FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Projekt u sklopu predmeta “Bioinformatika” 2019/2020

Poboljšanje djelomično sastavljenog genoma dugim očitanjima

**HERA algoritam**

Dominik Ivošević

Eduard-Edi Jerković

Marin Smoljanić

Sadržaj

[1. Uvod 3](#_Toc30374560)

[Usporedba dvije glavne klase algoritama sastavljanja genoma (OLC i DBG) 3](#_Toc30374561)

[2. HERA - Izgradnja preklapajućeg grafa 4](#_Toc30374562)

[3. Heuristički algoritam za kreiranje putanje između dva sidrena čvora 4](#_Toc30374563)

[Prvi pristup 4](#_Toc30374564)

[Drugi pristup 4](#_Toc30374565)

[Treći pristup 5](#_Toc30374566)

[4. Stvaranje konsenzusnih sekvenci za povezane putanje 5](#_Toc30374567)

[5. Konstrukcija povezanog grafa te spajanje sekvenci 6](#_Toc30374568)

[6. Implementacija 6](#_Toc30374569)

[Alat Minimap 6](#_Toc30374570)

[Implementacija HERA algoritma 9](#_Toc30374571)

[Zaključak 10](#_Toc30374572)

[Literatura 11](#_Toc30374573)

# 1. Uvod

Sastavljanje kompletnog genoma je ključan korak za provođenje postupka mapiranja i kloniranja gena kao i za identifikaciju strukturnih varijacija genoma. Genomi složenih eukariotskih organizama sadrže veliki broj ponavljajućih sekvenci što uvelike komplicira postupak sastavljanja genoma. Većina današnjih očitanja genoma su fragmentirana te daju nekompletnu sliku genoma. Cilj HERA algoritma je sastaviti nedostajuće DNK sekvence genoma koje se nalaze između postojećih DNK sekvenci, taj postupak uključuje novi koncept *preklapajućeg grafa* kao i algoritme za izgradnju takvog grafa.

## Usporedba dvije glavne klase algoritama sastavljanja genoma (OLC i DBG)

OLC stoji za **Overlap-Layout-Consensus** dok DBG za **De-Bruijn-Graph**.

OLC algoritam generalno radi u tri koraka: prvo se pronalaze preklapanja između svih očitanja (O), a zatim se određuje finalni *layout* (L) informacija svih očitanja i preklapanja na grafu te se konačno definira konsenzus (C). Algoritam je inicijalno kreiran od strane Stadena (1980), a potom je proširen i razrađen od strane brojnih znanstvenika.

DBG je algoritam koji cijepa početna očitanja u puno manje k-torke iz kojih u konačnici sastavlja genom. Algoritam su izvorno kreirali Raman M. Indury i Michael S. Waterman 1995. godine. U početcima je DBG algoritam bio uspješan samo na malim genima kao recimo od bakterije, a kasnije modificiran postao primjenjiv i na velikim genomima.

OLC algoritam je vrlo sličan Lander-Watermanovom modelu. Oba algoritma dijele definiciju parametra **T**. Lander-Watermanov model kaže da konačni broj kontiga ovisi o četiri parametra: duljina očitanja (**L**), duljina odsječka preklapanja (**T**), dubina sekvenciranja (**c**) te veličina genoma (**G**). **G** ima konstantnu vrijednost tako da se u principu barata sa tri parametra. Kako bi u DBG algoritmu mogli osigurati povezivanje k-torki minimalno preklapanje dva očitanja ne smije biti manje od duljine k-torke. Parametar duljine k-torke se označava sa **K** i odgovara parametru T iz Lander-Watermanova modela.

Metode korištene za izdvajanje informacija o preklapanjima razlikuju se između algoritama OLC i DBG. U OLC algoritmu identifikacija preklapanja između svakog para očitanja je eksplicitna, tipično radeći uspoređivanje očitanja svaki-sa-svakim. U DBG algoritmu implicitno se definira relacija preklapanja između susjednih k-torki. Postupak se provodi sjeckanjem svih očitanja u k-torke sa istovremenim bilježenjem njihova susjedna odnosa.

U našoj implementaciji koristili smo “modifikaciju” osnovnog OLC kao komponentu HERA algoritma te je ona u nastavku detaljnije opisana.

# 2. HERA - Izgradnja preklapajućeg grafa

Preklapajući graf G, je obojani neusmjereni graf sa dvije boje čvorova te dvije boje bridova. Čvorove čine **sidreni čvorovi** (NA) za sidrene sekvence tj. za *pre-assembled* kontige, a ***read nodes***(NR) za očitanje sekvenci. EO je tip brida koji se naziva **preklapajućim bridom** jer predstavlja preklapanje između dva očitanja i/ili *anchoring sequences* (sidrene sekvence). EC je tip brida koji se naziva ***coupling edge*** te predstavlja sekvencu čiji krajevi su predstavljeni dvama čvorovima. Svaki *read node* se smije spojiti sa najviše jednim sidrenim čvorom sa najvišom preklapajućom vrijednošću. Povezivanje sidrenih čvorova se može dobiti obilaskom grafa G koristeći DFS ili BFS. Sljedeći uvjet mora biti zadovoljen prilikom obilaska grafa: svaki puta kada se dođe u čvor preko **preklapajućeg brida** mora se izaći preko **coupling brida** i obrnuto.

# 3. Heuristički algoritam za kreiranje putanje između dva sidrena čvora

Počevši iz nasumičnog sidrenog čvora, **{NAs}**, pokušavamo identificirati sve njegove direktno povezane sidrene čvorove **{NAj}** unutar G te ispisati skup puteva obilazeći tako cijeli graf. Ovdje koristimo “direktnu povezanost” kako bi naglasili da je sidreni čvor povezan sa drugim bez mogućnosti susretanja nekog drugog sidrenog čvora na putanji. Produljivanje puta završava onda kada se dosegne očitanje koje povezuje sa slijedećim contigom. Računalno je zahtjevno izračunati sve dozvoljene puteve koji proizlaze iz NAs, te se stoga ograničava broj staza koje se odabiru za algoritam. Uzimamo u obzir samo puteve koji predstavljaju sekvence čija duljina ima predefiniranu maksimalnu vrijednost označenu sa **Lex**. Također postoji uvjet da *read node* može biti korišten samo jednom u konstrukciji putanje. Kako bi povećali šansu za kreiranje valjanog puta između dva sidrena čvora koristimo kombinaciju fiksnih shema bodovanja te Monte Carlo pristupa konstrukciji skupa najbolje-ocijenjenih puteva koji spajaju dva sidrena čvora.

## Prvi pristup

U prvom koraku proširenja sva očitanja koja se spajaju na **NAs** odabiru se za proširenje **NAs**. Za danja proširenja samo očitanja sa najvišom preklapajućom ocjenom se odabiru. Ako su ocjene za dva očitovanja iste uzima se očitanje sa višim ID-em ili većom duljinom niza. Treba napomenuti kako sve sekvence generirane u prvom koraku ne mogu naposlijetku biti prosirene na neki drugi sidreni čvor. U slučaju *dead enda,* situacije gdje nije moguće pronaći očitanja koja vode do proširenja, proširujući put se vraća na prethodni čvor te se proširuje sa očitanjem sa najvećom ocjenom.

## Drugi pristup

Isti kao prvi pristup osim toga da se u svakom koraku proširenja odabire očitanja s najvećom ocjenom produljenja.

## Treći pristup

Monte Carlo pristup se koristi za slučajni odabir povezujućeg očitanja za svako proširenje. Vjerojatnost da će očitanje biti izabrano za spajanje proporcionalna je ocjeni.

Konačno, kreiran je skup puteva kojima se dolazi do skupa sidrenih čvorova **{NAj}** iz **NAS** .Prva dva pristupa generiraju fiksan broj puteva dok treći pristup proizvoljan broj. U konačnici se iz skupa svih puteva brišu duplikati kako bi se dobio skup puteva iz **NAS** koji dosežu sidrene čvorove **NAj, {PSj}.**

# 4. Stvaranje konsenzusnih sekvenci za povezane putanje

Između svih puteva varijabilne duljine između para sidrenih čvorova, potrebno je pronaći reprezentativnu (konsenzus) sekvencu. Ako duljine putanja nisu velike, manje od 10kb, onda se sve putanje stavljaju u jednu grupu. U suprotnome putevi se dijele u različite grupe ovisno o duljini, gdje svaka grupa predstavlja moguću povezujuću sekvencu. Konkretno, sve sekvence se sortiraju prema duljini od najkraće do najdulje koristeći prozor od 1kb. Svaki prozor se uspoređuje sa prethodnim te idućim prozorom, a prozor sa najvećom sumom frekvencije puteva označava se kao vršni prozor, a ako ima najmanju sumu frekvencije puteva, onda je dolinski prozor. Ako je najmanja suma frekvencija puteva u dolinskom prozoru manja od 90% iznosa najviše sume frekvencije puteva desnog vršnog prozora, onda duljina frekvencije puteva iz dolinskog prozora s najmanjom frekvencijom se koristi se za podjelu čitavog skupa puteva u različite grupe.

U svakoj grupi, svi putevi koji imaju frekvenciju puteva manju od polovice iznosa najviše frekvencije puteva se odbacuju. Sekvence preostalih puteva su poravnate jedna s drugom, a jedan od puteva sa najvišom frekvencijom puteva koja odgovara najvećem broju drugih staza odabire se kao konsenzusna sekvenca. Broj puteva koji se podudara sa konsenzusnom sekvencom postavlja se kao validni broj staze u toj grupi.

Kada se pronađu višestruke grupe puteva između para sidrenih čvorova konsenzusna sekvenca će biti generirana za obje grupe. Višestruke konsenzusne sekvence ukazuju na postojanje sličnih ponavljajućih jedinica. Ako je duljina regije poznata, odabire se konsenzusna sekvenca koja odgovara poznatoj duljini. U suprotnome, počevši od grupe sa najvišom duljina puteva njezina frekvencija se uspoređuje sa susjednom grupom s duljim putevima, iterativno. Ako postoje samo dvije grupe ili validni broj puta u duljoj grupi je viši od polovice onoga u skupini kraćih puteva, odabire se grupa dužih puteva. U suprotnome se odabire grupa kraćih puteva. Konačno, konsenzusna sekvenca odabrane grupe se uzima kao konsenzusna sekvenca cijele regije.

Ako su duljine puteva u pojedinoj grupi raspodijeljene duž širokog prozora (>100kb), bez jasno definiranog vrha, tada će se ponavljanje temeljnih puteva promatrati kao kompleksno ponavljanje. Takve vrste ponavljanja se ne koriste za spajanje sidrenih sekvenci.

# 5. Konstrukcija povezanog grafa te spajanje sekvenci

Nakon što je određena konsenzusna sekvenca između svakog para sidrenih čvorova, povezani graf se konstruira se na čvorovima pomoću konsenzusne sekvence koja povezuje čvorove kao bridove. Zbroj validnih brojeva puteva (NP) u oba smjera između svakog para sidrenih čvorova bilježi se na svakom bridu.

Definiramo **konfliktni index** (CI) za svaki sidreni čvor. Za bilo koji sidreni čvor **NAi**,za svaki od njegovih vezanih sidrenih čvorova **{NAj}**, validni broj puta je **NPij.** Ako dva povezana sidrena čvora **NAm** i **NAn** imaju prva dva najveća povezujuća broja puta **NPim** i **NPin,** odnosno, između svih **NPij,** definiramo konfliktni index za **NAi** kao

**CIi E = NPin/NPim**. Sidreni čvor ima konfliktne veze ako mu je konfliktni index veći od predefinirane vrijednosti, **CImax**. Ako je sidreni čvor **NAs** povezan s drugim sidrenim čvorom **{NAi}** bez konfliktnih veza, uzimamo sidreni čvor iz **{NAi}** sa najvećim brojem puta kao odgovarajući spojni sidreni čvor od **NAi.**

Sidreni čvorovi s konfliktnim vezama neće se koristiti u vezama sekvenci ukoliko konflikt ne bude razriješen. Ne-konfliktni sidreni čvorovi su spojeni sa svojim odgovarajućim sidrenim čvorovima sa konsenzusnom sekvencom između njih kako bi generirali kontige.

# 6. Implementacija

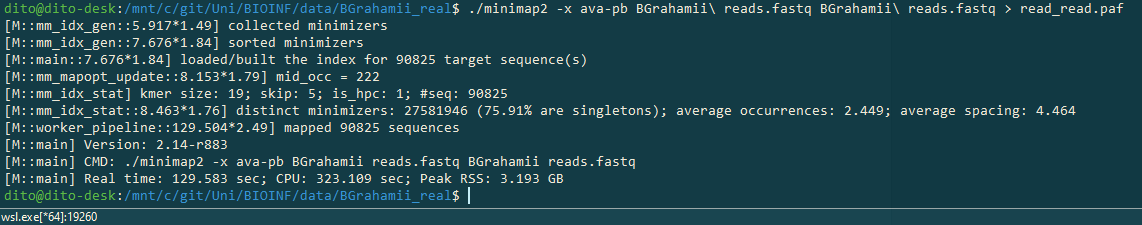
Implementacija HERA algoritma je pisana u C#-u. Podaci koji su korišteni za testiranje su dobiveni od fakulteta. Dobiveni su očitanja, već prethodno složeni contigovi i referentna, složena sekvenca za 3 bakterijska genoma: *Campylobacter jejuni*, *Bartonella grahamii* i *Escherichia coli* s time da su sintetski podaci dobiveni za *E. coli*.

Primarno smo radili s E. coli podacima. Alati koji su korišteni uz našu implementaciju su minimap (https://github.com/lh3/minimap2) i Gepard (<https://github.com/univieCUBE/gepard>).

## Alat Minimap

Alat Minimap je korišten za izračun preklapanja između parova očitanja i očitanja i između parova očitanja i contigova. On bi uzeo 2 .fasta datoteke i generirao .paf datoteku s preklapanjima.

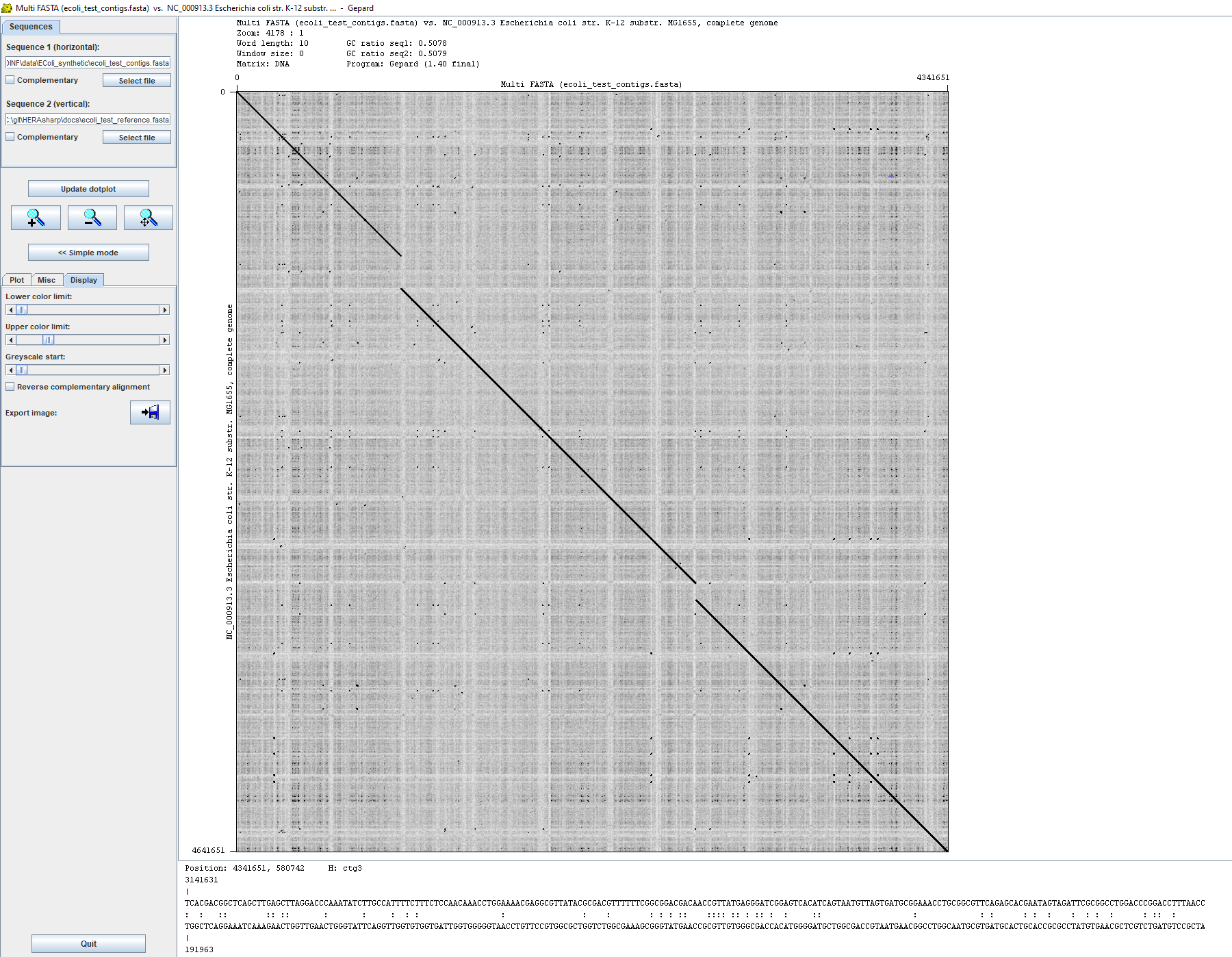
Na slici se vidi način korištenja alata tako što B. graham očitanja dajemo Minimapu i on vraća read\_read.paf datoteku.



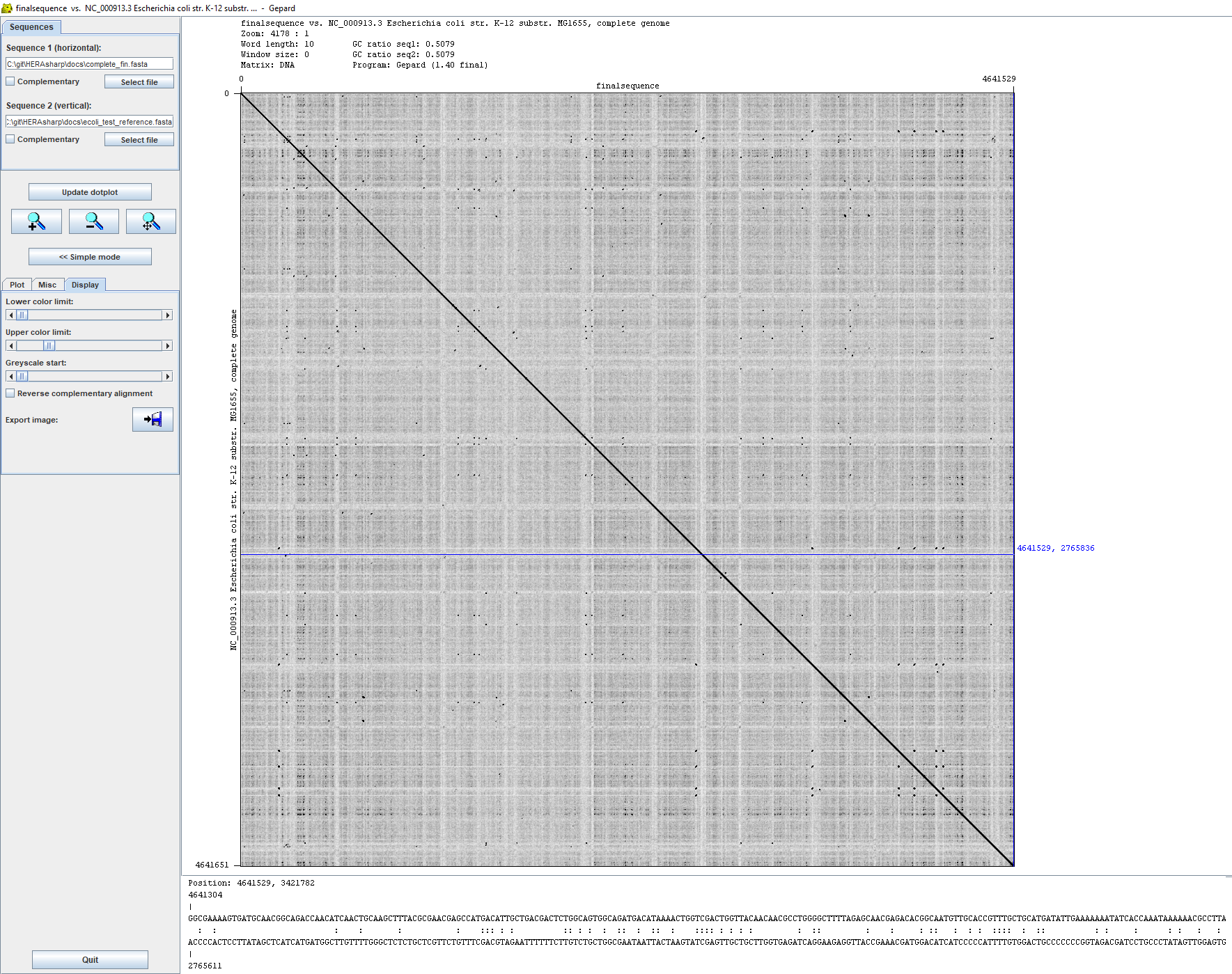
Alat Gepard

Alat Gepard se koristi za vizualizaciju i isprobavanje uspješnosti finalnog slaganja sekvence. GEPARD je bakronim za Genome Pair – Rapid Dotter). On koristi lokalnu usporedbu dvije sekvence nukleotida ili amino kiselina iz zadanih datoteka.

Na slici se može vidjeti kako izgleda usporedba za E.coli između referentne sekvence i 3 contiga koja imamo na raspolaganju u testnim podacima.



Na slijedećoj slici se vidi finalna usporedba između sekvence dobivene našom implementacijom HERA algoritma i referentne sekvence.

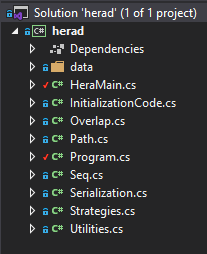


Kao što se može vidjeti iz priloženih slika, sekvenca E. coli je uspješno spojena pomoću naše implementacije.

## Implementacija HERA algoritma

Sama implementacija HERA algoritma je srž ovog rada. Glavni cilj je uspješno ispunjen tako što smo uspješno sastavili E. coli genom.

Trajanje izvršavanja rješenja za dobivanje finalne sastavljanje sekvence za E. coli je 3:02 minute i korišteno je do 2gb RAM-a. Nakon što smo uveli jednostavnu paralelizaciju gdje smo paralelizirali na razini generiranja konsenzus sekvenci od svakog contiga u release buildu smo postigli vrijeme izvršavanja od 62 sekunde na Intel i5-6600k.

Kod je organiziran unutar jednog projekta kao što se može vidjeti na slici

U klasi HeraMain se nazali metoda MainLogic u kojoj metode odgovaraju koracima kako su opisani u izvornom radu u kojem je opisan algoritam HERA.

Prvi korak je generiranje konsenzus sekvenci. Iteriramo kroz sve raspoložive contige te za svaki koristeći strategije ocjene Overlap score, Extension score i Monte carlo. Najveći izazov u ovom koraku je kako optimalno provjeriti je li neko preklapanje već korišteno. Odabrali smo metodu lookup tablice budući da u ovom slučaju raspolažemo sa nešto manje od 10 000 očitanja te to ne predstavlja veliki izazov za memoriju te je mnogo brže od korištenja hash tablica.

U metodi GetConsensusSequenceAndItsGroup primamo cijeli popis puteva koji se generiraju koristeći prethodno navedene strategije te podijelimo puteve po njihovoj duljini u grupe po kriteriju duljina gdje su unutar svake grupe duljine u razmaku od 10 000 000 baza. Zatim unutar grupe s najvećom frekvencijom očitanja odabiremo onu sekvencu koja ima najbolji prosjećni Overlap score.

U slijedećem koraku moramo odabrati put kroz sve contigove. U metodi GetConnectionGraph sortiramo consensus sequence prema njihovoj ocjeni te po principu sličnom Kruskalovom algoritmu provjeravajući cikličnost i prethodno pojavljivanje polu-naivno odabiremo redoslijed.

Nakon što je dobiven redoslijed posjećivanja contigova, u metodi GetFinalPathOverlaps generiramo finalnu listu preklapanja tako što pospajamo consensus sequence prema tome kako smo izgradili put kroz graf u prethodnom koraku.

Tu dobivenu listu preklapanja pretvaramo u konačnu sekvencu baza unutar metode BuildSequenceFromOverlaps te sekvencu zapisujemo u datoteku.

# Zaključak

Naša implementacija HERA algoritma ispunjava osnovne zahtjeve uspješnosti te su performanse zadovoljavajuće, barem za ovako mali genom kao što je to E. coli. Ovakva paradigma sastavljanja i ispravljanja očitanja je otvorena za mnoga dodatna eksperimentiranja budući da se metode koje se u koracima koriste nisu optimalne nego su samo jedna od mogućih heuristika koje se mogu odabrati. Jedan od najvećih izazova u samom radu je bio razumjeti što su u izvornom radu mislili u određenim koracima jer nisu išli do kraja u detalje objasniti svaku od tehnika te smo u tim slučajevima odabirali vlastite metode.

U daljnjem radu bi se najviše išlo ka dodatnom optimiranju koda kako bi se mogli efikasni Unit testovi uvesti budući da sa početnim trajanjem od 4 minute nije imalo smisla isprobavati na unit testovima, a i nismo imali problema sa regresijom budući da smo cijelo vrijeme radili na E. coli genomu. Ima mnogo prostora za istražiti koje su efikasnije strukture podataka koje bi se koristile za pohranu i pretraživanje u koracima.

Drugi smjer u kojem bi se ovaj rad mogao unaprijediti je daljnje poboljšavanje kvalitete očitanja s time da se iz naizgled veoma sličnih puteva, tj. nizova preklapanja napravi konsolidacija i pomoću neke metode kao što je glasovanje dodatno ispravlja neke baze.

Treći smjer u kojem bi se ovaj rad mogao unaprijediti je korištenje pametnije algoritma za izgradnju finalne staze kroz contigove. Daljnja eksperimentiranja s raznim metodama ocjenjivanja samih preklapanja, puteva i consensus sequenca bi moglo lučiti bolje rezultate, pogotovo na skupu podataka koji ima više grešaka.

# Literatura

[1] Huilong Du, Chengzhi Liang; Assembly of chromosome-scale contigs by efficiently resolving repetitive sequences with long reads, bioRxiv 345983; <https://www.biorxiv.org/content/10.1101/345983v1>

[2] Razni autori; Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph; <https://academic.oup.com/bfg/article/11/1/25/191455>