SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 735

Računalne metode za poboljšanje i validaciju sastavljenih genoma

Vanessa Županović

Zagreb, lipanj 2014.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem svojoj majci što me uvijek bodrila i bila uz mene do zadnjeg trenutka kao potpora, prijateljica i oslonac onda kada mi je to najviše trebalo. Hvala ti mama...

Zahvaljujem i doc. dr. sc. Mili Šikiću na iskazanom povjerenju, tehničkoj podršci te strpljenju.

SADRŽAJ

1. Uvod

Napredak u tehnologijama sekvenciranja genoma te jednostavniji pristup sekvenciranim podacima doveli su do ponovnog porasta interesa k načinima (algoritmima) za sastavljanje genoma. Unutar ovoga rada fokus je stavljen na zadnji korak sastavljanja - skafolding (eng. "scaffolding") gdje se već parcijalno rekonstruirane sekvence pokušavaju smjestiti u kontekst cijelog genoma. Preciznije, pokušat će se razviti heurističke metode za poboljšanje postojećih skafolda. Budući da je rad nastao u suradnji s GIS-om ($eng.$"$Genome\ Institue\ of\ Singapore$" skraćeno "$GIS$") kao ulaz, odabrani su skafoldi generirani skafolderom OPERA, a sami će se postupci temeljiti na vrlo sličnoj matematičkoj pozadini koju koristi i OPERA, tj. optimizaciji kriterija skladnih (harmoničkih) bridova. Za spomenuti je skafolder poznato da daje rješenje optimalno prema tom kriteriju, a unutar rada će se dokazati da dobiveno rješenje nije jedino moguće rješenje, tj. da postoji više međusobno ekvivalentnih optimalnih rješenja te da se veći dio strukturalnih pogrešaka u skafoldu može ukloniti upravo detektiranjem i odstranjivanjem nejednoznačnosti unutar pojedinih regija koje su najčešće uzrokovane veličinom inserata te činjenicom da ponekad uparena očitanja ne pružaju dovoljno informacija da točno smjestimo mali kontig unutar skafolda. Prije nego se detaljno obradi spomenuti problem bit će dat kratak osvrt na trenutno postojeće metode sekvenciranje te načine sastavljanja genoma, iako, kao što je već rečeno, najveći dio ovog rada bavit će se upravo skafoldingom, tj. heurističkim metodama za poboljšanje već postojećih skafolda. Pri samom kraju rada prikazat će se i svi važniji rezultati te iznjeti neke od ideja za potencijalna poboljšanja u budućnosti. Naravno, valjanost razvijenih metoda će biti ispitana cjevovodom GAGE te će se, također, voditi računa i o tome je li moguće popuniti velike procjepe koji nastaju kao posljedica uklanjanja nejednoznačnih regija.

2. Sekvenciranje genoma

Sekvenciranje predstavlja postupak kojim se unutar određene sekvence utvrđuje točan redoslijed pojedinih baza. Pojava sekvenciranja značajno je promijenila prirodu biomedicinskih istraživanja te medicine. Redukcija u cijeni, kompleksnosti te vremenu potrebnom za sekvenciranje velikih količina DNA kao i poboljšanja u mogućnosti sekvenciranja bakterijskih te eukariotskih genoma, u budućnosti će imati vrlo značajan kulturološki i ekonomski utjecaj. Projekti velikih razmjera vezani uz sekvenciranje kao što je primjerice Sekvenciranje nasumičnih probira (eng. "Whole-Genome Sequencing", skraćeno WGS), obično zahtijevaju kloniranje DNA fragmenata u bakterijske vektore, amplifikaciju i purifikaciju pojedinih predložaka te tzv. Sangerovo sekvenciranje Schebye-Alsing et al. (2009) (važno je napomenuti da postoje i alternativne tehnike sekvenciranja - Maxam i Gilbertova metoda, pirosekvenciranje te jednomolekularno sekvenciranje temeljeno na egzonukleazama França et al. (2002)).

2.1. Metode sekvenciranja genoma

Osnovnu procedura kod sekvenciranja jest izolacija genomske DNA ili RNA (reverzno transkribirana u kodirajuću DNA). Izolirana se DNA zatim klonira u vektore (kao što su plazmidi i BAC-ovi) koji imaju sposobnost stabilne propagacije u odgovarajućim stanicama domaćina (npr. bakterija $E. - coli$). Razvijeno je nekoliko različitih klonirajućih sustava unutar kojih veličina inserata varira od nekoliko stotina parova baza pa sve do reda veličine megabaza. Idealna biblioteka klonova za sekvenciranje ima sljedeće karakteristike: klonovi su krajnje redundantni tako da višestruko prikrivaju kompletan genom (tipično od 6 do 10 puta), klonovi genom prikrivaju u potpunosti nasumično ne dajući prednost niti jednoj regiji genoma posebno, klonovi su stabilni te ne dolazi do intreagiranja između istih i okoline u procesu propagacije. Nakon propagacije određeni klonovi se selektiraju te započinje proces sekvenciranja temeljen na četiri nezavisne reakcije DNA polimeraze. Esencijalna značajka sekvenciranja jest da se na svaku bazu stavljaju vrijednosti kvalitete očitanja koje su korespondentne s

vjerojatnošću da je promatrana baza točno određena. U procesu sastavljanja genoma spomenute vrijednosti će pomoći u razlikovanju pravog DNA polimorfizma od pogrešaka prilikom sekvenciranja, no činjenica je da većina trenutno postojećih alata za sastavljanje genoma ove vrijednosti u potpunosti ignorira.

Današnje DNA komercijalne platforme za sekvenciranje genoma uključuju *Genome Sequencer* (Roche 454 Life Sciences) , *Solexa Genome Analyzer* (Ilumina), *SOLiD System, Ion Torrent* (Applied Biosystems), *Pacbio* (Pacific Biosciences), *The Helioscope* (Helicos) te komercijalizirani *Polonator*. Ono što navedene platforme razlikuje od nekadašnjih jest to da se iste ne oslanjaju na tzv. Sangerovu kemiju što je karakteristika prve generacije uređaja korištenih u svrhu sekvenciranja (*Applied Biosystems Prism 3370*, *The Molecular Dynamics MegaBACE*). Druga generacija uređaja je karakterizirana visoko paraleliziranim operacijama, većim prinosom, te znatno nižom cijenom po očitanju. Novost je i ta da su očitanja kod ove generacije sekvenciranja vidno kraća u odnosu na prethodnu generaciju što dodatno komplicira i direktno utječe na odabir algoritma za sastavljanje genoma, naime, pojavljuju se alati za sastavljanje genoma čija se paradigma temelji na de Brujinovim grafovima (više u poglavljima što slijede). Današnje uređaje za sekvenciranje često se naziva i uređajima za sekvenciranje nove generacije (eng. "Next generation sequencers" – NGS) s duljinom očitanja u prosjeku od 100 ili čak i manje parova baza (Solexa i Solid uređaji) do 400 pb (Roche 454). Samo za usporedbu, duljina očitanja kod prve generacije uređaja za sekvenciranje varirala je od 500 do 1000 pb, dok se točnost samih očitanja gotovo i nije promijenila te iznosi oko 99%, no kraća očitanja donose dodatne probleme kod sastavljanja genoma budući da pružaju puno manje informacija poglavito ukoliko se radi o ponavljajućim regijama. Sastavljanje genoma iz kratkih očitanja zahtjeva visoku prekrivenost kako bi uopće bilo moguće detektirati preklapanja. Visoka prekrivenost samo povećava složenost cijelog postupka i inducira probleme vezane uz velike skupove podataka. Rezultati sekvenciranja (očitanja) pohranjuju se unutar FASTA, FASTQ, SFF (binarni format za kodiranje očitanja nastalih na 454 platformi), SAM/BAM, FASTG te VCF. Važno je napomenuti da se unutar FASTQ datoteka uz samu sekvencu pohranjuje i kvaliteta svake baze (QV). Iako je QV u osnovi korisna informacija te može pomoći prilikom otkrivanja potencijalnih SNP-ova većina trenutno postojećih alata za sastavljanje genoma ju ne koristi budući da povećava količinu zauzeća memorije prilikom izvršavanja programa te procesorske zahtjeve.

2.2. Osnovne strategije sekvenciranja genoma

Sekvenciranje genoma je disciplina koja je proteklih godina bila podlegnuta progresivnom razvoju. Uvođenje novih masivno paralelnih metoda sekvenciranja otvara čitav niz novih polja primjene. Eksperimentalna tehnika u većini projekata sekvenciranja nove generacije viših organizama, terminiranje DNA lanca, izumljena je prije tridesetak godina te još uvijek čini samu bazu postupka, dok su određene izmjene vidljive samo u povećanoj automatizaciji istog. Najranija strategija sekvenciranja genoma tzv. Sekvenciranje nasumičnim probirom (eng. "shotgun sequencing") se pojavljuje još davne 1981. te se tijekom godina što slijede njezina primjena proširuje na sve veće i veće DNA sekvence klonirane unutar plazmida, kozmida te umjetnih kromosoma (bakterije). HTS sekvenciranje (eng. "High Throughput Sequencing") kodirajuće DNA se javlja nešto kasnije te se paralelno s istim uvodi i eng. "Expressed Sequence Tag" – EST. Kolekcija EST-ova daje prvu dobru aproksimaciju raznolikosti kodirajućih regija te s vremenom postaje važan alat čija je namjena usko vezana uz analizu i otkrivanje gena. Količina podataka dobivena sekvenciranjem je zadivljujuća, npr. sekvenciranje ljudskog genoma rezultiralo je s 23 bilijuna baza u Međunarodnom konzorcijumu za sekvenciranje ljudskog genoma (eng. "International Human Genome Sequencing Consortium") te 27 bilijuna baza u tzv Celera projektu. Naravno, takva ogromna količina podataka ne može biti smjesta konkatenirana kako bi se dobila finalna sekvenca, već postoje različite strategije sastavljanja čija uspješnost zavisi o ulaznim podacima. Neke strategije bolje rukuju duljim, a neke kraćim očitanjima, tj. odabir odgovarajućeg algoritma za sastavljanje genoma direktno ovisi o načinu sekvenciranja. Jednako tako količina (broj) očitanja kao i organizam čiji se genom sekvencira, igraju, također, vrlo značajnu ulogu. Kratak osvrt na trenutno postojeće strategije za sastavljanje genoma bit će izložen kroz naredna poglavlja s posebnim naglaskom na zadnji korak sastavljanja - skafolding.

2.2.1. Sekvenciranje nasumičnim probirom (eng. "Shotgun sequencing")

Kod Sekvenciranja nasumičnim probirom tipično se uzima jedna individualna DNA sekvenca kao izvor. Za fragmente koji imaju manje od 98% sličnosti pretpostavlja se da dolaze iz različitih regija polaznog genoma. Samo sekvenciranje ugrubo razlikuje dva osnovna principa: Cjelovito sekvenciranje nasumičnim probirom (eng. "Whole-Genome Shotgun" - skraćeno WGS) te Hijerarhijsko (eng. "Hierarchical Shotgun

Sequencing" - skraćeno HSS).

Sekvenciranje cijelog genoma nasumičnim probirom - WGS

Kod ove strategije sekvenciranja čitav se polazni genom razbije na niz malih fragmenata koji se zatim kloniraju u vektore. Inserti se potom procesiraju kako bi se dobila očitanja. Dobivena očitanja nastojimo poravnati s ciljem rekonstruiranja polazne sekvence.

Hijerarhijsko sekvenciranje nasumičnim probirom - HGS

Ova strategija se često naziva i Map-based, BAC-based ili Clone-by-clone sekvenciranje. Osnovna razlika u odnosu na prethodno opisanu strategiju je ta što ovaj slučaj uključuje generiranje velikih klonova (inserata) koji prekrivaju cijeli genom (tipično 100 do 200 kb po insertu) nakon čega se na svaki pojedini insert razbije na niz malih fragmenata, tj. provodi se "shotgun" sekvenciranje nad insertima. Kod ove strategije je dobro što je pomoću određenih metoda moguće utvrditi redoslijed inserata što značajno olakšava sastavljanje polaznog genoma.

2.2.2. Kombinirano sekvenciranje (eng. "Mixed Strategy Sequencing", skraćeno MSS)

Često se rabi prilikom sekvenciranja vrlo velikih kompleksnih genoma. Koristi djelove iz Hijerarhijskog i WGS sekvenciranja pri čemu se Hijerarhijsko sekvenciranje ponaša kao okosnica za WGS sekvenciranje. Uspješno je primjenjeno na genom štakora.

2.2.3. Sekvenciranje reducirane reprezentacije (eng. "Reduced Representation Sequencing" skraćeno RRS)

Predstavlja varijantu WGS sekvenciranja s tom razlikom što se kod ove strategije ne sekvencira čitav genom, već je moguće odabrati koji dijelovi genoma će biti sekvencirani kako bi se izbjeglo sekvenciranje velikih regija koje nas u osnovi ne zanimaju.

2.2.4. Sekvenciranje temeljeno na izraženim tagovima

Izraženi tagovi - eng. "Expressed Sequence Tags" (ESTs) jesu sekvence koje reprezentiraju gene koji potječu iz specifičnih tkiva. U EST sekvenciranju vrši se jedno automatsko sekvenciranje s jednog ili oba kraja inserata kodirajuće DNA (cDNA).

Namjena EST sekvenciranja najčešće nije utvrđivanje kompletnih inserata kodirajuće DNA, već parcijalnih sekvenci transkribiranih gena.

3. Sastavljanje genoma

Jedino je računalno moguće efikasno sastaviti sekvencirane fragmente DNA. Naravno, rekonstrukcija polazne sekvence je vrlo zahtjevna te nikako nije trivijalan zadatak. Izuzetak čine slučajevi kada se sastavlja vrlo jednostavan genom (npr. mali genom virusa ili transkriptosoma) i duljina očitanja daleko premašuje duljinu ponavljajućih regija, dakle, težina zadatka je u korespondenciji s odnosom između duljine očitanja i ponavljajućih regija unutar sekvence koja se sastavlja. Ponavljajuće regije kod sisavaca čine 50% sekvence dok je taj postotak ukoliko se radi o bakterijama značajno manji i iznosi tek 5%. Važno je naglasiti da gotovo nikad rezultirajući sastavljeni genom neće biti kontinuiran. U većini slučajeva rezultantni genom će biti fragmentiran i veoma sklon pogreškama. Kao što se već može naslutiti, jedna od glavnih poteškoća kod računalnog sastavljanja genoma predstavlja upravo razvoj takvih algoritama koji su u mogućnosti detektirati pružanja repetitivne DNA bez da uzrokuju krive relokacije spomenutih dijelova DNA. Međutim, ponavljajuće regije nisu jedini problemi s kojim se susreću algoritmi za sastavljanje genoma. Greške prilikom određivanja pojedinih baza također predstavljaju ozbiljan problem, nadalje, dodatne komplikacije u cijeli proces sastavljanja genoma unosi i mogućnost postojanja SNP-ova eng. "Single nucleotide polymorphysm" (jednonukleotidnih polimorfizama), kimernih fragmenata te raznih drugih kontaminacija.

3.1. Principi sastavljanja genoma

Svi se alati za sastavljanje genoma u osnovi zasnivaju na tri paradigme:

1. pohlepna – bazira se na heurističkim metodama koje pohlepno ujedinjuju dva fragmenta s najvećim preklapanjem. Jasno je da će ova paradigma davati najslabije rezultate, stoga se danas gotovo i ne primjenjuje, ali je zgodna kao ilustracija.

2. Overlap Layout Consensus (skraćeno OLC) – temelji se na teoriji grafova, efi-

kasna pri korištenju u kombinaciji s dužim i poprilično netočnim očitanjima (>200 pb, Roche 454, Sanger te Pacbio platforma).

3. paradigma zasnovana na de Brujinovim grafovima – pogodna je za vrlo kratka i točna očitanja ($\lesssim 100pb$, sekvenciranje druge generacije).

3.2. Osnovni pojmovi – kontizi i skafoldi

Prije nego što se da detaljan pregled prethodne tri spomenute paradigme, valjda dati kratak osvrt na neke osnovne koncepte, tj. podatke kojima algoritmi za sastavljanje genoma "rukuju". Kada na ulaz alata za sastavljanje genoma stavimo datoteku s prethodno generiranim očitanjima na izlazu ćemo u najvećem broju slučajeva dobiti datoteku koja se sastoji od više djelomično sastavljenih nizova reda veličine od po nekoliko očitanja. Svaki takav dobiveni niz nazivamo kontig (iz eng. "contig"). Preciznije, kontig je kontinuirana DNA sekvenca sastavljena od niza manjih fragmenata (očitanja) koja se međusobno preklapaju. Kontizi se pohranjuju unutar najobičnijih fasta datoteka i to na način da se unutar zaglavlja koje počinje znakom ">" navodi identifikator kontiga -ID (jedini obavezan atribut, no iza njega po potrebi mogu doći i drugi atributi, primjerice duljina kontiga ili indikator o kojem se organizmu radi) te u novom redu i sam kontig u obliku stringa čija je abeceda ograničena na A,T,G,C. Poneki alati na temelju orijentacije redoslijeda i udaljenosti kontige slažu u još veće uređene strukture koje nazivamo skafoldi (iz eng. "scaffolds") , tj. skafold predstavlja točno poredani niz kontiga unutar kojeg je međusobna udaljenost te orijentacija istih poznata. Kao što je iznad i spomenuto važno je naglasiti da svi alati za sastavljanje genoma ne daju na svojem izlazu i skafolde, iako postojanje takvih datoteka dosta pomaže kod evaluacije kvalitete sastavljenog genoma , nadalje, postoji vrlo malo *stand-alone* skafoldera koji koriste informacije dobivene iz pre-asemblanih kontiga i parova očitanja. Skafoldi se pohranjuju unutar jednostavnih fasta datoteka koje vizualno odgovaraju datotekama s nizom kontiga s tom razlikom što su generirani skafoldi ipak duži, naravno, kontig može činiti zaseban skafold i on se tada naziva singleton. Na generiranje skafolda se često gleda kao na zaseban završni korak sastavljanja genoma u kojem se kontizi dobiveni poravnanjem niza kratkih očitanja moraju smjestiti unutar konteksta čitavog genoma. Skafolde se katkad naziva i super-kontizi, te je razvijeno više alata (algoritama) koji su specijalizirani isključivo za ovaj korak. Veći dio njih se temelji na grafovima i vjerojatnosnim metodama (jedan od pristupa temeljnih na grafovima bit će detaljno opisan u nastavku rada budući da je korišten za poboljšavanje izlaza ska-

foldera OPERA Gao et al. (2011)). Važno je naglasiti da skafoldi nikako nisu savršena reprezentacija polaznog genoma budući da između njih i dalje mogu postojati procijepi (eng. "gaps"). Sekvence unutar porcijepa najčešće nisu poznate, stoga je genom i nakon ovog koraka fragmentiran, što ostavlja prostora za dodatna poboljšanja. Na samom kraju procesa sastavljanja genoma potrebno je pokušati zatvoriti procijepe, tj. umetnuti u njih odgovarajuće kratke kontige za što je ponovo razvijen niz pristupa od kojih će jedan biti odabran te demonstriran u sklopu ovog rada.

3.3. Osnovne strategije za sastavljanje genoma

Prije nego se fokus prebaci na skafolding (eng. "scaffolding") potrebno je dati kratak osvrt na trenutno postojeće metode za sastavljanje genoma (parcijalno sastavljen genom ulazi u proces skafoldinga) budući da su skafolderi najčešće ugrađeni u same asemblere stoga se svi problemi vezani uz korak sastavljanja očitanja u kontige reflektiraju i na dobivene skafolde. Postoji tek nekoliko stand-alone skafoldera Bambus Pop et al. (2004), SOPRA Dayarian et al. (2010) i Opera Hunt et al. (2014) iako potpuno odvajanje skafoldinga od generiranja kontiga nedvojbeno pruža veću fleksibilnost poglavito kada se na ulazu kombiniraju podaci nastali različitim platformama za sekvenciranje. Alati koji barataju lokalnim sastavljenim genomom (popunjavanje procjepa – "in silico" završna dorada) su vrlo značajni za validaciju i poboljšanje sastavljenog genoma te su kao takvi često dio modernih tzv. eng. "assembly" cjevovoda. Unatoč opsežnoj matematičkoj analizi problema sastavljanja genoma, asembleri se danas nastavljaju oslanjati na heuristike i ostale *ad hoc* tehnike umjesto na rigorozne algoritme činidbenih jamstava. Razlog tomu je taj što je vrlo teško osmisliti realističan matematički model kojim bi se opisali ulazni podaci. Najveći broj dosadašnjih formulacija upućuje na to da bi dobivanje optimalnog rješenja (genoma) moglo koštati nepojmljivu količinu resursa , tj. s današnjom opremom jest neostvarivo, međutim najnovija istraživanja ukazuju na to da su optimalna rješenja ipak moguća u određenim koracima sastavljanja genoma kao što su skafolding i završna dorada genoma eng. "Genome finishing" – (zatvaranje procjepa). Na jednom takvom modelu zasniva se i egzaktan skafolder OPERA koja garantira pronalazak jednog od optimalnih rješenja (skafolda) međutim i dalje za pojedine genome generira dosta pogrešaka (no ipak, znatno manje od ostalih postojećih skafoldera poglavito na simuliranim testnim skupovima). Značajno poboljšanje izlaza spomenutog skafoldera uspjelo se postići optimizacijom određenog kriterija vezanog uz bridove unutar grafa i to na način da se unutar skafolda pronalaze dvosmislene regije te uklanjaju oni kontizi koji tvore jednu

takvu regiju (kontizi za čiju poziciju unutar skafolda nismo u potpunosti sigurni, stoga se isti mogu bez narušavanja ograničenja zamijeniti s nekim drugim kontigom unutar istog skafolda, tj. moguće im je zamijeniti pozicije).
Kao što je već ranije napomenuto postoje tri osnovne strategije za sekvenciranje genoma. Prethodno je, također, dat kratak osvrt na to kada se koja strategija u osnovi koristi, no važno je napomenuti da su s vremenom razvijeni i takvi alati za sastavljanje genoma temeljeni na de Brujinovim grafovima koji uspješno rukuju i duljim očitanjima. Ključan korak u sastavljanju kod tog slučaja je pretprocesiranje unutar kojeg se nastoje ispraviti pogreške nastale prilikom sekvenciranja, naravno sada već postoje i OLC asembleri koji se mogu nositi i sa kraćim očitanjima. Moderni alati za sastavljanje genoma moraju moći efikasno analizirati ogromne skupove podataka te biti relativno imuni na pogreške nastale prilikom sekvenciranja, također, moraju uspješno identificirati ponavljajuće regije unutar genoma. Važno je napomenuti da se alati za sastavljanje genoma koji se ugrubo zasnivaju na istoj paradigmi međusobno vrlo razlikuju. Primjerice ukoliko govorimo o OLC asemblerima, postoji čitav niz različitih metoda za pronalaženje podudaranja između fragmenata određene sekvence. Neke od njih se baziraju na BLAST-u a kod drugih nailazimo na u potpunosti različit princip. Nadalje, na temelju toga kako dolazimo do samih očitanja i neke druge informacije nam mogu biti dostupne kao što su parovi očitanja (pair reads, upareni parovi), podaci o BAC-ovima te npr. kvaliteta očitanja svake baze. U nastavku je dat kratak opis svake od tri osnovne paradigme na kojima se zasniva sastavljanje genoma.

3.3.1. Pohlepna paradigma

Radi se o jednostavnoj heurističkoj metodi koja uvijek spaja dva fragmenta (niza) ukoliko oni trenutno donose najveću dobit, tj. imaju trenutno najbolje moguće preklapanje, primjerice ukoliko imamo niz ATC njega možemo spojiti s TCG budući da se preklapaju u dvije baze T i C pa dobivamo ATCG. Valja napomenuti da se ovom metodom uvijek spajaju dva očitanja koja se najbolje preklapaju dokle god takvo spajanje ne uvodi kontradikciju u do sada sastavljeni genom. Dakle, odluke koje se ovdje donose su isključivo lokalno optimalne i globalno gledano mogu biti u potpunosti krive. Pohlepna paradigma je zapravo najraniji pokušaj rekonstrukcije fragmentiranog genoma i javlja se kod ranijih generacija alata za sastavljanje genoma kao što su TIGR asembler i PHRAP, te se danas zbog svojih dosta loših performansi gotovo i ne koristi. Samo neki od problema s kojima se susreće ova metoda jesu: nejedinstvenost uslijed kratkih očitanja, nejedinstvenost uslijed heterogenosti, nepotpuno preklapanje, pogreške

u sekvenciranju, nejasnost uslijed ponavljanja, trošak računanja itd. Neki od tih problema se mogu pokušati riješiti uvođenjem dužih i uparenih očitanja te sastavljanjem istih jedino ako postoje značajna preklapanja. Jednako tako zbog nejedinstvenosti uslijed heterogenosti (može se dogoditi da uzorak DNA sadrži dvije verzije sekvence koje se primjerice razlikuju u jednoj bazi) valja prilikom sastavljanja uzeti u obzir višestruke moguće načine sastavljanja očitanja, dok je problem nepotpunog preklapanja djelomično moguće zaobići generiranjem što većeg broja očitanja, što zahtjeva puno više početnog kapitala te računalnih i ostalih resursa. Naravno, proces sastavljanja genoma otežavaju i greške prilikom sekvenciranja. Smetnje istih je moguće umanjiti korištenjem parova očitanja (ako očitanje sadrži pogrešku puno je manja vjerojatnost da će biti točno sastavljeno) ili uporabom statistika za procjenu vjerojatnosti pogreške. Dakle ova paradigma predstavlja najmanje uspješan princip na kojem se potencijalno mogu zasnivati alati za sastavljanje genoma te se zbog svoje mogućnosti zaglavljivanja u lokalnom optimumu u modernim alatima gotovo niti ne koristi.

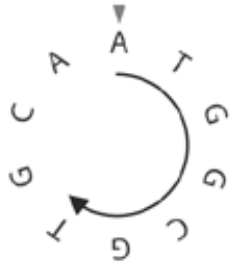
3.3.2. OLC princip (od eng. "Overlap Layout Consensus")

Ovaj se pristup temelji na teoriji grafova te ga je u osnovi (kao što je vidljivo i iz samog naziva) moguće podijeliti na tri osnovna koraka: *Overlap* – pronađi preklapanja među fragmentima poravnanjem istih, *Layout* – razmjesti očitanja na temelju poravnanja, *Consensus* – postizanje konsenzusa spajanjem svih sekvenci očitanja ujedinjavanjem preklapanja. U kontekstu cijelog genoma analiziraju te trilijuni preklapanja između očitanja. U tu svrhu većina OLC asemblera koristi algoritme za poravnanje koji su generalne modifikacije Needleman – Wunsch, Smith – Waterman te Gotoh algoritma Schebye-Alsing et al. (2009). Inicijalna detekcija preklapanja se najčešće provodi na način da se pronalaze identične podsekvence (k-meri) između očitanja prije nego što se uopće izvede poravnanje. Pronađene identične podsekvence se poslije koriste za identifikaciju potencijalnih preklapajućih sekvenci, koje se onda mogu međusobno poravnati s ciljem utvrđivanja predstavljaju li iste pravo preklapanje. Veličina k-mera (podsekvence) varira od metode do metode, a u nekim je slučajevima čak i dinamična, nadalje identični k-meri se također mogu grupirati na različite načine. Sam OLC princip bi se ukratko mogao opisati na način: kako se OLC asembleri temelje na grafovima, svaki čvor predstavlja očitanje. Postoji brid od čvora A do čvora B ako se sufiks od čvora A značajno preklapa s prefiksom od čvora B. Cilj je pronaći put koji posjećuje svaki čvor točno jednom (problem pronalaska Hamiltonovog puta – NP potpun problem, ne možemo riješiti u polinomnom vremenu). Ovakva struktura poda-

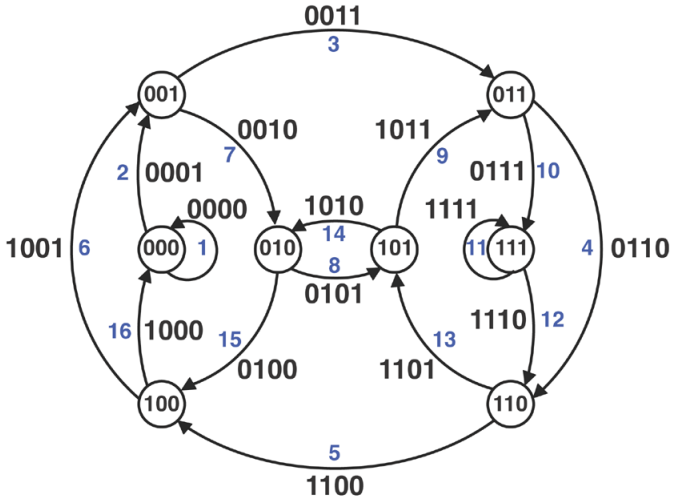
taka omogućava razvoj vrlo kompleksnih algoritama za sastavljanje genoma pomoću kojih je moguće uspostaviti globalan odnos između očitanja. Jedna od varijanti ovog pristupa – string graf pojednostavljuje globalni graf preklapanja na način da uklanja redundantne informacije – tzv. tranzitivne bridove. Nakon višestrukog poravnanja sekvenci generira se konsenzus i to na način da se najčešće uzima najučestaliju bazu na svakoj poziciji, naravno, postoje i drugi načini kao što je primjerice razbijanje višestrukog poravnanja u k-mere i analiziranje relativne frekvencije onih gdje je prisutnost alternativnih transkripata detektirana preko frekvencije k-mera (područja alternativnog konsenzusa).

3.3.3. Paradigma zasnovana na de Brujinovim grafovima

Začetak ove ideje može se pratiti 300 godina u prošlost kada je postavljen problem Konigsbergovih mostova (današnji Kaliningrad – Rusija) koji glasi: 7 mostova spaja 4 dijela grada koji su smješteni na međusobno suprotnim obalama rijeke, je li moguće prijeći preko svih mostova točno jednom i vratiti se u točku iz koje je obilaženje i počelo. Problem je riješio Euler 1735. na način da je svako kopneno područje predstavio vrhom (čvorom) a svaki most bridom. Na taj je način dobio graf pa se problem sveo na traženje staze (eng. "path") koja prolazi svakim bridom točno jedanput. Eulerovu ideju usvojio je danski matematičar Nicolaas de Brujin i primijenio na pronalaženje cikličke sekvence slova nad određenom abecedom za koju se svaka moguća riječ duljine k pojavljuje kao string uzastopnih znakova unutar cikličke sekvence točno jednom, ili jednostavnije rečeno, cilj ove paradigme (pristupa) jest pronaći najkraći cirkularni nadniz koji sadrži sve moguće podnizove duljine k (k-torke dane abecede). De Brujinovi grafovi su se pokazali izuzetno korisnima i na polju molekularne biologije unutar koje je potrebno sastaviti čitav genom od bilijuna kratkih očitanja. Važno je naglasiti da su alati temeljeni na ovoj paradigmi namijenjeni upravo sastavljanu genoma prvenstveno iz kratkih očitanja.



Slika 3.1: Mali cirkularni genom



Slika 3.2: Primjer de Brujinovog grafa nad abecedom 0,1 i k = 4. Ovaj graf zasigurno sadrži Eulerovu stazu budući da je broj bridova koji ulaze u čvor jednak broju bridova koji izlaze iz čvora (2). Ukoliko pratimo bridove numerirane plavom bojom (1 - 16) možemo iščitati Eulerovu stazu : 0000, 0001, 0011, 0110, 1100, 1001, 0010, 0101, 1011, 0111, 1111, 1110, 1101, 1010, 0100, 1000. Nadalje, ako obratimo pažnju na prvi znak unutar labela na svakom bridu, lako možemo iščitati ciklički nadniz: 0000110010111101.

Da bi ilustrirali kako se ovaj graf može koristiti u svrhu sastavljanja genoma promotrimo sljedeći jednostavan primjer sa tek nekoliko kratkih očitanja (CGTGCAA, ATGGCGT, CAATGGC, GGCGTGC i TGCAATG, **sliku 3.1**) dobivenih sekvenciranjem malog cirkularnog genoma ATGGCGTGCA (uzmimo za usporedbu da tehnologije za sekvenciranje nove generacije generiraju očitanja duljine cca 100 pb). Ukoliko bi se odlučili sastaviti genom iznad na klasičan OLC način svako bi očitanje predstavili čvorom a preklapanje između očitanja usmjerenim bridom koji povezuje ta dva čvora. U našem primjeru dva čvora koji predstavljaju očitanja bit će povezana usmjerenim bridom ukoliko se sami čvorovi preklapaju u minimalno 5 nukleotida. U tom bi se slučaju naš problem ponovnog sastavljanja genoma sveo na traženje Hamiltonovog ciklusa unutar jednog takvog grafa, no ukoliko se vodimo idejom o de Brujinovim grafovima, problem možemo značajno pojednostavniti i to tako da svako očitanje razbijemo na k-mere veličine npr. 3 baze te uvedemo čvorove veličine k – 1 (u našem slučaju dva) koji zapravo predstavljaju sve prefikse i sufikse prethodno generiranih k-mera. Postojanje brida između dva čvora upućuje na to da se određeni prefiks i sufiks nalaze unutar istog k-mera. Primjerice brid ATG (k-mer) povezuje čvorove AT i TG. Ovako formuliran problem omogućuje nam pronalaženje Eulerove staze umjesto Hamiltonovog puta što značajno olakšava polazni problem. Iako netom opisani princip uvodi određene novosti

te pri tom olakšava postupak sastavljanja genoma, prilikom istog se, također, javljaju brojne poteškoće kao što su: nejedinstvenost (istrošeno uže – eng. "frayed rope") - očitanja nužno ne pokrivaju čitavu polaznu sekvencu (nedostaju pojedina očitanja), nepotpuno pokrivanje – može rezultirati nepovezanim grafom, uzastopna ponavljanja – pojava ciklusa, greške na krajevima očitanja tzv. mamuze (redundantne "stršeće" grane) i greške u središtu očitanja – mjehuri (mjehure izaziva i polimorfizam u središtu očitanja, tj. ne mora nužno biti riječ o pogreški). Generalno gledano sve poteškoće iznad opisane posljedično doprinose složenosti grafa (povećava se broj bridova stoga takav graf postaje vrlo teško razriješiti). U kontekstu teorije grafova konačni korak u sastavljanju genoma svodi se na problem redukcije grafa. Većina optimalnih redukcija grafova potpada pod klasu NP potpunih problema za koje trenutno ne postoji efikasno jednoznačno rješenje, već se naprotiv koriste razne heuristike i aproksimacije s ciljem smanjenja redundancije, redukcije kompleksnosti, produživanja jednostavnih putova unutar grafa te simplifikacije grafa općenito.

3.4. Metode evaluacije i problemi prilikom sastavljanja genoma

Pogreške prilikom sastavljanja, dakle, postoje u nacrtima i u potpunosti sastavljenim genomima te je pri tom važno napomenuti da se većina genoma ostavlja upravo u formi nacrta ("skice") što negativno utječe na ukupan broj pogrešaka (broj pogrešaka je veći u nacrtima). Situacija je tim gora što ne postoje opće prihvaćene, univerzalne metode za validaciju sastavljenog genoma te se unutar istih učestalo rabe mjere koje se isključivo zasnivaju na veličini sastavljenih kontiga (odnosno skafolda) na način da se preferiraju veći kontizi (skafoldi) što zapravo ne govori ništa o samoj kvaliteti (točnosti) sastavljenog genoma budući da poneki asembleri forsiraju kontinuiranost (veće kontige) nauštrb same točnosti sastavljanja. Evaluacija točnosti sastavljenog genoma predstavlja ozbiljan problem budući da je doista vrlo teško ocijeniti točnost sastavljenog genoma poglavito ukoliko ne postoji referentni genom na kojeg bi se poravnali dobiveni kontizi (skafoldi). Današnje najučestalije tehnike za sekvenciranje se ugrubo mogu raščlaniti na tri zasebna koraka: prvo se DNA nasumično razbije na više manjih fragmenata, zatim se kraj svakog fragmenta sekvencira što rezultira generiranjem dva očitanja po svakom fragmentu te se naposljetku polazna sekvenca rekonstruira iz dobivenih očitanja. Nove strategije sekvenciranja koje su još u nastajanju također prate netom spomenuti model samo se strategije primijenjene po svakom koraku razlikuju.

Prva dva koraka su već sada visoko automatizirana, no zadnji korak je i dalje težak izazov za sve tehnologije sekvenciranja. Sastavljanje genoma bi bio trivijalan zadatak da je svako očitanje moguće smjestiti na točno jedno određeno mjesto, ali organizmi (pa čak i oni najjednostavniji) sadržavaju ponavljajuće regije unutar svojih genoma. Spomenute ponavljajuće regije "zbunjuju" alate za sastavljanje genoma budući da očitanja koja potječu iz različitih kopija ponavljajućih regija djeluju identično. Nadalje, za gotovo jednake ponavljajuće regije vrlo je teško razlikovati greške nastale prilikom sekvenciranja od polimorfizma, stoga se lako može dogoditi da asembler krivo pozicionira ponavljajuća očitanja što rezultira krivo sastavljenim genomom. Sparivanje očitanja dobivenih sa suprotnih krajeva istog fragmenta pomaže smanjenju dvosmislenosti smještanja istih okolo ponavljajućih regija. Generalno gledano, dobro sastavljeni genom mora zadovoljavati sljedeće uvjete: sekvence preklapajućih očitanja se moraju podudarati (iznimke su greške prilikom sekvenciranja i poliploidi organizmi), udaljenost između parova očitanja mora biti konzistentna s veličinom fragmenata nastalih nasumičnim razbijanjem DNA (iznimke su kimerni fragmenti, kimerna DNA - hibridna DNA molekula nastala ligiranjem DNA restrikcijskih fragmenata iz različitih izvora), parovi očitanja moraju biti orijentirani jedni prema drugima, tj. moraju dolaziti s različitih strana sekvencirane DNA, te naposljetku, razmještaj očitanja unutar sastavljenog genoma mora odgovarati Poissonovoj distribuciji.

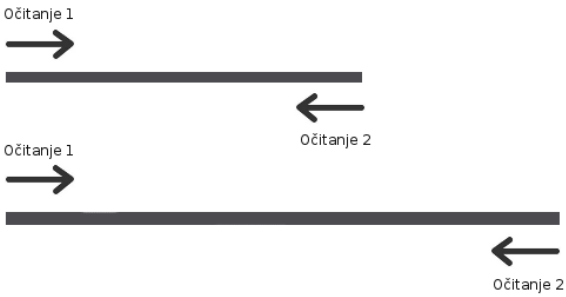
3.5. Zadnji korak u sastavljanju genoma – eng. "scaffolding" (skafolding)

Problem sastavljanja genoma se ugrubo može podijeliti na dva velika koraka – međusobno poravnavanje kontiga te smještanje kontiga u kontekst cijelog genoma, tj. spajanje kontiga u veće jedinice koje se nazivaju skafoldi. Skafolding - problem utvrđivanja slijeda, udaljenosti te orijentacije kontiga unutar genoma tipično koristeći informacije dobivene uparenim očitanjima jest krucijalan korak u sastavljanju visoko kvalitetnih nacrta istog. Iako su se metode sekvenciranja u skorije vrijeme značajno poboljšale postojeći skafolderi se većinom zasnivaju na heuristikama bez garancije o kvaliteti dobivenog rješenja (iznimku predstavlja skafolder OPERA), nadalje algoritmi za skafolding najčešće su ugrađeni u alate za sastavljanje genoma (tzv. asemblere) i ne mogu biti nezavisno kontrolirani. Općenito gledano, sam problem utvrđivanja redoslijeda i orijentacije kontiga jest NP potpun stoga se algoritmi koji se bave ovim korakom sastavljanja genoma zasnivaju na heuristikama, kao što je primjerice eng. "greedy" –

pohlepna metoda koja iterativno spaja skafolde povezane s najvećim brojem uparenih očitanja ili pohlepna metoda koja se bazira na ograničenjima udaljenosti i sl.. Dodatno uz uparena očitanja kao izvor informacija prilikom skafoldinga može se koristiti i informacija o sličnosti s referentnim genomom (ukoliko isti postoji) te restrikcijske mape. Spomenuti izvori informacija mogu dakle, doprinijeti parcijalnom utvrđivanju redoslijeda kontiga što dovodi do pojednostavljenja polaznog problema, no isti također uvode i dodatne komplikacije, naime sastavljanje genoma koje se oslanja na referentni genom potencijalno može baratati malicioznim sintetskim informacijama, dok druga spomenuta metoda često dovodi do pojave dvosmislenih regija o kojima će nešto više biti rečeno u nastavku. Valja naglasiti da se prilikom analize kvalitete sastavljenog genoma preferiraju duži skafoldi, no to nikako ne smije biti jedina relevantna mjera kvalitete budući da neki od algoritama žrtvuju točnost u korist kontinuiranosti.

3.5.1. Osvrt na postojeće metode

Veličine kontiga određene su količinom ponavljajućih regija, prekrivenošću i varijacijama u istoj te naravno duljinom samih očitanja i tehnologijom kojom su ista dobivena. Kontinuiranost sastavljenog genoma, kao što je već ranije rečeno, može se značajno poboljšati povezivanjem kontiga u skafolde.



Slika 3.3: U tekstu što slijedi vrlo će se često spominjati pojam *uparena očitanja*. Važno je naglasiti da će se pojam odnositi i na eng. "pair end reads" te eng. "mate pairs". Razlika između spomenutih vrsta očitanja je između ostalog i u dužini inserta (način dobivanja je u osnovi drukčiji - korak cirkularizacije za "mate pairs", što uvjetuje i različitu orijentaciju očitanja). Kod druge skupine očitanja radi se u duljim insertima. I jedan i drugi tip očitanja (kao što je prikazano na slici) dobivaju se sekvenciranjem s oba kraja (današnje tehnologije za sekvenciranje su u mogućnosti generirati očitanja duljine od 600 do 1200 pb, zbog čega u sredini često imamo "rupu" - insert). Unutar svih testnih skupova rabit ćemo kombinaciju obije vrste očitanja (radi veće pokrivenosti). Dulji inserti omogućavaju da očitanja "premošćuju" veće udaljenosti što nam pomaže kod rukovanja ponavljajućim regijama i služi kao svojevrsna "zaštita" od teških strukturalnih pogrešaka. Kraća očitanja pomažu popuniti "rupe" između dužih očitanja. Kombinacija jednih i drugih, dakle, posljedično uvjetuje bolji konsenzus.

U nastavku ovog poglavlja, sukladno popisu skafoldera, dat je kratak osvrt na postojeće metode koje koriste informacije iz uparenih očitanja dobivenih sa svakog kraja fragmenta DNA kako bi se "premostile" regije genoma koje nije moguće pročitati (sekvencirati) ili ih je vrlo teško sastaviti. U zavisnosti o načinu na kojeg su različite biblioteke za sekvenciranje (eng. *"sequencing libraries"*) dizajnirane uparena očitanja mogu "premošćivati" procjepe unutar sekvence reda veličine od nekoliko stotina do desetaka tisuća parova baza (vidi **sliku 3.3.**).
Informacije dobivene iz spomenutih očitanja koriste se kako bi se odredila točna orijentacija i redoslijed kontiga kao i za estimaciju duljine procjepa između kontiga (točnije, procijepi će nakon svakog određenog koraka algoritma biti re-estimirani maksimizacijom funkcije najveće izglednosti, vidi poglavlje *4.6.3. Utvrđivanje veličine procjepa*). Očekivano, problemi kod skafoldinga također, proizlaze iz ponavljajućih regija koje algoritmi na neki način moraju prepoznati i riješiti brojne linkove proizašle iz spomenutih ponavljanja. Kao što je već ranije napomenuto problem skafoldinga može biti formaliziran korištenjem teorije grafova, i to na način da svaki kontig predstavlja čvor unutar grafa, dok su očitanja koja povezuju (podržavaju) određena dva kontiga

odgovarajući brid. Trenutno postojeći skafolderi imaju različite pristupe za dobivanje približnih rješenja. Primjerice kao dodatak redoslijedu i orijentaciji, očekivana udaljenost između očitanja poznata iz načina na kojeg je konstruirana biblioteka, može biti korištena za estimaciju udaljenosti između kontiga (vidi poglavlje *3.5.2. Metoda zasnovana na grafovima (teorijski model na kojem se bazira skafolder OPERA)*), stoga će veličina procjepa unutar dobivenih skafolda biti približno dobre veličine. Naravno, idealni output skafoldera bi bio jedan skafold po kromosmu sa procjepima točne veličine koji razdvajaju kontige. Povećane mogućnosti sekvenciranja nove generacije (NGS) dovele su do intenzivnog razvoja algoritama koji su u mogućnosti efikasno sastaviti milijune kratkih očitanja, te naposljetku objediniti dobivene kontige u skafolde. Iako većina alata za sastavljanje genoma ima "build in" skafolding algoritam, standardna je procedura pokrenuti taj korak zasebno s ciljem dobivanja veće kontinuiranosti.

Tablica 3.1: Trenutno postojeći skafolderi i metode na kojima se isti zasnivaju. Podaci se odnose na 2013. godinu, stoga treba uzeti u obzir činjenicu da se polje verzija ne mora nužno odnositi na zadnju aktualnu verziju.

Skafolder	Verzija	Datum objave	Metoda	Maper za očitanja
ABySS	1.3.6	27/02/2009	Grafovi	proizvoljan
Bambus2	3.1.0	16/09/2011	Grafovi	proizvoljan
GRASS	0.003	06/04/2012	Grafovi	BWA/NovoAlign
MIP	0.5	13/10/2011	Grafovi	proizvoljan
Opera	1.2	19/09/2011	Grafovi	Bowtie/BWA
Scarpa	0.22	29/12/2012	Grafovi	proizvoljan
SGA	0.9.43	07/12/2011	Grafovi	proizvoljan
SOAPdenovo2	r223	27/12/2012	Grafovi	SOAP2
SOPRA	1.4.6	24/06/2010	Grafovi	proizvoljan
SSPACE	2(basic)	07/12/2010	Pohlepna	Bowtie/BWA

Prvi korak bilo kojeg algoritma za skafolding je utvrđivanje kamo smjestiti koje očitanje u kontekstu kontiga, tj. mapirati očitanja na kontige. Gotovo svi skafolderi za ovaj korak koriste već postojeće mapere 3.1. Ukoliko maper nije integriran u skafolder, korisnik će sam morati odvrtjeti mapiranje te dobivene rezultate dati na ulaz skafoldera u SAM ili BAM formatu Hunt et al. (2014) – u ovom slučaju korisnik je slobodan odabrati preferirani maper. Kada su dobivene koordinate svakog kontiga (pozicija) slijedi sam skafolding. U ovom koraku skafolderi se međusobno razlikuju, primjerice BAMBUS koristi pohlepnu paradigmu kako bi međusobno spojio kontige s

najvećim brojem linkova te pri tom ignorira korespondentne bridove koji su u konfliktu s željenim spajanjem Pop et al. (2004). SSPACE koristi sličan princip, no izgradnju prvog skafolda započinje tako što koristi najduži kontig te nastavlja spajati kontige toliko dugo dok je spajanje podržano od strane većine bridova Boetzer et al. (2011). Ostali skafolderi se ne oslanjaju na pohlepnu metodu, već umjesto nje izgrađuju skafold grafove (eng. "scaffold graph") i prema različitim principima lome grafove na manje regije te posljedično smanjuju problem budući da se obrada svede na određeni podgraf koji najčešće nema puno čvorova (jedna od metoda smanjena grafa koja se također, koristi unutar implementacije opisane u ovom radu jest lomljenje grafa po tzv. rubnim kontizima – eng. "border contigs"). Za razliku od SSPACE -a, BAMBUS2 prvo identificira ponavljajuće regije i uklanja ih sa skafold grafa, no zadržava kontige koji proizlaze iz varijacija, zatim utvrđuje redoslijed i orijentaciju istih te naposljetku provodi korak detekcije varijacija Koren et al. (2011). SOPRA iterativno uklanja nekonzistentne bridove u grafu te čvorove (kontige) koji povećavaju broj "podmetnutih" veza te pri tome koristi statističke optimizacije Dayarian et al. (2010) s ciljem povećanja praktičnosti same metode. Spomenuti koraci se ponavljaju sve dok broj kontiga koje je je moguće izbaciti iz skafolda ne padne na nulu, a svi bridovi i čvorovi postanu konzistentni. Nadalje, ABySS ograničava svoju pretragu za linkovima između kontiga koristeći gornje ograničenje na dozvoljenu distancu između bilo koja dva kontiga. SCARPA utvrđuje redoslijed i orijentaciju kontiga u dvije odvojene faze koristeći restrikcije koje problem čine traktabilnim i za razliku od ostalih skafoldera uključuje i korak verifikacije koji razbija kontige koji su prilikom sastavljanja krivo pozicionirani Donmez i Brudno (2013). SGA koristi vrlo konzervativan pristup, tj. u osnovi ne dozvoljava ikakve konflikte unutar grafa te pri tome izbjegava heuristike što rezultira malim brojem pronađenih valjanih bridova u svakom koraku. MIP particionira graf u podgrafove ograničene veličine te transformira svaki dobiveni podgraf u eng. "Mixed Integer Programming Problem" (MIP) Salmela et al. (2011). Svaki se MIP rješava zasebno bez nametnutih ograničenja vezanih uz orijentaciju kontiga. GRASS također koristi MIP, no kao i SOPRA do konačnog skafolda dolazi iterativno Gritsenko et al. (2012). Metoda i teorijski model na kojem se zasniva OPERA bit će opisani zasebno budući da je upravo izlaz OPERA-e ulaz programa koji je napravljen u sklopu ovog rada te mu je osnovna namjena poboljšavanje izlaza netom spomenutog skafoldera (tj. smanjenje broja teških strukturalnih pogrešaka – translokacija, inverzija, relokacija te indela).

3.5.2. Metoda zasnovana na grafovima (teorijski model na kojem se bazira skafolder OPERA)

Definicije

U tipičnom WGS-u, nasumično nasjeckani fragmenti DNA se sekvenciraju korištenjem jedne ili više tehnologija za sekvenciranje koje su danas na raspolaganju. Nakon sekvenciranja dobivena očitanja se poravnavaju *"in silico"* kako bi se dobile dulje kontinuirane sekvence – kontizi. Dodatno, očitanja su najčešće dobivena sekvenciranjem krajeva dugih fragmenata čija je približna dužina poznata iz jedne ili više biblioteka). Te se informacije koriste za utvrđivanje međusobne povezanosti, redoslijeda te orijentacije kontiga.

Promotrimo skup kontiga $C = \{c_1, c_2, .., c_n\}$, takav da je svakom $c_i \in C$ pridružena vrijednost orijentacije koja može biti +1 ili -1, skafold je tada definiran kao predznačena permutacija kontiga i procjepa koji se nalaze među susjednim kontizima. Ovdje valja uočiti da bi naivna pretraga za optimalnim skafoldom morala proći kroz sve moguće kombinacije procjepa i kontiga uzimajći pri tom u obzir i orijentaciju istih, stoga primjerice uzmimo graf G(V, E) gdje su V vrhovi i E bridovi (prisjetimo se da vrhovi predstavljaju kontige), tada mogućih skafolda imamo $2^{|V|}|V|!(|V| - broj\ kontiga)$, dakle sasvim je jasno da zbog velikog broja kontiga naivno rješenje nikako nije primjenjivo. Za bilo koja dva kontiga c_i i c_j povezana uparenim očitanjem njihova relativna orijentacija može biti predočena dvostruko usmjerenim bridom unutar grafa, dakle problem skafoldinga se unutar ovog modela može formalno definirati na sljedeći način:

Definicija 1. Za postojeći skafold graf G, pronađi takav skafold S koji maksimizira broj skladnih (harmoničnih) bridova u grafu.

Za formalnu definiciju skladnih bridova vidi poglavlje *4.2.1 Uvjet valjanih (skladnih) - harmoničnih bridova*. Definicija iznad je bitna budući da ćemo reskafolding vršiti upravo onda kada pronađemo takvu višeznačnu regiju koja ne narušava uvjet maksimalnosti, tj. izbacivat ćemo iz skafolda one kontige čija pozicija unutar skafolda nije jednoznačno određena već ih je zbog nedostatka informacija moguće pozicionirati na više mjesta i to tako da broj skladnih bridova unutar svih rasporeda bude maksimalan.

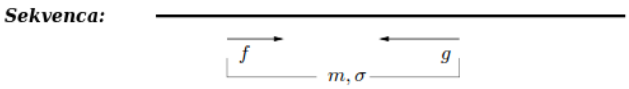
Rukovanje ponavljajućim regijama

Ponavljajuće regije unutar genoma se najčešće sastavljaju kao jedan kontig. U koraku skafoldinga informacije iz uparenih očitanja mogu pomoći da se takvi kontizi

smjeste na različite lokacije unutar skafolda, no u slučaju OPERA-e takve se regije filtriraju prije počinjanja skafoldinga, a kako alat razvijen u svrhu poboljšanja postojećih skafolda barata izlazima OPERA-e, njima se unutar implementacije nećemo posebno baviti jer pretpostavljamo da su inicijalno filtrirani.

Skafold graf

Prije no što se opiše sama struktura skafold grafa valja objasniti neke od osnovnih pojmova kako bi se lakše shvatio proces konstruiranja bridova istog. Naime, sva očitanja koja će se koristiti kao ulaz metoda za poboljšanje skafolda dobivena su na *Ilumina* platformi. Za svaku biblioteku unaprijed je poznat iznos približne udaljenosti te relativne orijentacije između neka dva sekvencirana fragmenta (vidi **sliku 3.4**). Vrijednosti na koje se najčešće može naići su - 10k, 20k, 150k itd. Mi ćemo kod procesiranja ponekad rabiti samo jednu, a najčešće više biblioteka, naravno maksimalna veličina inserata svakog testnog skupa bit će istaknuta pored njegovog imena npr. $"drosophila" - 20k(200bp)$ znači da maksimalna veličina inserata koje koristimo iznosi 20 kilobaza, a najmanji kontizi koje skafoldamo 200 pb [1]. Srednju vrijednost μ biblioteke onda dobijemo lako i to tako da uprosječimo sve međusobne udaljenosti fragmenata unutar iste biblioteke (npr. 10k biblioteke), jednom kada znamo srednju vrijednost, tada vrlo lako dobijemo i odgovarajuću standardnu devijaciju σ.



Slika 3.4: Dva fragmenta f i g koji tvore upareno očitanje sa poznatom srednjom vrijednošću udaljenosti m te standardnom devijacijom σ. Valja obratiti pažnju i na njihovu međusobnu orijentaciju unutar sekvence.

Za svaki skup kontiga, uparena očitanja mapiramo na elemente istog kako bi dobili bridove dvostruko usmjerenog multigrafa unutar kojeg (kao što je već ranije spomenuto) kontizi predstavljaju čvorove povezane bridovima koji su podržani uparenim očitanjima sa sličnom udaljenošću i odgovarajućom orijentacijom. Dakle, na neka dva čvora (kontiga) možemo mapirati više očitanja. Ono što želimo dobiti jest "objedinjeni" brid. Proces objedinjavanja (eng. "bundling") bridova činimo tako da uzmemo sva očitanja koja povezuju promatrani par kontiga i računamo odgovarajuće parametre

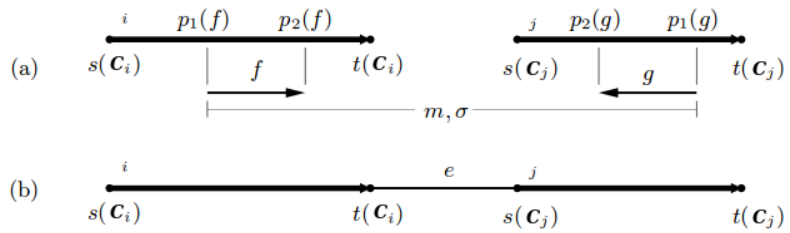
[1] prag na veličinu kontiga kojeg koristi OPERA iznosi 500 pb, sukladno tome ukoliko je najmanja duljina kontiga kojeg skafoldamo 500pb, to nećemo istaknuti uz ime, katkad iznos praga može biti i veći od 500 pb kao što je primjerice slučaj kod simuliranog ljudskog genoma i iznosi 700 pb.

$l(e)$ (duljina objedinjenog brida) te $\sigma(e)$ (standardna devijacija objedinjenog brida) na način:

$$\sigma(e) = \frac{\sigma}{\sqrt{n}} \tag{3.1}$$

$$l(e) = \frac{\sum_{i=1}^{n} l(e_i)}{n} \tag{3.2}$$

gdje su σ - standardna devijacija biblioteke te $l(e_i)$ duljina pojedinog tvorbenog brida Huson et al. (2001). Dobivene vrijednosti pohranjuju se unutar datoteke i to tako da jedan redak predstavlja jedan objedinjeni brid. Za bridove je poznat iznos udaljenosti između vrhova (distanca), veličina – broj očitanja koja podržavaju brid, standardna devijacija te orijentacija svakog pojedinog vrha (kontiga). Valjda napomenuti da su orijentacije "+ +" i "- -" te "- +" i "+ -" ekvivalentne budući da nije poznato s koje strane "promatramo" brid te one moraju odgovarati orijentacijama kontiga unutar skafolda. Nakon što je korak izgradnje bridova završen dobivene bridove je prema određenim kriterijima potrebno filtrirati. Nama će biti zanimljivi samo oni bridovi koji su podržani od strane 5 ili više očitanja (prag ne mora nužno biti 5 očitanja, no simulacijom kimernih očitanja je utvrđeno da je optimalna vrijednost ovog parametra upravo 5 Gao et al. (2011)) čija je orijentacija valjana (ne kosi se s orijentacijom kontiga unutar skafolda) te niti jedan od bridova ne smije premošćivati tzv. rubne kontige te posljedično skafolde (postoje bridovi koji to čine no za njih smo sigurni da nisu harmonični pa ih filtriramo). Filtracija bridova se radi ponajviše zato kako bi se smanjio broj nevaljalih bridova koji daju kriva ograničenja, no ista nam je potrebna i za smanjenje složenosti grafa, budući da manja složenost grafa povlači i bolje performanse reskafoldera (manji je prostor stanja kojeg trebamo pretražiti unutar svakog parcijalnog skafolda).



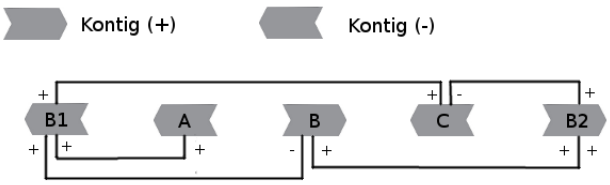
Slika 3.5: (a) Dva fragmenta f i g koji tvore upareno očitanje sa poznatom srednjom vrijednošću udaljenosti m te standardnom devijacijom σ koji "pogađaju" kontige C_i i C_j (b) što uvjetuje brid (e) između njih. U ovom slučaju duljina brida je jednaka: $l(e) = m - (l(B_i) - p_1(f)) + p_1(g))$

Problem skafoldinga kako smo ga definirali ranije izravno ne ocrtava strukturu

skafold grafa. U praksi je skafold graf ograničen činjenicom da biblioteke uparenih očitanja imaju gornju granicu τ te kontizi imaju minimalnu duljinu l_{min} što određuje limit na broj kontiga koji mogu biti razapeti očitanjem – w širina biblioteke gdje je $w \leq \frac{\tau}{l_{min}}$. Unutar implementacije fiksiramo parametar w te na taj način možemo vršiti reskafolding u polinomnom vremenu. Ovdje valja uočiti da mali kontizi vrlo negativno utječu na vrijeme izvođenja, što je jedan od razloga zašto OPERA postavlja prag na duljinu kontiga od 500pb. Polinomno vrijeme jest posljedica potrebe za re-estimacijom procjepa nakon svakog značajnog koraka algoritma koja se pod pretpostavkom da se procjepi ravnaju prema normalnoj razdiobi svodi na maksimizaciju funkcije najveće izglednosti za procjepe. Spomenuta se funkcija logaritmiranjem može svesti upravo na problem minimizacije kvadratne funkcije (ekvivalentan problem - želimo maksimizirati funkciju log izglednost, no implementacijski mnogo jednostavniji) o čemu će nešto više biti rečeno u nastavku rada.

Kontig graf - skafold

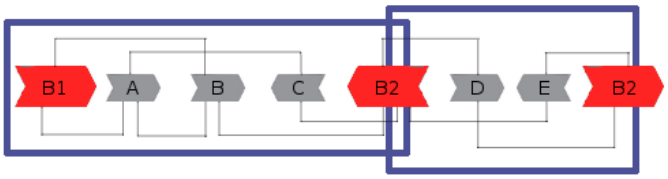
Uz skafold graf vrlo bitnu strukturu u memoriji predstavlja i sam skafold – jednostavan graf unutar kojeg svaki od čvorova ima stupanj dva izuzev prvog i zadnjeg čvora čiji je stupanj jedan. Prilikom reskafoldinga višestruko će se mijenjati upravo ova struktura i to pomoću informacija dobivenih iz skafold grafa. Potrebno je napomenuti da ove strukture nisu nezavisne, budući da su unutar obije čvorovi isti kontizi. Zbog svoje jednostavnosti ovu strukturu možemo reproducirati običnom listom.



Slika 3.6: Primjer parcijalnog skafolda te odgovarajućeg skafold grafa omeđenog rubnim kontizima (B_1 i B_2) svaki kontig unutar parcijalnog skafolda ima određenu orijentaciju (+/-) orijentacija kontiga unutar brida skafold grafa mora odgovarati orijentaciji kontiga unutar skafolda, u protivnom, brid nije valjan. Na slici je uvjet na orijentaciju bridova zadovoljen. Prilikom promjene redoslijeda unutarnjih kontiga moramo paziti da ne narušimo valjanost orijentacije, stoga nikada nećemo mijenjati redoslijed kontiga direktno povezanih bridom budući da smo sigurni da na taj način uvjetujemo porast broja neskladnih bridova.

Kontrakcija grafova (smanjenje prostora pretraživanja)

Kontizi sastavljeni od fragmenata nastalih WGS (vidi prvo poglavlje) sekvenciranjem dolaze u različitim veličinama. Najčešće dobar asembler generira nekoliko kontiga duljih od praga biblioteke τ. Za određenu veličinu biblioteke takve kontige označavamo kao rubne – eng. "border contig", tj. kontig je rubni ako i samo ako je njegova veličina veća ili jednaka od: $\mu + t * \sigma (t = 6)$ gdje su parametri μ i σ maksimalna srednja vrijednost duljina biblioteka te odgovarajuća standardna devijacija (prisjetimo se, prilikom skafoldinga rabimo očitanja iz jedne ili više biblioteka), tj. uzimamo onaj par μ i σ koji maksimizira prethodno napisanu formulu, dok vrijednosti spomenutih parametara odgovaraju vrijednostima dobivenima iz uparenih očitanja (tj. procijenjene su korištenjem određenog broja uparenih očitanja, očitanja su u ovisnosti o platformi smještena u zasebne datoteke). Reskafolding čitavog skafolda (grafa), dakle, možemo ograničiti na iteriranje i odgovarajuće izmjene po prostoru omeđenom s dva rubna kontiga, budući da smo sigurni da ne postoji niti jedan valjani (harmoničan) brid koji bi mogao premostiti rubni kontig (dokazano u Gao et al. (2011) **Lema 4**.), stoga sve izmjene utječu samo na relativno mali ograničeni dio grafa što značajno pojednostavljuje polazni problem i čini ga računalno manje zahtjevnim. Sličan modul u sebi sadrži i OPERA s tom razlikom što mi nećemo kontrahirati čvorove unutar parcijalnog skafolda u jedan super-čvor po završetku procesiranja istog budući da kreiranje super-čvorova obavlja drugi modul (više u nastavku) te mu je namjena u potpunosti drukčija nego u slučaju OPERA-e. Konačan optimalni skafold, dakle, nastaje unijom manjih ograničenih dijelova. Naravno, može se dogoditi da unutar skafolda ne postoji niti jedan rubni kontig (kao što je recimo slučaj kod realnog ljudskog genoma) tada reskafolding moramo vršiti na čitavom skafoldu što je računalno mnogo zahtjevnije te naravno, ovisi o samom genomu.



Slika 3.7: Prikaz dijela skafolda koji se pomoću rubnih kontiga može particionirati. Valja uočiti da jedan rubni kontig istovremeno može spadati u najviše dva parcijalna skafolda (označeno plavim okvirima). Budući da valjani bridovi ne prelaze rubne kontige jasno je da se problem reskafoldinga može rješavati iterativno, dok je konačno rješenje – skafold, jednako uniji svih parcijalnih rješenja. Naravno slučaj kod kojeg unutar skafolda ne postoje rubni kontizi jest sasvim legitiman te u tom slučaju kreiramo virtualni parcijalni skafold koji odgovara čitavom skafoldu te ga tretiramo na isti način kako bi tretirali običan parcijalni skafold budući da ne postoje valjani bridovi koji mogu premostiti dva skafolda, dakle smanjenje prostora pretraživanja rubnim kontizima nije nužno, no činimo ga ponajviše zato da smanjimo računalne zahtjeve potrebne za procesiranje velikih genoma.

4. Heurističke metode za poboljšanje postojećih skafolda

4.1. Princip rada metoda

Već je ranije spomenuto da OPERA garantira da će rezultirajući skafold biti optimalan prema kriteriju *skladnih bridova*, no ono što je važno napomenuti jest da ne postoji samo jedan optimalan skafold, već je katkad zbog nedostatka informacija dobivenih iz uparenih očitanja moguće konstruirati više optimalnih rješenja od kojih će jedno, naravno, biti i rezultat procesa skafoldinga. Sukladno rečenom, naše će se heuristike temeljiti na pretpostavci da veći dio grešaka prilikom sastavljanja genoma proizlazi upravo iz nejednoznačnosti koja se najčešće javlja kod pokušaja utvrđivanja pozicije malih kontiga unutar skafolda te velikih inserata. Dakle, osnova algoritama koji će se koristiti za poboljšanje postojećih skafolda bit će identifikacija višeznačnih regija te izdvajanje malih kontiga čija pozicija nije jednoznačno određena u zaseban skafold. Na ovaj način nećemo previše gubiti na kontinuiranosti, dapače, mjera N50 gotovo da se i neće promijeniti, iako, povećat ćemo broj singletona. U praksi se pokazalo da na ovaj način "uklanjamo" od 0.1% do 3% genoma što je vrlo malo, a drastično smanjuje broj pogrešaka (od 40% do čak 70% za pojedine genome) te istovremeno povećava mjeru ispravljeno N50.

Kako se spomenute metode temelje na grafovima, prirodno je kao ulaz bilo odabrati skafolde dobivene "$stand - alone$" skafolderom OPERA, nadalje performanse spomenutog skafoldera su se pokazale kao značajno bolje od većine preostalih skafoldera poglavito na simuliranim podacima, što povlači činjenicu da ukoliko je moguće poboljšati te rezultate, onda će se isti pokazati kao bolji od rezultata većine trenutno postojećih metoda. Važno je napomenuti da za sada ovo nisu univerzalne metode već su prvenstveno zamišljene kao ekstenzija iznad spomenutom skafolderu kako se zasnivaju na jednakom teorijskom modelu, stoga bi jedan od budućih izazova mogao biti

upravo provjera mogu li obrađene metode poboljšati izlaz bilo kojeg skafoldera, no to izlazi iz okvira ovog rada te u poglavljima što slijede neće biti razmatrano. Sve su metode implementirane u programskom jeziku C++ i zamišljene kao dio zasebnog alata koji učitava postojeći skafold te skafold graf iz datoteka u memoriju nakon čega prema određenim kriterijima optimira postojeći skafold (vidi kriterij skladnih bridova), a rezultat optimizacije pohranjuje u obliku .scaf datoteke te pripadne .fasta datoteke koja će biti korištena pri validaciji s GAGE cjevovodom Salzberg et al. (2012).

4.2. Osvrt na odabrana rješenja problema

Sa ciljem smanjenja pogrešaka unutar rezultirajućih skafolda osmišljeno je više međusobno sličnih heurističkih metoda koje se vode činjenicom o postajanju više međusobno ekvivalentnih optimalnih rješenja – skafolda. Kao što je već ranije rečeno, cilj svake od metoda jest ukloniti nejednoznačnosti iz pojedinih regija skafolda (što iste i čine), no ono što ih međusobno razlikuje jest upravo odabir heuristike za izbacivanje kontiga koji uzrokuju nejednoznačnosti. Prva, naivna i najjednostavnija metoda izbacuje sve one kontige koji uzrokuju nejednoznačnosti te pri tom ne vodi računa o tome što će se dogoditi s povezanošću skafolda – tj. možemo dobiti nepovezan graf što onemogućava reastimaciju procjepa posredstvom Goldfarb i Idnanijeve metode (Goldfarb i Idnani (1983)) te neke od postupaka evaluacije. Druga metoda se zasniva na višestrukim prolascima kroz parcijalni skafold (ili skafold ukoliko nema rubnih kontiga) sve dok se zamjenama mjesta dva susjedna kontiga može utvrditi nejednoznačnost. Ukoliko se utvrdi nejednoznačnost, uklanja se samo jedan manji kontig po iteraciji. Posljednja metoda, također, višestruko prolazi kroz skafold te uklanja komponente povezanosti grafa dokle god postoje nejednoznačnosti. Kod ove metode u jednoj iteraciji moguće je ukloniti više kontiga (pod iteracijom se podrazumijeva zamjena dva susjedna čvora – kontiga i sve popratne operacije), te je ista "agresivnija" od prethodne dvije, budući da njenom primjenom uklanjamo značajno više kontiga po iteraciji što ne mora uvijek nužno biti dobro. U nastavku je dat iscrpan pregled svake od metoda kao i osvrt na neka dodatna moguća poboljšanja.

4.2.1. Uvjet valjanih (skladnih) – harmoničnih bridova

Prije no što se svaka od metoda zasebno detaljno obradi valja skrenuti pažnju i na kriterij koji u svakom koraku algoritma mora biti zadovoljen (koristimo isti kriterij kojeg optimira i OPERA), ukoliko to nije slučaj, zamjenu dva čvora nećemo prihvatiti i

morat ćemo algoritam vratiti korak u nazad. Dakle, kao što je ranije napomenuto, ideja o poboljšanju postojećih skafolda temelji se na detekciji i uklanjanju višeznačnosti iz pojedinih regija. Višeznačnu regiju formalno definiramo na sljedeći način:

Definicija 2. Promatrani skafold sadrži višeznačnu regiju ukoliko promjena redoslijeda neka dva kontiga unutar regije (primjerice parcijalnog skafolda omeđenog rubnim kontizima) ne narušava uvjet valjanih (skladnih) bridova skafold grafa koji glasi:

$$\sum_{i=1}^{k} Ci < \mu_e + t * \sigma_e(t = 6) \tag{4.1}$$

dakle, nakon svake promjene redoslijeda neka dva kontiga moramo ispitati ako je došlo do kršenja iznad napisanog pravila što je računalno zahtjevno poglavito ukoliko imamo "velik" parcijalni skafold i mnogo bridova koji povezuju kontige unutar spomenutog parcijalnog skafolda, stoga to moramo raditi u linearnom vremenu te ispitivati samo one parove udaljenosti za koje smo sigurni da bi promjena redoslijeda mogla uvjetovati kolaps (za opis parametara vidjeti poglavlje *3.5.2. Metoda zasnovana na grafovima (teorijski model na kojem se bazira skafolder OPERA)*). Temeljna razlika između heurističkih metoda i OPERA-e je u tome što OPERA nastoji pronaći skafold takav da broj harmoničnih bridova bude maksimalan, dok heurističke metode nastoje prema istom kriteriju detektirati sva optimalna rješenja (rješenja unutar kojih je broj skladnih bridova maksimalan) te iz njih ukloniti nejednoznačne regije s ciljem dobivanja samo jednog optimalnog rješenja - skafolda s nadom da će tako dobiveni skafold biti bolji od polaznog.

4.2.2. Formati ulaznih i izlaznih datoteka

Prije nego što se detaljno opiše načelo rada te strukture podataka koje koristi ova metoda potrebno se osvrnuti na formate ulaznih i izlaznih datoteka. U osnovi unutar implementacije će se rukovati s dvije velike strukture podataka koje su inicijalno pohranjenje u zasebnim datotekama. Skafold graf pohranjen je u datotekama s bridovima oblika:

$633471 \quad - \quad 633473 \quad - \quad -561728 \quad 997 \quad 1$
$633469 \quad - \quad 633473 \quad - \quad -557546 \quad 997 \quad 1$
$633469 \quad - \quad 633471 \quad + \quad -465455 \quad 997 \quad 1$
$633469 \quad - \quad 633473 \quad + \quad -537637 \quad 997 \quad 1$

Svaki brid zapisan je u zasebnom retku. Bridovi imaju definirane vrhove (prvi i drugi kontig), orijentaciju (+/-), udaljenost (procijenjen broj baza između dva čvora), stan-

dardnu devijaciju te veličinu (broj uparenih očitanja koja podržavaju brid). U primjeru iznad slučajno je ispalo da je svaki brid podržan samo jednim očitanjem, takve bridove filtriramo. Metoda na ulaz mora primiti sve odgovarajuće datoteke s bridovima (minimalno dvije) neovisno o tome jesu li bridovi unutar datoteka valjani, stoga je uz implementaciju metode priložena i Python skripta na čiji ulaz treba predati putanje do svih odgovarajućih datoteka s bridovima na temelju kojih će ista generirati jednu datoteku sa svim bridovima koja se zatim predaje na ulaz programu. Uz skafold graf u memoriju je potrebno smjestiti i sam skafold koji je inicijalno pohranjen u .scaf datoteci sljedećeg formata:

$> opera\_scaffold\_1\ length:\ 8565007\ cov:\ 42.0$

$631041\ \ BE\ \ 28920\ \ -12$

$632985\ \ EB\ \ 93574\ \ -41$

$621797\ \ EB\ \ 578\ \ 64$

$623913\ \ BE\ \ 986\ \ -41$

$621799\ \ EB\ \ 578\ \ -39$

$632753\ \ BE\ \ 76590\ \ 60$

$630723\ \ EB\ \ 24595\ \ -40$

$627129\ \ BE\ \ 3817\ \ -41$

$624015\ \ EB\ \ 1013\ \ -41$

Redoslijed kontiga unutar ove datoteke odgovara redoslijedu kontiga unutar genoma uz pretpostavku da je skafold generiran bez pogrešaka (što u praksi nikad nije tako, vidjeti poglavlje *4.6.4. Rezultati i diskusija*). Svakom kontigu pridružena je orijentacija gdje BE označava pozitivnu (+) orijentaciju (skraćeno od eng. "begin") dok EB označava negativnu. Uz orijentaciju poznata je i vrijednost veličine procjepa, negativne vrijednosti u ovom slučaju samo govore da se dva kontiga preklapaju. Naš reskafolder na izlazu generira i ovakvu datoteku koja je zapravo modifikacija ulaza s različitim vrijednostima procjepa budući da svaki puta kada promijenimo redoslijed kontiga i uklonimo kontig moramo re-estimirati procjepe. Važno je napomenuti da prva metoda koja će biti opisana u nastavku rada ne omogućava točnu re-estimaciju procjepa, budući da je za istu potrebno očuvati informacije o povezanosti grafa, stoga se veličine procjepa u tom slučaju re-estimiraju vrlo jednostavnom heuristikom. Uz .scaf datoteku, program generira i uobičajenu fasta datoteku sa skafoldima koja se prvenstveno koristi prilikom validacije GAGE cjevovodom koji na temelju iste i referentnog genoma procjenjuje ukupan broj pogrešaka (inverzija, translokacija, indela, relokacija) te računa neke od osnovnih mjera kontinuiranosti kao što su N50, ispravljeno N50, broj procjepa > 1000, srednja pogreška procjepa itd. Osim spomenutih datoteka, program na izlazu generira

i datoteku sa statistikama za svaki pojedini skafold s ciljem lakšeg praćenja promjena koje su se dogodile na skafoldu tijekom rada programa. Ispod je dat primjer jedne takve datoteke koja je nastala reskafoldingom simuliranog ljudskog genoma.

$Old\ scaff.\ ID|Num.\ of\ removed|Remained\ length|Initial\ length|Length\ frac.$

1 111 53208803 53284166 0.998586

2 102 41483916 41557907 0.99822

3 103 39198686 39271268 0.998152

4 106 38474853 38551474 0.998013

5 86 35489196 35549057 0.998316

Također, program bilježi i parove onih kontiga koje je moguće zamijeniti bez narušavanja uvjeta o skladnim bridovima. Iz parova je moguće iščitati i koji su kontizi "izbačeni" iz skafolda budući da uvijek mičemo kraći kontig kako ne bi puno gubili na duljini rezultirajućih skafolda.

$Scaffold\ ID|First\ contig|Length|Gap\ size|Second\ contig|Length|Gap\ size$

1 621797 578 64 623913 986 -41

1 623913 986 601 621799 578 -39

1 626373 2639 -41 623451 841 4516

4.3. Naivno poboljšanje metode zasnovane na grafovima bez očuvanja informacije o povezanosti grafa

4.3.1. Osnovno načelo rada

Osnovno načelo rada ove metode je vrlo jednostavno, ista jednom prolazi kroz svaki parcijalni skafold (ili skafold ukoliko ne postoje rubni kontizi) te pri tom mijenja redoslijed svaka dva susjedna kontiga vodeći pri tom računa o ograničenju postavljenom na bridove skafold grafa (*Kriterij skladnih-valjanih bridova*). Ukoliko je novi dobiveni skafold prema kriteriju o valjanim bridovima ekvivalentan polaznome, metoda uklanja kraći kontig iz skafolda u zaseban skafold te na taj način smanjuje broj višeznačnih regija. Naravno, ovakva metoda ima i svoje nedostatke budući da kao što je ranije spomenuto uklanjanjem kontiga možemo dobiti nepovezan graf što onemogućava točnu re-estimaciju veličine procjepa, no u nastavku je predloženo i rješenje koje "pazi" na povezanost grafa provođenjem BFS-a između čvorova (unutarnjih kontiga) prije svakog potencijalnog uklanjanja te do istog dolazi onda i samo onda ako smo sigurni da

uklanjanjem promatranog kontiga nećemo dobiti nepovezan graf.

4.3.2. Algoritam

U svrhu boljeg razumijevanja metode priložen je krajnje pojednostavljen algoritam u obliku pseudokoda. Važno je napomenuti da će isti pseudokod uz manje modifikacije biti baza i za preostale metode za poboljšanje koje slijede u nastavku rada.

Algoritam 1: Pojednostavljeni algoritam naivne metode.

Input : Objedinjena klaster datoteka, datoteka sa standardnom devijacijom i srednjom vrijednošću biblioteke - lib.txt, datoteka s kontizima

Output: Obrađena skafold datoteka, datoteke s nužnim statistikama

Data: skafold graf - (dobiven iz klaster datoteke), kontig graf - skafold (dobiven iz skafold datoteke)

Filtriraj neželjene bridove;

Generiraj skafold graf;

Generiraj skafold;

if *(rubni kontizi postoje)* **then**
    particijoniraj skafolde po rubnim kontizima;
else
    Za svaki skafold kreiraj virtualni parcijalni skafold, postavi rubne kontige na NULL

while *(ne obiđeš sve parcijalne skafolde)* **do**
    kreiraj matricu bridova za trenutni parcijalni skafold;
    kreiraj matricu distance za trenutni parcijalni skafold;
    for $i \leftarrow 0$ **to** $duljina(polje\_unutarnjih\_kontiga)$ **do**
        zamjeni ($kontig(i)$, $kontig(i+1)$);
        osvježi podatke u matrici distance;
        provjeri uvjet za valjane(harmonične) bridove;
    if *(uvjet zadovoljen)* **then**
        prihvati zamjenu;
        ukloni kraći kontig iz skafolda $\rightarrow$ detektirana je nejednoznačna regija;
    else
        vrati matricu distance u početno stanje;
        $(i++)$;

4.3.3. Strukture podataka i vremenska složenost

Kako bi se pseudokod iznad mogao razumjeti potrebno je opisati strukture podataka kojima isti rukuje. Osnovnu gradivnu jedinicu za sve strukture predstavlja objekt kon-

tig unutar kojeg su pohranjene sve informacije dobivene iz .scaf datoteka te sve metode kojima se omogućuje povezivanje kontiga s preostalim strukturama podataka. Jedino je kontig fizički "stvarna" struktura, sve ostale su samo spremnici za pokazivače na objekte tipa Kontig. Implementacija je realizirana tako da bi se promjene unutar jedne strukture automatski reflektirale i na sve ostale, te kako bi se spriječilo nepotrebno kopiranje. Dvije, također, važne strukture jesu skafold i skafold graf (definirani ranije), skafold graf je fiksna struktura koja se nakon inicijalnih filtracija ne mijenja već služi za pohranu i dohvaćanje svih bitnih informacija vezanih uz bridove te je ostvaren kao neuređena mapa (eng. "unordered map") kako bi se omogućio što brži dohvat elemenata. Vrijeme dohvaćanja je u prosjeku konstantno, no u najgorem slučaju može biti i linearno u zavisnosti o veličini spremnika. Skafold je struktura koja se neprestano mijenja te u sebi između ostalog sadrži vektor pokazivača na kontige te pripadne parcijalne skafolde ukoliko oni postoje. Na temelju podataka iz spomenute dvije strukture konstruiramo sljedeće matrice:

- matrica bridova
- matrica distanci

Koje će nam poslužiti za detekcija narušavanja uvjeta skladnih (harmoničnih) bridova, strukturu matrica najjednostavnije je objasniti kroz sljedeći primjer:

Primjer 1. Neka je zadan parcijani skafold:

...

$630723 \quad EB \quad 24595 \quad -40$

$627129 \quad BE \quad 3817 \quad -41$

$624015 \quad EB \quad 1013 \quad -41$

$631611 \quad EB \quad 38170 \quad -41$

...

Za konstrukciju matrice distanci važna nam je udaljenost između svaka dva kontiga, stoga radi preglednosti izdvajamo u zasebnu tablicu identifikator te odgovarajuću duljinu kontiga. Budući da su srednja vrijednost te standardna devijacija biblioteke jednaki redom 9982 te 997, kontizi 630723 te 631611 zbog svoje veličine predstavljaju rubne kontige. Prisjetimo se, kontig je rubni ako i samo ako ima duljinu l_c za koju vrijedi $l_c \geq \mu + 6 * \sigma$, što u našem slučaju iznosi 15964.

Na temlju informacija iz tabličnog prikaza neposredno iznad konstruiramo matricu distanci prikazanu slikom 4.1.

	630723	627129	624015	631611
630723	0	0	3817	4830
627129	0	0	0	1013
624015	0	0	0	0
631611	0	0	0	0

Slika 4.1: Matrica distanci. Odabrani smjer promatranja skafolda jest s lijeva na desno. Udaljenost(distanca) između kontiga i i j jednaka je zbroju dužina l_k svih kontiga koji se nalaze između kontiga i i j.

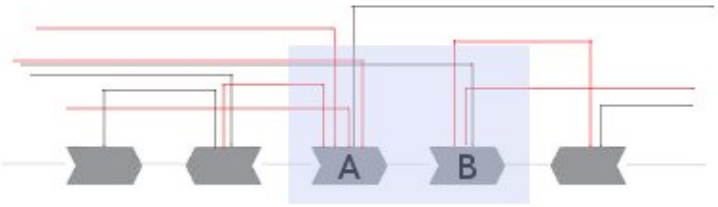
Nadalje, iz datoteke s bridovima dobivamo listu bridova karakterističnih za promatrani parcijalni skafold:

624015	+	630723	+	3616	229	19
627129	-	630723	+	-37	122	67
630723	-	631611	-	2296	449	92
624015	-	631611	-	-150	208	23

	630723	627129	624015	631611
630723	-10000	695	4990	1734
627129	695	-10000	-10000	1013
624015	4990	-10000	-10000	0
631611	-10000	1734	1098	-10000

Slika 4.2: Matrica bridova. Valja obratiti pažnju da iznosi unutar matrice odgovaraju rezultatu izraza $\mu_e + t\sigma_e$ gdje su μ_e i σ_e srednja vrijednost i standardna devijacija odgovarajućeg brida.

Matricu bridova za razliku od matrice distance generiramo samo jednom za svaki parcijalni skafold (ili skafold) te se ona ne mijenja (fiksna je struktura). Matrica distance generira se jednom po ekspanziji parcijalnog skafolda. Pod ekspanzijom se podrazumijeva kreiranju sve moguće djece parcijalnog skafolda zamjenama uzastopnih kontiga (jedna za roditelja). Svaka zamjena kontiga reflektira se na vrijednosti zabilježene unutar iste, no promjene vrijednosti vrše se u linearnom vremenu, budući da svaka zamjena mjesta dva kontiga ne utječe na sve bridove unutar parcijalnog skafolda već na točno određen podskup kojeg je moguće obići u linearnom vremenu.



Slika 4.3: Prikaz dijela parcijalnog skafolda. Želimo zamijeniti uzastopne kontige A i B na pozicijama i te $i+1$. Prilikom promjene moramo ispitati uvjet harmoničnosti samo za podskup bridova označen crvenom bojom. Naime, ukoliko promotrimo uvjet lako je uočiti da do narušavanja istog neće doći ukoliko se međusobna udaljenost neka dva promatrana kontiga smanjuje, što je u našem konkretnom slučaju udaljenost kotiga A od C_k za svaki $k > i + 1$, budući da se iz udaljenosti "gubi" veličina kontiga B, također, isto vrijedi i za udaljenost kontiga B od C_k, za svaki $k < i$. Dakle, nužno je i dovoljno prilikom svake zamjene kontiga provjeriti podskup bridova koji nije sadržan u iznad spomenutim podskupovima. Važno je napomenuti da zamjenu dva uzastopna kontiga nikako ne smijemo vršiti ukoliko su isti povezani bridom iz skafold grafa budući da bi na taj način povećali broj neskladnih bridova u grafu.

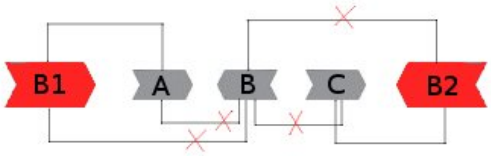
Primjerice za odabrani parcijalni skafold generiramo prvo dijete zamjenom unutarnjih kontiga na pozicijama 0 i 1 te dobivamo parcijalni skafold:
630723, 624015, 627129, 631611
Čija je odgovarajuća matrica distanci sada prikazana slikom 4.4.

	630723	**627129**	**624015**	**631611**
630723	0	**1013**	0	4830
627129	0	0	0	0
624015	0	0	0	**3817**
631611	0	0	0	0

Slika 4.4: Matrica distanci nakon zamjene unutarnjih kontiga na pozicijama 0 i 1 (627129 te 624015). Lako je uočiti da potencijalno narušavanje uvjeta harmoničnih bridova mogu izazvati samo vrijednosti u zasivljenim poljima budući da je iznos unutar tih polja porastao, stoga je dovoljno nakon zamjene provjeriti uvjet samo za te bridove (u našem slučaju dva) što se može učiniti u linearnom vremenu (podskup bridova za koje iznos lijeve strane izraza (4.1) poraste, trivijalno je poznat u svakom trenutku). U našem konkretnom slučaju, usporedbom odgovarajućih polja unutar matrice bridova te matrice distance, dolazimo do zaključka da zamjena neće biti prihvaćena jer ne zadovoljava uvjet skladnih bridova, zbog čega moramo poništiti zamjenu te povući promjene unutar matrice distance i prijeći na sljedeći par uzastopnih kontiga.

4.3.4. Nedostaci

Kao što je ranije napomenuto glavni nedostatak ove metode jest upravo činjenica da se prilikom uklanjanja kontiga nigdje ne vodi računa o povezanosti skafold grafa. Skafold graf te skafold su u osnovi međusobno zavisne strukture te sadrže identične čvorove, tj. ne postoji nikakva fizička odvojenost između te dvije strukture već je ona virtualna i temeljena na različitoj prirodi bridova koje promatramo u određenom trenutku (bridove nastale u procesu skafoldinga – jednostavan graf u kojem je stupanj svakog čvora najviše 2 ili bridove nastale mapiranjem uparenih očitanja na kontige – bridovi skafold grafa). Razlika je stvorena prvenstveno kako bi se nekako logički razdvojilo strukturu po kojoj se iterira od strukture koja se mijenja, stoga brisanje čvorova iz skafolda uvjetuje i brisanje čvorova iz skafold grafa. Brisanjem određenih čvorova gubimo informacije o povezanosti, stoga postaje nemoguće dobro re-estimirati procjepe. Za re-estimaciju veličine procjepa u ovom se slučaju vodimo vrlo jednostavnom heuristikom, tj. pretpostavljamo da je veličina novog procjepa jednaka upravo veličini procjepa između kontiga kojeg brišemo te prvog prethodnog uvećana za duljinu kontiga kojeg brišemo te desnog procjepa kontiga koji se briše. Ovakav način reastimacije procjepa dovodi do porasta broja vrlo velikih procjepa (većih od 1000 pb) te negativno utječe na srednju pogrešku veličine procjepa, također, valja napomenuti da ova metoda kao i sve metode što slijede u nastavku nejednoznačnosti unutar skafolda rješava isključivo brisanjem kontiga iz originalnog skafolda te smještanjem istog u zaseban skafold, što je svojevrstan nedostatak budući da ne znamo bi li skafold bio bolji ukoliko bi samo prihvatili zamjenu pozicija dvaju kontiga bez uklanjanja jednog od njih. Kako god, iako ima mnogo nedostataka, naivna metoda ipak pruža dobro idejno rješenje koje će bez nekih većih modifikacija biti usvojeno i u svim ostalim metodama.



Slika 4.5: Jednostavan primjer povezanog parcijalnog skafolda. Uklanjanje čvora B uvjetuje nepovezanost budući da jedini put s kraja na kraj (od rubnog kontiga B_1 do rubnog kontiga B_2) i to takav da se posjete svi unutarnji čvorovi vodi preko čvora B.

4.3.5. Osvrt na rezultate naivne metode

Naivna metoda je testirana na 5 simuliranih testnih skupova: vinskoj mušici (lat. *"drosophila"*) – za biblioteku 10k, 20k, nematodi (lat. *"c. elegans"*) – biblioteke (10k, 20k) te čovjeku. Dobiveni skafoldi validirani su GAGE cjevovodom (vidi poglavlje *4.4. Validacija koristeći GAGE cjevovod*). Od realnih skupova odabran je genom čovjeka. U praksi je pokazano da kod obije vrste odabranih skupova metoda značajno smanjuje broj pogrešaka te povećava mjeru ispravljeno N50 za i do 70%, što nam daje dobru okvirnu sliku o valjanosti postupaka temeljenih na sličnim heuristikama. Radi ilustracije neki od rezultata priloženi su u nastavku.

Tablica 4.1: Rezultati naivne metode primjenjene na genom nematode – 10k ($lat.$"$c.elegans$"). U prvom retku nalaze se statistike izračunate na skafoldima generiranima OPERA-om, dok su u drugom retku prikazane statistike nakon obrade OPERA-inih skafolda naivnom metodom. Iz rezulata je jasno vidljivo da metoda uklanja velik dio pogrešaka te pozitivno utječe na mjeru ispr. N50, no također popratno povećava broj velikih procjepa te srednju pogrešku procjepa što je i očekivano budući da procjepe reastimiramo naivno te uklanjamo određen broj kontiga na čijem se mjestu nakon uklanjanja pojavljuje procjep. Štoviše, srednja je pogreška procjepa nakon obrade porasla s 159.57 na 400.69.

Metoda	Indeli	Inverzije	Translokacije	Relokacije	Ispr. N50
OPERA	101	0	0	201	1007522
Naivna m.	35	0	0	63	1294918

Tablica 4.2: Rezultati naivne metode primjenjene na genom vinske mušice – 10k ($lat.$"$drosophila$"). Kao i u slučaju prethodnog testnog skupa, metoda ponovo uklanja velik dio pogrešaka, no u ovom slučaju mjera N50 ostaje ne promijenjena.

Metoda	Indeli	Inverzije	Translokacije	Relokacije	Ispr. N50
OPERA	10	0	1	14	2711941
Naivna m.	4	0	1	2	2711941

Tablica 4.3: Prikaz djelovanje naivne metode na skafolde genoma čovjeka (simulirani testni skup). Iz tablice je jasno vidljivo da je postotak kontiga koji se uklanjaju iz svakog pojedinog skafolda vrlo mali (za primjer su dana 4 najdulja skafolda) te iznosi tek nekoliko stotina kontiga po skafoldu zbog čega više od 99% duljine istih ostaje očuvano. Dakle, i uklanjanjem vrlo malog djela sekvence možemo drastično smanjiti ukupan broj pogrešaka nastalih prilikom skafoldinga.

Identifikator	Uklonjeno	Pr. duljina	Poč. duljina	Frakcija
1	111	53208803	53284166	0.998586
2	102	41483916	41557907	0.99822
3	103	39198686	39271268	0.998152
4	106	38474853	38551474	0.998013

4.4. Validacija koriteći GAGE cjevovod

GAGE – (eng. "Genome Assembly Gold – standard Evaluations) predstavlja sustav koji omogućuje evaluaciju dobivenih kontiga/skafolda onda kada je poznat referentni genom. Valja naglasiti da do sada nije razvijena niti jedna univerzalna metoda evaluacije genoma poglavito onda kada nemamo poznat referentni genom, nadalje sve trenutno postojeće metode evaluacije kao primarnu ocjenu kvalitete uzimaju mjeru N50 što je u potpunosti krivo budući da se kod takve procjene u obzir ne uzimaju pogreške prilikom sastavljanja, a mnogi asembleri u praksi žrtvuju točnost u korist postizanja duljih kontiga. Sastavina (eng. "assembli") dobivena na taj način zapravo je vrlo loša bez obzira na postojanje velikog broja dugih kontiga. Uz mjeru N50 GAGE računa i mjeru ispravljeno N50 koja daje nešto bolju ocjenu kvalitete dobivenih kontiga (ili u našem slučaju skafolda) i to tako da se svaki kontig razbije na manje dijelove na mjestu pogreške. Pogreške se detektiraju na način da se kontizi (ili skafoldi) razbiju po procjepima i mapiraju na referentni genom koristeći MUMMER Salzberg et al. (2012). Osim mjere ispravljeno N50 nama su također vrlo bitni iznosi pojedinih tipova pogrešaka. GAGE ukupno definira 4 tipa pogreške:

- Inverzije: dio kontiga ili skafolda je inverzan referentnom genomu, tj. umjesto "ispravno" orijentiranog dijela kontiga umetnut je njegov reverzni komplement.
- Relokacije ("preuređenja" unutar kromosoma): određeni se skafold/kontig premješta na krivu lokaciju unutar kromosoma.
- Translokacije: Slično kao relokacije, samo što se radi o premještanju na krivu lokaciju izvan matičnog kromosoma.

– Indeli: Indel pogreška unutar skafolda govori da je kontig (>200 pb u duljini) ili krivo obrisan ili krivo umetnut.

Sva četiri tipa pogrešaka se mogu svrstati u kategoriju eng. "misjoins" – krivih združivanja, koja predstavljaju ozbiljnu strukturalnu pogrešku te se najčešće javljaju kada asembler pogrešno združi dvije udaljene lokacije na genomu. Cilj nam je svesti broj ovakvih pogrešaka na minimum. Uz spomenute pogreške važni su nam i podaci o procjepima, stoga moramo obratiti pažnju na broj velikih procjepa (većih od 1000 pb) te na srednju vrijednost apsolutnih razlika pretpostavljenih veličina procjepa i stvarnih veličina procjepa unutar genoma (srednja pogreška procjepa). Sve metode implementirane u sklopu ovog rada, dakle, kao osnovnu metodu evaluacije koriste upravo GAGE cjevovod, budući da je vrlo jednostavan za korištenje, a opet daje sve one statistike koje su nam potrebne da bi ocijenili kvalitetu dobivenih skafolda.

Primjer 2. Izlaz GAGE cjevovoda:

```
Scaffold Stats
Total units:  11812
Reference:  100272276
BasesInFasta:  103442107
Min:  201
Max:  13632832
N10:  13632832 COUNT: 1
N25:  12921912 COUNT: 2
N50:  8009142 COUNT: 5
N75:  2849561 COUNT: 10
E-size:7610081.6
Corrected Scaffold Stats
Indels, Inversions, Translocations, Relocations, Gaps >= 1000,
Mean Gap Error, Contigs, Corrected N50, Corrected E-size
67,0,0,123,72,74.17,8,2598658,4413216.54
```

4.5. Ostale metode evaluacije

Osim validacije kvalitete skafolda koristeći GAGE, za svaki testni skup bit će izmjereno vrijeme koje je potrebno metodi da u potpunosti obradi sve skafolde, nadalje, maksimalno memorijsko zauzeće također će biti evidentirano za svaki testni skup, no važno je naglasiti da spomenute veličine ne daju nikakve informacije o kvaliteti same

metode već mjere optimalnost implementacije iste, koja se, naravno, u ovisnosti o raspoloživom vremenu uvijek može poboljšati, poglavito zato što su spomenute metode unutar ovog rada implementirane prvi puta bez znanja hoće li iste uopće funkcionirati te su nadograđivane u hodu što dovodi do činjenice da su neki od modula ipak mogli i biti optimalnije implementirani.

4.6. Iterativna metoda poboljšanja postojećih skafolda temeljena na grafovima – nadogradnja naivne metode

Kao jedno od poboljšanja naivne metode uvodi se iterativna metoda zasnovana na grafovima. Metoda je prozvana iterativnom budući da za razliku od naivne metode koja kroz svaki parcijalni skafold/skafold prolazi samo jednom iterativna višestruko prolazi (iterira) kroz isti parcijalni skafold sve dok je zamjenama neka dva kontiga unutar parcijalnog skafolda moguće utvrditi postojanje nejednoznačnih regija ili zbog određenih ograničenja na povezanost grafa više nije moguće vršiti izmjene. Ovom je metodom ujedno i dokazano da veći broj pogrešaka u skafoldima nastaje upravo zbog nejednoznačnosti koja se najčešće javlja pri pokušaju smještanja manjih kontiga o čijoj točnoj lokaciji nemamo dovoljno informacija dobivenih iz uparenih učitanja. Valjda naglasiti da sve pogreške ipak nisu posljedica nejednoznačnosti unutar pojedinih regija, no nažalost, još uvijek ne znamo što ih uvjetuje.

4.6.1. Osnovno načelo rada

Kao što je već spomenuto u prethodnom poglavlju iterativna se metoda zasniva na naivnoj s tom razlikom da se svaki parcijalni skafold (ili čitav skafold ukoliko ne postoje rubni kontizi) višestruko obrađuje i to upravo onoliko puta koliko je potrebno da se iz skafolda izbace sve nejednoznačnosti. Naravno, ponekad uklanjanje svih nejednoznačnosti ipak neće biti moguće budući da bi progresivnim izbacivanjem kontiga iz skafolda mogli izgubiti povezanost skafold grafa. Kako se to ne bi dogodilo prije svakog izbacivanja kontiga iz skafolda provodimo BFS između unutarnjih čvorova parcijalnog skafolda (unutarnji članovi parcijalnog skafolda su svi oni koji se nalaze između dva promatrana rubna kontiga. Ukoliko rubni kontizi ne postoje, tada se svi kontizi unutar skafolda tretiraju kao unutarnji) i to tako da kontig koji planiramo ukloniti označimo "nevidljivim" te promatramo je li postoji put između rubnog kontiga B_i do rubnog

kontiga B_j (koji ne vodi preko nevidljivog kontiga) i to takav da se posjeti svaki od unutarnjih kontiga. U praksi se pokazalo da će to velikom broju slučajeva biti moguće, no i kada nije moguće obići sve čvorove, moguće je doći s kraja na kraj te se tada rodila ideja o uklanjanju čitave komponente povezanosti grafa koja će biti obrađena u zasebnom poglavlju.

4.6.2. Algoritam

Uz manje modifikacije, pojednostavljeni algoritam ove metode u osnovi je sličan algoritmu naivne metode, naime naivna metoda je poslužila kao osnovni gradivni blok za sve ostale metode. Inicijalna filtriranja bridova kao i strukture podataka nad kojima iterativna metoda operira jednake su onima koji se koriste i kod naive metode stoga neće biti posebno razmatrane. Nadalje, boldani dijelovi algoritma ne postoje kod naivne metode te su karakteristični za ovu metodu. Kao dodatak metodi uvodi se i mogućnost kontrakcije grafova definiranjem parametra t koji nam samo govori da čvorove na udaljenosti $\leq t$ treba tretirati kao jedan super-čvor (za detalje vidjeti poglavlje *4.7. Dodatna poboljšanja - problem "bliskih kontiga"*).

Algoritam 2: Pojednostavljeni algoritam iterativne metode.

Input : Objedinjena klaster datoteka, datoteka sa standardnom devijacijom i srednjom vrijednošću biblioteke - lib.txt, datoteka s kontizima, parametar t

Output: Obrađena skafold datoteka, datoteke s nužnim statistikama

Data: skafold graf - (dobiven iz klaster datoteke), kontig graf - skafold (dobiven iz skafold datoteke)

Filtriraj neželjene bridove; Generiraj skafold graf; Generiraj skafold;

if *(*$t! = NO\_THRSH$*)* **then**
    spoji kontige (C_i, C_j) - za koje vrijedi $d_{i,j} < t$ (kreiranje super-čvorova); osvježi bridove skafold grafa;
else
    preskoči kreiranje super-čvorova

if *(rubni kontizi postoje)* **then**
    particijoniraj skafolde po rubnim kontizima;
else
    za svaki skafold kreiraj virtualni parcijalni skafold, postavi rubne kontige na NULL;

while *(ne obiđeš sve parcijalne skafolde)* **do**
    while *(postoje nejednoznačne regije* || *ima promjena u skafoldu)* **do**
        kreiraj matricu bridova i distance za trenutni parcijalni skafold;
        for $i \leftarrow 0$ **to** $duljina(polje\_unutarnjih\_kontiga)$ **do**
            zamjeni ($kontig(i)$, $kontig(i+1)$);
            osvježi podatke u matrici distance;
            provjeri uvjet za valjane(harmonične) bridove;
        if *(uvjet zadovoljen)* **then**
            prihvati zamjenu; učini kraći kontig nevidljivim;
            BFS om provjeri povezanost grafa zanemarujući sve bridove koji idu preko nevidljivog kontiga;
            if *(graf povezan)* **then**
                prihvati zamjenu $\rightarrow$ detektirana je nejednoznačna regija;
                ukloni kraći kontig; evidentiraj promijenu unutar skafolda;
            else
                poništi zamjenu; vrati matricu distance u početno stanje; $(i++)$;
            ukloni kraći kontig iz skafolda $\rightarrow$ detektirana je nejednoznačna regija;
        else
            vrati matricu distance u početno stanje; $(i++)$;
    if *(ima promjena unutar parcijalnog skafolda* **then**
        reastimiraj veličine procjepa;
    else
        pređi na sljedeći parcijalni skafold;

4.6.3. Utvrđivanje veličine procjepa

Kao što je vidljivo iz iznad priloženog algoritma, ukoliko je došlo do bilo kakvih promjena unutar parcijalnog skafolda, valja ponovno izračunati veličine procjepa između odgovarajućih kontiga. U tu svrhu koristit ćemo metodu koja je implementirana i unutar skafoldera OPERA, a zasnovana je na $Goldfarbi Idnani jevoj$ metodi za rješavanje striktno konveksnih kvadratnih funkcija (Goldfarb i Idnani (1983)).
Kod ovog koraka ponovo se koriste ograničenja proizašla iz uparenih očitanja (bridovi skafold grafa). Naravno, prilikom re-estimacije procjepa za određeni parcijalni skafold, mi nećemo u obzir uzimati sve bridove, već samo one koji povezuju kontige unutar promatranog parcijalnog skafolda budući da smo sigurni da ne postoje skladni bridovi koji premošćuju više parcijalnih skafolda. Svaki brid predstavlja jedno ograničenje.
Osnovna pretpostavka kojom ćemo se voditi prilikom re-estimacije procjepa je da se bridovi skafold grafa (tj. veličina istih) međusobno nezavisni te se ravnaju prema normalnoj razdiobi s parametrima (μ_i i σ_i). Tada za skup bridova E te skup veličina procjepa G problem možemo reformulirati na način da nam je cilj odrediti najizglednije parametre μ_i i σ_i takve da uzorkovanje primjera (brida) E_i bude najvjerojatnije moguće. U nedostatku dodatnih spoznaja jedino što možemo pretpostaviti je da je baš za naš uzorak E (skup bridova) bilo najvjerojatnije da bude izvučen iz populacije i da je to razlog zašto je realiziran baš taj uzorak. Budući da su bridovi E_i iid (eng. "$independently\ and\ identically\ distributed$") gustoća vjerojatnosti uzorka E, jednaka je umnošku gustoća vjerojatnosti pojedinačnih primjera. Dakle, za izračunavanje veličine procjepa usvajamo princip najveće izglednosti, stoga imamo:

$$\max_G p(E|G) = \max_G \Pi_{i \in E} = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(s_i(G)-\mu_i)^2}{\sigma_i^2}} \tag{4.2}$$

Gdje je E skup bridova skafold grafa, G skup procjepa, a $S_i(G)$ uočena separacija za brid i utvrđena iz veličina procjepa. Ukoliko je c_i ukupna duljina sekvenci kontiga razapetih bridom, a G_i skup procjepa također razapetih istim bridom tada možemo problem re-estimacije procjepa preformulirati na način da umjesto maksimizacije funkcije iznad, minimiziramo sljedeću kvadratnu funkciju:

$$\sum_{i \in G} \frac{((c_i + \sum_{j \in G_i} g_j) - \mu_i)^2}{\sigma_i^2} \tag{4.3}$$

što unutar implementacije i činimo. Za ovaj optimizacijski problem pokazalo se da ima pozitivno definitnu matricu Q s jedinstvenim rješenjem koje se može pronaći uz

pomoć "$Goldfarb\,i\,Idnanijeve$" metode u polinomnom vremenu.

4.6.4. Rezultati i diskusija

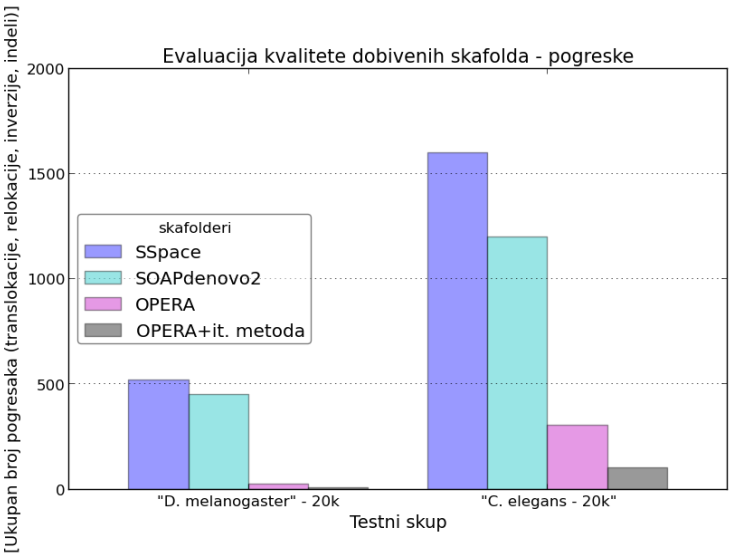
U svrhu dokaza da metoda valjano obavlja svoj zadatak priloženo je više tabličnih prikaza iz kojih je jasno vidljivo da ista značajno smanjuje ukupan broj pogrešaka. Radi usporedbe unutar svake tablice priloženi su rezultati skafoldera OPERA za kojeg je poznato da na odabranim skupovima daje bolje rezultate od svih ostalih postojećih metoda.

Tablica 4.4: Rezultati djelovanja iterativne metode na 7 odabranih testnih skupova. Iz priložene tablice je jasno vidljivo da uklanjamo vrlo mali dio sekvence, dok ćemo u prikazima što slijede pokazati da je to dovoljno za značajno smanjenje ukupnog broja pogrešaka.

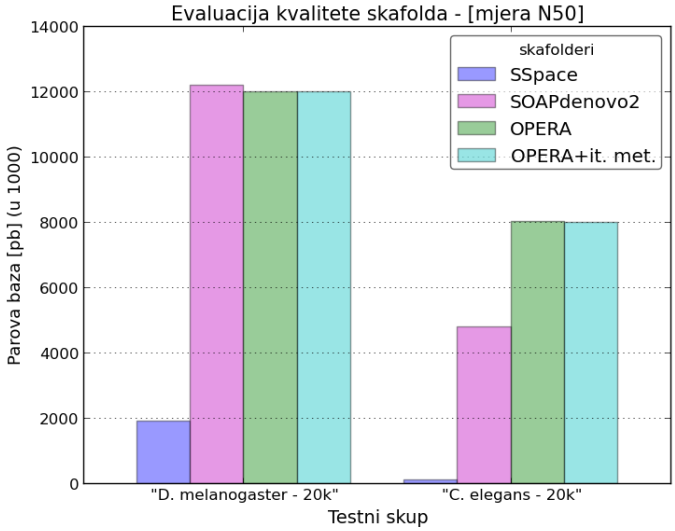
Testni skup	Nejednoznačnih regija ukupno	Uklonjeno regija[duljina%]
"$drosophila-10k$"	428	358[0.313%]
"$drosophila-20k$"	261	225[0.145%]
"$drosophila-20k-200bp$"	690	664[0.214%]
"$celegans-3k$"	647	582[0.473%]
"$celegans-10k$"	4293	2829[3.264%]
"$celegans-20k$"	687	647[0.556%]
"$h.sapiens-20k(700bp)$"	9746	8631[0.321%]
"$h.sapiens*-20k$	13189	7696[0.150%]

Tablica 4.5: Rezultati djelovanja iterativne metode na 7 proizvoljno odabranih testnih skupova. Razlika u odnosu na prethodnu tablicu jest ta što sada mjerimo broj teških strukturalnih pogrešaka - indela, translokacija, relokacija i inverzija po testnom skupu. Dodatno, data je i usporedba mjere ispravljeno N50 prije i nakon obrade iterativnom metodom. Ono na što je važno skrenuti pažnju je činjenica da bi velik inicijalni iznos mjere ispr. N50 kod skafoldera OPERA mogao biti posljedica ignoriranja malih kontiga prilikom procesiranja (<500pb) - naime, vrlo je teško samo na temelju informacija iz uparenih očitanja ispravno smjestiti mali kontig što je unutar ovoga rada i pokazano. Jedini testni skup kod kojeg OPERA spušta prag na 200 pb jest "$drosophila - 20k(200bp)$", no iz rezultata u tablici ispod bez obzira na relativno visok inicijalni iznos ispr. N50 se lako može uočiti ponovni porast spomenute mjere u gotovo svim testnim skupovima (izuzetak je "$drosophila - 10k$" budući da je genom iste već inicijalno bio vrlo dobro sastavljen). Zvjezdica kod zadnjeg testnog skupa označava da se radi o realnom testnom skupu, ostali testni skupovi su simulirani.

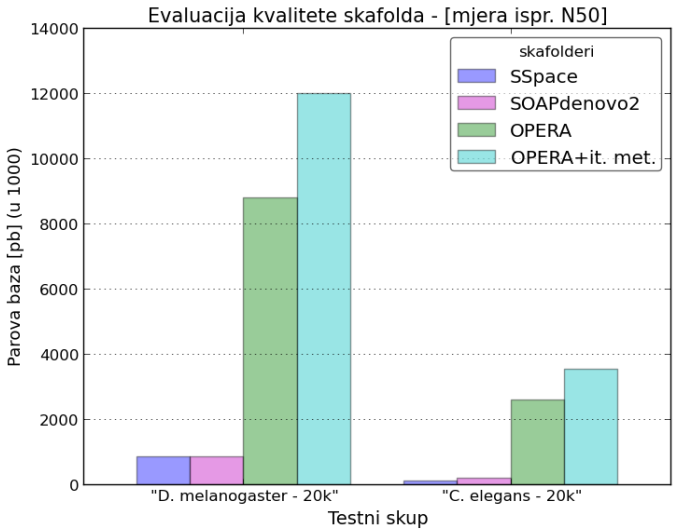
Testni skup	Br. pogrešaka(OPERA/it. met.)	Ispr. N50 (prije/nakon obrade)
"$drosophila - 10k$"	25/**7** [-72.00%]	2711941/**2711941** [+0.00%]
"$drosophila - 20k$"	60/**14** [-76.67%]	8789252/**11982009**[+26.64%]
"$drosophila - 20k(200bp)$"	262/**137** [-47.70%]	2246959/**3165815**[+29.02%]
"$celegans - 3k$"	66/**36**[-45.45%]	1867201/**1868348**[+0.06%]
"$celegans - 10k$"	302/**101** [-66.55%]	1007522/**1290796**[+21.94%]
"$celegans - 20k$"	190/**84** [-55.78%]	2246959/**3535185**[+36.44%]
"$h.sapiens - 20k(700bp)$"	12209/**9570**[-21.61%]	1457082/**1085971**[-25.46%]
"$h.sapiens * -20k$"	16621/**12304** [-25.97%]	649642/**668977** [+2.89%]



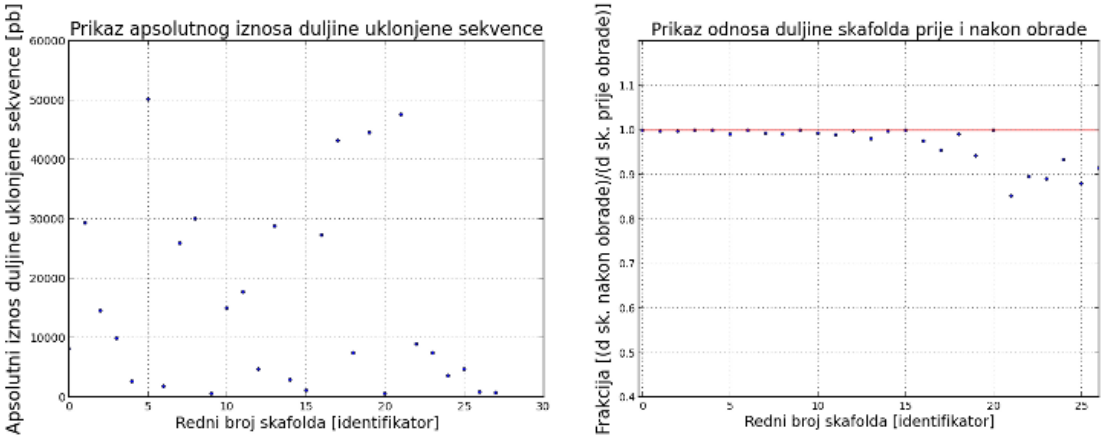
Slika 4.6: Prikaz ukupnog broja pogrešaka (indela, translokacija, relokacija te inverzija) za 3 proizvoljno odabrana skafoldera: SSPACE, SOAPdenovo2 i OPERA. Sivi stupci predstavljaju kombinaciju OPERA-e i iterativne metode. Pogreške su mjerene na skafoldima generiranima iz dva testna skupa - "$drosophila" - 20k$ i "$c.elegans" - 20k$. Iz prikaza je jasno vidljivo da su skafoldi dobiveni skafolderom OPERA puno manje podložni pogreškama od skafolda dobivenih s preostala dva skafoldera, štoviše, ukoliko se na iste primjeni iterativna metoda, broj pogrešaka na oba testna skupa postaje zanemariv.



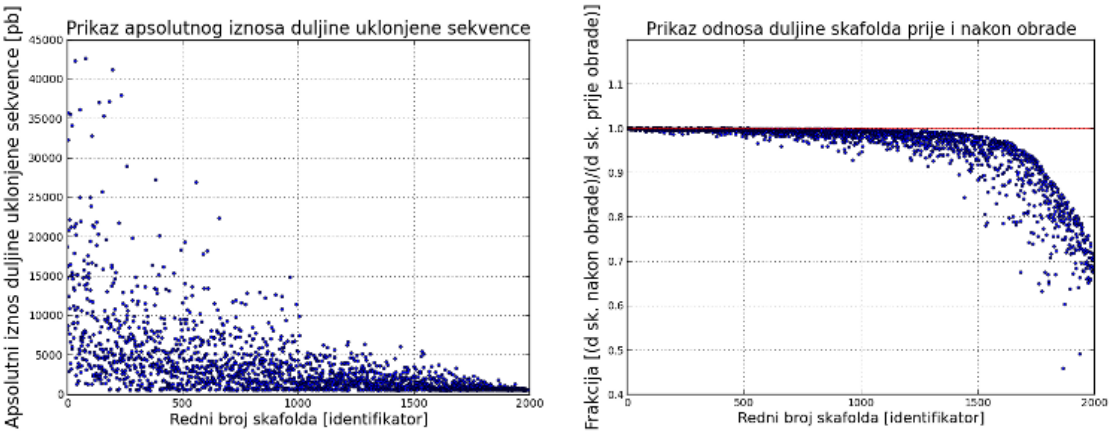
Slika 4.7: Prikaz iznosa mjere N50 za 3 proizvoljno odabrana skafoldera: SSPACE, SOAPdenovo2 i OPERA, koja je također mjerena na skafoldima generiranima iz dva testna skupa - "$drosophila$" $-$ $20k$ i "$c.elegans$" $-$ $20k$. Svijetlo plavi stupci, predstavljaju kombinaciju OPERA-e i iterativne metode. Iz prikaza je jasno vidljivo da najveći iznos mjere N50 postiže SOAPdenovo2, no to zapravo ništa ne govori o samoj točnosti dobivenih skafolda, budući da mjera N50 samo ukazuje na kontinuiranost. Dakle, forsiranje visoke mjere N50 katkad može dovesti do vrlo loših skafolda što će biti pokazano u idućem grafičkom prikazu. Najslabiji iznos N50 postiže SSPACE. OPERA i OPERA u kombinaciji s iterativnom metodom daju vrlo slične iznose N50, iako je OPERA za oba testna skupa neznatno bolja.



Slika 4.8: Prikaz analogan prethodnom s tom razlikom što se sada umjesto mjere N50 prati iznos mjere ispravljeno N50. Iz istog je jasno vidljivo da velik iznos N50 može biti vrlo kriv pokazatelj kvalitete sastavljenih skafolda budući da su sada oni skafolderi koji su forsirali velik iznos N50 pali na samo začelje. Kombinacija OPERA + iterativna metoda se u ovom slučaju pokazala kao daleko najbolja.



Slika 4.9: Utjecaj iterativne metode na veličinu skafolda testnog skupa lat "$c.elegans$" $-20k$. Skafoldi su poredani prema veličini (veći skafoldi imaju manji identifikator), te oni unutar kojih nije došlo do nikakvih promjena nisu niti prikazani. Na desnom grafu je ucrtan pravac $y = 1$ na kojem ćemo dobiti točku ukoliko je veličina skafolda prije procesiranja jednaka veličini skafolda nakon procesiranja, prisjetimo se, takvi skafoldi nisu prikazani što objašnjava zašto imamo "rijedak" graf, nadalje iz grafičkih prikaza je jasno vidljivo da i kod ovog testnog skupa veličina skafolda ostaje gotovo ne promijenjena budući da gotovo sve prikazane vrijednosti "plešu" oko pravca $y = 1$. Najveće se varijacije javljaju kod velikih skafolda što je očekivano.



Slika 4.10: Prikaz omjera duljina sekvence unutar svakog pojedinog skafolda prije i nakon obrade iterativnom metodom za simulirani ljudski genom. Unutar ovog genoma dolazi do većih varijacija, tj. postoje skafoldi iz kojih uklanjamo i do 50% sekvence, no globalno gledano procesiranjem nismo uklonili niti 1% ukupnog sastavljenog genoma (točnije 0.351%)

Detaljan prikaz rezultata po testnim skupovima

Unutar ovog poglavlja (radi lakše usporedbe) detaljno su raspisani rezultati za svaki testni skup - broj indela, translokacija, relokacija, inverzija te iznos mjere ispravljeno N50 za skafolder OPERA i iterativnu metodu. Rezultati su radi jednostavnosti prikazani u tabličnom formatu.

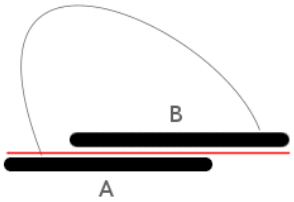
Tablica 4.6: Globalni prikaz objedinjenih rezultata za neke od testnih skupova (broj teških strukturalnih pogrešaka po skupu). U prvom redu svakog retka, nalaze se rezultati dobiveni evaluacijom OPERA-inih skafolda, dok se u drugom redu nalaze rezultati dobiveni evaluacijom skafolda nakon procesiranja iterativnom metodom. Zvjezdica kod zadnjeg testnog skupa označava da se radi o realnom testnom skupu.

Testni skup	Indeli	Inverzije	Translokacije	Relokacije	Ispr. N50
$Drosophila$	10	0	1	14	2711941
$(10k)$	4	0	1	2	2711941
$Drosophila$	30	0	1	29	8789252
$(20k)$	7	0	1	6	11982009
$Drosophila$	76	49	10	127	2246959
$(20k - 200bp)$	41	36	9	51	3165815
$C.elegans$	15	0	0	51	1867201
$(3k)$	8	0	8	28	1868348
$C.elegans$	101	0	0	201	1007522
$(10k)$	37	0	0	64	1290796
$C.elegans$	67	0	0	123	2598658
$(20k)$	22	0	0	62	3535185
$H.sapiens$	7297	60	94	4758	1462141
$(20k - 700bp)$	5901	60	94	3515	1085971
$H.sapiens*$	8085	269	330	7577	649642
$(20k)$	6252	260	320	5472	668977

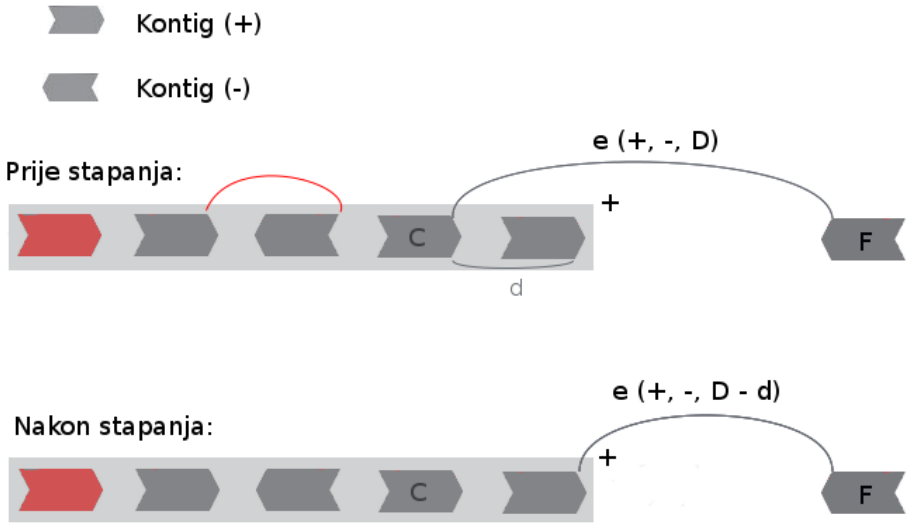
4.7. Dodatna poboljšanja – problem "bliskih kontiga"

Ponekad se prilikom sastavljanja genoma ovisno o asembleru kojim se genom sastavlja (mi ovdje koristimo SOAP) može dogoditi da neki par kontiga koji bi zbog svoje međusobne blizine (vidi **sliku 4.11**) trebao biti sastavljen kao jedan kontig ipak bude sastavljen kao dva zasebna kontiga linkana uparenim očitanjem. U tom slučaju brid koji ih povezuje sprječava da po skafoldu proizvoljno selimo jedan od članova para s

ciljem utvrđivanja postojanja nejednoznačnih regija. Ovakav problem javlja se prvenstveno kod složenijih genoma kao što je primjerice genom čovjeka. Kako bi riješili taj problem, uvodimo prag na udaljenost kontiga potrebnu da bi se neka dva promatrana kontiga tretirala kao dvije zasebne jedinice, nakon čega prolazimo skroz svaki skafold i kreiramo "slijepljene regije" i to tako da prvi kontig unutar skafolda predstavlja korijen regije. Korijen proširujemo novim kontizima dokle god ne naiđemo na kontig čija je udaljenost (desni procjep) od zadnjeg člana regije veća od unaprijed definiranog praga t ili ne postoji brid između promatranog kontiga i bilo kojeg člana regije. Nakon kreiranja regija, iste pohranjujemo u datoteku budući da ćemo naposljetku prije kreiranja skafolda u fasta formatu morati rekonstruirati članove unutar svake pojedine regije. Ono što je bitno napomenuti jest da procjepe unutar regija ne re-estimiramo, izuzev zadnjeg člana regije čiji desni procjep postaje jednak desnom procjepu odgovarajućeg super-čvora.



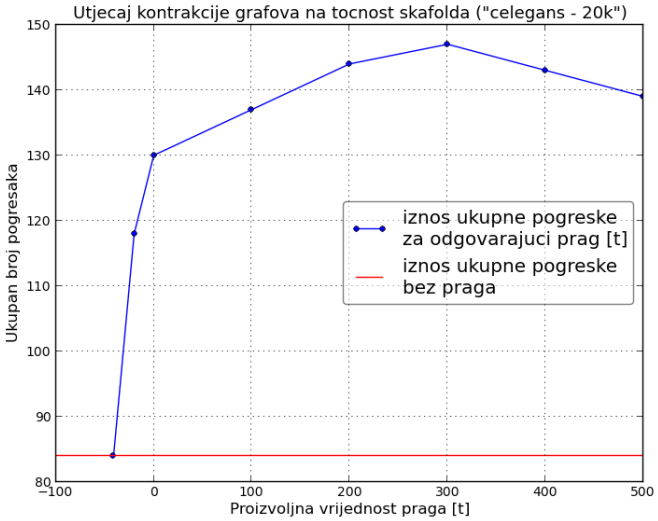
Slika 4.11: Primjer dva međusobno bliska kontiga povezana bridom. Ovako "slijepljene" kontige ne možemo seliti unutar skafolda budući da bi ih trebalo seliti u paru. S ciljem rješavanja tog problema definiramo tzv. "prag bliskosti" koji nam zapravo govori da sve one kontige čija je udaljenost manja ili jednaka pragu prilikom procesiranja tretiramo kao jedan super-kontig.



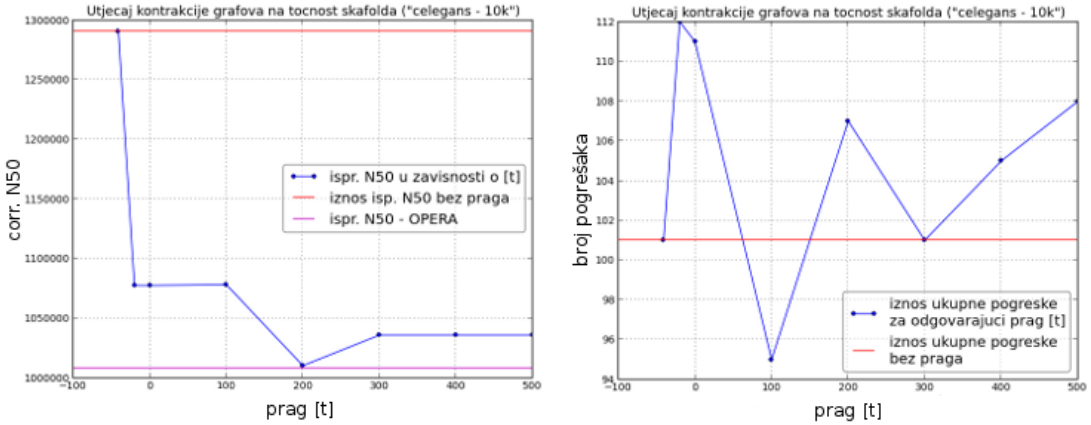
Slika 4.12: Pojednostavljeni primjer kreiranja super-čvorova. Radi jednostavnosti unutar super-čvora istaknut je samo jedan kontig s odgovarajućim bridom. Unutarnji brid označen je crvenom bojom te će nakon kontrakcije skafold grafa nestati. Lijeve i desne bridove valja preusmjeriti na odgovarajuće krajeve super-čvora. Je li brid lijevi ili desni ovisi o orijentaciji unutar super čvora, orijentaciji kontiga u polaznom bridu te činjenici je li promatrani kontig izvorište (START) ili kraj (END) brida. Orijentaciju super kontiga određujemo proizvoljno te će ona u našem slučaju uvijek biti pozitivna BE (na slici +). Kontig označen slovom C sa slike unutar super-čvora ima pozitivnu orijentaciju, nadalje, unutar promatranog brida također ima pozitivnu orijentaciju, a i izvorište je polaznog brida, stoga je brid koji izvire u istom desni i valja ga preusmjeriti na desni kraj super čvora. Globalno gledano postoji 8 kombinacija (orijentacija + pozicija - START/END) na temelju kojih određujemo je li brid lijevi ili desni. Brid je desni ako orijentacija kontiga unutar super čvora odgovara orijentaciji super čvora , te je orijentacija kontiga unutar brida pozitivna a isti početna pozicija ili negativna a isti završna pozicija. Također, brid je opet desni ukoliko se orijentacije super-čvora i kontiga unutar super-čvora ne podudaraju, a orijentacija unutar početnog brida je negativna a promatrani kontig startna pozicija ili pozitivna, a promatrani kontig završna pozicija. U svim ostalim (preostala 4) slučaja brid je lijevi. Jednom kada je utvrđeno radi li se o lijevom ili desnom bridu, istog je moguće preusmjeriti na odgovarajući kraj i to tako da se vrijednosti distance i desnog procjepa ponovno računaju. Nova distanca će biti jednaka početnoj distanci umanjenoj za iznos lijeve (ili desne) udaljenosti ovisno o kojem se bridu radi, dok je sama udaljenost jednaka zbroju duljina unutarnjih kontiga i procjepa između njih. U našem slučaju radi se o desnom bridu pa ga preusmjeravamo desni kraj super čvora. Novi iznos distance brida e postaje jednak iznosu stare distance D umanjene za iznos desnog procjepa kontiga C i duljine zadnjeg kontiga unutar super-čvora $D - d$

4.7.1. Kontrakcija grafa

Svaka se promjena unutar skafolda (kreiranje super-čvorova) reflektira i na skafold graf. Prisjetimo se, skafold i skafold graf jesu dvije međusobno zavisne strukture čiji se čvorovi podudaraju, dakle, prilikom kreiranja super-čvorova stapanjem dvaju ili više bliskih kontiga, dolazi do kontrakcije skafold grafa. Kontrakcijom gubimo sve bridove koji povezuju čvorove C_i i C_j budući da se isti sada nalaze unutar jednog super-čvora $C_{...ij...}$, dok je sve "vanjske" bridove koji izlaze ili ulaze u neki od tvorbenih jedinica super-čvora potrebno preusmjeriti na krajeve super-čvorova vodeći pri tome računa je li polazni brid bio lijevi ili desni (lijeve bridove preusmjeravamo na lijevi kraj, analogno za desne bridove), sukladno tome potrebno je osvježiti i informacije o bridu kao što su: distanca, iznos desnog procjepa te orijentacija (vidi **sliku 4.12**). Usmjerenost brida računamo na temelju orijentacije kontiga (vrha brida) unutar skafolda (koja je jednaka orijentaciji kontiga unutar super-čvora. Radi jednostavnosti orijentaciju super čvora uvijek početno uzimamo kao pozitivnu), inicijalne orijentacije kontiga u bridu te pozicije (početna - brid "izvire" u promatranom kontigu, završna - brid "završava" u promatranom kontigu). Jednom kada znamo usmjerenost svakog brida, isti vrlo lako možemo "preusmjeriti" na odgovarajući kraj super čvora. Kontrakciju grafa obavlja poseban modul "$preprocessor$" te ju je moguće uključiti u proces reskafoldinga, no nije nužno budući da se u praksi pokazalo da kod većine genoma ovakvo smanjenje broja kontiga negativno utječete na kvalitetu skafolda jer kontrakcijom gubimo velik dio informacija a već sama OPERA koristi prag od 500 pb na minimalnu veličinu kontiga unutar skafolda.



Slika 4.13: Prikaz ovisnosti ukupnog broja pogrešaka o proizvoljno odabranom pragu za kontrakciju grafa - t. Horizontalnom crvenom linijom označen je broj pogrešaka koji dobijemo bez da mijenjamo polazni graf. Kod ovog testnog skupa kontrakcija grafova utječe negativno na ukupan broj pogrešaka. Maksimalan prag koji testiramo iznosi 500pb budući da je minimalna veličina kontiga koja se može pojaviti unutar polaznog skafolda upravo 500 pb (OPERA zanemaruje kontige manje od 500 pb zbog čega se isti ne pojavljuju u polaznom skafoldu). Dakle, korištenjem većeg praga zasigurno bi gubili informacije budući da na mjesto procjepa tada zasigurno možemo umetnuti kontig. Mjera ispravljeno N50 nije ucrtana na grafu jer je za sve odabrane pragove konstantna i iznosi 2598648 što je opet neznatno više nego kod inicijalnih skafolda.



Slika 4.14: Prikaz ovisnosti ukupnog broja pogrešaka (desno) i mjere ispravljeno N50 (lijevo) o proizvoljno odabranom pragu - t za testni skup "$c.elegans$" $-$ $10k$. Iz grafičkog prikaza je jasno vidljivo da se najmanja ukupna pogreška postiže za $t = 100$, no pri tom iznosu parametra t vrijednost mjere ispravljeno N50 ipak nije najveća budući da tada imamo veliku srednju pogrešku procjepa. Ne postoji značajnija korelacija između iznosa praga t i vrijednosti ispravljeno N50 (ili broja pogrešaka), naravno, pri odabiru dovoljno velikog praga iznos spomenute mjere počinje padati budući da gubimo informacije iz grafova te stvaramo vrlo veliki broj rubnih kontiga koje zbog svoje veličine nije moguće seliti unutar skafolda.

4.7.2. Kako korištenjem praga t smanjiti broj pogrešaka?

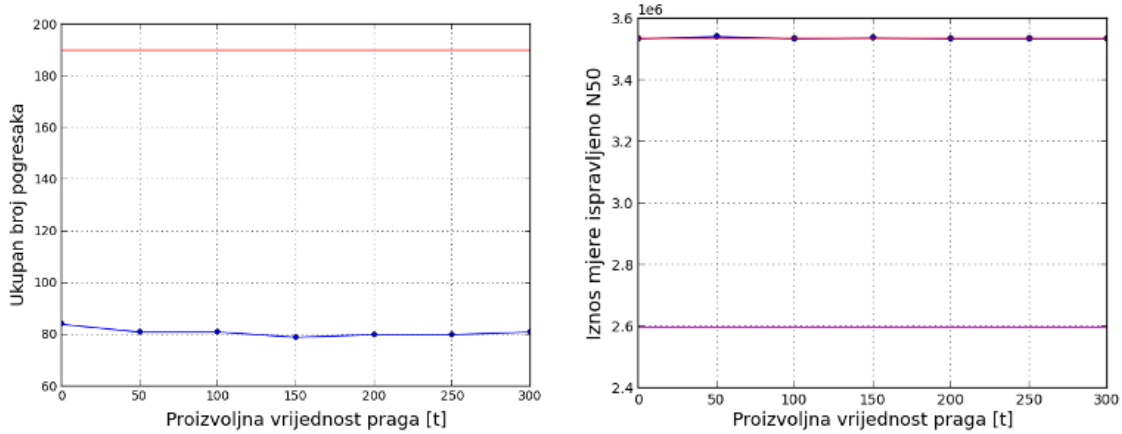
Iz grafičkih prikaza unutar prošlog poglavlja moglo se krivo zaključiti da broj pogrešaka nije moguće smanjiti korištenjem proizvoljnog praga t na udaljenost kontiga.To bi bilo točno u slučaju da iterativnu metodu pokrećemo direktno s pragom, što se u praksi pokazalo da ne smijemo činiti budući da se može dogoditi da dobijemo dva super-čvora unutar kojih već postoje pogreške od inicijalnog sastavljanja (nakon prolaska OPERA-e) stoga bi kreiranjem super čvorova zapravo sačuvali pogreške unutar regija zajedno s regijama. Nadalje, kada kreiramo super-čvorove može doći i do porasta imobilnih regija (pojava slijepljenih super-čvorova). Spomenute probleme rješavamo na način da prvo pokrenemo iterativnu metodu bez praga da riješimo sve "mikroregije", nakon čega isprobavamo različite vrijednosti praga t kako bi pronašli onu vrijednost koja minimizira broj pogrešaka. U ovom slučaju ne čuvamo broj pogrešaka unutar regija budući da znamo da su jednoznačnosti iz polaznih skafolda svedene na minimum.

4.7.3. Pregled rezultata uz optimalnu vrijednost parametra t

Tablica 4.7: Globalni prikaz objedinjenih rezutata za neke od testnih skupova.

Metoda	Indeli	Inverzije	Translokacije	Relokacije	Ispr. N50
"Drosophila-10k"					
OPERA	10	0	1	14	2711941
It. metoda	4	0	1	2	2711941
It metoda [t=100]	1	0	1	1	2711941
"Drosophila-20k"					
OPERA	30	0	1	29	8789252
It. metoda	7	0	1	6	11982009
It metoda [t=100]	4	0	1	6	11982009
"Drosophila-20k$(200bp)$**"**					
OPERA	76	49	10	127	2246959
It. metoda	41	36	9	51	3165815
It metoda [t=100]	39	36	9	47	3165815
"C. elegans-10k"					
OPERA	101	0	0	201	1007522
It. metoda	37	0	0	64	1290796
It metoda [t=200]	30	0	0	50	1025802
"C. elegans-20k"					
OPERA	67	0	0	123	2598658
It. metoda	22	0	0	62	3535185
It metoda [t=50]	22	0	0	59	3542077

Iako smanjuje broj pogrešaka, činjenica je da nam iterativna metoda u kombinaciji s proizvoljnim pragom ipak ne omogućuje da povećamo iznos mjere ispravljeno N50. Razlog za to bi mogao biti taj što kreiranje super-čvorova ne vršimo spajajući čvorove po iznosu distance na bridu već po iznosu desnog procjepa.



Slika 4.15: Varijacije u broju pogrešaka za testni skup "$c.elegans$" $-$ $20k$. Prvo procesiranje vrši se bez praga kako bi se uklonile pogreške unutar "mikroregija". Dobiveni skafoldi postaju ponovni ulaz u metodu s različitim iznosima praga t. Na lijevom grafu prikazana je ovisnost ukupnog broja pogrešaka o pragu t. Crvenom linijom označen je broj pogrešaka dobiven evaluacijom OPERA-inih skafolda. Na desnom grafu pratimo iznos mjere ispravljeno N50 o spomenutom pragu, dok smo crvenom linijom označili iznos mjere prije postavljanja praga. Ljubičasta horizontalna linija predstavlja iznos isp. N50 OPERA-inih skafolda.

4.8. Statistike vezane uz implementaciju

U do sada prikazanim statistikama fokus je bio na pokušaju procjene valjanosti implementiranih metoda budući da za iste inicijalno nije bilo znano niti hoće li funkcionirati. Kako je ovako nešto po prvi puta implementirano unutar ovog rada jasno je da implementacija nije najbolja moguća, štoviše mnoge su odluke i ideje donesene u hodu te trenutno unutar iste ne postoji nikakva paralelizacija što je veliki nedostatak naročito prilikom obrade većih genoma budući da se relativno dugo treba čekati na rezultate. Nadalje, re-estimacija procjepa se odvija u polinomnom vremenu, što također pogoduje sporosti implementacije, no polinomno vrijeme je posljedica metode odabrane za re-estimaciju procjepa, a ne loše implementacije. Kontrakciju grafova (nije vremenski kritična, odvija se samo jednom) trenutno obavlja modul napisan u Pythonu - *preprocessor* zbog čega (ako se korisnik odluči kreirati super-čvorove) ista iziskuje vrlo veliku količinu memorije. U koliko je riječ o većem genomu kao što je genom čovjeka maksimalno zauzeće može biti i do 40 Gb, nadalje više od $\frac{1}{3}$ ukupnog vremena izvođenja se troši na učitavanje podataka iz datoteka te upis podataka u iste.

Tablica 4.8: Statistike vezane uz kvalitetu implementacije (kreiranje super-čvorova se nije izvodilo). Zabilježeno je ukupno vrijeme izvođenja implementacije po svakom testnom skupu (učitavanje i ispis podataka su uključeni u vrijeme izvođenja) te maksimalno memorijsko zauzeće po svakom testnom skupu. Važno je napomenuti da cilj ovoga rada nije bio napraviti maksimalno optimalnu implementaciju već dokazati (ili opovrgnuti) ideju o pogreškama kao posljedici postojanja nejednoznačnih regija unutar skafolda. Mjerenja vezana uz implementaciju vršena su na računalu: *intel xeon x7550 - 512 Gb (RAM).*

Testni skup	Vrijeme izvođenja	Maks. memorijsko zauzeće (RSS)
"$drosophila - 10k$"	19.39 sec	23.59 Mb
"$drosophila - 20k$"	20.27 sec	39.50 Mb
"$drosophila - 20k(200bp)$"	281.61 sec	199 Mb
"$celegans - 3k$"	31.42 sec	100 Mb
"$celegans - 10k$"	39.33 sec	240 Mb
"$celegans - 20k$"	59.26 sec	140 Mb
"$h.sapiens - 20k(700bp)$"	4540.9 sec	9.04 Gb
"$h.sapiens * -20k$"	15754.4 sec	30.5 Gb

4.9. Pokušaj popunjavanja procjepa

Zadnji korak u obradi skafolda jest pokušaj da se procjepi nastali kao posljedica obrade svakog pojedinog skafolda iterativnom metodom popune određenim kontizima. Kako bi dobili statistike koliko je kontiga zapravo moguće vratiti (pokušaj zatvaranja procjepa), netom kreirane skafolde ćemo provući kroz sustav FinIs (Gao et al., 2012) te tako "popunjene" skafolde ponovo validirati GAGE cjevovodom. Rezultati pokušaja zatvaranja procjepa za svaki pojedini testni skup su prikazani u nastavku:

Tablica 4.9: Rezultati popunjavanja svih postojećih procjepa sustavom FinIs.

Dataset	Ukupan broj procjepa	Broj procjepa koje je mog. popuniti
"$drosophila - 10k$"	16798	6876
"$drosophila - 20k$"	5153	4078
"$celegans - 3k$"	19626	16374
"$celegans - 10k$"	?	?
"$celegans - 20k$"	?	?

Tablica 4.10: Rezultati popunjavanja procjepa sustavom FinIs samo za one procjepe koji su generirani tjekom reskafoldinga (sigurni smo da uklanjamo kontige veće od $500pb$, budući da je to prag koji koristi OPERA, stoga na ulaz nikako ne možemo dobiti kontig manji od spomenutog praga). Iz rezultata je jasno da odabrana metoda još uvijek ima problema s popunjavanjem velikih procjepa koji nastaju kao posljedica reskafoldinga, no važno je napomenuti da je ista u razvoju te da se na njoj trenutno aktivno radi.

Dataset	Broj novih procjepa	Popunjeno	Dodano sekvence [ukupno]
"$drosophila - 10k$"	358	0	2701862 bp
"$drosophila - 20k$"	664	0	0 bp
"$celegans - 3k$"	582	0	4707390 bp
"$celegans - 10k$"	?	?	?
"$celegans - 20k$"	?	?	?

Važno je naglasiti da iako je procjepe moguće popuniti, odabrana metoda popunjavanja povećava ukupan broj pogrešaka (posebice indela), stoga skafoldi neočekivano postaju lošiji nego oni inicijalno dovedeni na ulaz iterativne metode. Štoviše srednja pogreška procjepa umjesto da se smanji naraste i za nekoliko redova veličine što uvjetuje drastično smanjenje mjere ispravljeno N50, no budući da odabrana metoda nije razvijena u sklopu ovog rada već je korištena samo u svrhu provjere mogu li se pomoću trenutno postojećih metoda popuniti procjepi nastali uklanjanjem kontiga, njome se detaljnije nećemo baviti. Vrijednosti zamjenjene upitnikom govore da je tijekom obrade došlo do određenih pogrešaka (kao što je segmentacijska greška) te se ista nije mogla provesti u cjelosti. Nadalje, za testne skupove koji fale metoda se još uvijek ne može pokrenuti (još uvijek je u razvoju). Jedno od potencijalnih rješenja problema bi bilo odabrati drugu metodu, no to zbog ograničenog vremena na raspolaganju nije učinjeno.

4.10. Osvrt na metodu zasnovanu na uklanjanju komponente povezanosti grafa - nadogradnja na naivnu metodu

Metoda zasnovana na uklanjanju komponente povezanosti grafa je u osnovi idejno vrlo slična iterativnoj metodi, s krucijalnom razlikom u sljedećem koraku:
Nakon što detektiramo nejednoznačnu regiju i odlučimo koji kontig želimo ukloniti vršimo BFS (kao i kod iterativne metode) da bi utvrdili hoće li uklanjanje kontiga

rezultirati nepovezanim skafold grafom. Ukoliko to jest slučaj, provjeravamo je li tijekom BFS-a ipak bilo moguće doći s jednog kraja skafolda na drugi (primjerice iz rubnog kontiga B_1 do rubnog kontiga B_2). Ukoliko je to moguće učiniti, tada kako ne bi dobili nepovezan skafold graf ne uklanjamo samo jedan kontig već jednu cijelu komponentu povezanosti koje je član i kontig koji uzrokuje nejednoznačnost. Postupak ponavljamo za svaki parcijalni skafold onoliko puta koliko je potrebno da bi se uklonile sve nejednoznačnosti ili dokle god postoji promjena unutar parcijalnog skafolda (katkad nije moguće ukloniti sve nejednoznačnosti zbog uvjeta na povezanost skafold grafa). Svi ostali koraci kod ove metode jednaki su onima unutar iterativne, no ista je metoda u praksi pokazala nešto slabije rezultate od iterativne metode budući da se može dogoditi da prilikom uklanjanja komponenata povezanosti gubimo i one kontige koje ne bi trebali (iste bi u budućnosti bilo moguće seliti unutar skafolda i time smanjiti broj nejednoznačnih regija). Obzirom na svoje relativno slabije rezultate, metodu zasnovanu na uklanjanju komponente povezanosti u nastavku rada nećemo detaljnije obrađivati već će ostati samo idejno opisana.

Tablica 4.11: Rezultati primjene metode zasnovane na uklanjanju komponente povezanosti na testni skup "$drosophila - 10k$". Dobiveni rezultati su jednaki onima koje daje iterativna metoda.

Metoda	Indeli	Inverzije	Translokacije	Relokacije	Ispr. N50
Iterativna	4	0	1	2	2711941
Komp. pov.	4	0	1	2	2711941

Tablica 4.12: Rezultati primjene metode zasnovane na uklanjanju komponente povezanosti na testni skup "$drosophila - 20k$".

Metoda	Indeli	Inverzije	Translokacije	Relokacije	Ispr. N50
Iterativna	7	0	1	6	11982009
Komp. pov.	8	0	1	6	11972306

Tablica 4.13: Rezultati primjene metode zasnovane na uklanjanju komponente povezanosti na testni skup "$celegans - 10k$".

Metoda	Indeli	Inverzije	Translokacije	Relokacije	Ispr. N50
Iterativna	37	0	0	64	1290795
Komp. pov.	44	0	0	70	1268256

Tablica 4.14: Rezultati primjene metode zasnovane na uklanjanju komponente povezanosti na testni skup "$celegans - 20k$".

Metoda	Indeli	Inverzije	Translokacije	Relokacije	Ispr. N50
Iterativna	22	0	0	62	1290795
Komp. pov.	37	0	0	64	3527533

4.11. Ideje koje potencijalno mogu dovesti do dodatnih poboljšanja

4.11.1. Kriterij donje granice

Sve dosadašnje metode za poboljšanje skafolda zasnivaju se na uklanjanju kontiga iz višeznačnih regija, dakle kada obavimo zamjenu kontiga C_i i C_j provjerimo uvjet skladnih bridova te uklanjamo kraći kontig u zaseban skafold, no tim postupkom možda katkada žrtvujemo valjanu zamjenu, tj. ukoliko imamo sekvencu $..C_1, C_2, C_3, A, B, C_4, ...$ i zamijenimo redoslijed kontiga A i B, primjenom prethodno opisanih metoda kao rezultat nikako ne možemo dobiti $..C_1, C_2, C_3, B, A, C_4, ...$ već će ili kontig A ili kontig B (ovisno o svojoj veličini) biti izbačeni iz skafolda u zaseban skafold. Kako bi mogli prihvatiti zamjenu bez izbacivanja manjeg kontiga možemo uvesti novi kriterij na bridove skafold grafa (tzv. "$kriterij\ donje\ granice$"):

$$\sum_{i=1}^{k}(C_i + g_i) > \mu_e + t * \sigma_e(t = 6) \tag{4.4}$$

Iz gornjeg izraza je jasno da je uvjet vrlo sličan uvjetu harmoničnih bridova s tom razlikom što više ne gledamo samo sumu duljina kontiga između neka dva trenutno promatrana kontiga već i iznose njihovih desnih procjepa, dakle u postojeći algoritam bi trebalo dodati i provjeru ovog uvjeta i to ako i samo ako je uvjet harmoničnih bridova zadovoljen. Tada u ovisnosti o tome prolazi li ili ne kriterij gornje granice (kriterij prolazi ukoliko je broj bridova koji inicijalno narušavaju kriterij veći od broja bridova koji narušavaju ovaj kriterij nakon zamjene) zamjenu možemo prihvatiti i smatrati dobiveni skafold boljim od inicijalnog skafolda. Ukoliko kriterij ne bi prošao tek tada bi brisali neki od kontiga iz čega slijedi da bi uklanjali još manji postotak sekvence, a mogli bi dobiti jednako dobre ili čak bolje skafolde. Glavni problem kod ovog pristupa je činjenica da moramo re-estimirati procjepe nakon svake zamjene, a kako se re-estimacija odvija u polinomnom vremenu za kompliciranе genome (kao što je genom čovjeka)

vrlo vjerojatno ne bi niti mogli dobiti rezultate zbog suviše dugog vremena izvođenja. Nadalje, budući da re-estimiramo procjepe nakon svake zamjene, a re-estimaciju nije moguće vršiti sa sto postotnom točnošću svakim bi korakom sve više akumulirali pogrešku početne re-estimacije što bi u konačnici moglo rezultirati lošim skafoldima jer koristimo iznos procjepa pri računanju kriterija donje granice.

4.11.2. Proširenje na "$pacbio$" očitanja

Jedna od ideja za potencijalna poboljšanja također je i korištenje *pacbio* očitanja kao svojevrsnu smjernicu prilikom procesa reskafoldinga. Zbog svoje duljine spomenuta bi očitanja mogla pomoći pri rješavanju nejednoznačnosti i to na način da prije no što počnemo s procesiranjem skafolda, kontige mapiramo na očitanja. Tada bi se naš problem zapravo sveo na pokušaj minimizacije broja neskladnih bridova unutar novodobivenog skafold grafa. Kako podaci (očitanja) nisu bili spremni, a vrijeme za pisanje rada je ograničeno, ideja nije implementirana, već ostaje otvorena kao mogućnost za potencijalna poboljšanja.

5. Zaključak

Unutar ovoga rada dokazano je da se postojeći skafoldi mogu značajno poboljšati koristeći relativno jednostavan kriterij skladnih (harmoničnih) bridova. Identifikacija i uklanjanje nejednoznačnosti pomoću spomenutog kriterija u praksi smanjuju broj teških strukturalnih pogrešaka unutar skafolda za i do 70% kod pojedinih testnih skupova. Naravno, smanjenje pogrešaka za sobom povlači i povećanje iznosa mjere ispravljeno N50 koja je vrlo dobar pokazatelj kvalitete dobivenih skafolda. Kako ovaj rad zapravo predstavlja prvi pokušaj da se koristeći metode zasnovane na grafovima optimiraju već postojeći skafoldi, a rezultati dobiveni unutar istog su obećavajući, prostor za potencijalna poboljšanja je vrlo velik i izazovan, no ne treba smetnuti s uma činjenicu da je trenutak kada ćemo razviti takve tehnologije sekvenciranja i računala da nam bilo kakve heuristike prilikom sastavljanja genoma više neće biti potrebne sve izvjesniji i polako prestaje biti samo dio imaginarne domene, no do tada se ipak moramo služiti raznim matematičkim trikovima i heuristikama kako bi se približili konačnoj formi genoma pojedinih organizama.

Računalne metode za poboljšanje i validaciju sastavljenih genoma

Sažetak

Sekvenciranje jest postupak kojim se unutar određene sekvence utvrđuje točan redoslijed pojedinih baza. Kako genome nije moguće sekvencirati u cijelosti služimo se različitim računalnim postupcima (algoritmima) kako bi rekonstruirali (sastavili) polazni genom. Zadnji korak u sastavljanju genoma predstavlja skafolding koji se u osnovi može svesti na problem utvrđivanja orijentacije te redoslijeda već prethodno djelomično sastavljenih sekvenci - kontiga, tj. pokušavamo smjestiti kontige u kontekst čitavog genoma, naravno, trenutno ne postoje metode koje bi to učinile sa sto postotnom točnošću, stoga unutar ovoga rada razvijamo više heurističkih postupaka koji bi trebali smanjiti broj pogrešaka pri sastavljanju. Svi razvijeni postupci zasnivaju se na činjenici da je katkad vrlo teško jednoznačno odrediti točnu poziciju malog kontiga unutar genoma, stoga ga se može smjestiti na više pozicija u genomu bez da se naruši ijedan od uvjeta na bridove skafold grafa. Dakle, ono što činimo u svakom od razvijenih postupaka jest: detekcija višeznačnih regija i uklanjanje nejednoznačnosti micanjem kontiga koji uzrokuje nejednoznačnost u zaseban skafold. Naravno, prilikom micanja kontiga na njegovoj prvotnoj poziciji stvaramo procjep, stoga je važno da na kraju procesiranja provjerimo je li moguće ponovno popuniti procjepe nastale kao posljedica istog. Implementirane metode validirane su cjevovodom GAGE, a rezultati validacije pokazali su se kao vrlo obećavajući , zbog čega će metode zasigurno poslužiti kao osnova za neka buduća istraživanje te potencijalna poboljšanja.

Ključne riječi: GAGE, OPERA, sastavljanje genoma, sekvenciranje, skafolding, skafold graf

Computational approaches for refinement and validation of genome assemblies

Abstract

Sequencing is the process of determining the right order of nucleotides within the DNA molecule. However, we can not sequence the entire genome of certain organism at once, so we are forced to use different computational approaches (genome assembly methods) in order to reconstruct the initial sequence from the large amount of relatively small sequencing fragments. The last step in genome assembly is called scaffolding. Scaffolding can be visualized as the problem of ordering and orienting contigs and it is a crucial step in the assembly of high-quality draft genomes. Currently, there are no scaffolding methods that can produce 100% accurate scaffolds, so in this work we are introducing different heuristic algorithms which can significantly reduce the number of errors inside the existing scaffolds. All of our approaches are based on the fact that sometimes it is really difficult to correctly place small contigs inside genome which leads us to the problem od ambiguous order that we are trying to slove by detecting ambiguous regions inside scaffolds and removing contigs that are causing the ambiguity. After the removing wrongly scaffolded contigs, we must check if the gap (which was caused by contig removal) can be filled. The developed methods were evaluated using the GAGE pipeline. The results of evaluation seemed to be very promising which led as to the conclusion that our methods would make a good start for some future research and development of other similar approaches.

Keywords: GAGE, OPERA, genome assembly, genome sequencing, scaffolding, scaffold graph

LITERATURA

Marten Boetzer, Christiaan Henkel, Hans Jansen, Derek Butler, i Walter Pirovano. Scaffolding pre-assembled contigs using sspace. *Bioinformatics*, 27(4):578–579, 2011.

Adel Dayarian, Todd Michael, i Anirvan Sengupta. Sopra scaffolding algorithm for paired reads via statistical optimization. *BMC bioinformatics*, 11(1):345, 2010.

Nilgun Donmez i Michael Brudno. Scarpa scaffolding reads with practical algorithms. *Bioinformatics*, 29(4):428–434, 2013.

Lilian TC França, Emanuel Carrilho, i Tarso BL Kist. A review of dna sequencing techniques. *Quarterly reviews of biophysics*, 35(02):169–200, 2002.

Song Gao, Wing-Kin Sung, i Niranjan Nagarajan. Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *Journal of Computational Biology*, 18(11):1681–1691, 2011.

Song Gao, Denis Bertrand, i Niranjan Nagarajan. Finis: Improved in silico finishing using an exact quadratic programming formulation. 7534:314–325, 2012. doi: 10.1007/978-3-642-33122-0_25. URL `http://dx.doi.org/10.1007/978-3-642-33122-0_25`.

Donald Goldfarb i Ashok Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical programming*, 27(1):1–33, 1983.

Alexey Gritsenko, Jurgen Nijkamp, Marcel Reinders, i Dick Ridder. Grass a generic algorithm for scaffolding next-generation sequencing assemblies. *Bioinformatics*, 28(11):1429–1437, 2012.

Martin Hunt, Chris Newbold, Matthew Berriman, i Thomas Otto. A comprehensive evaluation of assembly scaffolding tools. *Genome Biology*, 15(3):R42, 2014. ISSN

1465-6906. doi: 10.1186/gb-2014-15-3-r42. URL http://genomebiology.com/2014/15/3/R42.

Daniel H. Huson, Knut Reinert, i Eugene Myers. The greedy path-merging algorithm for sequence assembly. stranice 157–163, 2001. doi: 10.1145/369133.369190. URL http://doi.acm.org/10.1145/369133.369190.

Sergey Koren, Todd Treangen, i Mihai Pop. Bambus 2 scaffolding metagenomes. *Bioinformatics*, 27(21):2964–2971, 2011.

Mihai Pop, Daniel Kosack, i Steven Salzberg. Hierarchical scaffolding with bambus. *Genome research*, 14(1):149–159, 2004.

Leena Salmela, Veli Makinen, Niko Valimaki, Johannes Ylinen, i Esko Ukkonen. Fast scaffolding with small independent mixed integer programs. *Bioinformatics*, 27(23):3259–3265, 2011. doi: 10.1093/bioinformatics/btr562. URL http://bioinformatics.oxfordjournals.org/content/27/23/3259.abstract.

Steven L Salzberg, Adam M Phillippy, Aleksey Zimin, Daniela Puiu, Tanja Magoc, i Koren. Gage: A critical evaluation of genome assemblies and assembly algorithms. *Genome research*, 22(3):557–567, 2012.

K. Schebye-Alsing, S.Hoffman, A.Frankel, i P.Jensen. Sequence assembly. *Computational Biology and Chemistry*, stranice 121–136, 2009. URL http://www.sciencedirect.com/science/article/pii/S1476927108001497.