

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

# Poboljšanje djelomično sastavljenog genoma dugim očitanjima

Dan Ambrošić

Stjepan Dugonjić

Mihaela Bošnjak

Zagreb, siječanj 2019.

# SADRŽAJ

1. Uvod	1
2. Podaci i problemi	2
3. Opis metode i implementacija	3
4. Rezultati	4
5. Zaključak	5

# 1. Opis problema i podataka

Cilj ovog projekta je poboljšati djelomično sastavljen genom koristeći duga očitanja. Ukoliko genom ima puno ponavljajućih sekvenci, pogotovo ako su iste duže od duljine očitanja, teško ga je sastaviti u potpunosti. Ulazni podaci su skup sastavljenih sekvenci (contig-a), koje su dobivene nekim od alata za sastavljanje genoma, te skup dugih očitanja. Zadatak je napisati program koji pokušava sastaviti contig-e u jednu sekvencu koristeći dobivena očitanja.

Za provjeru rada implementacije korištena su 3 genoma:

1. EColi - sintetski podaci (3 contig-a)
2. CJejuni - stvarni podaci (6 contig-a)
3. BGrahamii - stvarni podaci (6 contiga-a)

Svi podaci se sastoje od datoteke s očitanjima te datoteke s već sastavljenim sekvencama koje su u FASTA formatu. Uz njih algoritmu su potrebna i preklapanja između contig-a i očitanja, te međusobna preklapanja očitanja koja su dobivena alatom Minimap2<sup>1</sup> u PAF formatu. Konačno, dobivena je i datoteka koja sadrži referentnu sekvencu s kojom se uspoređuje krajnji rezultat.

---

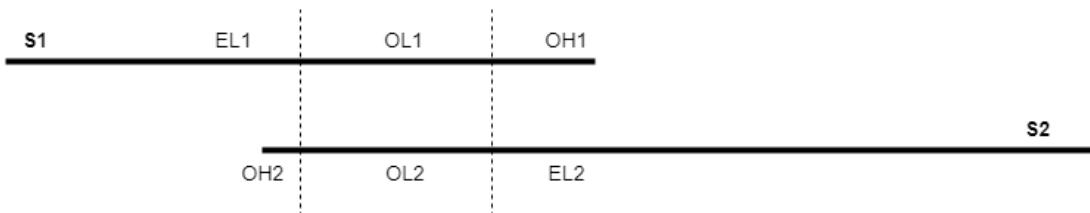
<sup>1</sup><https://github.com/lh3/minimap2>

## 2. Opis implementacije

Implementacija uglavnom slijedi algoritam HERA (engl. *Highly Efficient Repeat Assembly*) opisan u radu [?] uz nekoliko manjih modifikacija. Algoritam se sastoji od tri glavnih koraka. Prvo se gradi graf preklapanja u kojem se traže putevi između contig-a. Nakon što su pronađeni mogući putevi, za svaki par contig-a se traži reprezentativna sekvenca te se u konačnici sastavlja jedna sekvenca između povezanih contig-a.

Alat Minimap2, ukoliko pronađe preklapanje između dvije sekvence (očitanje ili contig), da informacije o indeksima početka i kraja preklapanja za obje sekvene. Prvi korak je odbacivanje preklapanja koja zadovoljavaju barem jedan od sljedećih uvjeta:

- preklapanje je između dvije iste sekvene,
- preklapanje u kojem jedna sekvenca u potpunosti sadrži drugu,
- SI (engl. *sequence identity*) mjera je ispod određene granice (primjerice 40%).



Slika 2.1: Primjer preklapanja između S1 i S2

Za preostale sekvene se računaju mjere preklapanja i produživanja. Na slici ?? je prikazano preklapanje između sekvenca  $S_1$  i  $S_2$  gdje se sekvenca  $S_2$  nalazi nakon sekvence  $S_1$ . U sredini između isprekidanih linija je regija preklapanja čija je duljina  $OL_1$  i  $OL_2$ , ovisno koju sekvencu gledamo, a izvana su  $OH$  (engl. *overhang length*) i  $EL$  (engl. *extension length*). Koristeći navedene duljine, računa se mjera preklapanja  $OS$  i mjera produživanja  $ES_1$  i  $ES_2$  prema izrazima ?? -

???. Pri tome je potrebno voditi računa preklapaju li se sekvene na istim ili suprotnim lancima. Primjerice da se  $S_1$  i  $S_2$  preklapaju na suprotnim lancima, potrebno bi bilo zamijeniti vrijednosti  $OH_2$  i  $EL_2$ .

$$OS = \frac{(OL_1 + OL_2) * SI}{2} \quad (2.1)$$

$$ES_1 = OS + \frac{EL_1}{2} - \frac{OH_1 + OH_2}{2} \quad (2.2)$$

$$ES_2 = OS + \frac{EL_2}{2} - \frac{OH_1 + OH_2}{2} \quad (2.3)$$

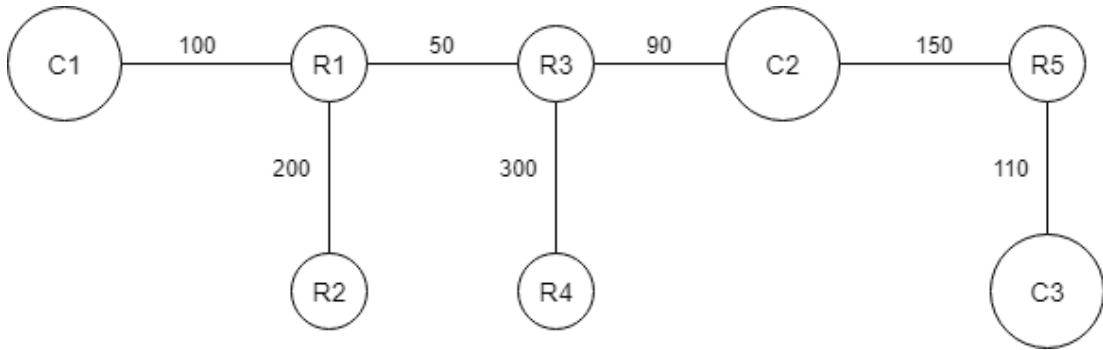
Nakon što su sve mjere izračunate, odbacuju se preklapanja gdje je zbroj  $OH_1$  i  $OH_2$  veći od 20% duljine produživanja  $EL_1$  ili  $EL_2$  te je moguće prijeći na izradu grafa preklapanja.

## 2.1. Graf preklapanja

Čvorovi u grafu predstavljaju contig-e i očitanja, a grane preklapanja, s time da grana može biti povezana na glavu ili rep čvora. Drugim riječima, svaki čvor ima svoje prefikse i sufikse. Sljedeći korak je traženje puteva između dva contig-a koji se obavlja pretraživanjem u dubinu (engl. *DFS*) uz nekoliko pravila. Postupak kreće iz svakog contig-a i zaustavlja se kada dođe do očitanja koje je povezano s drugim contig-om ili je trenutni put veći od maksimalne dopuštene duljine. Dodatno ograničenje je da se jedno očitanje ne može više puta pojaviti u istom putu.

Za izgradnju puteva koriste se tri pristupa. Prvi pristup iz početnog contiga izabere sve produžetke, ali za svaki sljedeći produžetak bira onaj koji ima najveću mjeru preklapanja  $OS$ . Ukoliko se dođe do čvora koji nema produžetke, vraća se jedan korak nazad i bira sljedeći najbolji produžetak. Drugi pristup radi isto kao prvi jedino koristi mjeru produživanja  $ES$ . Konačno, treći pristup nasumično bira produživanja proporcionalno mjeri produživanja  $ES$ , s time da je potrebno odrediti koliko puta će se iz svakog contig-a pokušati pronaći put ovom metodom. Po završetku postupka pronalaženja puteva između contig-a potrebno je samo odbaciti duplike i moguće je prijeći na sljedeći korak algoritma.

Na slici ?? je primjer jednostavnog grafa preklapanja gdje brojevi na granama predstavljaju mjeru preklapanja  $OS$ . Prepostavimo da tražimo put iz contig-a  $C_1$  prvim pristupom. Prvo ćemo produžiti put u  $R_1$ , nakon toga u  $R_2$  jer ima veću mjeru preklapanja od puta koji vodi u  $R_3$ . Međutim, kako dalje nema čvorova



Slika 2.2: Primjer grafa preklapanja

moramo se vratiti nazad i ići u  $R_3$ . Iako je po mjeri preklapanja veći put u  $R_4$ , za sljedeći čvor biramo  $C_2$  jer predstavlja contig i time je jedan put završen.

## 2.2. Generiranje konsenzusa

Jednom kada je konstruiran graf preklapanja, od potencijalno velikog broja pronađenih puteva između svakog od parova contig-a potrebno je odabratи jednu reprezentativnu (konsenzusnu) sekvencu (engl. *consensus sequence*).

Proces odabira takve sekvence započinje grupiranjem pronađenih puteva u tzv. konsenzusne grupe temeljem distribucije njihovih duljina, pri čemu razlikujemo 3 različita slučaja.

1. Prvi slučaj nastupa ukoliko su duljine svih pronađenih puteva usko distribuirane (unutar 10 kilobaza). Tada se svi putevi slažu u istu grupu, a kao reprezentativna sekvenca odabire se ona s najvećom prosječnom *sequence identity* mjerom.
2. Ukoliko su duljine distribuirane šire od 10 kb, no ne šire od 100 kb, pokušava se formirati više konsenzusnih grupa. Na početku se putevi razvrstavaju (temeljem njihove duljine) u 10 podgrupa veličine 10 kb, koje pokrivaju čitavo područje distribucije. Zatim se za svaku podgrupu koja sadrži barem jedan put izračuna suma globalnih frekvencija pojavljivanja sadržanih puteva određene duljine. Sve podgrupe čija je suma frekvencija manja od 20% najveće sume označavaju se kao doline (engl. *valleys*) dok u suprotnom postaju vrhovi (engl. *peaks*). Nad tako označenim podgrupama tada se traže sve doline okružene s vrhovima te ukoliko iste postoje, duljina puta s najmanjom globalnom frekvencijom pojavljivanja svake od njih postaje granična duljina

cijepanja glavne grupe (u kojoj su prvotno sadržani svi putevi). Pronalazak  $n$  takvih dolina dovodi do cijepanja glavne grupe na  $n + 1$  dijelova, od kojih svaki postaje zasebna konsenzusna grupa čija se reprezentativna sekvenca bira identično kao i u prvom slučaju.

3. Ukoliko su duljine distribuirane šire od 100 kb, u potpunosti se odbacuje mogućnost odabira reprezentativne sekvene između tog para contig-a.

Nakon stvaranja svake od grupa, iz istih se filtriraju putevi čija je frekvencija pojavljivanja duljine manja od 50% najveće frekvencije. Jednom kada su stvorene grupe između svih parova contig-a prelazi se na zadnji korak algoritma.

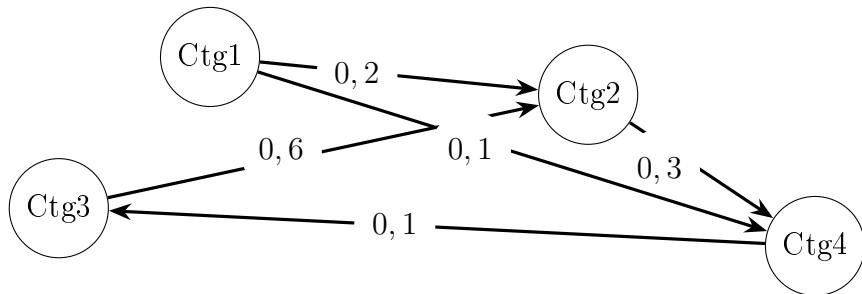
### 2.3. Spajanje spojenih contiga

Za svaki par spojenih contiga dobijemo n-grupa te uzmememo path koji je najbolji s obzirom na neku mjeru. Implementrali smo dvije mjerne koje koristimo a to su:

- prosjek svih SI
- medijan svih SI

Nakon odabira najboljeg puta između dvije contige radimo još jedan graf. Graf koji radimo radimo radi pronađenja najboljeg puta koji prolazi kroz sve contige ili kroz njih maksimalno koliko može.

Graf je usmjeren jer razlikujemo poveznicu  $Ctg1 \rightarrow Ctg2$  i  $Ctg2 \rightarrow Ctg1$ . Contiga 1 je povezana sa svojim krajom s početkom contige 2. Dok Contiga 2 je povezana sa svojim krajom s početkom contige 1. Ako idemo kroz graf nakon što uđemo kroz glavu moramo proći kroz rep pa je tako potrebno da ovo bude ili usmjereni graf ili kompleksnija struktura. Osim što je usmjeren graf je i težinski. Težine su ili medijan SI ili prosjek svih SI između te dvije contige. Primjer na slici ?? je ilustracija kako graf može izgledati ako imamo 4 spojene contige. Za ovaj primjer još je relativno jednostavno pronaći Hamiltonov put ali kada imamo više konekcija potrebno je proći kroz sve moguće kako bismo odabrali najbolju poveznicu svih contiga.



**Slika 2.3:** Graf contiga - pronalazak Hamiltonovog puta

Nakon što smo sagradili graf moramo pronaći Hamiltonov put. Pronalazak Hamiltonovog puta je implementiran kao rekurzija. Putovi koje dobijemo nisu nužno Hamiltonovi putovi ali nakon sortiramo tako da na kraju vektoru budu oni koji su najbliže toj definiciji. Sortiramo sve puteve po duljini a nakon toga po "dobroti"(suma medijana ili prosjeka). Odabiru se trenutno 3 najbolja puta i šalju na ispis.

Ispis je implementiran tako da pazi na kojem smo contigu te prva contiga nam je referentna strana. Prilikom ispisa obilazimo contige redoslijedom iz puta. Contige i readove koji su na suprotnoj strani okrećemo i uzimamo komplement kako bi bili na istoj DNA zavojnici. Razrješavanje overlapova radimo tako da uzmemo odmah sljedeći jer nam različiti pristupi nisu mijenjali uveliko rezultate.

Krajnji rezultat su maksimalno 3 datoteke koje u svojim imenima sadrže redoslijed contiga koje je algoritam pronašao.

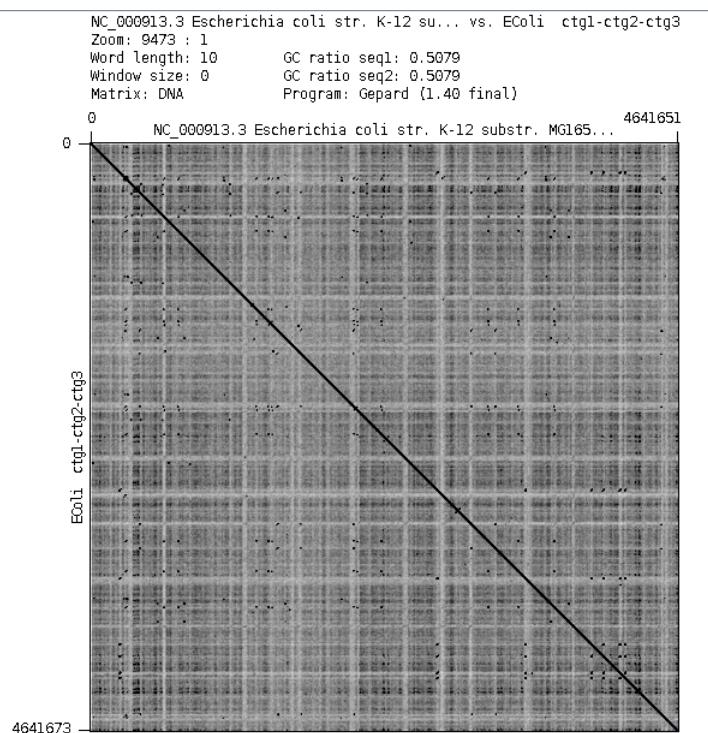
### 3. Rezultati

Dana su nam 3 genoma za testiranje:

1. EColi - sintetski podaci (3 contig-a)
2. CJejuni - stvarni podaci (6 contig-a)
3. BGrahamii - stvarni podaci (6 contiga-a)

#### 3.1. EColi

Kod EColi smo uspjeli ostvariti skoro savršenu pododarnost kao što vidimo na slici ??.



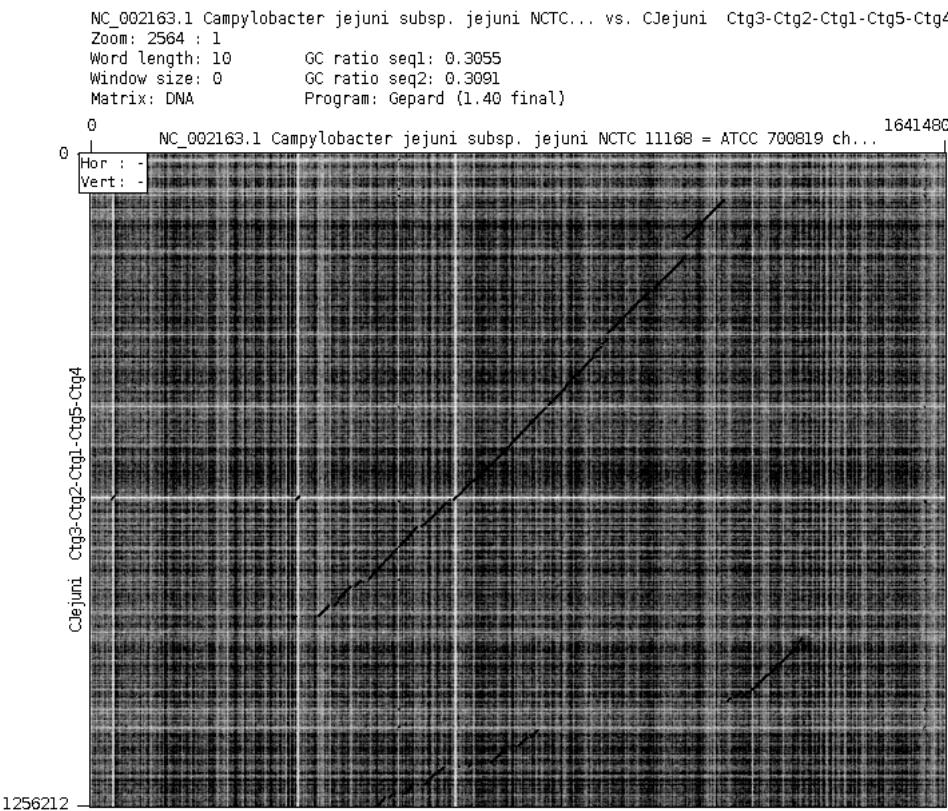
Slika 3.1: EColi - rezultati s Geparda

Duljina referentne sekvence je 4,641,651, a naše 4,641,673 baza. GC omjer je za obje sekvence jednak. Naravno da i podudarnost nismo dobili na bazu, ali smo uspjeli dobiti jednu konzistentnu sekvencu. Količina vremena koja je potrebna da se sastavi genom je u prosjeku 10 sekundi, a zauzima otprilike 50 MiB memorije.

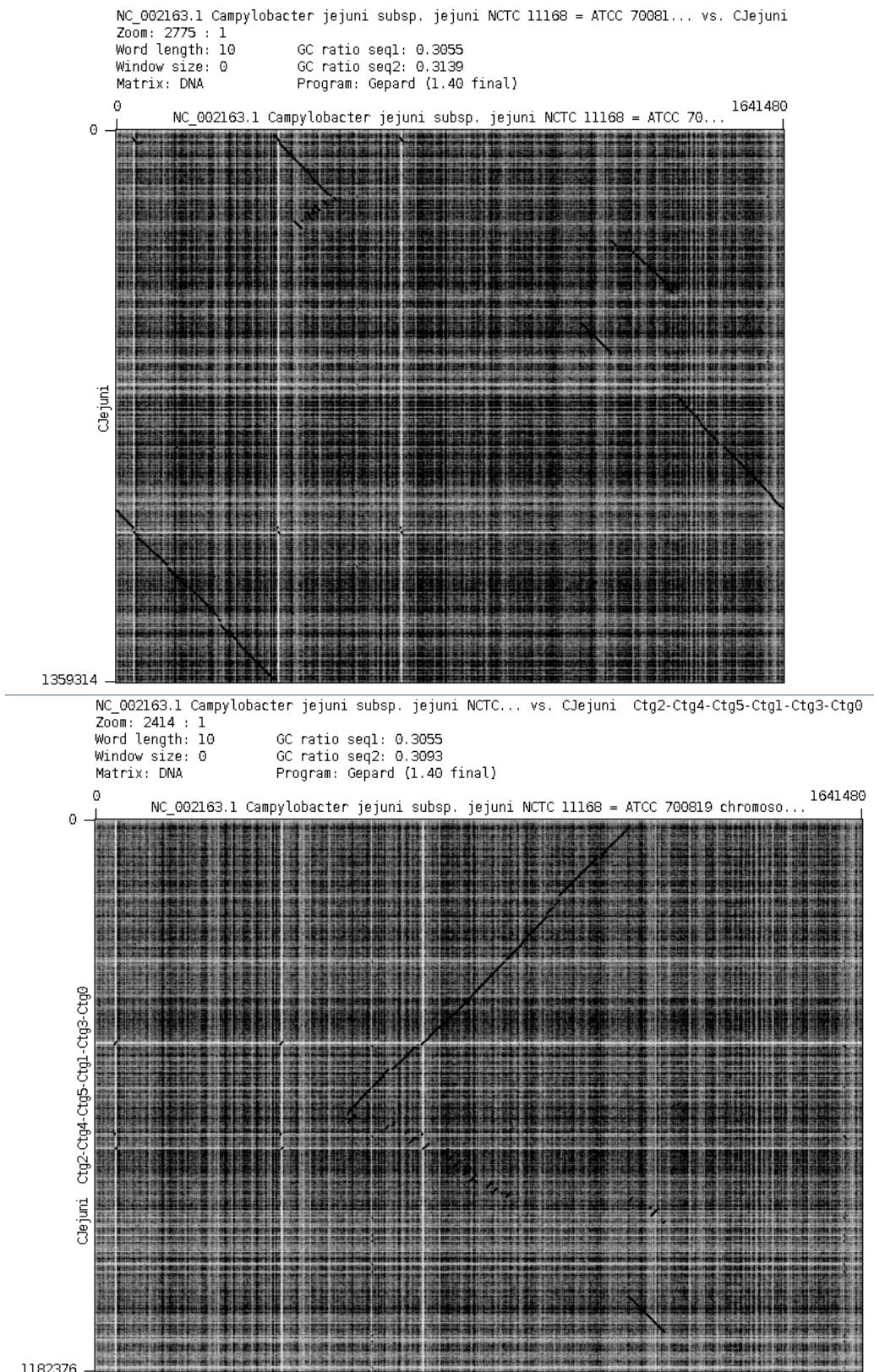
### 3.2. CJejuni

Kod CJejuni nismo uspjeli dobiti sekvencu koja je u potpunosti u skladu s referencom. U nastavku slijedi nekoliko vrijednih pokušaja. Za procesiranje ovih podataka je potrebna 1 minuta i 200 MiB memorije.

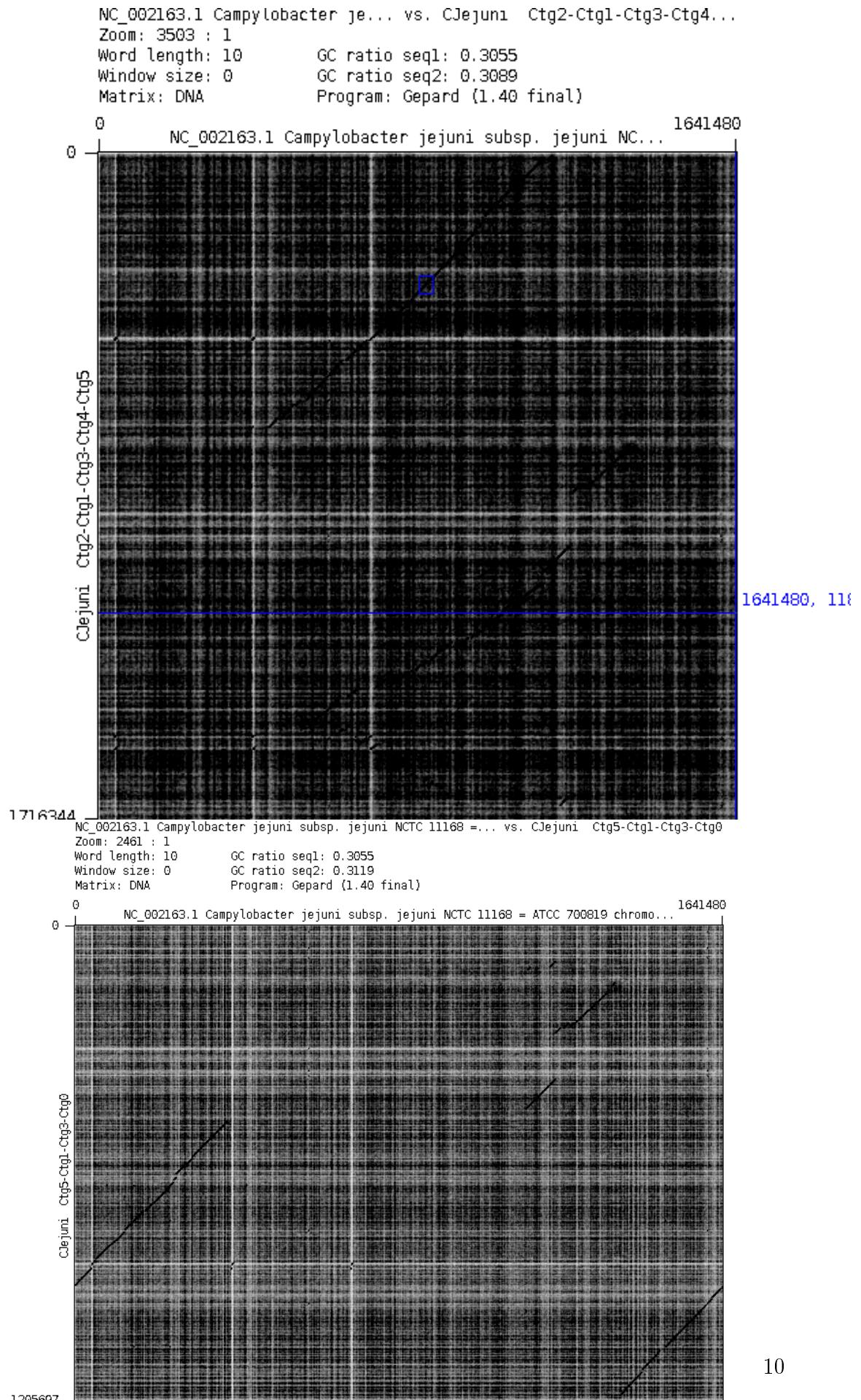
Najdulja sekvencia koju smo dobili je Ctg2-Ctg4-Ctg5-ctg1-Ctg3-Ctg0. Iako smo povezali sve contige u jednu sekvencu, nisu povezani potpuno točno. Uspjeli smo puno bolje naći manje dijelove od par sekvenci i povezati ih kao što vidimo na narednim slikama.



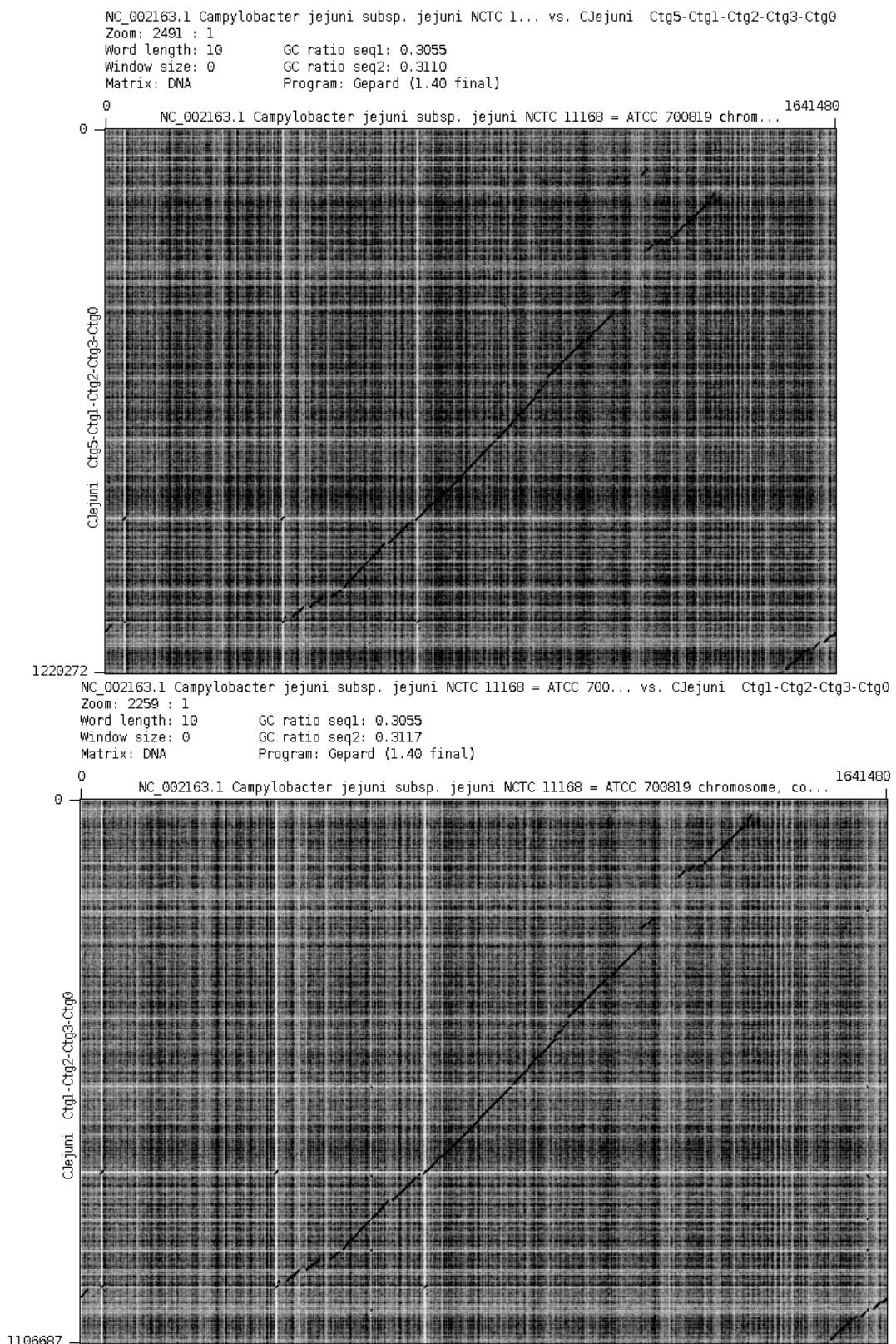
Slika 3.2: CJejuni - rezultati s Geparda 1



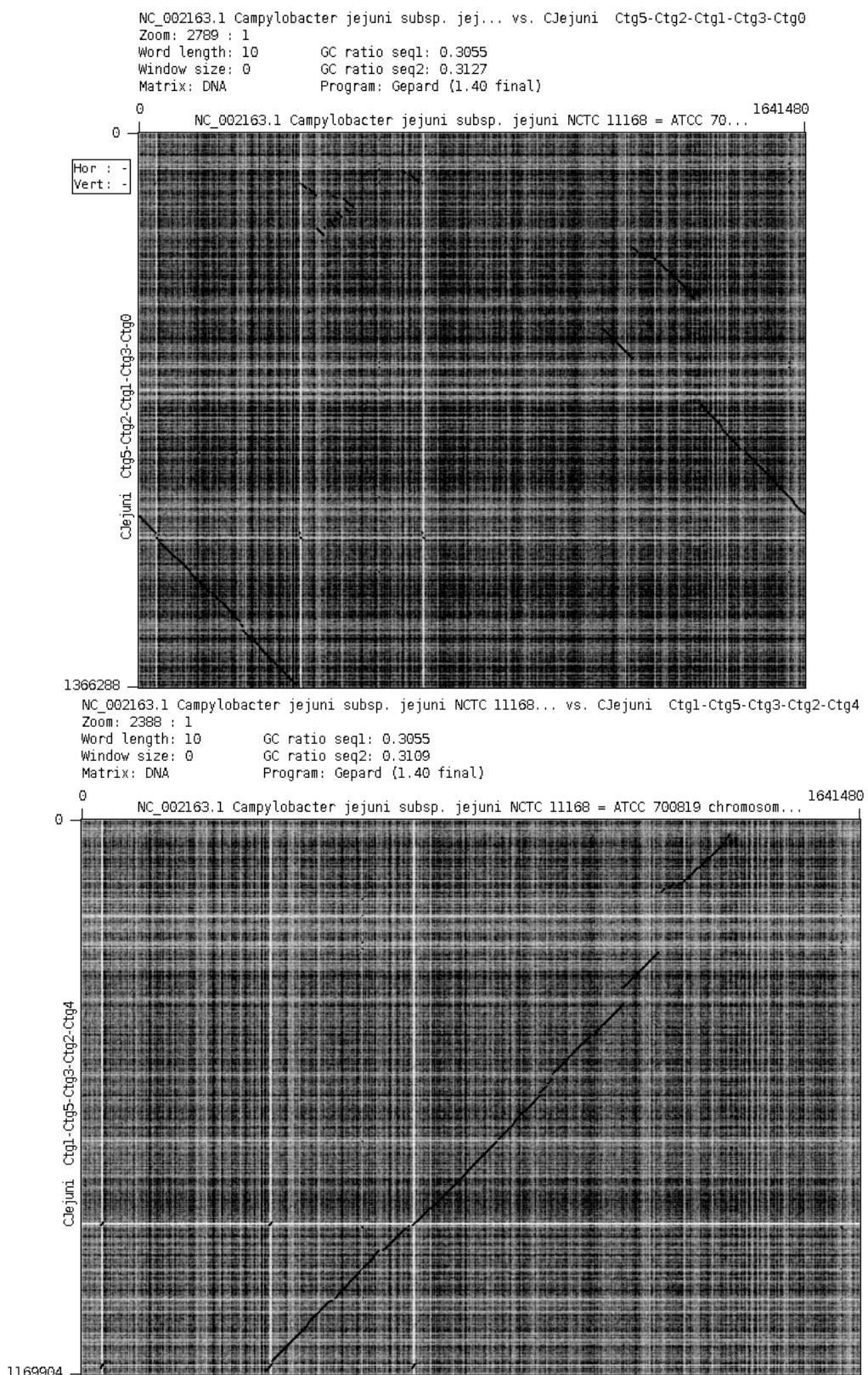
**Slika 3.3:** CJejuni - rezultati s Geparda 2



Slika 3.4: CJejuni - rezultati s Geparda 3



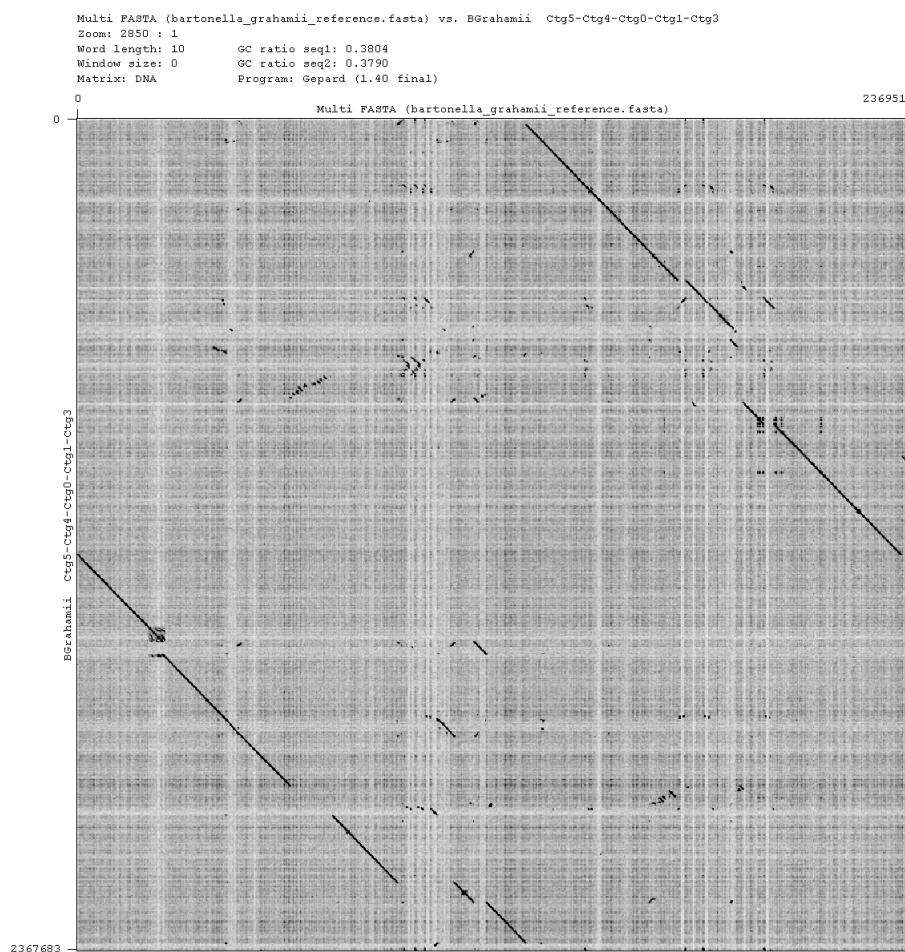
**Slika 3.5:** CJejuni - rezultati s Geparda 4



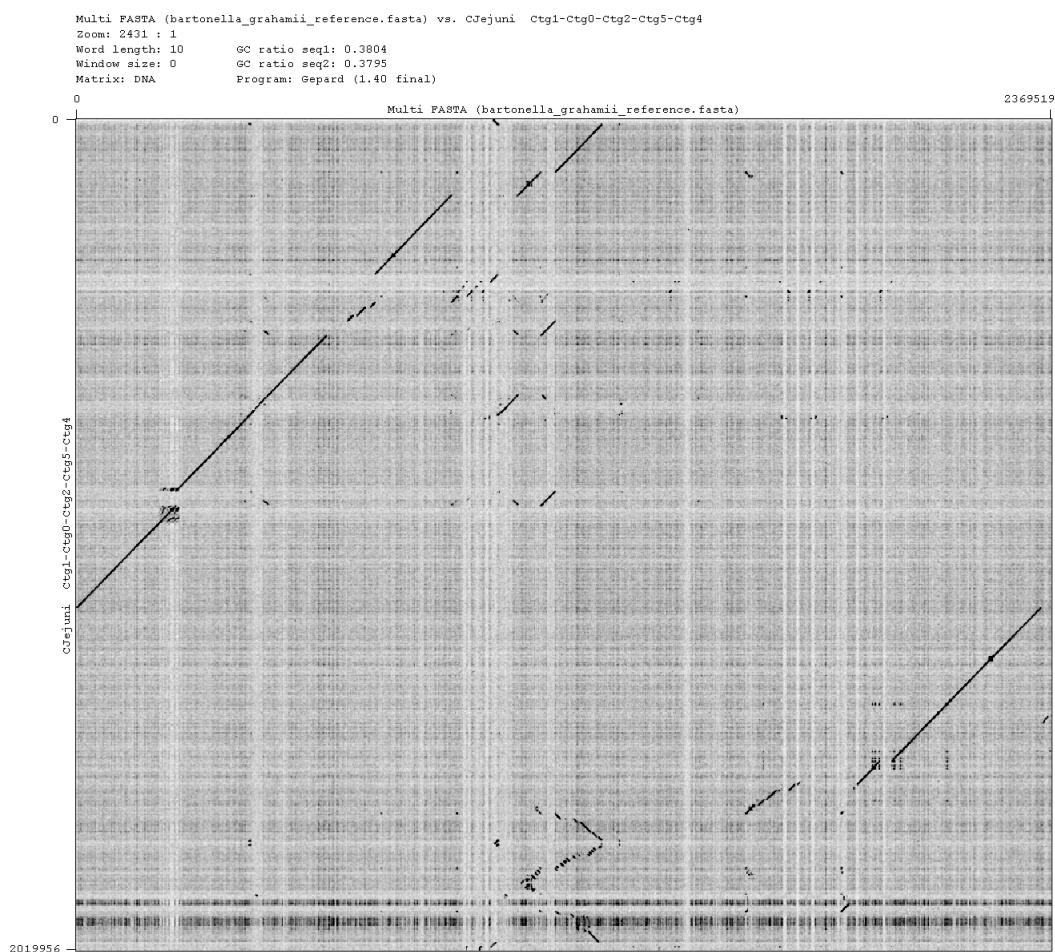
Slika 3.6: CJejuni - rezultati s Geparda 5

### 3.3. BGrahamii

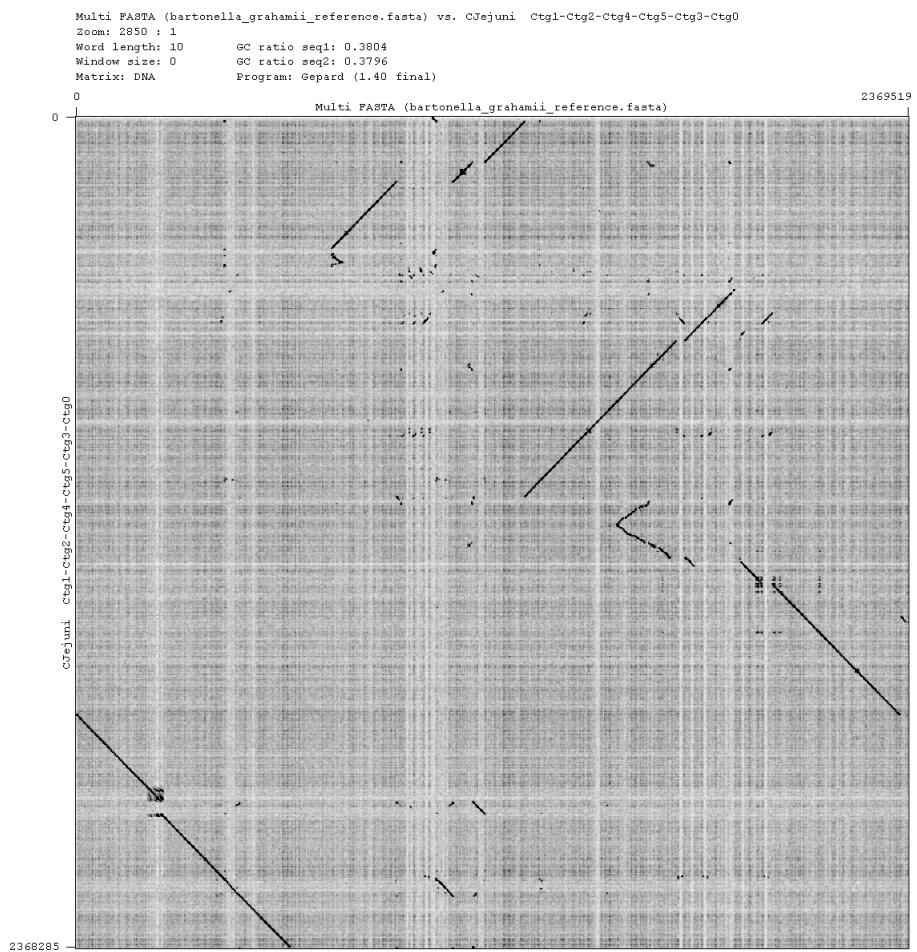
Rezultati kod BGrahamii genoma isto se ne poklapaju u potpunosti s referentnom sekvencom. Za procesiranje ovih podataka je potrebno 3-10 minuta i 2 GiB memorije (ovisno o parametrima programa). Sastavljene sekvence su vidljive na slikama u nastavku.



Slika 3.7: BGrahamii - rezultati s Geparda 1



**Slika 3.8:** BGrahamii - rezultati s Geparda 2



**Slika 3.9:** BGrahamii - rezultati s Geparda 3

## 4. Zaključak

Implementacija pomalo obskurnog i loše napisanog algoritma nam je donijela brojne probleme i natjerala nas da se fokusiramo više na maštu i vlastitu reimplementaciju. Napravili smo dosta promjena u radu. Neke iz neshvaćanja, a neke jer su nam naše implementacije bolje radile. Poigravali smo se s različitim načinima filtriranja te razrješavanjem konsenzusa grupa. Sami rezultati nisu bajni, ali postoji još puno mesta za napredak. Nismo se pozabavili stvarima poput drugačijih mjera za razješavanje konsenzusa grupa te nismo pretraživali u suprotnom smjeru od contige jer u tom slučaju bismo morali započeti pretragu ne samo od glave do repa već i od repa do glave svake contige što bi dodatno zakompliciralo implementaciju te prodljilo sam rad algoritma. Isto tako samo promatramo SI mjeru prilikom razrješavanja puteva i nije nam se isplatilo komplikirati implementaciju u slučaju da putevi imaju jednaku važnost.