměnné pak odpovídají genům v chromozomu. Prvním krokem je vytvoření populace kandidátních řešení. Každému kandidátnímu řešení v počáteční populaci je přiřazena míra kvality pomocí tzv. fitness funkce. Každý evoluční krok spočívá v náhodném výběru

neboli selekci rodičovských kandidátních řešení. Aplikací genetických operátorů vznikne kombinací vybraných řešení nové kandidátní řešení, které je rovněž ohodnoceno mírou kvality pomocí fitness funkce. Nová generace vzniká kombinací populace rodičů a potomků. Celý algoritmus je ukončen ve chvíli, kdy je dosažen maximální počet generací dané populace nebo je nalezeno optimální kandidátní řešení. Efektivita a schopnost najít optimální řešení spočívá zejména ve vhodném zakódování problému do chromozomu a v aplikaci vhodných genetických operátorů

Jednotlivé evoluční techniky se liší zejména v typu selekce, aplikací genetických operátorů, počtem kandidátních řešení v populaci a způsobem vzniku nové generace z populace rodičů a potomků. Společnou vlastností je stochastický přístup při selekci i rekombinaci, což je proces vzniku nového jedince kombinací jeho rodičů.

Diferenciální evoluce je přímé stochastické prohledávání, jehož cílem je globální optimalizace [50]. Principiálně se velmi podobá optimalizační heuristické metodě PSO (*Particle Swarm Optimization*), což je optimalizace inspirovaná hejnem ptáků při hledání potravy.

Algoritmus diferenciální evoluce zahrnuje zachování populace, která je iterativně vystavena změnám v důsledku selekce, evaluace pomocí fitness funkce a rekombinace. Od standardního genetického algoritmu se liší zejména při rekombinaci. Přístup spočívá v náhodném výběru 3 kandidátních řešení. Nové kandidátní řešení $S$ vznikne podle následujícího vzorce 6.1.

$$S = P_3 + F \times (P_1 - P_2) \tag{6.1}$$

Proměnné $P_i$ odpovídají vybraným kandidátním řešením k rekombinaci a $F$ je váha připočtené změny ke kandidátnímu řešení $P_3$. Proces vzniku nového kandidátního řešení se nazývá perturbace a v kombinaci se selekcí samoorganizuje prostor problému a ohraničuje v něm oblasti, kde se mohou nacházet hledaná řešení.

Diferenciální evoluci a její konfiguraci lze popsat speciálním řetězcem $DE/x/y/z$, kde $x$ označuje řešení, které má být perturbováno. Klasicky se jedná o náhodné či nejlepší řešení v populaci. Proměnná $y$ odpovídá počtu rozdílových vektorů použitých při perturbaci $x$. Rozdílový vektor je dán rozdílem dvou náhodně vybraných kandidátních řešení z populace. Parametr $z$ značí operátor rekombinace (binomický, exponenciální).

Celý algoritmus lze zapsat pseudokódem, znázorněným na obrázku 6.1

**Require:** Population$_{size}$, Problem$_{size}$, Weighting$_{factor}$, Rekombination$_{rate}$
1: Population = InitPopulation(Population$_{size}$, Problem$_{size}$)
2: Evaluate(Population)
3: S$_{best}$ = GetTheBestOne(Population)
4: **while** !StopCondition() **do**
5: NewPopulation()
6: **for** P$_i$ in Population **do**
7: S$_i$ = Perturb(P$_i$,Population, Problem$_{size}$, Weighting$_{factor}$, Rekombination$_{rate}$)
8: NewPopulation.Insert(Better(S$_i$, P$_i$))
9: **end for**
10: Population = NewPopulation
11: EvaluatePopulation(Population)
12: S$_{best}$ = GetTheBestOne(Population)
13: **end while**
14: **return** S$_{best}$

Obrázek 6.1: Algoritmus diferenciální evoluce [50].

V tabulce 6.1 jsou popsány použité proměnné a funkce z pseudokódu na obrázku 6.1.

| Název proměnné / funkce | Význam |
|---|---|
| Population$_{size}$ | Počet jedinců v populaci |
| Problem$_{size}$ | Počet proměnných k optimalizaci |
| Weighting$_{factor}$ | Váhový faktor rozdílu hodnot při rekombinaci |
| Rekombination$_{rate}$ | Míra aplikace rekombinace |
| S$_{best}$ | Nalezené optimální řešení |
| Population | Populace kandidátních řešení |
| InitPopulation | Funkce vytvoří počáteční populaci kandidátních řešení |
| Evaluate | Funkce ohodnotí kvalitu řešení v populaci |
| GetTheBestOne | Funkce vybere nejlepší kandidátní řešení z populace |
| StopCondition | Ukončující podmínka diferenciální evoluce |

Tabulka 6.1: Použité proměnné a funkce v algoritmu diferenciální evoluce.

Hlavní částí algoritmu je perturbace Perturb, jejíž pseudokód je znázorněn na obrázku 6.2:

**Require:** $P_0$, Population, $\text{Problem}_{size}$, $\text{Weighting}_{factor}$, $\text{Rekombination}_{rate}$
1: **repeat**
2: $P_1 = \text{randomMember(Population)}$
3: **until** $P_1 \neq P_0$
4: **repeat**
5: $P_2 = \text{randomMember(Population)}$
6: **until** $P_2 \neq P_1$ and $P_2 \neq P_0$
7: **repeat**
8: $P_3 = \text{randomMember(Population)}$
9: **until** $P_3 \neq P_2$ and $P_3 \neq P_1$ and $P_3 \neq P_0$
10: $\text{cutPoint} = \text{randomPosition}(\text{Problem}_{size})$
11: **for** i to $\text{Problem}_{size}$ **do**
12: **if** i == cutPoint or random() $< \text{Rekombination}_{rate}$ **then**
13: $S_i = P_{3i} + \text{Weighting}_{factor} \times (P_{1i} - P_{2i})$
14: **else**
15: $S_i = P_{0i}$
16: **end if**
17: **end for**
18: **return** S

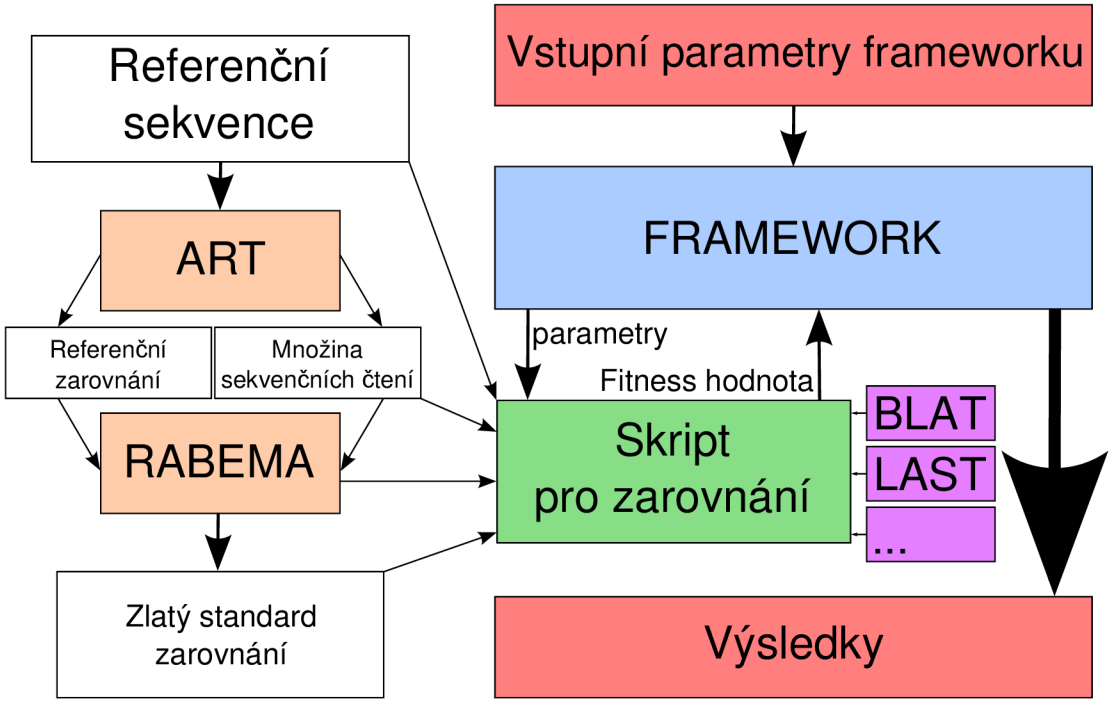
Obrázek 6.2: Pseudokód funkce perturbace Perturb [50].

Pro jednotlivé parametry, které mění povahu evoluce a rychlost konvergence k optimálnímu řešení, lze vybrat hodnoty z doporučených intervalů. Velikost populace a možný omezený počet generací přímo souvisí s délkou vektoru kandidátního řešení. V případě vyššího počtu proměnných je třeba více jedinců v populaci i větší počet generací. Dalším klíčovým parametrem je rozsah hodnot těchto proměnných, který může být volitelně omezen. Váhový faktor rozdílu hodnot při rekombinaci ($\text{Weighting}_{factor}$) je vhodné zvolit z intervalu $\langle 0, 2\rangle$, přičemž doporučená standardní hodnota je 0,8. Pravděpodobnost, s jakou dochází k rekombinaci ($\text{Rekombination}_{rate}$), je standardně 0,9 a může nabývat hodnot z intervalu $\langle 0, 1\rangle$. Pro ukončení běhu algoritmu je nutné dosáhnout ukončující podmínky, kterou lze realizovat více možnými způsoby. Jedním z těchto způsobů může být nalezení optimálního kandidátního řešení. Další možností je explicitní omezení počtu generací. V průběhu evoluce ovšem dochází k postupnému snižování míry rozdílnosti kandidátních řešení uvnitř populace, což vede až na vznik populace složené z identických kandidátních řešení a k dalšímu zlepšení už nemůže dojít.

## 6.2 Implementace

Návrh celého systému je znázorněn na obrázku 6.3. Pomocí programu ART [18] se z referenční sekvence vygeneruje množina sekvenčních čtení včetně referenčního zarovnání. Dalším krokem je vytvoření zlatého standardu zarovnání (viz kapitola 7). Tím je skončena iniciační fáze, která předchází spuštění experimentů. Druhá fáze začíná nastavením vstupních parametrů pro optimalizační framework a spuštění tohoto frameworku. Fitness funkce je realizována externím skriptem. Průběh výpočtu fitness funkce je ovlivněn vstupními parame-

try externího skriptu. Výsledky optimalizace jsou generovány v textové podobě. Jednotlivé součásti celého systému budou podrobně popsány v následujících sekcích.




Obrázek 6.3: Schéma optimalizačního systému.

Výpočetní framework je implementací algoritmu diferenciální evoluce. K implementaci byl použit programovací jazyk Python verze 2.7 a byla využita paradigmata objektově orientovaného programování, a to zejména dědičnosti. Cílem bylo rozdělení celého optimalizačního systému na oddělené části, které se liší svojí funkčností a které lze v případě potřeby nahradit novějšími verzemi. Každá část je reprezentována samostatnou třídou z následujícího seznamu tříd:

- `Member`,
- `Population`,
- `Evolution`.

Objekt třídy `Member` reprezentuje jedno kandidátní řešení a zapouzdřuje jeho konfiguraci, což je seznam hodnot optimalizovaných parametrů. Míra kvality každého kandidátního řešení je stanovena dvěma fitness hodnotami. Primární fitness hodnota reprezentuje počet správně zarovnaných sekvenčních čtení, proto je cílem nalézt její maximální hodnotu. Sekundární fitness hodnota odpovídá době běhu zkoumaného zarovnání v sekundách a má význam pouze v případě existence více odlišných kandidátních řešení se stejnou primární fitness hodnotou. V takovém případě se vybere rychlejší varianta. Proměnné odpovídající optimalizovaným parametrům jsou intervalové, tedy je možné provést jejich rozdíl podle požadavku diferenciální evoluce. Hodnoty jsou celočíselné a nezáporné kvůli snadnějšímu počítání rozdílů. V případě, že je třeba jako hodnotu parametru dosadit zápornou hodnotu či nějaký řetězec, což je případ binární šablony u jednoho z parametrů programu LAST,

je nutné před aplikací původní hodnotu změnit například podle dodatečné tabulky. Tento proces lze uskutečnit také proto, že jazyk Python patří do skupiny slabě typovaných jazyků, u kterých probíhá typová kontrola dynamicky za běhu programu, díky čemuž lze proměnnou interpretovat jako číslo nebo jako řetězec.

Objekt třídy `Population` slouží k reprezentaci populace kandidátních řešení. Zapouzdřuje seznam kandidátních řešení a odkaz na nejlepší kandidátní řešení v populaci. Klíčovou funkcí je metoda pro výpočet fitness funkce všech kandidátních řešení pomocí externího skriptu pro operační systém Linux. Tento skript se spustí paralelně pro všechny kandidátní řešení v dané populaci kvůli celkovému urychlení výpočtu. Následně program aktivně čeká na dokončení výpočtu všech hodnot fitness. Výpočet hodnoty fitness je optimalizovaný tak, že pro každé kandidátní řešení se zkoumá, zda již nebylo v průběhu evoluce ohodnoceno jiné kandidátní řešení se stejnou konfigurací hodnot parametrů. V případě, že je takové dřívější řešení nalezeno, není třeba spouštět externí skript pro opakované zarovnání a ohodnocení a hodnota fitness se pouze zkopíruje.

Hlavním objektem, který realizuje algoritmus diferenciální evoluce, je objekt třídy `Evolution`. Při vytváření instance této třídy lze nastavit různé parametry běhu evoluce, které jsou zobrazeny v tabulce 6.2.

| Název atributu | Typ atributu | Popis atributu |
|---|---|---|
| popSize | integer | Velikost populace |
| genCount | integer | Počet generací evoluce |
| problemLow | list | Seznam horních hranic hodnot proměnných |
| problemHigh | list | Seznam dolních hranic hodnot proměnných |
| problemName | list | Seznam názvů proměnných |
| weightingFactor | float | Váhový faktor při perturbaci |
| crossingFactor | float | Pravděpodobnost perturbace |
| best_fitness | float | Nejlepší hodnota fitness funkce |
| script_test | string | Cesta ke skriptu s fitness funkcí |
| aligner | string | Název zarovnávacího programu |
| reads_type | string | Typ sekvenčních čtení (single-end/pair-end) |
| param_def | string | Znak příkazového řádku mezi parametrem a hodnotou |
| read_length | string | Délka sekvenčního čtení |
| source | string | Typ dat (reálná/simulovaná) |
| sec_param | list | Seznam sekundárních parametrů |

Tabulka 6.2: Seznam atributů, kterými lze nastavit běh optimalizačního frameworku.

Z algoritmu diferenciální evoluce vyplývá, že v populaci musí být minimálně čtyři kandidátní řešení, aby bylo možné realizovat perturbaci, která vyžaduje vedle perturbovaného řešení ještě tři vzájemně unikátní kandidátní řešení. Sekundární parametry slouží pro případ, kdy zarovnávací program potřebuje pro svůj běh spuštění dalších doprovodných programů. Tyto programy lze pak spustit se sekundárními parametry, které nepodléhají evoluci. Jednotlivé etapy probíhající evoluce je možné ovlivnit standardním způsobem pomocí vytvoření potomka této třídy a aktualizace kódu libovolné metody. Pokud se jedná pouze o jednoduchou modifikaci vstupních či výstupních dat, lze použít pomocné metody. Každá metoda této třídy má dvě pomocné metody, z nichž jedna se provede před hlavní metodou a druhá se provede po ní. Příkladem je psudokód na obrázku 6.4.

```
...
self.preActualizeBestSoFar(self.pop.bestMember())
self.actualizeBestSoFar(self.pop.bestMember())
self.postActualizeBestSoFar(self.pop.bestMember())
...
```

Obrázek 6.4: Volání pomocných metod.

Před každým voláním metody `actualizeBestSoFar` se volá metoda `preActualizeBestSoFar` a za ní se volá metoda `postActualizeBestSoFar`. Výhodou je snadnost dodatečné modifikace před začátkem nebo po skončení hlavní metody.

Ukončující podmínka je implementována pouze omezeným počtem generací. V průběhu testování se ukázalo vhodné regulovat počet generací přímo, aby nedocházelo ke zbytečným prodlevám v důsledku již neefektivního pokračování výpočtu. Díky stochastickému přístupu může docházet ke zdlouhavému procesu úplného vyčištění populace od rozdílných kandidátních řešení. V případě programu LAST je rozsah parametrů v řádu desetitisíců a počet generací by proto enormně narůstal.

## 6.3 Obecné použití

Základním požadavkem při tvorbě frameworku je možnost širokého použití na optimalizaci parametrů různých nástrojů. Proto je nutné, aby modifikace při změně cílového nástroje byla co možná nejsnazší. Framework je tvořen třemi třídami, ze kterých se pomocí dědičnosti vytvoří potomci. U tříd `Population` a `Evolution` je nutné modifikovat konstruktory ostatních tříd, aby se vytvořila instance potomka místo předka. Při vytvoření objektu evoluce je nutné zadat parametry diferenciální evoluce a zarovnávaných dat. Pomocí seznamů se definují přípustné rozsahy optimalizovaných parametrů a důležitým parametrem je skript pro zarovnání, jehož standardním výsledkem jsou hodnoty fitness funkcí vypsané na standardní výstup a oddělené znakem konce řádku. Pro správný běh evoluce je pak třeba provést následující sekvenci příkazů:

1. `Evolution()`
2. `initEvolution()`
3. `runEvolution()`

V průběhu evoluce dochází k průběžnému výpisu detailních informací o výsledcích v každé generaci diferenciální evoluce. Volitelně lze vypsat i dodatečné informace pomocí metod, které jsou popsány v programové dokumentaci.

Optimalizované nástroje musí splňovat určité požadavky, které odpovídají implementaci frameworku. Hlavním požadavkem je typ hodnoty parametrů. Standardně se jedná o intervalový typ, aby bylo možné provádět rozdíl hodnot při perturbaci. V případě řetězce je nutný převod do celočíselné nezáporné podoby pomocí tabulky. Při následném spuštění programu s parametry za účelem vyhodnocení fitness funkce se pomocí tabulky provede reverzní převod hodnoty na odpovídající řetězec nebo záporné číslo.

Dalším požadavkem je výstupní formát. Používané formáty v rámci protokolu pro komunikaci mezi jednotlivými částmi systému jsou popsány v následující kapitole. Pokud není výstupním formátem zarovnávacího nástroje standardní formát SAM, je nutné použít nebo vytvořit konvertor do tohoto formátu. V opačném případě nelze použít jako fitness funkci program RABEMA [15] (viz sekce 6.5).

## 6.4 Použité formáty

Jednotlivé části výpočetního frameworku spolu vzájemně komunikují speciálně navrženým protokolem, který sestává převážně ze standardních používaných formátů. Tyto formáty jsou čitelné pro všechny dostupné zarovnávací programy, čímž lze například u výsledku zarovnání porovnat a ohodnotit výsledky dvou různých zarovnávacích nástrojů.

### 6.4.1 FASTA

Sekvence DNA, RNA nebo proteiny lze uložit ve více možných formátech v závislosti na množství doprovodných informací. Jedním z nejjednodušších a nejpoužívanějších formátů je formát FASTA. Soubor ve formátu FASTA sestává z jednotlivých FASTA záznamů. Každý záznam je rozdělen na hlavičku, která obsahuje libovolný řetězec s informacemi, a primární strukturu sekvence. V případě této práce se jedná o sekvenci DNA. Příkladem je následující FASTA záznam:

```
>hg19_ensGene_ENST00000399012 range=chrX:200855-216002
ATGGGTGGGCAGGTGAGCGCTTCCAACAGCTTCTCGAGGCTGCACTGCAG
AAATGCCAACGAGGACTGGATGTCGGCACTGTGTCCCCGGCTCTGGGATG
```

Hlavička záznamu začíná symbolem „>“. Z uvedeného záznamu vyplývá, že se jedná o sekvenci genu na chromozomu X. Dále je známa pozice genu na tomto chromozomu a primární struktura DNA, což je posloupnost jednotlivých nukleotidů.

### 6.4.2 FASTQ

Formát FASTQ je podobný formátu FASTA. Soubor sestává ze záznamů, které kromě informace o primární struktuře sekvence obsahují informaci o kvalitě jednotlivých nukleotidů. Tato kvalita souvisí se zdrojem sekvence, kterým je jedna ze sekvenačních platforem. Formát FASTQ se stal standardním výstupním formátem sekvenačních přístrojů. Příkladem je následující FASTQ záznam:

```
@LAST_Gen-99990
GTGAGATGACGAGGACACCTGCCGCCCCAGCCCCCACCTGATCCACCACA
+
B0C?@<BCCF9E@CDD?ED.;BG:A?D5>?:9@7@2:8*5522?)+81+2
```

První řádek je jako v případě FASTA záznamu nositelem popisné informace o sekvenčním čtení na následujícím řádku. Za oddělovačem následuje 4. řádek s informací o kvalitě a správnosti odpovídající báze. Kvalita je kódována ASCII znakem a rozsah znaků je upraven několika možnými formáty, které se většinou označují nejnižším možným znakem v hexadecimální podobě. Příkladem je Phred33 nebo Phred64. Z celkového rozsahu, který končí hexadecimálním číslem 126, bývá využito pouze asi 40 znaků.

### 6.4.3 SAM

Formáty FASTQ i FASTA slouží pro uložení referenční sekvence či sekvenčního čtení. Formát SAM je standardní formát pro popis zarovnání sekvenčního čtení k referenční sekvenci. Vedle textového formátu SAM ještě existuje jeho binární podoba BAM. Pomocí vhodných nástrojů, jako je například Samtools [31], který je použit i v této práci, lze tyto formáty

mezi sebou převádět. Soubor ve formátu SAM sestává z hlavičky a SAM záznamů v tabulce bez záhlaví. Výhoda textového SAM formátu spočívá v přehlednosti a snadné interpretaci v porovnání s binárním BAM formátem. Následující příklad je krátký soubor ve formátu SAM:

```
@HD VN:1.4 SQ:unsorted
@SQ SN:CHRX LN:100000
@SQ SN:CHRY LN:100000
CHRX-1  0 CHRX 15505 255 10=1X10= * 0 0 GGTAACGCTTGGGAATTACAT *
CHRY-1 16 CHRY 66353 255 21= * 0 0 GACGTTCTGTGCAAGCAGAGG *
```

První řádek je začátek hlavičky obsahující verzi a informaci o seřazení záznamů. Za prvním řádkem následuje databáze referenčních sekvencí. Povinnými poli jsou jak název této sekvence, tak její délka. Součástí záznamů mohou být i další pole, která nejsou povinná a která nejsou pro tuto práci důležitá. Nepovinnými údaji v hlavičce jsou údaje o zdrojovém programu a vzniku souboru se zarovnáním, údaje o skupinách sekvenčních čtení a dodatečný jednořádkový komentář. Dále následuje seznam SAM záznamů. Každý záznam začíná názvem sekvenčního čtení a druhým údajem je číselná reprezentace vlastnosti zarovnání. V této práci jsou použita pouze čísla 0 pro zarovnání k přímému vláknu a číslo 16 pro zarovnání k reverznímu vláknu. Dalším údajem je název referenční sekvence. Následuje pozice sekvenčního čtení na referenční sekvenci, kvalita mapování, řetězec CIGAR, informace o párových sekvenčních čteních a jejich pozicích, primární struktura sekvenčního čtení a dodatečné informace o kvalitě. Zmíněné informace nejsou v práci využity, jen řetězec CIGAR je vytvořen speciálním skriptem ze zarovnání párových sekvenčních čtení ve formátu MAF. Tento řetězec popisuje shody („=“), neshody („X“), inzerce („I“) a delece („D“) po celé délce sekvenčního čtení. Význam řetězce CIGAR u prvního SAM záznamu v ilustračním příkladě je 10 shod následovaný jednou neshodou a opět 10 shod.

Problémem zůstává více možných výstupních formátů u řady zarovnávacích nástrojů, mezi které patří BLAT i LAST. U obou nástrojů není SAM formát výchozím formátem pro výpis zarovnání, a proto je nutná konverze z použitého formátu MAF, který oba používají, do formátu SAM, který byl díky zmíněným výhodám vybrán jako cílový.

## 6.5 Fitness funkce

Klíčovou součástí diferenciální evoluce i všech evolučních algoritmů je vhodně zvolená fitness funkce. Výsledkem zarovnání je množina zarovnaných sekvenčních čtení v různých formátech, které jsou převeditelné do nejpoužívanějšího formátu SAM (viz sekce 6.4). Tento formát je použit i v této práci a program RABEMA, pomocí něhož je realizována fitness funkce, vytváří tzv. zlatý standard zarovnání. Tento zlatý standard se vytváří z referenčního zarovnání ve formátu SAM a referenční sekvence ve formátu FASTA. Cílem je pro každý SAM záznam vytvořit odpovídající záznam zlatého standardu, který odpovídá následujícímu popisu:

```
@GSI VN:1.1
@MATES SEP:/ TYPE:01
#QNAME SCORE RNAME RDIR SPOS EPOS
BLAT_Gen-1 1 BLAT_Gen R 48462 48462
BLAT_Gen-2 1 BLAT_Gen F 67982 67982
```

```
BLAT_Gen-3 1 BLAT_Gen F 52123 52123
BLAT_Gen-4 3 BLAT_Gen F 91016 91016
```

První dva řádky obsahují hlavičku, která pro další použití není důležitá. Následuje tabulka záznamů s hlavičkou, která popisuje význam jednotlivých sloupců. Pole QNAME odpovídá názvu zarovnávaného sekvenčního čtení. Pole SCORE odpovídá počtu chyb na stupnici od 0 do 8, což je standardní rozlišovací rozsah při vytváření zlatého standardu. Jedná se o počet neshod, delecí a inzercí po celé délce zarovnaného čtení. Rozlišovací rozsah je omezen na 8 chyb s ohledem na kvalitu dat i doporučené nastavení programu RABEMA. Z tabulky 6.3 vyplývá, že pro délku sekvenčního čtení v prvním řádku je procento odpovídajících zarovnání minimálně 79 % pro délku sekvenčního čtení 100 bází při použití standardní rozlišovací schopnosti programu RABEMA.

| Délka sekvenčního čtení | 50 | 65 | 75 | 90 | 100 |
|---|---|---|---|---|---|
| Procento referenčních zarovnání | 1,00 | 0,99 | 0,99 | 0,92 | 0,79 |

Tabulka 6.3: Procento referenčních sekvenčních čtení v dané rozlišovacím intervalu v závislosti na délce sekvenčního čtení.

Pro sekvenční čtení délky více než 100 bází by bylo lépe použít vyšší rozlišovací schopnost, aby mohla být zkoumána i sekvenční čtení s výskytem více než 8 chyb v zarovnání.

Další záznamy v tabulce záznamů zlatého standardu odpovídají po řadě názvu referenční sekvence, označení přímého či reverzního vlákna a pozicím na tomto vlákně.

Při ohodnocení zkoumaného zarovnání se provádí srovnání s vytvořeným zlatým standardem. Základním požadavkem je zarovnání ve formátu SAM včetně správně sestavené hlavičky. K porovnání je opět použit program RABEMA. Výsledkem porovnání je výstupní formát RABEMA. V příloze B je celý výstup programu RABEMA. Jeho zjednodušená forma s pouze důležitými údaji je popsána následujícím příkladem:

```
Intervals to find:              13304
Intervals found:                11051
Intervals found [%]             83.0652
Invalid alignments:             465

Number of reads:                13330
Number of reads with intervals: 13304

  ERR     #max   #found   %found
-----------------------------------------------------
    0      737      737   100.00
    1     2334     2254    96.57
    2     3189     2893    90.72
    3     3035     2543    83.79
    4     2135     1617    75.74
    5     1152      715    62.07
    6      484      219    45.25
    7      184       58    31.52
    8       54       15    27.78
```

Z tohoto záznamu vyplývá, že je možné zarovnat celkem 13 330 sekvenčních čtení, přičemž rozsah chyb v zarovnání splňuje pouze 13 304. V hodnoceném zarovnání ve formátu SAM bylo nalezeno 11 051 správných zarovnání, což odpovídá 83,06 % úspěšnosti. Ze zbylých sekvenčních čtení bylo 465 zarovnáno nesprávně. V tabulce pod celkovými statistikami jsou rozepsaná sekvenční čtení podle třídy počtu neshod v zarovnání.

Výstupní formát ohodnocení zarovnání programem RABEMA je zdrojem vstupních údajů pro fitness funkci. Existuje více možností, jak z těchto údajů určit míru kvality zkoumaného zarovnání a jemu odpovídajícího nastavení parametrů. V průběhu vývoje nástroje byly vyzkoušeny tři přístupy, z nichž byl vybrán jeden pro následné uskutečnění experimentů.

Prvním přístupem je součet hodnot ve sloupci %found. Maximální hodnota fitness funkce je v tomto případě 900, které lze dosáhnout v případě 100 % úspěšnosti zarovnání v každé kategorii. Výhodou tohoto přístupu je, že v případě nízkého maximálního počtu sekvenčních čtení v dané kategorii vysoký nárůst fitness funkce při zarovnání jednoho nového sekvenčního čtení. Při maximálním počtu 2 sekvenčních čtení by nalezení jednoho sekvenčního čtení znamenal nárůst výstupní hodnoty fitness funkce o 50. Použití tohoto přístupu způsobuje vysoký nárůst hodnoty fitness v průběhu evoluce. Nevýhodou je zbytečné zvýhodnění kategorií s nižším počtem možných sekvenčních čtení.

Druhý přístup se od prvního liší v případě prázdné kategorie. Pokud v kategorii není žádné sekvenční čtení z referenčního zarovnání, pak zde ani žádné nelze nalézt a hodnota sloupce %found je v tomto případě 0, čímž dochází k nepříznivému ovlivnění výpočtu a maximální hodnoty 900 nelze dosáhnout. To lze vyřešit připočítáním hodnoty 100 namísto 0, čímž lze opět dosáhnout hodnoty 900. Díky nerovnoměrnému nárůstu hodnoty fitness ovšem nelze dvě zkoumaná řešení vzájemně porovnat stejně jako v předchozím případě.

Třetí fitness funkce již nebere v úvahu možnost rychlého nárůstu hodnoty, ale upřednostňuje možnost vzájemného porovnání dvou zkoumaných zarovnání. Výsledkem fitness funkce je celkový počet správně zarovnaných sekvenčních čtení. Každá dvě zkoumaná zarovnání lze jednoznačně porovnat v případě, že je použita stejná sada sekvenčních čtení, stejné referenční zarovnání a stejná referenční sekvence. V průběhu testování byla nakonec použita tato varianta, protože původní dvě poskytovaly nevyhovující výsledky. Maximální hodnota této fitness funkce odpovídá prvnímu údaji v zjednodušeném výstupním formátu programu RABEMA.

Součástí implementace je vedle primární fitness funkce, jejímž výsledkem je počet správně zarovnaných sekvenčních čtení i sekundární fitness funkce. Výslednou hodnotou je v této práci doba běhu v sekundách. Cílem evoluce je nalézt co nejvyšší hodnotu primární fitness. V případě, že existují dvě odlišná kandidátní řešení se stejnou primární fitness hodnotou, rozhoduje sekundární fitness hodnota a uplatní se rychlejší řešení.

# Kapitola 7

# Experimenty

V předchozí kapitole byla popsána implementace optimalizačního frameworku a jeho možnost aplikace na množinu zarovnávacích nástrojů. Cílem provedených experimentů je ověření přínosu optimalizace, což lze provést pomocí srovnání přesnosti zarovnání při aplikaci optimálních hodnot parametrů v porovnání s aplikací výchozích hodnot. Experimenty lze také ověřit vývoj nejlepšího řešení v průběhu evoluce a jeho zlepšení ve smyslu nárůstu počtu správně zarovnaných sekvenčních čtení. Součástí experimentů je modifikace různých parametrů a sledování následného vlivu na výsledky. Mezi tyto parametry patří v první řadě délka sekvenčního čtení a jejich typ. Výsledky se mohou různit, když se délka sekvenčního čtení změní z výchozích 50 bází na dvojnásobek. Použití jednoduchých sekvenčních čtení (single-end) se také může lišit od varianty s párovými sekvenčními čteními (pair-end). Programy LAST ani BLAT a mnohé další nejsou přímo sestrojeny pro zarovnání párových sekvenčních čtení, nicméně lze maximalizovat přesnost při individuálním zarovnání obou čtení v páru.

Vedle možností při změně parametrů sekvenčních čtení lze měnit i délku referenční sekvence, což ovšem souvisí s problémem výpočetní náročnosti. V případě delších referenčních sekvencí se výrazně zpomaluje proces vytváření indexu, čímž se zarovnání stává časově neefektivní. Díky tomu dojde ke zpomalení výpočetního procesu diferenciální evoluce, což znesnadní průběh experimentů.

## 7.1 Metacentrum

Proces samotného zarovnání sekvenčních čtení a vyhodnocení fitness funkce je časově a výpočetně náročný. V případě algoritmu diferenciální evoluce se tyto náročné výpočty sekvenčně opakují v každé generaci výpočtu. Za tímto účelem byly provedeny optimalizace minimalizující počet těchto operací (viz sekce 6.5). Dále je výpočet hodnot fitness funkce všech kandidátních řešení v dané populaci realizován pomocí spuštění paralelních procesů. V případě platformy s možností paralelního zpracování více vláken se proces výpočtu několikanásobně urychlí.

Za tímto účelem byla využita distribuovaná výpočetní infrastruktura MetaCentra[1]. Jedná se o systém rezervací výpočetních zdrojů za účelem provádění náročných výpočtů. Implementace a jednotlivé součásti výpočetního frameworku jsou navrženy tak, aby bylo možné jednoduché spuštění na MetaCentru a následný zisk výsledků v textové podobě. Pro spuštění úlohy je nutné nejprve nahrát všechny komponenty potřebné ke spuštění výpočtu

[1] http://metavo.metacentrum.cz/cs/

na cílové úložiště rezervovaných počítačů a následně spustit program či výpočetní skript. Pro rezervaci se používá rezervační řetězec obsahující mimo dalšího nastavení zejména počet jader nutných pro výpočet, velikost operační paměti a velikost místa na disku. Současně je nutné vybrat vhodnou frontu, do které bude požadavek na spuštění úlohy umístěn. Fronty se liší maximální délkou běhu spuštěné úlohy. Pomocí opakovaných spuštění bylo zjištěno, že pro spuštění optimalizace je vhodné zadat parametry rezervačního řetězce, uvedené v tabulce 7.1.

| Počet CPU | RAM | HDD | Fronta |
|---|---|---|---|
| popSize + 1 | 40 GB | 100 GB | q_1d |

Tabulka 7.1: Parametry pro nastavení rezervačního řetězce.

Na první pohled vyšší požadavky vyplývají zejména z výpočetní náročnosti vytvoření indexu a zarovnání u programu LAST. Velikost paměti RAM souvisí i s počtem CPU, jejichž počet však během experimentů nikdy nepřekročil 16. Typ fronty q_1d odpovídá frontě s maximální dobou běhu skriptu 1 dne. V případě menšího počtu CPU nebo méně paměti RAM může dojít k vyčerpání zdrojů a předčasnému neúspěšnému ukončení výpočtu. V případě testování programu BLAT na reálných datech získaných z přístroje Illumina je nutné použít frontu s maximální dobou běhu 1 týden.

V průběhu experimentů bylo spuštěno 1 199 úloh s celkovou spotřebou 1 272,7 dnů procesorového času. Součástí těchto úloh byla i pokusná spuštění za účelem ladění výpočetního frameworku.

## 7.2 Generovaná sekvenční čtení

Za účelem provádění experimentů byla vytvořena datová sada sekvenčních čtení, která nesou empirické vlastnosti sekvenčních čtení přístroje Illumina. Existuje množství generátorů, které z referenční sekvence ve formátu FASTA vygenerují množinu sekvenčních čtení zadané délky. Jedním z požadavků je možnost explicitně zadané hustoty pokrytí referenční sekvence vzniklými sekvenčními čteními. Generátory lze využít pro generování jak jednoduchých, tak párových sekvenčních čtení. Velkou výhodou těchto generátorů je rychlost, která je oproti sekvenačním přístrojům mnohem vyšší. Dále je možné zvolit libovolnou referenční sekvenci, libovolnou délku sekvenčních čtení a libovolně násobné pokrytí. Nevýhodou je softwarový přístup, díky kterému se vlastnosti sekvenčních čtení pouze blíží simulovaným originálům. V této práci byl použit program ART, který oproti jiným testovaným má výhodu přímého vytvoření referenčního zarovnání ve formátu SAM. Referenční zarovnání je následně použito pro vytvoření zlatého standardu programem RABEMA (viz sekce 6.5). Množina sekvenčních čtení je standardně uložena v souboru ve formátu FASTQ.

Trénovací datové sady sekvenčních čtení pro optimalizaci jednotlivých nástrojů se vzájemně liší. Hlavní odlišností je délka referenční sekvence. Za účelem testování byla použita referenční sekvence lidského chromozomu X (NC_000023.11) z databáze GenBank[2] a pro další použití byly použity některé její části. Nejdříve byla v obou případech použita referenční sekvence stejné délky 100 kilobází. V obou případech se doba zarovnání vytvořených sekvencí pohybovala v řádu vteřin. V případě programu BLAT však vlivem změn hodnot parametrů docházelo k nárůstu doby běhu až na několik minut. V případě programu LAST nebyl nárůst tak zásadní, proto byla použita delší referenční sekvence délky 500 kilobází.

[2] http://www.ncbi.nlm.nih.gov/genbank/

S tím souvisí i nárůst počtu sekvenčních čtení, protože v obou případech byly při generování použity stejné parametry pokrytí. Pro testování různých délek sekvenčních čtení bylo vybráno 5 zástupců délek v rozmezí od 50 do 100 bází. Pro všechny délky byla provedena analýza maximálního počtu zarovnaných sekvenčních čtení při výchozí rozlišovací schopnosti programu RABEMA a výsledky jsou v tabulce 6.3 (viz kapitola 6.5). Celkový přehled parametrů obou datových sad je v tabulce 7.2

| Název programu | LAST | BLAT |
|---|---|---|
| Název referenční sekvence | LAST_Gen | BLAT_Gen |
| Délka referenční sekvence | 500 kilobází | 100 kilobází |
| Počet sekvenčních čtení (50 bází) | 99 990 | 20 000 |
| Počet sekvenčních čtení (65 bází) | 76 910 | 15 380 |
| Počet sekvenčních čtení (75 bází) | 66 660 | 13 330 |
| Počet sekvenčních čtení (90 bází) | 55 550 | 11 110 |
| Počet sekvenčních čtení (100 bází) | 49 995 | 10 000 |

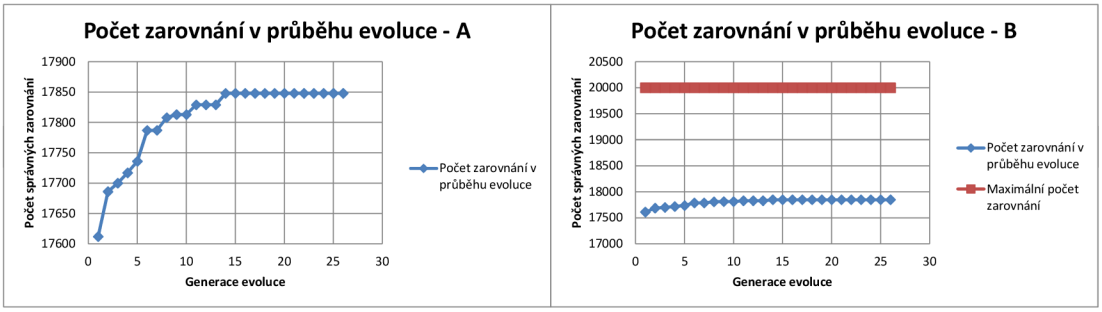
Tabulka 7.2: Přehled počtu sekvenčních čtení v jednotlivých datových sadách.

Referenční sekvence byla sestavena z náhodně vybraných genů lidského chromozomu X. Tyto geny byly sestaveny za sebou do dvou referenčních sekvencí zmíněných délek. Při generování sekvenčních čtení z těchto referenčních sekvencí pomocí programu ART bylo použito desetinásobné pokrytí. Jelikož se jedná o softwarový generátor a výsledné pokrytí je rovnoměrné, čímž se výsledky liší od reálných sekvenčních čtení, produkovaných sekvenačními přístroji.

Vytvořené datové sady byly použity pro experimenty s optimalizačním frameworkem. Dostatečný počet sekvenčních čtení k zarovnání je klíčovým požadavkem pro správnost výsledků. Pro následné ověření této správnosti je nutné provést stejné experimenty i s datovou sadou vytvořenou z reálných sekvenčních čtení.

### 7.2.1 Experimenty s programem BLAT

Na obrázku 7.1 (A) je znázorněn vývoj nejlepšího kandidátního řešení v průběhu evoluce. Při experimentu byla použita datová sada 20 000 nepárových sekvenčních čtení délky 50 bází. Zarovnávacím nástrojem byl program BLAT. V první generaci bylo správně zarovnáno 17 612 sekvenčních čtení a tato hodnota byla v průběhu evoluce zlepšena na 17 848, což je v porovnání s maximálním počtem nárůst o 1,2 %. Toto srovnání s maximálním počtem je znázorněno na obrázku 7.1 (B).



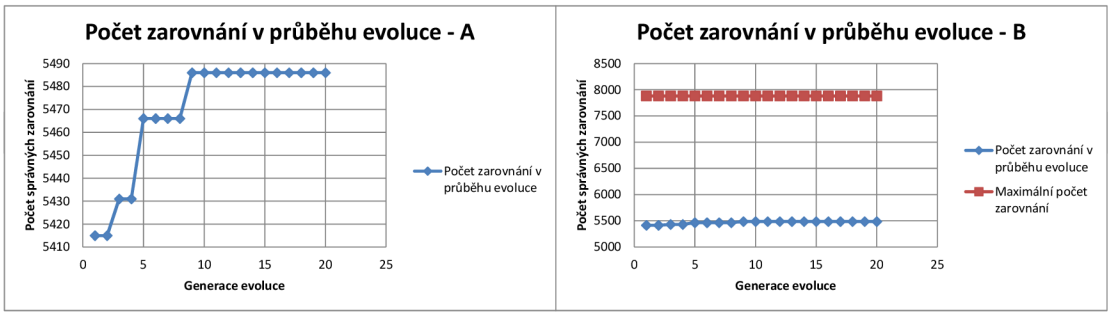

Obrázek 7.1: Vývoj nejlepšího kandidátního řešení v průběhu optimalizace programu BLAT.

Bylo provedeno 7 běhů tohoto experimentu s občasnými modifikacemi omezení hodnot parametrů. Nejlepší kandidátní řešení jsou v následující tabulce 7.3. Rozdíl mezi počtem zarovnaných sekvenčních čtení u obou nalezených řešení je v porovnání s vysokým celkovým počtem zanedbatelný. Rozdíl v době běhu zarovnání je však značný. V tabulce je pro srovnání i výsledek běhu zarovnání s výchozími parametry. Rozdíl v počtu zarovnaných sekvenčních čtení oproti běhu s optimálním nastavením parametrů je průměrně 1 295 sekvenčních čtení, což v porovnání s celkovým počtem sekvenčních čtení vykazuje nárůst přesnosti o 6,4 %.

| Pořadí | Konfigurace | Fitness hodnota | Doba zarovnání |
|---|---|---|---|
| 1. | tileSize=6 stepSize=1 oneOff=0 | 17 848 | 2,3 h |
| 2. | tileSize=6 stepSize=4 oneOff=0 | 17 787 | 14 min |
| výchozí | tileSize=11 stepSize=11 oneOff=0 | 16 522 | 10 s |

Tabulka 7.3: Výsledky optimalizace programu BLAT na datové sadě sekvenčních čtení délky 50 bází.

Při dalších experimentech bylo manipulováno zejména s délkou sekvenčních čtení a jejich typem. Ve všech případech dochází k velkému nárůstu doby zarovnání v případě maximálního zvýšení citlivosti při vytváření indexu. Na grafu na obrázku 7.2 (A) je zopakován stejný experiment jako v případě obrázku 7.1 s tím rozdílem, že délka sekvenčního čtení je 100 bází. Celkový počet sekvenčních čtení je podle tabulky 7.2 10 000, z čehož pouze 7 882 sekvenčních čtení lze v rámci dané rozlišovací schopnosti testovat na správnost zarovnání. Srovnání s maximálním počtem zarovnání je znázorněno na vedlejším grafu na obrázku 7.2 (B).



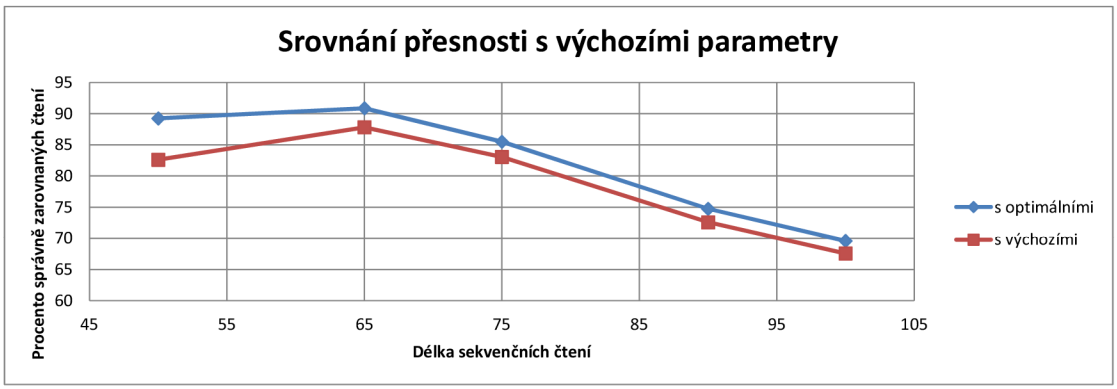

Obrázek 7.2: Vývoj nejlepšího kandidátního řešení v průběhu optimalizace programu BLAT.

Jako i v předešlém případě bylo provedeno 7 experimentů, které se lišily nastavením povolených rozsahů parametrů. Kompletní výsledky na všech datových sadách nepárových sekvenčních čtení různých délek jsou v tabulce 7.4.

| Délka | Pořadí | Konfigurace | Fitness | Max. | Čas |
|---|---|---|---|---|---|
| 50 bází | 1. | tileSize=6 stepSize=1 oneOff=0 | 17 848 | 20 000 | 2,3 h |
| 50 bází | 2. | tileSize=6 stepSize=4 oneOff=0 | 17 787 | 20 000 | 14 min |
| 50 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 16 522 | 20 000 | 10 s |
| 65 bází | 1. | tileSize=6 stepSize=1 oneOff=0 | 13 972 | 15 375 | 2,8 h |
| 65 bází | 2. | tileSize=6 stepSize=4 oneOff=0 | 13 945 | 15 375 | 24,3 min |
| 65 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 13 503 | 15 375 | 10 s |
| 75 bází | 1. | tileSize=6 stepSize=1 oneOff=1 | 11 375 | 13 304 | 3,2 h |
| 75 bází | 2. | tileSize=6 stepSize=4 oneOff=1 | 11 340 | 13 304 | 23,6 min |
| 75 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 11 051 | 13 304 | 12 s |
| 90 bází | 1. | tileSize=6 stepSize=1 oneOff=0 | 7 631 | 10 209 | 3,6 h |
| 90 bází | 2. | tileSize=6 stepSize=4 oneOff=1 | 7 600 | 10 209 | 40,4 min |
| 90 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 7 410 | 10 209 | 12 s |
| 100 bází | 1. | tileSize=6 stepSize=1 oneOff=0 | 5 486 | 7 882 | 3,9 h |
| 100 bází | 2. | tileSize=6 stepSize=4 oneOff=0 | 5 457 | 7 882 | 29,8 min |
| 100 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 5 327 | 7 882 | 12 s |

Tabulka 7.4: Výsledky optimalizace programu BLAT na všech datových sadách sekvenčních čtení.

Ve všech případech je znatelný rozdíl fitness hodnoty mezi zarovnáním s optimálními parametry a zarovnáním s výchozími parametry. V porovnání s maximálním počtem zarovnaných sekvenčních čtení dochází ve všech případech délky sekvenčního čtení k nárůstu přesnosti zarovnání o několik procent. Nejlepším řešení je v téměř všech případech kandidátní řešení s konfigurací parametrů tileSize=6, stepSize=1 a oneOff=0. Porovnání procentuální úspěšnosti pro všechny zkoumané délky sekvenčních čtení je znázorněno grafem na obrázku 7.3. Červená křivka odpovídá zarovnání s výchozími parametry a modrá křivka odpovídá zarovnání s optimálními parametry. Průměrně dochází ke zlepšení přesnosti o 3,27 % a v extrémním případě u sekvenčních čtení délky 50 bází o 6,63 %. S rostoucí délkou sekvenčního čtení klesá vliv optimalizace při zarovnání programem BLAT.

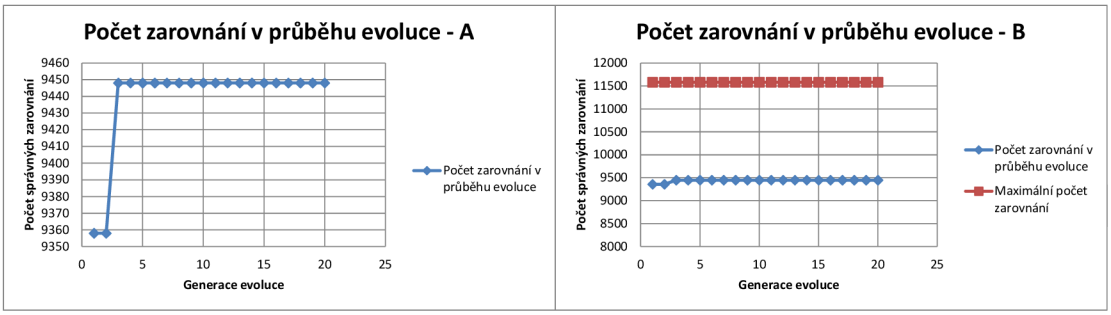



Obrázek 7.3: Srovnání úspěšnosti optimálních a výchozích parametrů.

Další experimenty byly prováděny s párovými sekvenčními čteními. Parametry generátoru sekvenčních čtení ART zůstaly stejné. Při vytváření párových sekvenčních čtení byla zvolena délka fragmentu 200 bází, což odpovídá vlastnostem reálných sekvenčních čtení, kdy se velikost fragmentu pohybuje od 200 do 500 bází (viz kapitola 3). Střední odchylka

velikosti fragmentu byla nastavena na 10 bází. Pokrytí referenční sekvence zůstalo desetinásobné a délky sekvenčních čtení odpovídají délkám z předchozího experimentu. Velikost datové sady je stejná jako v případě nepárových sekvenčních čtení. Počet párů odpovídá polovičnímu počtu nepárových čtení.

Na obrázku 7.4 (A) je znázorněn graf vývoje nejlepšího kandidátního řešení v průběhu evoluce a na vedlejším obrázku 7.4 (B) je srovnání s maximálním počtem zarovnaných sekvenčních čtení. V případě párových sekvenčních čtení není přímo využito znalosti o velikosti fragmentu, proto je každé sekvenční čtení v páru zarovnáno samostatně a výsledkem je množina takových zarovnání, která sestávají z obou zarovnaných čtení v páru. Tato množina je následně filtrována a příliš vzdálená sekvenční čtení v páru jsou odstraněna.




Obrázek 7.4: Výsledky optimalizace programu BLAT na datové sadě sekvenčních čtení délky 50 bází.

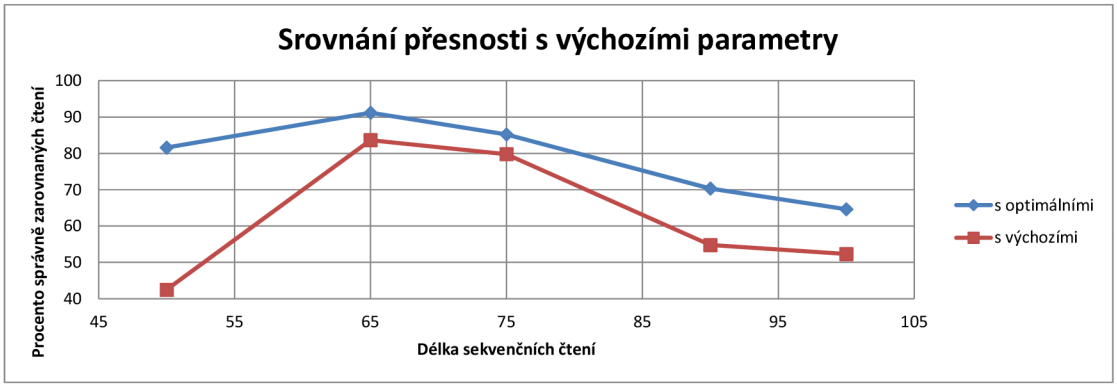
V této kategorii bylo provedeno 5 opakovaných běhů, které se jako i v předešlém případě lišily zejména nastavením rozsahů hodnot optimalizovaných parametrů. Další změnou byla délka sekvenčních čtení, vybíraná z rozsahu od 50 do 100 bází. V případě párových sekvenčních čtení byly očekávány stejné výsledné parametry jako v případě nepárových sekvenčních čtení z důvodu nezávislého zarovnání obou čtení v páru. V tabulce 7.5 jsou znázorněny výsledky běhů nástroje BLAT při zarovnání párových sekvenčních čtení různých délek.

| Délka | Pořadí | Konfigurace | Fitness | Max. | Čas |
|---|---|---|---|---|---|
| 50 bází | 1. | tileSize=6 stepSize=1 oneOff=0 | 9 448 | 11 581 | 2,2 h |
| 50 bází | 2. | tileSize=6 stepSize=4 oneOff=0 | 9 289 | 11 581 | 9,9 min |
| 50 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 4 918 | 11 581 | 10 s |
| 65 bází | 1. | tileSize=6 stepSize=1 oneOff=0 | 11 848 | 12 993 | 2,9 h |
| 65 bází | 2. | tileSize=6 stepSize=4 oneOff=0 | 11 787 | 12 993 | 18,6 min |
| 65 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 10 869 | 12 993 | 10 s |
| 75 bází | 1. | tileSize=6 stepSize=1 oneOff=0 | 9 084 | 10 661 | 3,2 h |
| 75 bází | 2. | tileSize=6 stepSize=4 oneOff=0 | 9 056 | 10 661 | 30,9 min |
| 75 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 8 506 | 10 661 | 12 s |
| 90 bází | 1. | tileSize=6 stepSize=1 oneOff=1 | 4 644 | 6 602 | 3,6 h |
| 90 bází | 2. | tileSize=6 stepSize=4 oneOff=0 | 4 599 | 6 602 | 30,6 min |
| 90 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 3 617 | 6 602 | 12 s |
| 100 bází | 1. | tileSize=6 stepSize=1 oneOff=0 | 3 110 | 4 812 | 3,8 h |
| 100 bází | 2. | tileSize=6 stepSize=4 oneOff=0 | 3 086 | 4 812 | 36,6 min |
| 100 bází | výchozí | tileSize=11 stepSize=11 oneOff=0 | 2 517 | 4 812 | 12 s |

Tabulka 7.5: Výsledky optimalizace programu BLAT na všech datových sadách sekvenčních čtení.

Výsledky jsou podle předpokladu téměř totožné s výsledky experimentů s nepárovými sekvenčními čteními. Nejlepší kandidátní řešení má opět kofiguraci tileSize=6, stepSize=1, oneOff=0. Mezi prvním a druhým vítězem v testech je menší rozdíl v počtu správně zarovnaných sekvenčních čtení, ale rozdíl v době běhu je enormní jako i v případě nepárových sekvenčních čtení. U maximálního počtu zarovnaných sekvenčních čtení je u první datové sady mnohem nižší číslo, než bylo očekáváno, což je způsobeno rozlišovacím rozsahem programu RABEMA. V tomto případě je celkový počet sekvenčních čtení snížen na pouhých 55 %. I přes tento nedostatek je optimalizace úspěšná a na této datové sadě dochází k průměrnému zvýšení přesnosti vlivem optimalizace o 38 %. Tento výsledek se ovšem velmi liší od celkového průměru, proto jej lze považovat za náhodný a celkovou úspěšnost by neměl ovlivnit.

Na obrázku 7.5 je znázorněno srovnání běhu zarovnávacího nástroje s výchozími parametry, což znázorňuje červená křivka, a s optimálními parametry, čemuž odpovídá modrá křivka. Průměrný nárůst přesnosti ve smyslu počtu správně zarovnaných sekvenčních čtení je 10,2 %, což je víc než v případě nepárových sekvenčních čtení. Do výpočtu průměrného zlepšení není započtena extrémní hodnota nárůstu přesnosti 39,1 % v datové sadě sekvenčních čtení délky 50 bází.

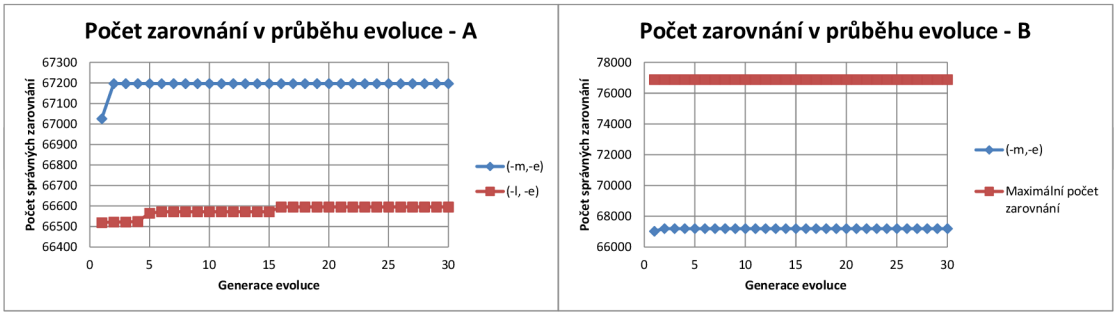



Obrázek 7.5: Srovnání úspěšnosti optimálních a výchozích parametrů.

V průběhu experimentů bylo manipulováno zejména s nastavením vlastností diferenciální evoluce. Velikost populace se pohybovala od 10 do 12 kandidátních řešení. Počet generací byl vzhledem k menšímu počtu možných kombinací hodnot parametrů volen mezi 20 a 30 generacemi. Výsledky potvrdily původní očekávání, že nejvyšší přesnosti dosáhne program při maximální povolené citlivosti při vytváření indexu. Výsledky pro nepárová i párová sekvenční čtení jsou pro jednotlivé délky zobrazeny v tabulkách 7.4 a 7.5. Za optimální řešení lze považovat obě nejlepší kandidátní řešení v každé kategorii, protože rozdíl v počtu správně zarovnaných sekvenčních čtení je v porovnání s celkovým počtem zanedbatelný.

### 7.2.2 Experimenty s programem LAST

Stejně jako u programu BLAT byl prvním experimentem test růstu přesnosti v průběhu evoluce. Program LAST se ovšem skládá ze dvou oddělených programů, u nichž probíhá optimalizace parametrů odděleně. Další odlišností je velikost datové sady sekvenčních čtení, jejichž počet odpovídá delší referenční sekvenci. Na obrázku 7.6 (A) je znázorněn graf vývoje nejlepšího kandidátního řešení při optimalizaci parametrů programu LASTAL, který provádí samotné zarovnání s využitím již vytvořeného indexu. K experimentu byla využita datová sada nepárových sekvenčních čtení délky 65 bází. Modrá křivka odpovídá zarovnání při optimalizaci parametrů mezní frekvence výskytu při rozšiřování semínka a minimálního skóre zarovnání. Červená křivka znázorňuje zarovnání při optimalizaci maximální délky semínka a minimálního skóre. Vedlejší graf na obrázku 7.6 (B) znázorňuje srovnání maximálního počtu správných zarovnání a zarovnání odpovídající lepšímu z obou na předchozím grafu na obrázku 7.6 (A).




Obrázek 7.6: Výsledky optimalizace programu LAST na datové sadě sekvenčních čtení délky 65 bází.

Z grafu a výsledků je zřejmé, že parametr omezující maximální povolenou velikost semínka při jeho rozšiřování není vhodné omezovat. Výchozí hodnota je neomezená a výsledky jsou lepší v případě, že zůstane neomezená. Experiment potvrdil výhodu použití neomezené adaptivní délky semínka před variantou s omezenou délkou. V tabulce 7.6 je znázorněno optimální řešení ve srovnání s výchozím nastavením parametrů.

| Řešení | Konfigurace | Fitness hodnota | Doba běhu |
|---|---|---|---|
| optimální | m1 e88 | 67 197 | 6 s |
| výchozí | m10 e40 | 66 595 | 23 s |

Tabulka 7.6: Výsledky optimalizace programu LAST na datové sadě sekvenčních čtení délky 65 bází.

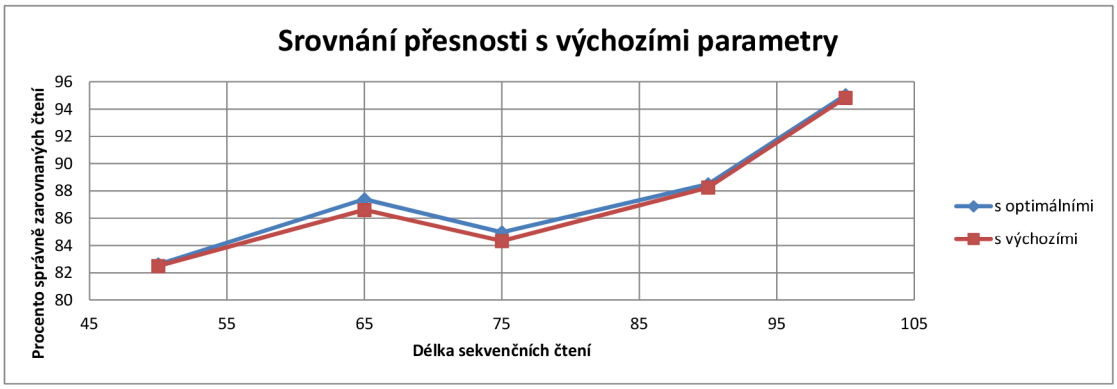
Zajímavou vlastností je zkrácení doby běhu na pouhých 6 sekund a současné zvýšení přesnosti. Mezní frekvence výskytu klesla z výchozích 10 výskytů na pouhý 1 výskyt, což podle očekávání sníží chybovost počátečního zarovnání semínka. Tato vlastnost ovlivnila pokles doby zarovnání i současné zvýšení přesnosti. Hodnota minimálního skóre se uplatní spíše při minimalizaci nesprávných zarovnání, která není součástí zvolené fitness funkce. Při tomto experimentu došlo k nárůstu přesnosti o 0,7 %. Program vykazuje i s použitím výchozích parametrů vysokou přesnost, proto k výraznějšímu zlepšení nedošlo.

V tabulce 7.7 jsou znázorněny výsledky, kterých bylo dosaženo na dalších datových sadách sekvenčních čtení různých délek.

| Délka | Řešení | Konfigurace | Fitness | Max. fitness | Čas |
|---|---|---|---|---|---|
| 50 bází | optimální | m1 e99 | 83 881 | 99 989 | 5 s |
| 50 bází | výchozí | m10 e40 | 82 491 | 99 989 | 15 s |
| 65 bází | optimální | m1 e88 | 67 197 | 76 890 | 6 s |
| 65 bází | výchozí | m10 e40 | 66 595 | 76 890 | 23 s |
| 75 bází | optimální | m1 e83 | 56 549 | 66 559 | 6 s |
| 75 bází | výchozí | m10 e40 | 56 130 | 66 559 | 28 s |
| 90 bází | optimální | m3 e99 | 45 468 | 51 381 | 14 s |
| 90 bází | výchozí | m10 e40 | 45 348 | 51 381 | 30 s |
| 100 bází | optimální | m3 e92 | 37 598 | 39 561 | 12 s |
| 100 bází | výchozí | m10 e40 | 37 519 | 39 561 | 46 s |

Tabulka 7.7: Výsledky optimalizace programu LASTAL na všech datových sadách sekvenčních čtení.

Následující graf na obrázku 7.7 znázorňuje srovnání optimálního a výchozího nastavení parametrů pomocí procentuální úspěšnosti. K nejvýraznějšímu zlepšení došlo na datové sadě sekvenčních čtení délky 65 bází o zmíněných 0,7 %, průměrně na všech sadách došlo ke zlepšení o 0,39 %.



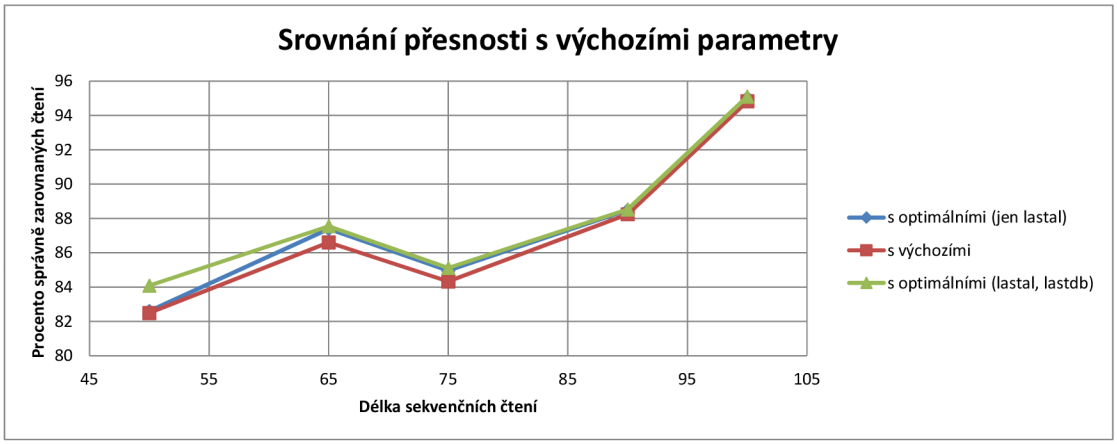

Obrázek 7.7: Srovnání úspěšnosti optimálních a výchozích parametrů.

Dalším krokem byla optimalizace vytvoření indexu. Zde už byla pouze jedna kombinace parametrů $m$ a $w$, což odpovídá prostorově rozšířenému semínku a indexačnímu kroku. V průběhu experimentů bylo manipulováno zejména s rozsahy jednotlivých parametrů, velikostí populace a počtem generací. Tabulka 7.8 obsahuje výsledky získané z běhů nad všemi použitými datovými sadami. Cílem experimentu bylo zejména nalezení optimálního semínka, u kterého není známá výchozí hodnota pro délky sekvenčních čtení větší než 50 bází. Při následném zarovnání bylo použito optimálních hodnot parametrů, které byly získány v předchozím experimentu.

| Délka | Řešení | Lastal | Lastdb | Fitness | Max. | Čas |
|---|---|---|---|---|---|---|
| 50 bází | optimální | m1 e99 | m11110111000110100101 w1 | 84 088 | 99 989 | 8 s |
| 50 bází | výchozí | m10 e40 | m1 w1 | 82 491 | 99 989 | 15 s |
| 65 bází | optimální | m1 e88 | m1110010110011101001 w1 | 67 318 | 76 890 | 10 s |
| 65 bází | výchozí | m10 e40 | m1 w1 | 66 595 | 76 890 | 23 s |
| 75 bází | optimální | m1 e83 | m111010011010100100 w1 | 56 656 | 66 559 | 13 s |
| 75 bází | výchozí | m10 e40 | m1 w1 | 56 130 | 66 559 | 28 s |
| 90 bází | optimální | m3 e99 | m1000110110010010110 w1 | 45 490 | 51 381 | 19 s |
| 90 bází | výchozí | m10 e40 | m1 w1 | 45 348 | 51 381 | 30 s |
| 100 bází | optimální | m3 e92 | m1101000100111000100 w1 | 37 623 | 39 561 | 26 s |
| 100 bází | výchozí | m10 e40 | m1 w1 | 37 519 | 39 561 | 46 s |

Tabulka 7.8: Výsledky optimalizace programu LAST na všech datových sadách sekvenčních čtení.

Z výsledků v tabulce 7.8 je zřejmé, že doba zarovnání je stále nižší v případě použití optimálních parametrů než v případě použití výchozích parametrů. V případě optimalizace obou programů jsou výsledky ještě lepší než v předchozím experimentu, kde byly optimalizovány parametry pouze u programu LASTAL. Výsledkem tohoto experimentu je průměrné zlepšení o 0,77 %, což je o něco více než v předchozím případě. Celkové srovnání s použitím výchozích hodnot parametrů je znázorněno grafem na obrázku 7.8. Zelená křivka odpovídá výsledkům tohoto experimentu, červená křivka běhu s výchozími parametry a modrá křivka reprezentuje výsledky z předchozího experimentu.




Obrázek 7.8: Srovnání úspěšnosti optimálních a výchozích parametrů.

Posledním experimentem v kategorii generovaných sekvenčních čtení je aplikace programu LAST při zarovnání párových sekvenčních čtení. Jejich množství odpovídá množství nepárových čtení, tedy počet párů je poloviční v porovnání s počtem nepárových čtení. Parametry generátoru jsou stejné jako v případě programu BLAT. Stejným způsobem se přistupuje i k výsledné množině zarovnání, která obsahuje pouze páry, které jsou zarovnané v povolené vzdálenosti, která odpovídá střední vzdálenosti 200 bází s odchylkou maximálně 10 bází. Současně také platí, že každý z obou čtení v páru je zarovnán na jiném ze dvou komplementárních vláken, což je základní vlastností párových čtení (viz kapitola 3).

V tabulce 7.9 jsou zobrazeny výsledky na jednotlivých datových sadách sekvenčních

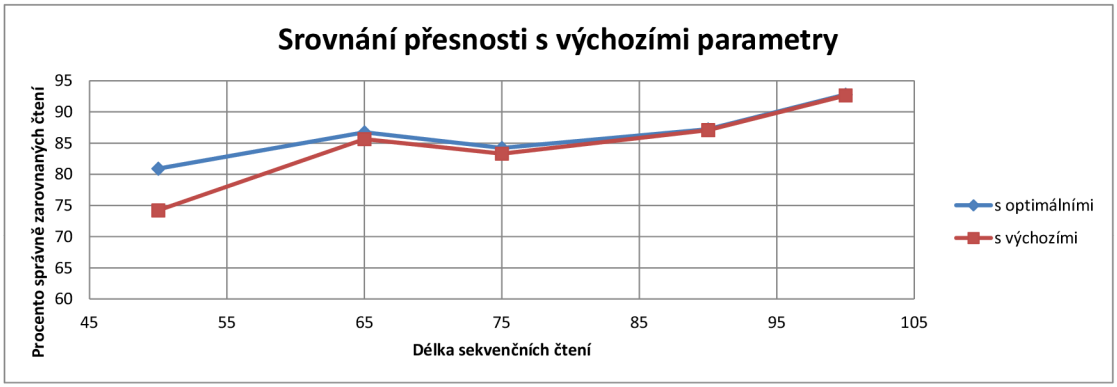
čtení různých délek. Nejprve proběhla optimalizace programu zarovnání a následně s již známými optimálními parametry tohoto programu proběhla optimalizace parametrů u programu vytvoření indexu. V tabulce je rovněž pro srovnání i výsledná přesnost v případě použití výchozích hodnot parametru.

| Délka | Řešení | Lastal | Lastdb | Fitness | Max. | Čas |
|---|---|---|---|---|---|---|
| 50 bází | optimální | m1 e57 | m11110101100011001101 w1 | 47 022 | 58 103 | 10 s |
| 50 bází | výchozí | m10 e40 | m1 w1 | 43 127 | 58 103 | 12 s |
| 65 bází | optimální | m1 e87 | m1111100101011000100 w1 | 56 416 | 65 039 | 8 s |
| 65 bází | výchozí | m10 e40 | m1 w1 | 55 718 | 65 039 | 40 s |
| 75 bází | optimální | m1 e68 | m111000111011000000101 w1 | 44 686 | 53 052 | 10 s |
| 75 bází | výchozí | m10 e40 | m1 w1 | 44 203 | 53 052 | 26 s |
| 90 bází | optimální | m5 e100 | m1001000011100001101 w1 | 28 439 | 32 599 | 21 s |
| 90 bází | výchozí | m10 e40 | m1 w1 | 28 393 | 32 599 | 29 s |
| 100 bází | optimální | m5 e92 | m11011100100100010100 w1 | 22 671 | 24 431 | 27 s |
| 100 bází | výchozí | m10 e40 | m1 w1 | 22 639 | 24 431 | 34 s |

Tabulka 7.9: Výsledky optimalizace programu LAST na všech datových sadách sekvenčních čtení.

Problém nízké citlivosti programu RABEMA v případě párových sekvenčních čtení délky 50 bází je stejný jako v případě experimentů s programem BLAT. Maximální počet zarovnaných sekvenčních čtení je i v tomto případě necelých 55 %, ale 58 103 sekvenčních čtení je stále dostatečně velká množina, na které lze najít optimální hodnoty parametrů. Ve všech případech se ukázalo nejlepší stanovit indexační krok na jeho výchozí hodnotu. Vytváření indexu je proto maximálně citlivé. Jednotlivá optimální prostorově rozšířená semínka jsou vytvořena pro každou délku sekvenčního čtení. V průběhu evoluce se však ukazuje více možných řešení se stejnou výslednou fitness hodnotou. Pro lepší rozhodnutí a výběr jediného vítězného kandidáta by bylo třeba zvolit jinou fitness funkci (viz sekce 7.4), nebo experiment opakovat a za optimální výsledek označit nejčastější výsledek.

Na obrázku 7.9 je znázorněno srovnání optimálních a výchozích parametrů. Stejně jako v předešlých případech se hodnotí procento správně zarovnaných sekvenčních čtení. Průměrně došlo ke zlepšení o 0,57 %, což je v porovnání s nepárovými sekvenčními čteními nepatrně méně. Do průměru ovšem opět nebyla zahrnuta první hodnota, která díky nízkému maximálnímu počtu správně zarovnaných sekvenčních čtení vykazovala zlepšení o 6,7 %.




Obrázek 7.9: Srovnání úspěšnosti optimálních a výchozích parametrů.

Počet generací se měnil v rozsahu od 50 do 120 generací a v populaci bylo maximálně 15 kandidátních řešení. Počet generací byl v porovnání s programem BLAT vyšší z důvodu velkého množství povolených hodnot parametru prostorově rozšířeného semínka. Celkově došlo ke zlepšení přesnosti zarovnání o 0,57 % na datových sadách párových sekvenčních čtení a o 0,77 % na sadách s nepárovými čteními. Míra zlepšení je v porovnání s programem BLAT nižší, avšak není kompenzována takovým nárůstem doby běhu. Při použití optimálních parametrů došlo i ke zrychlení zarovnání o několik vteřin, což není pro výsledek nijak důležité. Dalším přínosem provedených experimentů bylo potvrzení lepší přesnosti při použití adaptivní délky semínka v porovnání s aplikací omezené délky semínka pevně stanovenou hodnotou.

## 7.3 Reálná sekvenční čtení

Druhou skupinou experimentů je provádění zarovnání sekvenčních čtení získaných z přístroje Illumina. Za tímto účelem byl z evropského bioinformatického archivu sekvenčních čtení[3] (SRA) vybrán experiment (DRR000534). Jedná se o sekvenaci exonů lidského chromozomu X. Výsledkem je datová sada nepárových sekvenčních čtení délky 75 bází. Sekvenční čtení v této datové sadě vykazují 50,6 % zastoupení bází cytozinu a guaninu a 49,4 % zastoupení bází adeninu a tyminu. Z popisu experimentu vyplývá, že se jedná o experiment sekvenace pouze exonových oblastí, proto je možné jako referenční sekvence použít seznam anotovaných exonů.

V případě programu LAST, který je nesrovnatelně rychlejší v porovnání s programem BLAT, lze jako referenční sekvenci použít celý chromozom X. Problém nastává v případě programu BLAT, u kterého samotné vytvoření indexu trvá řádově desítky hodin pro jisté konfigurace parametrů. Jednou z možností je místo jedné referenční sekvence vytvořit množinu referenčních sekvencí, které by odpovídaly anotovaným exonům na chromozomu X. Tato možnost bohužel není použitelná díky programu RABEMA, u kterého dochází vlivem vysokého počtu referenčních sekvencí k interním chybám. Proto byla za tímto účelem sestavena podobná referenční sekvence jako v případě generovaných sekvenčních čtení a jednotlivé exony byly spojeny do jedné sekvence. Nevýhodou tohoto spojení je možná nepřesnost v cílovém zarovnání, ale jelikož neexistuje žádné referenční zarovnání a je nutné jej vytvořit až na vzniklé sekvenci, lze tuto variantu použít.

[3] http://www.ncbi.nlm.nih.gov/sra

Druhým problémem je velké množství sekvenčních čtení. Většina bioinformatických experimentů, které se týkají lidských chromozomů a lidského genomu, obsahují obrovské množství sekvenčních čtení. Z důvodu časové náročnosti nelze zarovnat všechna sekvenční čtení. Za tímto účelem je vytvořena omezená množina sekvenčních čtení, která se při zarovnání využije. V případě programu BLAT obsahuje množina 12 000 sekvenčních čtení a program LAST vyžije množinu s 20 000 sekvenčními čteními. Výběr je realizován náhodně z množiny 76 milionů sekvenčních čtení. Pro každý program je tímto způsobem náhodného výběru vytvořeno 10 datových sad a při zarovnání je náhodně vybrána jedna z nich.

Ke každé sadě je nutné vytvořit již popsaný zlatý standard (viz sekce 6.5). Problémem je absence referenčního zarovnání, které při vzniku reálných sekvenčních čtení nelze získat. Podle návodu programu RABEMA lze získat referenční zarovnání pomocí nástroje RazerS, který je spuštěn s parametry zaručujícími maximální přesnost. Tímto způsobem bylo vytvořeno referenční zarovnání pro každou datovou sadu reálných sekvenčních čtení, které slouží pro vytvoření zlatého standardu. Na každé datové sadě vyjde pro konkrétní kandidátní řešení odlišná hodnota fitness, proto je nutná úprava algoritmu. V případě, že testované kandidátní řešení již bylo ohodnoceno fitness funkcí, je výsledná hodnota fitness vypočítána jako průměr původní a nové. Tím vznikne pro obě kandidátní řešení nová hodnota fitness. Před následnou aktualizací nejlepšího kandidátního řešení v průběhu evoluce je jeho fitness hodnota nahrazena touto průměrnou hodnotou v případě, že se jedná o kandidátní řešení se stejnou konfigurací. Výsledkem je pak v ideálním případě průměrná fitness hodnota na několika náhodně zvolených datových sadách.

### 7.3.1 Experiment s programem BLAT

Experiment s reálnými sekvenčními čteními byl nejprve proveden s použití nástroje BLAT. Vzhledem k rozsáhlé referenční sekvenci probíhaly výpočty velmi pomalu, což je znázorněno výslednými časy v tabulce 7.10. Pro experiment byla použita populace s pouze 8 kandidátními řešeními a evoluce probíhala pouze 12 generací. Délka celého běhu byla více než 120 hodin a dosažený výsledek nemusí být zcela optimální. Na získání optimálního řešení by bylo třeba mnohem více generací.

| Řešení | Konfigurace | Fitness/Max. fitness | Doba běhu |
|---|---|---|---|
| optimální | tileSize=6 stepSize=8 oneOff=0 | 2 805/3 355 | 10,6 h |
| výchozí | tileSize=11 stepSize=11 oneOff=0 | 2 724/3 355 | 10 s |

Tabulka 7.10: Výsledky optimalizace programu BLAT na datových sadách reálných sekvenčních čtení.

Navzdory obtížnému testování programu BLAT byl experiment úspěšný a došlo v tomto případě ke zlepšení přesnosti o 2,4 %, což je stejné jako v případě generovaných sekvenčních čtení, kde došlo ke zlepšení na této délce sekvenčního čtení o 2,44 %. Vzhledem k tomu, že se během prvních 12 generací neobjevila konfigurace s indexačním krokem délky 1, je pravděpodobné, že tato konfigurace není optimální, jak je tomu u generovaných sekvenčních čtení, z čehož vyplývá, že index vytvořený s maximální citlivostí nemusí být vždy lepší.

Zůstává problém extrémního nárůstu doby zarovnání, způsobeného především konstrukcí indexu. Pro zarovnání většího počtu sekvenčních čtení a při použití delší referenční sekvence, která je v tomto případě dlouhá téměř 38 megabází, je běh výpočetního frameworku i samotné zarovnání časově velmi náročné.

### 7.3.2 Experiment s programem LAST

Závěrečným experimentem byla optimalizace parametrů nástroje LAST za použití datových sad reálných sekvenčních čtení. Postup byl stejný jako v případě generovaných sekvenčních čtení. Prvním krokem byla optimalizace zarovnávacího programu LASTAL a navazujícím krokem byla optimalizace programu LASTDB pro vytvoření indexu s následným zarovnáním s dříve optimalizovanými parametry. Referenční sekvencí je v tomto případě celý lidský chromozom X a je využito opět 10 různých datových sad, vytvořených z náhodně vybraných sekvenčních čtení.

V tabulce 7.11 jsou zobrazeny výsledky z tohoto experimentu. Se získanou optimální konfigurací parametrů bylo provedeno 12 běhů na náhodných datových sadách. Z těchto běhů byly získány výsledky fitness hodnot, referenční maximální fitness i doba zarovnání. Průměrné výsledky jsou přesnější než jeden výsledek na jedné z testovacích sad.

| Řešení | Lastal | Lastdb | Fitness | Max. fit. | Doba běhu |
|---|---|---|---|---|---|
| optimální | m1 e83 | m1110100100000110011 w5 | 7 920 | 13 351 | 1,1 min |
| generované | m1 e68 | m1110100110101000100 w1 | - | - | - |
| výchozí | m10 e40 | m1 w1 | 7 720 | 13 351 | 2,1 min |

Tabulka 7.11: Výsledky optimalizace programu LAST na datových sadách reálných sekvenčních čtení.

Z výsledků je zřejmé, že vlivem optimalizace došlo opět ke zkrácení celkové doby zarovnání jako u předešlých experimentů s programem LAST. Současně došlo ke zlepšení přesnosti o 1,4 %. Stejně jako v případě programu BLAT je hlavním rozdílem oproti výsledkům dosažených na datové sadě generovaných sekvenčních čtení velikost indexačního kroku. V případě generovaných sekvenčních čtení je nejvhodnější použít maximální citlivost, která v případě reálných sekvenčních čtení není optimální. Nárůst přesnosti není tak výrazný jako v případě programu BLAT, ale je výraznější než v případě programu LAST při aplikaci na generovaná sekvenční čtení. Pro srovnání je zobrazen i výsledek prostorově rozšířeného semínka u generovaných sekvenčních čtení délky 50 bází, které je z 57,9 % totožné se získaným semínkem v tomto experimentu.

Velikost populace byla při tomto experimentu stanovena na 15 kandidátních řešení. Parametr počtu generací se od jiných experimentů s programem LAST nelišil. Při optimalizaci programu vytvoření indexu bylo použito 120 generací, v případě programu zarovnání postačilo 20 až 30 generací.

### 7.3.3 Shrnutí

U obou testovaných programů došlo podle očekávání k nárůstu přesnosti vlivem optimalizace parametrů pomocí optimalizačního frameworku. V případě programu BLAT vzrostla přesnost o 2,4 %, což bylo cílem i za cenu poklesu rychlosti zarovnání. V případě programu LAST došlo ke zvýšení přesnosti o 1,4 % a současně došlo k poklesu doby zarovnání. U obou programů se výsledky liší od výsledků dosažených na datových sadách generovaných sekvenčních čtení, a to zejména v délce indexačního kroku, který je v případě reálných sekvenčních čtení delší. V případě kratšího indexačního kroku by mohla být některá sekvenční čtení zarovnána nesprávně.

## 7.4 Možná rozšíření

Optimalizační framework je navržen za účelem optimalizace ve smyslu maximalizace počtu správně zarovnaných sekvenčních čtení. Dalším možným hodnotícím kritériem může být minimalizace počtu nesprávných zarovnání. V takovém případě by bylo třeba změnit algoritmus výpočtu fitness funkce. Výsledek programu RABEMA obsahuje údaj o počtu nesprávných zarovnání, čili je možné ho pro tento účel využít. V případě vytvoření fitness funkce založené na jiné kvantitativně hodnocené vlastnosti výsledného zarovnání je možné zaměnit program RABEMA za jiný hodnotící program, který lépe vyhovuje požadavkům.

Dalším možným rozšířením je použití delších sekvenčních čtení. V práci byla za účelem provádění experimentů použita délka čtení v rozmezí od 50 do 100 bází. Současné sekvenační přístroje jsou schopné produkovat i delší sekvenční čtení a jedním z dlouhodobých cílů je jejich další prodlužování. Optimalizační framework je na tuto možnost připraven. Délka sekvenčních čtení je jedním z parametrů běhu, které lze libovolně měnit.

Zarovnání lze realizovat s jednou datovou sadou sekvenčních čtení, nebo lze vygenerovat více sad a náhodně vybírat mezi nimi, jak je tomu například při realizaci experimentů s reálnými sekvenčními čteními. Jinou alternativou by mohlo být vytvoření množiny sekvenčních čtení před každým spuštění fitness funkce. V takovém případě by ale bylo nutné vždy vytvořit odpovídající zlatý standard k vybranému referenčnímu zarovnání. Vytvoření zlatého standardu však může být při větším množství sekvenčních čtení nebo při použití delší referenční sekvence časově náročnějším procesem, který by zpomalil běh výpočetního frameworku.

# Kapitola 8

# Závěr

Pomocí sekvenačních metod lze získat důležité informace o zkoumaném genomu. V práci byly popsány různé sekvenační metody, které se všechny vyznačují masivním paralelismem a generováním obrovského množství dat. Jejich společným výstupem je knihovna nasekvenovaných párových či nepárových sekvenčních čtení. Následujícím krokem je zarovnání k referenční sekvenci nebo sestavení sekvence *de novo*. V obou případech se využívá zarovnávacích nástrojů, které jsou programovou realizací vyhledávacích algoritmů založených na principu hashovací tabulky nebo prefixových či sufixových polí (viz kapitola 4). Tyto zarovnávací nástroje využívají při zarovnání délku sekvenčních čtení a jejich kvalitu, která je specifická každému sekvenačnímu přístroji.

Cílem zarovnávacích nástrojů je provést zarovnání co nejpřesněji. Vlastnosti takového zarovnání lze ovlivnit nastavením parametrů, mezi které patří zejména velikost semínka pro počáteční zarovnání, povolení neshod v tomto zarovnání a délka indexačního kroku při vytváření indexu pro vyhledání shody. Pro účely optimalizace byly vybrány dva nástroje LAST a BLAT, u kterých je cílem nalézt optimální hodnoty zmíněných parametrů, aby bylo výsledné zarovnání co nejpřesnější.

Hlavním cílem práce bylo pak vytvořit optimalizační framework, jehož cílem je zvýšení přesnosti zarovnávacích nástrojů pomocí nalezení optimálních hodnot zvolených parametrů. Zvýšení přesnosti odpovídá v této práci maximalizaci počtu správně zarovnaných sekvenčních čtení. Framework byl navržen a implementován tím způsobem, aby bylo možné jednotlivé části v případě potřeby nahradit a také aby bylo možné jednoduchou úpravou spustit optimalizaci libovolného nástroje s různými parametry. Optimalizační jádro je implementací diferenciální evoluce, která, jak ukázaly výsledky experimentů (viz kapitola 7), je vhodně zvoleným optimalizačním prvkem. Proces optimalizace lze ovlivnit nastavením parametrů četnosti křížení či velikosti populace a počtu generací. Klíčovou roli hraje výpočet fitness funkce, který je zcela oddělen v jiném programu a který může být jednoduše modifikován či nahrazen.

S vytvořeným optimalizačním frameworkem bylo provedeno několik sad experimentů jak na datových sadách softwarově generovaných sekvenčních čteních, tak na sekvenčních čteních ze sekvenačního přístroje Illumina. Výsledky těchto experimentů byly demonstrovány pomocí grafů, které znázorňovaly především nárůst úspěšnosti při zarovnání sekvenčních čtení různých délek. V případě zarovnání programu BLAT na sadě generovaných čtení došlo k nárůstu přesnosti o 3,27 %, u párových čtení o 10,2 % (viz kapitola 7). K nárůstu přesnosti došlo za cenu výrazného zpomalení celkového zarovnání. V případě programu LAST došlo ke zlepšení přesnosti zarovnání o 0,77 % u nepárových sekvenčních čtení a o 0,57 % v případě párových (viz kapitola 7). U obou programů výsledky ukazovaly, že optimálním řešením je

vytvořit co možná nejdetailnější index. V obou případech měl indexační krok délku 1. Při použití sady reálných sekvenčních čtení však už toto řešení nebylo nalezeno jako optimální. Délka indexačního kroku je u programu BLAT 8 bází a u programu LAST 5 bází při délce nepárového čtení 75 bází. Problém rychlosti u programu BLAT byl díky delší referenční sekvenci ještě výraznější než v případě generovaných čtení, u programu LAST došlo naopak opět ke zrychlení. V případě programu BLAT došlo ke zlepšení přesnosti o 2,4 % a v případě programu LAST o 1,4 %.

Výsledkem experimentů bylo kromě ověření správnosti optimalizačního frameworku a jeho využitelnosti při optimalizaci zarovnání také nalezení optimálních prostorově rozšířených semínek u programu LAST, která byla doposud známá jen pro sekvenční maximální délky 50 bází. Výsledná šablona má v případě experimentů s reálnými sekvenčními čteními s délkou 65 bází tvar 1110100100000110011. Současně byla ověřena vhodnost použití adaptivní délky semínka namísto použití semínka s omezenou délkou.

# Literatura


[1] Abouelhoda, M. I.; Kurtz, S.; Ohlebusch, E.: Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, ročník 2, č. 1, 2004: s. 53–86.

[2] Altschul, S. F.; Gish, W.; Miller, W.; aj.: Basic local alignment search tool. *Journal of molecular biology*, ročník 215, č. 3, 1990: s. 403–410.

[3] Branton, D.; Deamer, D. W.; Marziali, A.; aj.: The potential and challenges of nanopore sequencing. *Nature biotechnology*, ročník 26, č. 10, 2008: s. 1146–1153.

[4] Burrows, M.; Wheeler, D. J.: A block-sorting lossless data compression algorithm. 1994.

[5] Cao, X.; Li, S. C.; Tung, A. K.: Indexing dna sequences using q-grams. In *Database Systems for Advanced Applications*, Springer, 2005, s. 4–16.

[6] Collins, F.; Lander, E.; Rogers, J.; aj.: Finishing the euchromatic sequence of the human genome. *Nature*, ročník 431, č. 7011, 2004: s. 931–945.

[7] Dressman, D.; Yan, H.; Traverso, G.; aj.: Transforming single DNA molecules into fluorescent magnetic particles for detection and enumeration of genetic variations. *Proceedings of the National Academy of Sciences*, ročník 100, č. 15, 2003: s. 8817–8822.

[8] Eid, J.; Fehr, A.; Gray, J.; aj.: Real-time DNA sequencing from single polymerase molecules. *Science*, ročník 323, č. 5910, 2009: s. 133–138.

[9] Erlich, Y.; Mitra, P.; de la Bastide, M.; aj.: Alta-Cyclic: a self-optimizing base caller for next-generation sequencing. *Nat Methods*, ročník 5, 2008: s. 679–682.

[10] Farrar, M.: Striped Smith–Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, ročník 23, č. 2, 2007: s. 156–161.

[11] Fedurco, M.; Romieu, A.; Williams, S.; aj.: BTA, a novel reagent for DNA attachment on glass and efficient generation of solid-phase amplified DNA colonies. *Nucleic acids research*, ročník 34, č. 3, 2006: s. e22–e22.

[12] Ferragina, P.; Manzini, G.: Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, IEEE, 2000, s. 390–398.

[13] Frazer, K. A.; Murray, S. S.; Schork, N. J.; aj.: Human genetic variation and its contribution to complex traits. *Nature Reviews Genetics*, ročník 10, č. 4, 2009: s. 241–251.

[14] Hoffmann, S.; Otto, C.; Kurtz, S.; aj.: Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS computational biology*, ročník 5, č. 9, 2009: str. e1000502.

[15] Holtgrewe, M.; Emde, A.-K.; Weese, D.; aj.: A novel and well-defined benchmarking method for second generation read mapping. *BMC bioinformatics*, ročník 12, č. 1, 2011: str. 210.

[16] Homer, N.; Merriman, B.; Nelson, S. F.: BFAST: an alignment tool for large scale genome resequencing. *PLoS One*, ročník 4, č. 11, 2009: str. e7767.

[17] Honzík, J. M.: Algoritmy IAL Studijní opora pro studenty FIT. 2014-01-06.

[18] Huang, W.; Li, L.; Myers, J. R.; aj.: ART: a next-generation sequencing read simulator. *Bioinformatics*, ročník 28, č. 4, 2012: s. 593–594.

[19] Illumina Inc.: Mate-pair Sequencing Assay. http://www.illumina.com/technology/mate_pair_sequencing_assay.ilmn, cit: 2014-01-10.

[20] Illumina Inc.: Paired-end Sequencing Assay. http://www.illumina.com/technology/paired_end_sequencing_assay.ilmn, cit: 2014-01-10.

[21] Jiang, H.; Wong, W. H.: SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics*, ročník 24, č. 20, 2008: s. 2395–2396.

[22] Jokinen, P.; Ukkonen, E.: Two algorithms for approxmate string matching in static texts. In *Mathematical Foundations of Computer Science 1991*, Springer, 1991, s. 240–248.

[23] Kent, W. J.: BLAT-the BLAST-like alignment tool. *Genome research*, ročník 12, č. 4, 2002: s. 656–664.

[24] Kiełbasa, S. M.; Wan, R.; Sato, K.; aj.: Adaptive seeds tame genomic sequence comparison. *Genome research*, ročník 21, č. 3, 2011: s. 487–493.

[25] Kurtz, S.; Phillippy, A.; Delcher, A. L.; aj.: Versatile and open software for comparing large genomes. *Genome biology*, ročník 5, č. 2, 2004: str. R12.

[26] Landegren, U.; Kaiser, R.; Sanders, J.; aj.: A ligase-mediated gene detection technique. *Science*, ročník 241, č. 4869, 1988: s. 1077–1080.

[27] Langmead, B.; Trapnell, C.; Pop, M.; aj.: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, ročník 10, č. 3, 2009: str. R25.

[28] Ley, T. J.; Mardis, E. R.; Ding, L.; aj.: DNA sequencing of a cytogenetically normal acute myeloid leukaemia genome. *Nature*, ročník 456, č. 7218, 2008: s. 66–72.

[29] Li, H.; Durbin, R.: Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, ročník 25, č. 14, 2009: s. 1754–1760.

[30] Li, H.; Durbin, R.: Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics*, ročník 26, č. 5, 2010: s. 589–595.

[31] Li, H.; Handsaker, B.; Wysoker, A.; aj.: The sequence alignment/map format and SAMtools. *Bioinformatics*, ročník 25, č. 16, 2009: s. 2078–2079.

[32] Li, H.; Homer, N.: A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in bioinformatics*, ročník 11, č. 5, 2010: s. 473–483.

[33] Li, H.; Ruan, J.; Durbin, R.: Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome research*, ročník 18, č. 11, 2008: s. 1851–1858.

[34] Li, R.; Li, Y.; Kristiansen, K.; aj.: SOAP: short oligonucleotide alignment program. *Bioinformatics*, ročník 24, č. 5, 2008: s. 713–714.

[35] Li, R.; Yu, C.; Li, Y.; aj.: SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, ročník 25, č. 15, 2009: s. 1966–1967.

[36] Lin, H.; Zhang, Z.; Zhang, M. Q.; aj.: ZOOM! Zillions of oligos mapped. *Bioinformatics*, ročník 24, č. 21, 2008: s. 2431–2437.

[37] Ma, B.; Tromp, J.; Li, M.: PatternHunter: faster and more sensitive homology search. *Bioinformatics*, ročník 18, 2002: s. 440–445.

[38] Metzker, M. L.: Emerging technologies in DNA sequencing. *Genome Res*, ročník 15, 2005: s. 1767–1776.

[39] Metzker, M. L.: Sequencing technologies - the next generation. *Nature Reviews Genetics*, ročník 11, č. 1, 2009: s. 31–46.

[40] Mount, D. W.: Comparison of the PAM and BLOSUM amino acid substitution matrices. *Cold Spring Harbor Protocols*, ročník 2008, č. 6, 2008: s. pdb–ip59.

[41] Munro, J. I.; Raman, V.; Rao, S. S.: Space efficient suffix trees. *Journal of Algorithms*, ročník 39, č. 2, 2001: s. 205–222.

[42] Myers, E. W.: AnO (ND) difference algorithm and its variations. *Algorithmica*, ročník 1, č. 1-4, 1986: s. 251–266.

[43] Nagalakshmi, U.; Wang, Z.; Waern, K.; aj.: The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, ročník 320, č. 5881, 2008: s. 1344–1349.

[44] Navarro, G.: A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, ročník 33, č. 1, 2001: s. 31–88.

[45] Ossowski, S.; Schneeberger, K.; Clark, R. M.; aj.: Sequencing of natural strains of Arabidopsis thaliana with short reads. *Genome research*, ročník 18, č. 12, 2008: s. 2024–2033.

[46] Rumble, S. M.; Lacroute, P.; Dalca, A. V.; aj.: SHRiMP: accurate mapping of short color-space reads. *PLoS computational biology*, ročník 5, č. 5, 2009: str. e1000386.

[47] Smith, A. D.; Xuan, Z.; Zhang, M. Q.: Using quality scores and longer reads improves accuracy of Solexa read mapping. *BMC bioinformatics*, ročník 9, č. 1, 2008: str. 128.

[48] Smith, A. D.; Xuan, Z.; Zhang, M. Q.: Using quality scores and longer reads improves accuracy of Solexa read mapping. *BMC bioinformatics*, ročník 9, č. 1, 2008: str. 128.

[49] Smith, T. F.; Waterman, M.: Identification of Common Molecular Subsequence. *Journal of molecular biology*, ročník 147, 1981: s. 195–197.

[50] Storn, R.; Price, K.: *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*. ICSI Berkeley, 1995.

[51] Trapnell, C.; Pachter, L.; Salzberg, S. L.: TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, ročník 25, č. 9, 2009: s. 1105–1111.

[52] Valouev, A. a. k.: A high-resolution, nucleosome position map of C. elegans reveals a lack of universal sequence-dictated positioning. *Genome research*, ročník 18, 2008: s. 1051–1063.

[53] Wang, Z.; Gerstein, M.; Snyder, M.: RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, ročník 10, č. 1, 2009: s. 57–63.

[54] Wu, W.; Stupi, B. P.; Litosh, V. A.; aj.: Termination of DNA synthesis by N6-alkylated, not 3'-O-alkylated, photocleavable 2'-deoxyadenosine triphosphates. *Nucleic acids research*, ročník 35, č. 19, 2007: s. 6339–6349.

# Dodatek A

# Obsah CD

bude doplněno...

# Dodatek B

# Výstup programu RABEMA

```
Intervals to find:              4812
Intervals found:                3110
Intervals found [%]             64.6301
Invalid alignments:             18109
Additional Hits:                0

Number of reads:                14528
Number of reads with intervals: 4665

Mapped reads:                   3055
Mapped reads [% of total]:      21.0284
Mapped reads [% of mappable]:   65.4877

Normalized intervals found:     3023.5
Normalized intervals found [%]: 64.8124

Found Intervals By Error Rate
```

| ERR | #max | #found | %found norm max | norm found | norm found [%] |
|---|---|---|---|---|---|
| 0 | 6 | 6 | 100.00 | 6.00 | 6.00 | 100.00 |
| 1 | 39 | 37 | 94.87 | 39.00 | 37.00 | 94.87 |
| 2 | 121 | 111 | 91.74 | 121.00 | 111.00 | 91.74 |
| 3 | 360 | 301 | 83.61 | 358.00 | 299.00 | 83.52 |
| 4 | 584 | 443 | 75.86 | 580.00 | 439.50 | 75.78 |
| 5 | 849 | 593 | 69.85 | 840.00 | 584.50 | 69.58 |
| 6 | 1009 | 649 | 64.32 | 981.00 | 630.50 | 64.27 |
| 7 | 966 | 564 | 58.39 | 924.00 | 541.00 | 58.55 |
| 8 | 878 | 406 | 46.24 | 816.00 | 375.00 | 45.96 |