

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Poboljšanje djelomično sastavljenog genoma dugim očitanjima

Domagoj Pluščec, Dunja Vesinger

Zagreb, siječanj 2019.

SADRŽAJ

1. Uvod	1
2. Opis algoritma	2
2.1. Ulazni podaci	2
2.2. Konstrukcija grafa preklapanja	3
2.3. Generiranje potencijalnih puteva	4
2.4. Određivanje konsenzus sekvenci	6
2.5. Konstrukcija grafa povezivanja	7
2.6. Generiranje izlaza	8
3. Rezultati	9
3.1. Analiza performansi prilikom promjene broja Monte Carlo iteracija	9
3.2. Analiza performansi prilikom promjene maksimalnog broja čvorova unutar puta	10
3.3. Usporedba dobivenih rezultata i referentne sekvence uporabom alata Gepard	13
3.3.1. Analiza rezultata početnih sekvenci	13
3.3.2. Analiza rezultata generiranih sekvenci	15
4. Zaključak	18
Literatura	19

1. Uvod

DNK sekvenciranje je proces određivanja redoslijeda nukleotida u DNK. Sekvenciranje cijelog genoma je ono sekvenciranje u kojem se nastoji odrediti redoslijed nukleotida čitavog genoma nekog organizma. Nažalost, današnje metode za očitavanje DNK ne omogućavaju da se odjednom očita cijeli genom. Umjesto toga, uređaji očitavaju samo kratka očitanja koja je potom potrebno sastaviti. Pored toga, nije poznat redoslijed dobivenih očitanja što dodatno otežava njihovo sastavljanje.

Sastavljanje očitanja vrši se na temelju međusobnih preklapanja između očitanja. Najveća prepreka ovakvom načinu sastavljanja je prisutnost dugačkih identičnih ili gotovo identičnih slijedova nukleotida u genomu koje nazivamo ponavljajuće regije. Druga prepreka sastavljanju genoma su pogreške koje mogu nastati pri očitavanju slijeda nukleotida. One mogu uzrokovati lažna preklapanja u očitanjima (Šikić i Domazet-Lošo, 2013).

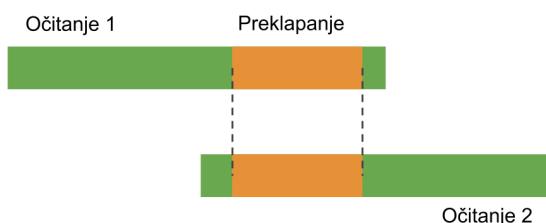
Sastavljanje očitanja još je uvijek otvoren problem na kojem rade mnogi znanstvenici. U ovom radu predstavljamo implementaciju algoritma koji omogućava efikasno sastavljanje genoma u kojima su prisutne ponavljajuće regije. Algoritam je predstavljen u radu (Du i Liang, 2018.). Algoritam koristi sirova očitanja i kontige nastale korištenjem postojećih alata za sastavljanje genoma te nastoji povezati nepovezane kontige korištenjem sirovih očitanja. Autori u svom radu navode kako je ova metoda drastično poboljšala rezultate sastavljanja u odnosu na dosad korištene metode.

2. Opis algoritma

Algoritam *Highly efficient repeat assembly* (skraćeno HERA) opisan je u radu (Du i Liang, 2018.). Njegov je cilj poboljšati kvalitetu sastavljenih očitanja. U tu svrhu algoritam koristi sirova očitanja te kontige nastale korištenjem postojećih alata za sastavljanje očitanja. Algoritam nastoji povezati dobivene kontige koristeći sirova očitanja. Za svaki par kontiga pronalaze se svi mogući putevi sastavljeni od sirovih očitanja koji povezuju ta dva kontiga. U konačnici se između svih dobivenih puteva između kontiga odabiru oni koji su najizgledniji i njima se kontizi međusobno povezuju. Detaljan pregled algoritma dan je u nastavku poglavlja.

2.1. Ulazni podaci

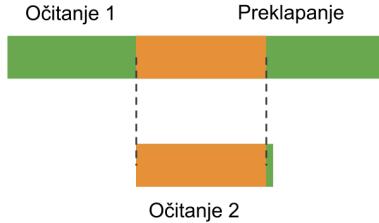
Algoritam kao ulazne podatke dobiva sirova očitanja i već sastavljene kontige u FASTA ili FASTQ formatu. Uz to, kao ulaz je potrebno proslijediti i preklapanja između kontiga i sirovih očitanja te međusobna preklapanja između očitanja u PAF formatu dobivena primjenom Minimap algoritma. Minimap algoritam i rezultirajući PAF format podataka opisani su u radovima (Li, 2017.) i (Li, 2018.).



Slika 2.1: Primjer preklapanja između dvaju kontiga koje je poželjno za rad HERA algoritma: kraj jednog očitanja preklapa se s početkom drugog očitanja.

Algoritam Minimap vratit će nam sva moguća preklapanja između očitanja, no HERA koristi samo ona preklapanja kod kojih se kraj jedne sekvene preklapa s početkom druge sekvene. (Pri tome *početak* i *kraj* sekvene ne znače da se stoga počinje od prvog nuk-

leotida u sekvenci i završava sa zadnjim nukleotidom, već označavaju približan položaj u očitanju.) Ovo se postiže filtriranjem sekvenci po mjeri koja se naziva identitet sekvene (engl. *sequence identity, SI*). Primjer poželjnog preklapanja za rad HERA algoritma dan je na slici 2.1. Slika 2.2 prikazuje preklapanje kod kojeg je jedna sekvenca posve sadržana u drugoj sekvenci. Ova vrsta preklapanja nije korisna za rad HERA algoritma te takva preklapanja želimo filtrirati.



Slika 2.2: Primjer preklapanja između dvaju kontiga koje nije poželjno za rad HERA algoritma: jedno očitnje je sadržano u drugom očitanju.

Nakon učitavanja i filtriranja podataka, najprije se konstuiru graf preklapanja.

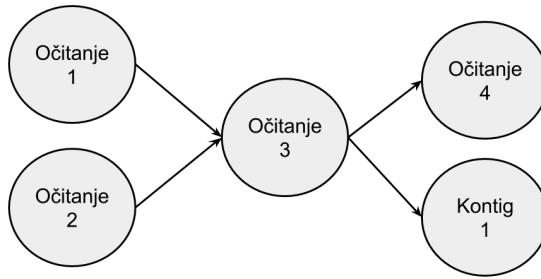
2.2. Konstrukcija grafa preklapanja

Graf preklapanja konstruira se na temelju preklapanja između sekvenci koja su dobivena Minimap algoritmom. U grafu preklapanja svaka sekvenca nukleotida (kontig ili sirovo očitanje) predstavlja jedan čvor grafa. Svako preklapanje između dvije sekvence, tj. dva čvora, predstavlja jedan brid u grafu. Kako smo prilikom učitavanja filtrirali preklapanja, preostala su samo ona u kojima se kraj jedne sekvence preklapa s početkom druge sekvene. Ovakva preklapanja jednoznačno definiraju poredak sekvenci pa je jasno da naš graf preklapanja u tom slučaju treba biti usmjereni graf.

Graf konstruiramo na sljedeći način. Najprije za svaku sekvencu nukleotida definiranu u ulaznim podacima stvaramo čvor. Potom prolazimo kroz sva preklapanja i za svako preklapanje dodajemo brid između čvorova koji predstavljaju početnu i završnu sekvencu preklapanja.

Slika 2.3 prikazuje primjer jednog segmenta mogućeg grafa preklapanja. Iz ovog grafa može se očitati kako postoje sljedeća preklapanja: kraj očitanja 1 preklapa se s početkom očitanja 3, kraj očitanja 2 preklapa se s početkom očitanja 3, kraj očitanja 3 preklapa se s početkom očitanja 4 i kraj očitanja 3 preklapa se s početkom kontiga 1.

Nakon što smo generirali graf svih preklapanja, nastojimo pronaći puteve u tom grafu koji mogu spojiti dva kontiga.

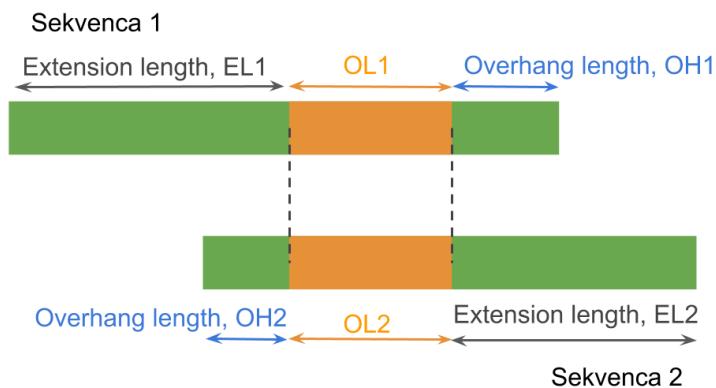


Slika 2.3: Primjer bridova za jedan čvor u grafu preklapanja.

2.3. Generiranje potencijalnih puteva

Prilikom generiranja potencijalnih puteva nastoje se pronaći putevi kojima bi se mogla povezati neka dva kontiga. Svaki put sastoji se od niza čvorova u grafu preklapanja koji su međusobno povezani bridovima. Svi bridovi na nekom putu moraju biti istog usmjerenja, a svi čvorovi unutar puta moraju biti jedinstveni, tj. mogu biti sadržani u putu najviše jednom.

Potencijalni putevi generiraju se zasebno za svaki kontig. Za kontig najprije uzmememo sve izlazne bridove prema drugim čvorovima. Potom uzmememo susjedne čvorove koji su na kraju tih bridova i za svaki susjedni čvor nastojimo generirati put od njega do nekog drugog kontiga. Taj put generiramo tako da za susjedni čvor odaberemo jedan brid koji vodi do nekog sljedećeg čvora. Potom iz tog čvora ponovo odabiremo jedan brid i tako sve dok ne dosegnemo jedan od uvjeta zaustavljanja. Bitno je naglasiti da se prilikom generiranja puta iz nekog čvora ne smijemo više puta obići isti čvor, stoga moramo pažljivo odabirati sljedeći brid na putu imajući to na umu. Postoje tri različite vrste selekcije na temelju kojih odabiremo sljedeći brid na putu. Za svaki izlazni brid koji je povezan s početnim čvorom koristimo sve tri metode selekcije.



Slika 2.4: Veličine kod preklapanja dvaju sekvenci definirane u PAF formatu.

Selekcije kojima se odabire sljedeći brid za nastavak puta međusobno se razlikuju po

kriteriju odabira brida. Prva selekcija odabire brid na temelju mjere koja se naziva *overlap score*, OS . Metoda izračuna ove mjere objašnjena je u nastavku. Neka su zadane dvije sekvene, S_1 i S_2 , koje imaju preklapanje. Preklapanje je definirano sljedećim veličinama: identitetom sekvene SI , duljinama *overhang*-ova OH_1 i OH_2 , duljinama preklapanja OL_1 i OL_2 te duljinama *extension*-a EL_1 i EL_2 . Sve navedene veličine dobivaju se kao rezultat Minimap algoritma i zapisane su u PAF zapisu preklapanja, a vizualno su prikazane na slici 2.4.

Overlap score preklapanja između sekvenci S_1 i S_2 računa se na sljedeći način:

$$OS = (OL_1 + OL_2) \cdot \frac{SI}{2}$$

Selekcija odabire onaj brid za koji je *overlap score* najveći.

Druga selekcija odabire sljedeći brid na temelju veličine koja se naziva *extension score*, ES . Ako postoji brid koji je usmjeren od sekvene S_1 prema sekveni S_2 , onda se *extension score* takvog preklapanja računa na sljedeći način:

$$ES = OS + \frac{EL_2}{2} - \frac{OH_1 + OH_2}{2}$$

Selekcija odabire onaj brid za koji je *extension score* najveći.

Treća selekcija koristi Monte Carlo metodu za odabir sljedećeg brida. Monte Carlo selekcija provodi se nad skupom svih bridova kojima je moguće nastaviti trenutni put i to proporcionalno s obziron na njihov *extension score*.

Prva i druga metoda selekcije su determinističke i svaka od njih može generirati najviše jedan put za neki izlazni brid iz početnog čvora. Treća metoda je nedeterministička te ju možemo pokrenuti proizvoljan broj puta i ona će vrlo vjerojatno prilikom svakog pokretanja generirati novi put.

Postoje tri načina zaustavljanja prilikom generiranja puteva. Prvi je da ne postoji niti jedan brid iz trenutnog čvora ili svi postojeći bridovi vode u čvorove koji su već obiđeni na trenutnom putu. U tom slučaju smo naišli na "slijepu ulicu" prilikom generiranja puta i odbacujemo putanju koju smo do sada generirali. Drugi način je da premašimo maksimalan dozvoljeni broj čvorova unutar puta. Generiranje potencijalnih puteva može biti vrlo zah-tjevno za računalne resurse te stoga ograničavamo maksimalnu duljinu generiranih puteva. U slučaju da put premaši maksimalnu dozvoljenu duljinu, odbacuje se. Konačno, put je us-pješno generiran kad odaberemo brid na čijem se kraju nalazi drugi kontig. U tom slučaju spremamo generirani put.

Jednom kad smo generirali puteve, potrebno je odabrati onaj najvjerojatniji, tj. odrediti konsenzus sekvene.

2.4. Određivanje konsenzus sekvenci

U konačnici je dva kontiga moguće povezati samo pomoću jednog puta. Stoga između svih puteva koje smo generirali za svaki par kontiga treba odabrati jedan za koji je najvjerojatnije da predstavlja slijed nukleotida u originalnom genomu. Algoritam HERA u tu svrhu najprije generira grupu puteva za koje je najvjerojatnije da odgovaraju stvarnom slijedu nukleotida, a potom se iz navedene grupe bira jedan reprezentativni put. U nastavku je opisan način generiranja takvih grupa koje sadržavaju najvjerojatnije puteve, a nazivaju se konsenzus sekvence.

Prepostavimo da želimo odrediti konsenzus sekvencu puteva od kontiga predstavljenog čvorom N_1 do kontiga predstavljenog čvorom N_2 . Najprije pronađemo sve puteve koji povezuju čvor N_1 s čvorom N_2 i grupiramo ih po duljini. Ako je razlika duljina između najkraćeg najduljeg puta manja od 10000, tada formiramo samo jednu grupu u kojoj se nalaze svi putevi. U suprotom formiramo grupe tako da najprije uzmemmo najkraći put i s njim grupiramo sve puteve čija je duljina od zadanoj puta veća za najviše 1000. Potom odabiremo najkraći od preostalih puteva (onih koje još nismo grupirali) i formiramo novu grupu na analogan način. Postupak ponavljamo sve dok ne grupiramo sve puteve.

Nakon toga za svaku grupu odredimo frekvencije duljina puteva koji se nalaze u toj grupi. Primjerice, ako grupa sadrži 10 puteva duljine 1000, tada je frekvencija puteva duljine 1000 u toj grupi jednaka 10. Jednom kad smo odredili sve frekvencije duljina, filtriramo svaku grupu s obzirom na najveću frekvenciju unutar nje. Iz grupe izbacujemo sve puteve čija je duljina takva da je frekvencija pojavljivanja te duljine unutar grupe manja od 50% najveće frekvencije.

Na primjer, recimo da imamo grupu koja se sastoji od 40 puteva duljine 1000, 30 puteva duljine 1100 i 10 puteva duljine 1200. Tada maksimalna frekvencija te grupe iznosi 40. Puteve duljine 1100 zadržavamo u grapi je jer njihova frekvencija veća od 50% najveće frekvencije: $30 > 0.5 \cdot 40$. Sve puteve duljine 1200 izbacujemo iz grupe jer im je frekvencija premala u odnosu na maksimalnu frekvenciju.

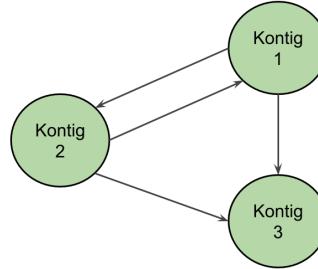
Ako imamo samo jednu grupu puteva od čvora N_1 do čvora N_2 , onda tu grupu proglašavamo konsenzus sekvencom. U slučaju da imamo dvije grupe, odabiremo onu koja ima dulje puteve i nju proglašavamo konsenzus sekvencom. Kada imamo više od dvije grupe, provodimo iterativni algoritam za odabir konsenzus sekvenca. Najprije odabiremo grupu koja ima najveću maksimalnu frekvenciju, tj. čija je maksimalna frekvencija veća od maksimalnih frekvencija svih ostalih grupa. Zatim promatramo prvu sljedeću grupu čiji su putevi dulji od odabrane grupe. Ako je njezina maksimalna frekvencija veća od 50% maksimalne frekvencije odabrane grupe, ona postaje nova odabrana grupa. U suprotnom, odabrana grupa postaje konsenzus sekvenca.

Kad smo odredili konsenzus sekvencu od čvora N_1 do čvora N_2 , potrebno je također odrediti konsenzus sekvencu od čvora N_2 do čvora N_1 . Ovaj postupak ponavljamo za svaki par kontiga u grafu. Ako između neka dva kontiga ne postoji niti jedan put, onda ne postoji niti konsenzus sekvencia između njih.

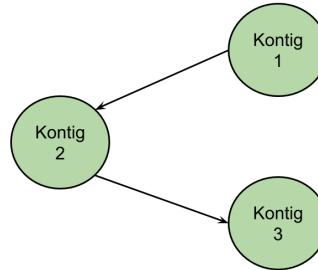
2.5. Konstrukcija grafa povezivanja

Nakon što smo odredili konsenzus sekvence između svih kontiga, možemo konstruirati graf povezivanja. Čvorovi grafa povezivanja su svi kontizi, a bridovi su predstavljeni konsenzus sekvencama. Važno je napomenuti kako su bridovi ovog grafa također usmjereni te između dva čvora, N_1 i N_2 , mogu postojati dva brida: brid od N_1 do N_2 te brid od N_2 do N_1 . U ovom grafu jedan kontig može imati bridove prema proizvoljnom broju drugih kontiga. No, u konačnom rješenju algoritma svaki kontig može imati najviše jedan ulazni i najviše jedan izlazni brid. Uz to, svaki se čvor u konačnom rješenju može pojaviti najviše jedanput. Stoga je potrebno eliminirati sve cikluse iz grafa.

Slika 2.5 prikazuje primjer jednog mogućeg grafa povezivanja za ulaz koji sadrži tri kontiga. Ovaj graf sadrži bridove koji ne mogu svi istovremeno postojati te je neke od njih potrebno odstraniti iz grafa. Slika 2.6 prikazuje moguće rješenje nakon eliminacije konfliktnih bridova. Način odabira bridova u grafu povezivanja opisan je u sljedećem potpoglavlju.



Slika 2.5: Primjer grafa povezivanja.



Slika 2.6: Primjer konačnog grafa iz kojeg se generira izlaz sustava.

2.6. Generiranje izlaza

Kako bismo odredili konačno rješenje, moramo odabratи bridove koje je potrebno eliminirati iz grafa povezivanja. U tu svrhu definiramo konflikt indeks, CI , za svaki čvor u grafu povezivanja. Za neki čvor N_1 , konflikt index se računa na sljedeći način. Svaka konsenzus sekvenca, tj. svaki brid u grafu povezivanja, ima definiran broj puteva (engl. *path number*, NP) koji su sadržani u njoj. Neka konsenzus sekvenca od čvora N_1 do čvora N_m ima najveći broj puteva od svih konsenzus sekvenci koje izlaze iz čvora N_1 , a konsenzus sekvenca od N_1 do N_n drugi najveći broj puteva. Označimo ove brojeve puteva s NP_{1m} i NP_{1n} . Konflikt indeks za čvor N_1 tada iznosi:

$$CI_1 = \frac{NP_{1n}}{NP_{1m}}$$

Ako je konflikt indeks nekog čvora velik, onda nije jasno definirano s kojim ga čvorom treba spojiti te se čvor ne spaja. Unaprijed definiramo konstantu C_{max} koja predstavlja najveći dozvoljeni konflikt indeks za koji će se izvršiti spajanje.

Kako bismo spojili čvorove, najprije ih poredamo po njihovim konfliktnim indeksima, od čvora s najmanjim konfliktnim indeksom do čvora s najvećim konfliktnim indeksom. Krećemo od čvora s najmanjim konflikt indeksom N_1 i odabiremo konsenzus sekvencu čvora N_1 koja ima najveći broj puteva te ga pomoću nje spajamo na čvor N_i na koji pokazuje konsenzus sekvenca. Pošto je početak čvora N_i sada spojen na čvor N_1 , mičemo čvor N_i iz liste čvorova na koju se moguće spojiti i ponovo računamo kolifikt indekse za preostale čvorove u grafu tako da najprije izbrišemo sve postojeće konsenzus sekvene koje vode prema čvoru N_i , a potom za preostale konsenzus sekvene provodimo izračun. Ovaj postupak se ponavlja sve dok konflikt indeksi svih preostalih čvorova ne budu veći od C_{max} .

Na ovaj način je određeno koji čvorovi trebaju biti povezani, tj. koje kontige je potrebno međusobno spojiti i kojim redoslijedom. Svaka spojnica između dvaju kontiga definirana je jednom konsenzus sekvencom. Konačno, za svaku konsenzus sekvencu odabiremo jedan put koji će predstavljati konkretnu sponicu dvaju kontiga. Najprije u konsenzus sekvenci dohvativamo sve puteve takve da je njihova duljina ona koja ima najveću frekvenciju unutar konsenzus sekvence, tj. da unutar konsenzus sekvene postoji najviše puteva upravo te duljine. Zatim od tih puteva nasumično odabiremo jedan koji će predstavljati konačnu spojnicu između dva kontiga.

3. Rezultati

U okviru ovog poglavlja prezentirani su rezultati prilikom korištenja razvijenog programa za poboljšanje djelomično sastavljenog genoma dugim očitanjima. Poglavlje je podijeljeno u tri potpoglavlja. U prvom potpoglavlju dana je analiza performansi programa prilikom promjene broja Monte Carlo iteracija, u drugom potpoglavlju dana je analiza performansi prilikom promjene maksimalnog broja čvorova unutar puta. U zadnjem potpoglavlju je dana usporedba rezultata programa s referencom uporabom alata Gepard (Krumseik et al., 2007.).

3.1. Analiza performansi prilikom promjene broja Monte Carlo iteracija

U okviru ovog potpoglavlja analizirane su performanse algoritma prilikom promjene broja Monte Carlo iteracija. Jedna Monte Carlo iteracija predstavlja traženje puta iz svakog izlaznog brida početnog čvora puta pri čemu početni čvor puta predstavlja kontig. Razmatran je raspon vrijednosti Monte Carlo iteracija između 0 i 100.

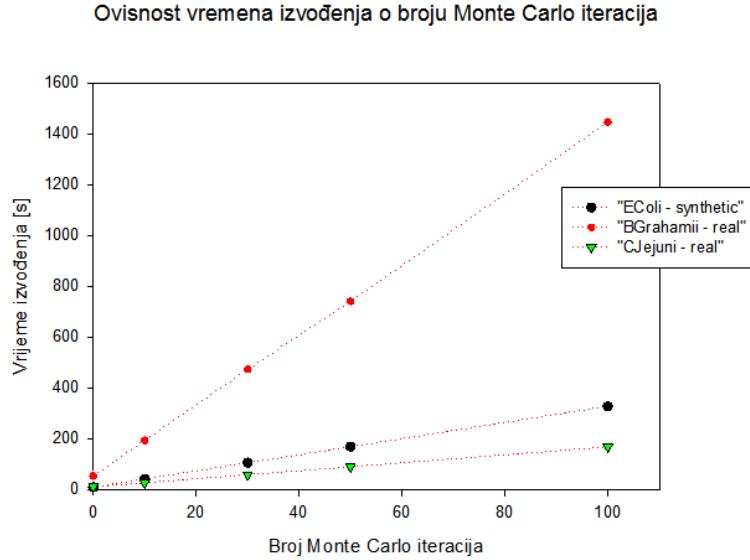
Promatrane performanse algoritma su maksimalno memorjsko zauzeće koje zauzima proces (engl. *maximum resident set size*), vrijeme izvođenja, ukupan broj generiranih puteva i broj generiranih sekvenci. Eksperiment je ponavljan deset puta te je u rezultatima prikazana prosječna vrijednost.

Prilikom izvođenja eksperimenta ostali parametri su bili fiksni i iznosili su: minimalni identitet sekvence $SImin = 0.5$, maksimalni identitet sekvence $SImax = 1$, maksimalni broj čvorova u putu $Np = 1000$, maksimalni konfliktni index $CImax = 0.7$.

Kako se količina zauzeća memorije mijenjala zanemarivo malo u odnosu na ukupno zauzeće memorije (red veličine 0.5 MB) promjenu zauzeća nismo analizirali. Prilikom testiranja na sekvenci “EColi - synthetic” maksimalno zauzeće je iznosilo 306.6 MB, za sekvencu “BGrahamii - real” zauzeće je iznosilo 1597.0 MB, a za sekvencu “CJejuni - real” zauzeće je iznosilo 475.4 MB.

Prilikom promjene broja Monte Carlo iteracija broj konačnih sekvenci uz dane parametre se nije mijenjao. Za sekvence “EColi - synthetic” i “CJejuni - real” generirana je jedna konačna sekvencia dok je za sekvencu “BGrahamii - real” generirano četiri sekvence.

Na grafu 3.1 je prikazana promjena vremena izvođenja programa za navedene tri sekvene prilikom promjene broja Monte Carlo iteracija. Iz grafa možemo primijetiti kako vrijeme trajanja značajno raste što je najvidljivije kod sekvene "BGrahamii - real".



Slika 3.1: Ovisnost vremena izvođenja o broju Monte Carlo iteracija.

Promjena ukupnog broja putova za navedene sekvene je navedena u nastavku. Za sekvencu "EColi - synthetic" generirano je 2673 putova za broj Monte Carlo iteracija manji od 10, dok je za veći iznos generirano 2674 putova. Za sekvencu "BGrahamii - real" generirano je 3781 put za broj Monte Carlo iteracija manji od 10, 3783 za broj iteracija manji od 30, te 3784 za veći broj iteracija. Za sekvencu "CJejuni - real" generirano je 1289 putova za broj Monte Carlo iteracija manji od 10, 1293 za broj iteracija manji od 50 te 1294 za veći broj iteracija.

Iz rezultata je vidljivo kako broj generiranih putova s porastom broja Monte Carlo iteracija raste izrazito sporo. Razlozi za takvo ponašanje su navedeni u nastavku. Prvi je razlog taj što je dio putova već uzet u obzir prilikom odabira pomoću najvećeg faktora produljenja. Drugi je razlog taj što se prilikom nasumičnog pretraživanja često dođe do čvorova koji nemaju djecu. Treći razlog je taj što prilikom takvog pretraživanja dolazimo i do predugačkih sekvenci.

3.2. Analiza performansi prilikom promjene maksimalnog broja čvorova unutar puta

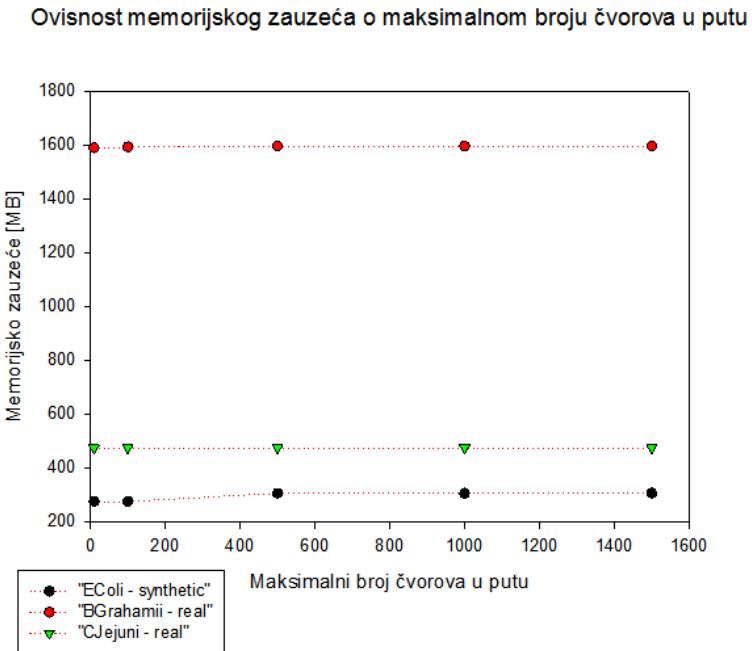
U okviru ovog potoglavlja analizirane su performanse algoritma prilikom promjene maksimalnog broja čvorova od kojih se put između dva kontiga može sastojati. Razmatran je raspon vrijednosti maksimalnog broja čvorova između 10 i 1500.

Prilikom izvođenja eksperimenta ostali parametri su bili fiksni iznosili su: minimalni identitet sekvence $SImin = 0.5$, maksimalni identitet sekvence $SImax = 1$, broj Monte Carlo iteracija $Nmc = 10$ i maksimalni konfliktni index $CImax = 0.7$.

Promatrane performanse algoritma su maksimalno memorijsko zauzeće koje zauzima proces, vrijeme izvođenja, ukupan broj generiranih puteva, broj generiranih sekvenci. Eksperiment je ponavljan deset puta te je u rezultatima prikazana prosječna vrijednost.

Za sekvencu "EColi - synthetic" generirane su tri sekvence za maksimalni broj čvorova manji od 500, dok je za veći maksimalni broj putova generirana jedna sekvencia. Za sekvencu "BGrahamii - real" generirane su tri sekvence za maksimalni broj čvorova manji od 500, dok je za veći broj čvorova generirano četiri sekvence. Za sekvencu "CJejuni - real" generirane su tri sekvence za maksimalni broj čvorova manji od 100, a za veći broj čvorova generirana je jedna sekvencia.

Na grafu 3.2 je prikazano kretanje memorijskog zauzeća prilikom povećanja maksimalnog broja čvorova unutar puta. Možemo primijetiti kako prilikom povećanja maksimalnog broja čvorova unutar puta memorijsko zauzeće raste te kako je rast veći nego što je bio prilikom promjene broja Monte Carlo iteracija.

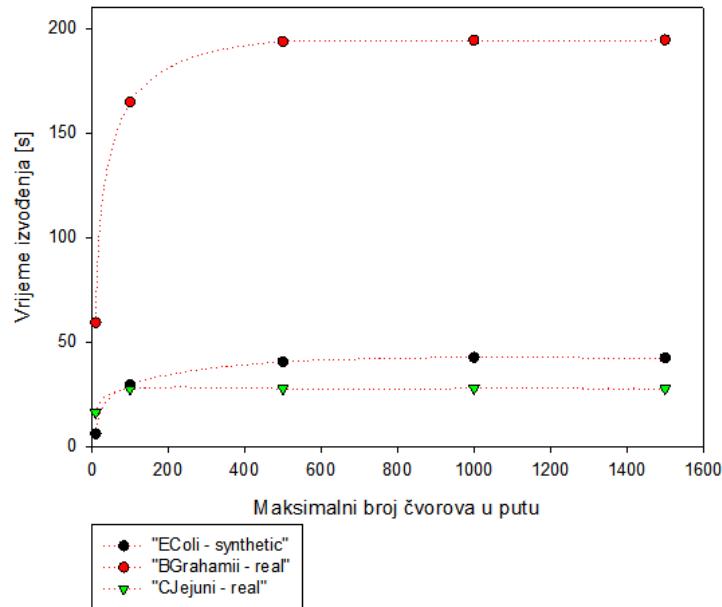


Slika 3.2: Ovisnost memorijskog zauzeća o maksimalnom broju čvorova u putu.

Na grafu 3.3 je prikazano vrijeme izvođenja programa za različiti maksimalni broj čvorova unutar puta. Možemo primijetiti kako prilikom povećanja maksimalnog broja čvorova unutar puta vrijeme izvođenja raste, no rast je manji u usporedbi s rastom vremena prilikom povećanja Monte Carlo iteracija. Vrijeme izvođenja počinje stagnirati kako se počinje smanjivati prirast generiranih putova, što možemo zaključiti prilikom usporedbe s grafom

ovisnosti broja generiranih putova o maksimalnom broju čvorova u putu.

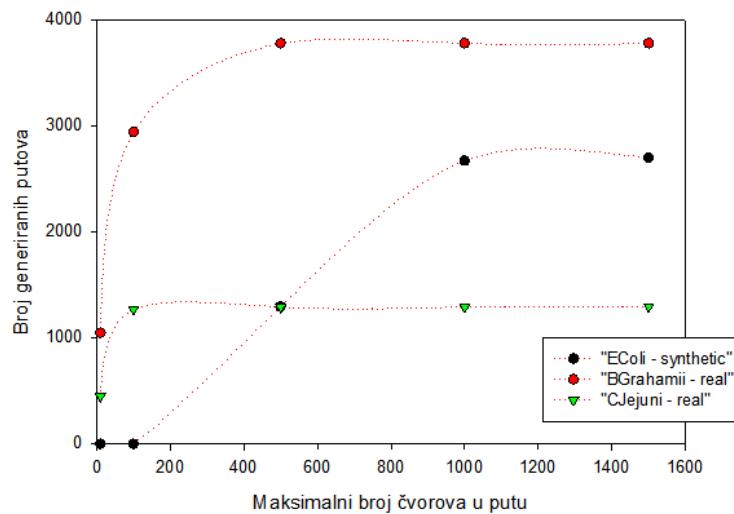
Ovisnost vremena izvođenja o maksimalnom broju čvorova u putu



Slika 3.3: Ovisnost vremena izvođenja o maksimalnom broju čvorova u putu.

Graf 3.4 prikazuje ovisnost ukupnog broja generiranih putova o maksimalnom broju čvorova unutar puta. Možemo primijetiti kako se prilikom promjene maksimalnog broja čvorova unutar puta broj puteva značajno mijenja te kako se prirast puteva smanjuje prilikom povećanja maksimalnog broja čvorova unutar puta što je i za očekivati jer je se smanjuje broj preostalih čvorova, a jedan čvor ne možemo dva puta iskoristiti. Također, broj puteva se neće mijenjati nakon što povećavamo maksimalni broj čvorova unutar puta iznad ukupnog broja čvorova.

Ovisnost broja generiranih putova o maksimalnom broju čvorova u putu

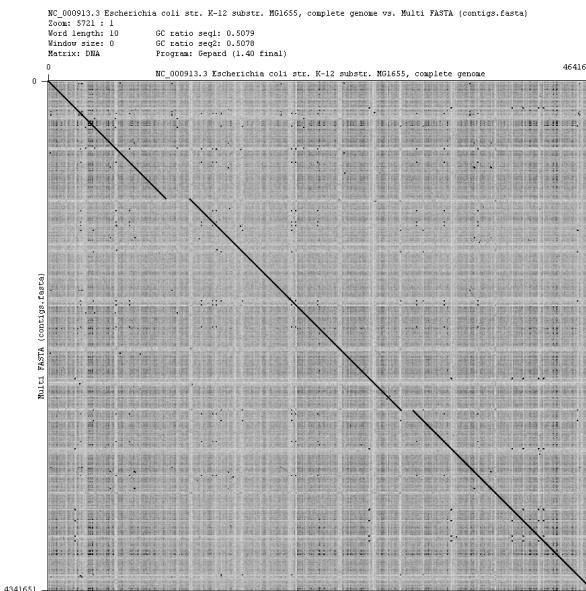


Slika 3.4: Ovisnost broja generiranih putova o maksimalnom broju čvorova u putu.

3.3. Usporedba dobivenih rezultata i referentne sekvene uporabom alata Gepard

U okviru ovog potpoglavlja grafički su analizirani rezultati algoritma. Rezultati su uspoređeni uporabom alata Gepard za prikaz sekvenci, usporedbom broja početnih kontiga i generiranih sekvenci te usporedbom duljine generirane sekvence i referentne sekvence.

3.3.1. Analiza rezultata početnih sekvenci



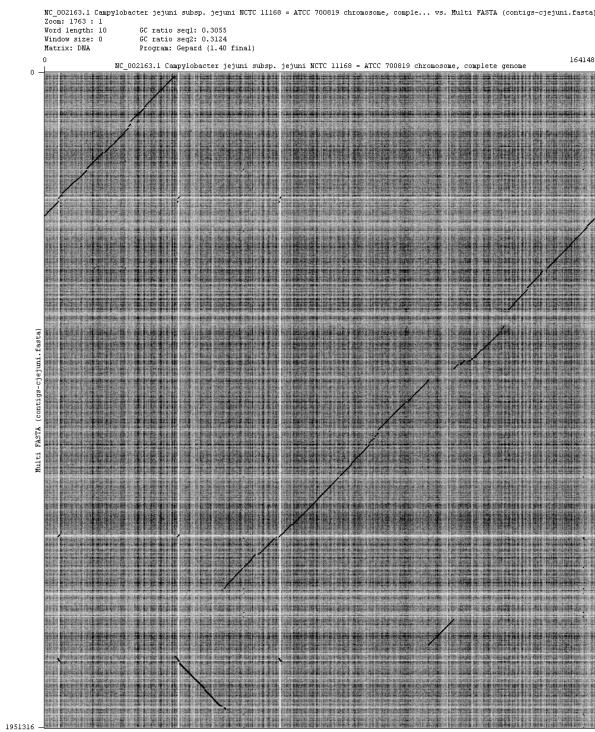
Slika 3.5: “EColi - synthetic” - usporedba kontig sekvenci s referentnom sekvencom.

Kako bi mogli analizirati uspješnost programa, potrebno je proučiti kakvi su rezultati ako bi kao sekvene uzeli ulazne kontige. Rezultati za tri promatrane sekvene dani su u nastavku. Apscisa prikazanih grafa odgovara referentnim sekvencama, a ordinata ulaznim kontizima.

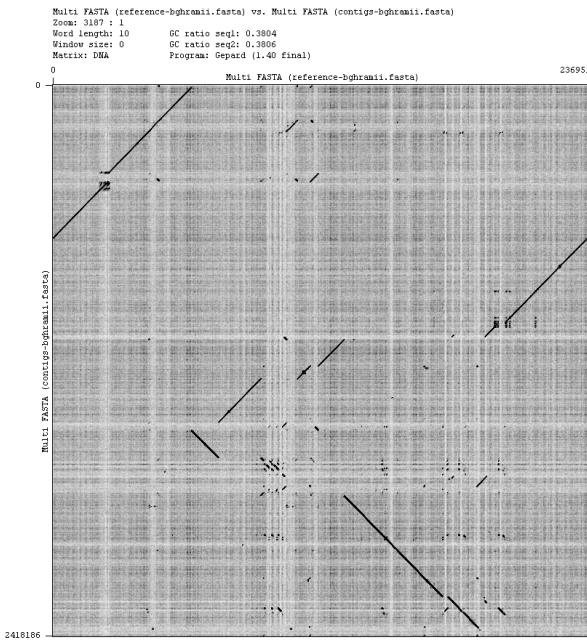
Slika 3.5 predstavlja usporedbu kontiga sekvene “EColi - synthetic” s referencom. Zbroja duljina kontiga iznosi 4.341.651, dok je duljina reference 4.641.651. Razlika duljina iznosi 300.000. Ulazna sekvenca se sastoji od tri kontiga.

Slika 3.7 predstavlja usporedbu kontiga sekvene “BGrahamii - real” s referencom. Zbroja duljina kontiga iznosi 2.418.186, dok je duljina reference 2.369.519. Razlika duljina iznosi –48.667 pri čemu je rezultat negativan jer je zbroj duljina kontiga veći od duljine reference. Ulazna sekvenca se sastoji od šest kontiga.

Slika 3.6 predstavlja usporedbu kontiga sekvene “CJejuni - real” s referencom. Zbroja duljina kontiga iznosi 1.951.316, dok je duljina reference 1.641.480. Razlika duljina iznosi –309.836, pri čemu je rezultat negativan jer je zbroj duljina kontiga veći od duljine reference. Ulazna sekvenca se sastoji od šest kontiga.



Slika 3.6: “CJejuni - real” - usporedba kontig sekvenci s referentnom sekvencom.



Slika 3.7: “BGrahamii - real” - usporedba kontig sekvenci s referentnom sekvencom.

3.3.2. Analiza rezultata generiranih sekvenci

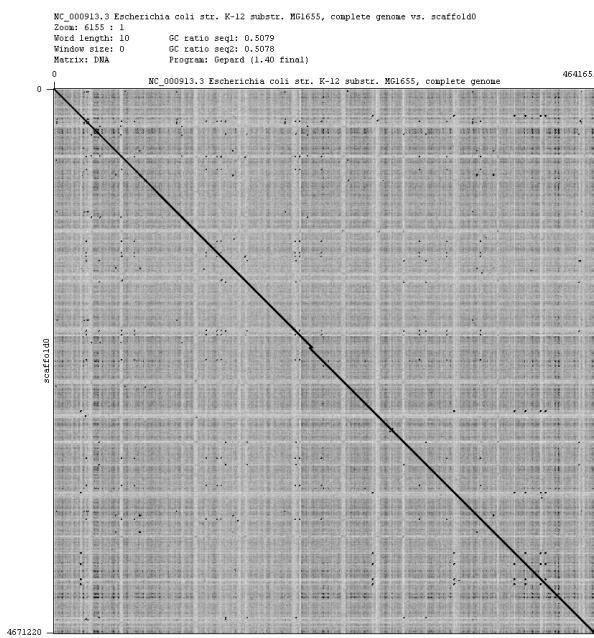
Eksperiment čiji su rezultati prikazani u nastavku je imao sljedeće iznose parametara: minimalni identitet sekvence $SI_{min} = 0.5$, maksimalni identitet sekvence $SI_{max} = 1$, broj Monte Carlo iteracija $N_{mc} = 10$, maksimalni broj čvorova u putu 1000 i maksimalni konfliktni index $CImax = 0.5$.

Prilikom prikazivanja grafova na apscisi se nalazi referentna sekvencia, dok se na ordinati nalaze generirane sekvence (skafoldi).

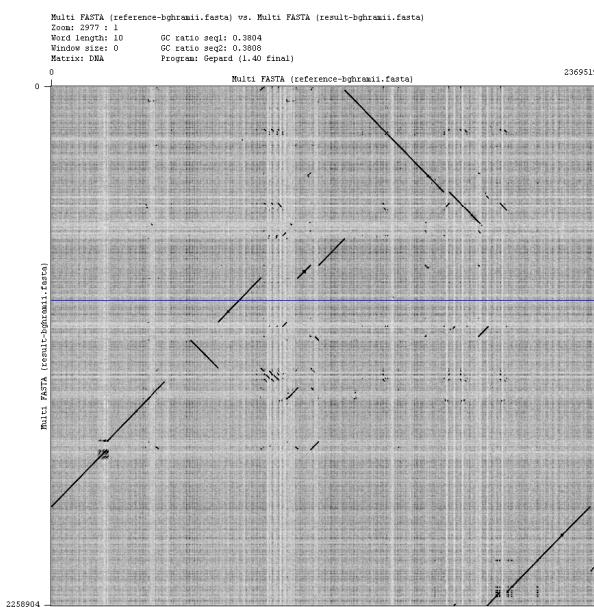
Rezultati za sekvencu “EColi - synthetic” prikazani su na slici 3.8. Program je generirao jednu sekvencu duljine 4.671.220. Apsolutna razlika između generirane sekvence i referentne sekvence iznosi 29569 što je deset puta manja razlika nego prilikom usporedbe sa ulaznim kontizima.

Rezultati za sekvencu “BGrahamii - real” prikazani su na slici 3.9. Program je generirao četiri sekvence duljine 2.258.904, pri čemu je smanjio broj sekvenci s izvornih 6. Apsolutna razlika između generirane sekvence i referentne sekvence iznosi 110.615.

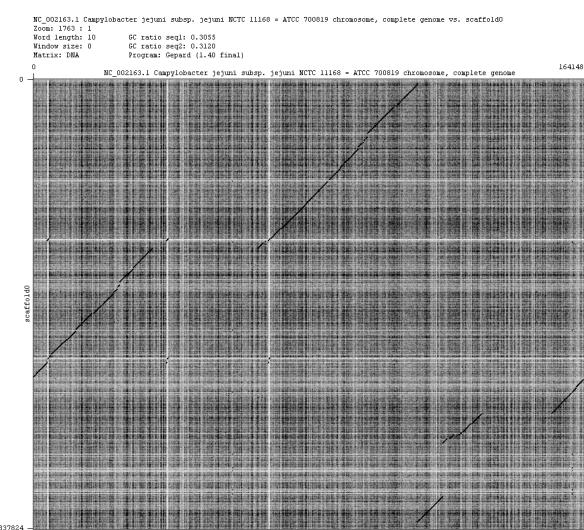
Rezultati za sekvencu “CJejuni - real” prikazani su na slici 3.10. Program je generirao jednu sekvencu duljine 1.337.824, pri čemu je smanjio broj sekvenci s izvornih 6. Apsolutna razlika između generirane sekvence i referentne sekvence iznosi 303.656.



Slika 3.8: “EColi -synthetic”: usporedba generirane sekvence sa referentnom sekvencom.



Slika 3.9: “B Grahamii - real”: usporedba generirane sekvence sa referentnom sekvencom.



Slika 3.10: “CJejuni - real”: usporedba generirane sekvence sa referentnom sekvencom.

4. Zaključak

U okviru ovog projekta predstavljena je implementacija algoritma Hera predstavljenog u radu (Du i Liang, 2018.). Algoritam je implementiran u programskom jeziku C++ te je dostupan javno na GitHub-u (Plušćec i Veisnger, 2018.) uz MIT licencu. Algoritam koristi sirova očitanja i kontige korištenjem postojećih alata za sastavljanje genoma te nastoji povezati nepovezane kontige korištenjem sirovih očitanja.

Program je testiran na dvije stvarne i jednoj sintetskoj sekvenci. Sintetska sekvencia je dobivena koristeći genom bakterije *Escherichia coli*. Algoritam je smanjio početni broj kontiga s tri na jednu sekvencu. Pri tome je i smanjio razliku u duljini sekvenci na jednu desetinu početne razlike. Kod prirodne sekvence *Bartonella grahamii* algoritam je smanjio početni broj sekvenci sa šest na četiri sekvence. Prirodnu sekvencu temeljenu na bakteriji *Campylobacter jejuni* algoritam je smanjio sa šest na jednu spojenu sekvencu.

U nastavku rada na projektu predlažemo sljedeća poboljšanja. Prvo poboljšanje je usmjereni na unaprjeđenje pretraživanja prilikom generiranja puta između dva kontiga. Ako prilikom pretrage dođemo do čvora koji nema djecu možemo se vratiti u prethodni čvor kako bi ponovno pokušali doći do drugog kontiga. Drugo poboljšanje se odnosi na prilagodbu usmjerenja sekvenci prilikom izgradnje konačne sekvence koristeći informaciju o usmjerenu iz PAF formata.

LITERATURA

- H. Du i C. Liang. Assembly of chromosome-scale contigs by efficiently resolving repetitive sequences with long reads, 2018.
- J. Krumsiek, R. Arnold, i T. Rattei. Gepard: A rapid and sensitive tool for creating dotplots on genome scale, 2007.
- H. Li. Minimap and minimap2: fast mapping and de novo assembly for noisy long sequences, 2017.
- H. Li. Minimap2: pairwise alignment for nucleotide sequences, 2018.
- D. Pluščec i D. Veisnger. Implementation of algorithm hera - highly efficient repeat assembly, 2018. URL github.com/domi385/bioinformatics.
- Mile Šikić i Mirjana Domazet-Lošo. *Bioinformatika*. 2013.